Du mouvement à l'interaction et au geste : études, techniques, outils et périphériques

Géry Casiez

N° ordre : 40862

HABILITATION À DIRIGER DES RECHERCHES

présentée par

Géry Casiez

Discipline : Informatique

Spécialité : Interaction Homme-Machine

Du mouvement à l'interaction et au geste : études, techniques, outils et périphériques

12 novembre 2012

Patrick Baudisch	Hasso Plattner Institut, Allemagne	
Michel Beaudouin-Lafon	Université Paris-Sud & IUF	Rapporteur
Kellogg S. Booth	University of British Columbia, Canada	Rapporteur
Stephen Brewster	University of Glasgow, Royaume-Uni	
Nicolas Roussel	Inria Lille Nord-Europe	
Wolfgang Stuerzlinger	York University, Canada	Rapporteur
Sophie Tison	Université Lille 1	

Habilitation à Diriger des Recherches préparée au sein de l'équipe Mint commune à l'Université Lille 1, au CNRS et au centre de recherche Inria Lille

Résumé

L'amélioration des outils informatiques devient davantage tributaire de l'enrichissement et du développement de nouvelles interfaces homme-machines que de l'augmentation de la puissance des machines. Les périphériques et techniques d'interaction font partie intégrante des interfaces homme-machines et sont au cœur de nos travaux. Nos recherches s'inscrivent dans cet axe de développement de l'interaction homme-machine en considérant à la fois les aspects matériels, logiciels et les capacités humaines liés aux interfaces homme-machines.

Dans ce cadre, nos travaux ont mis en évidence l'importance des fonctions de transfert pour les périphériques de pointage indirects et montrent qu'il existe une marge d'optimisation de ces fonctions. La prise en compte des caractéristiques physiques des périphériques, des capacités des utilisateurs et d'une reconsidération de la gestion des périphériques dans les systèmes actuels permettent en effet d'améliorer les tâches existantes et d'effectuer des tâches auparavant impossibles à réaliser avec des techniques d'interaction standards. Nous nous sommes également intéressés aux périphériques d'interaction directs pour lesquels nous avons montré que la prise en compte de l'occultation permet d'améliorer l'interaction sur écrans tactiles, tout comme une meilleure compréhension des capacités de séparation et d'intégration des degrés de liberté améliore les tâches de manipulation 3D sur écrans multipoints.

Le développement de nouvelles interfaces homme-machine est également au cœur de nos travaux. La compréhension des limites des périphériques isotoniques et élastiques existants nous a conduit à développer le périphérique hybride isotonique-élastique RubberEdge. Nos travaux portent également sur l'utilisation du retour tactile avec frottement programmable pour développer de nouvelles techniques d'interaction, l'amélioration des changements de mode sur écrans tactiles avec le développement du périphérique Conté et l'exploitation de l'interaction sur et au dessus d'une surface tactile avec le système Mockup Builder. Le dernier volet de nos travaux concerne l'estimation de la difficulté perçue d'exécution de gestes et l'analyse de la dynamique de gestes pour distinguer les tâches de sélection de celles de saisie d'objets dans le cadre d'interfaces sans contact.

Abstract

The enhancement of computer tools becomes more dependent on the improvement and development of new human-computer interfaces than the increase in the power of machines. Input devices and interaction techniques are a constitutive part of human-computer interfaces and are at the heart of our works. Our research is part of this line of development of human-computer interaction and considers hardware, software and human capabilities related to human-computer interfaces.

In this context, we highlighted the importance of transfer functions for indirect interaction. By taking into account the physical characteristics of input devices, user capabilities and by reconsidering the management of input devices in current systems we have shown how to improve existing tasks and perform tasks previously impossible to do with standard interaction techniques. In the context of direct interaction we have shown how hand occlusion can be accomodated to improve interaction with touch screens, while a better understanding of the capabilities for separating and integrating degrees of freedom can improve 3D manipulation tasks on multitouch screens.

The development of new human-computer interfaces is also at the heart of our work. The limitations of existing elastic and isotonic devices led us to develop an hybrid isotonic-elastic device called RubberEdge. Our work also focuses on the use of tactile feedback with programmable friction to develop new interaction techniques, improve mode switching on touch screens with the development of an input device called Conté and the use of the interaction on and above an interactive surface with Mockup Builder. The last part of our work concerns the estimation of the perceived difficulty of pen gestures and the trajectory and kinematic gesture analysis to discriminate between select, pick and release tasks with free-hand interfaces.

Remerciements

Je remercie tout d'abord Christophe Chaillou de m'avoir fait confiance, guidé, encouragé et soutenu au fil des années et sans qui je ne ferais sans doute pas ce que je fais aujourd'hui. J'adresse aussi mes remerciements à Ravin Balakrishnan pour m'avoir donné la possibilité de travailler dans une des meilleures équipes d'IHM au monde lors de mon postdoc au DGP. Merci également Laurent Grisoni, qui, à l'instar de Christophe pour l'équipe ALCOVE, a su créé au sein de l'équipe MINT des conditions de travail propices à l'épanouissement des idées de chacun.

Je remercie vivement les membres du jury : Michel Beaudouin-Lafon, Kellogg S. Booth et Wolfgang Stuerzlinger pour avoir accepté de rapporter mon travail en dépit d'emplois du temps déjà surchargés ; Patrick baudisch et Stephen Brewster qui ont amicalement accepté d'examiner mon travail. Je remercie enfin Sophie Tison d'avoir aimablement accepté de présider mon jury.

Merci à toutes les personnes avec qui j'ai collaboré ces dernières années, en particulier Daniel Vogel et Nicolas Roussel mais également : Jonathan Aceituno, Michel Amberg, Fabrice Aubert, Mélisande Biet, Ali Choumane, Andy Cockburn, Samuel Degrande, Rémi Cozot, Mathieu Cudmore, Bruno De Araùjo, Jean-Philippe Deblonde, Jérémie Gilliot, Frédéric Giraud, Carl Gutwin, Martin Hachet, Sébastien Hillaire, Joaquim Jorge, Anatole Lécuyer, Fabien Lotte, Damien Marchal, Antony Martinet, Paolo Olivo, Quin Pan, Thomas Pietrzak, Ludovic Potier, Philip Quinn, Jean-Baptiste Sauvan, Betty Semail, Romuald Vanbelleghem, Radu Vatavu et Quan Xu.

Je remercie mes parents de m'avoir toujours soutenu et encouragé au cours de mes études. J'ai également une pensée particulière pour mon oncle Claude subitement disparu cet été.

Enfin, la réalisation de tout ce travail n'aurait pas été possible sans le soutien quotidien de Sonia et de mes deux filles Héloïse et Clarence.

Table des matières

Table des figures

Chapitre

1 Introduction

We are ocular centric, and displays are therefore much more mature. Input is still primitive, and wide open for improvement.

Bill Buxton [1]

Ce constat de Bill Buxton se justifie pleinement quand on observe l'évolution des technologies d'affichage et des périphériques d'interaction ces 30 dernières années. Les tailles et résolutions des écrans n'ont cessé de progresser au fil des années. Ceux-ci atteignent aujourd'hui des densités de pixels où il devient difficile de distinguer à l'œil nu, à une distance typique, les pixels individuels. Ainsi, selon Apple, les MacBook Pro sortis en juin 2012 qui sont équipés d'écrans *Retina display*, ont une densité de pixels de 220 pixels par pouce, densité qui serait supérieure à ce qu'est capable de distinguer l'œil humain. Les performances des cartes graphiques et des techniques de rendu ont également considérablement progressé, si bien qu'il est aujourd'hui possible de créer des images de synthèse avec un rendu photo-réaliste.

Les périphériques d'interaction n'ont pas connu la même progression régulière au fil des années. La souris est le premier périphérique d'interaction qui a permis un changement radical de notre façon d'interagir avec les ordinateurs. Sans elle, les interfaces graphiques ne seraient peut-être pas ce qu'elles sont aujourd'hui. Il a cependant fallu attendre 30 ans entre l'invention de la souris par Douglas Engelbart et son adoption massive en 1995 avec la sortie de Windows 95. Elle est ensuite restée le principal périphérique d'interaction jusqu'au milieu des années 2000. L'évolution des périphériques d'interaction a réellement pris son essor ces cinq dernières années, certainement poussée par les possibilités limitées d'innover en terme d'affichage. Le succès commercial d'un système interactif est aujourd'hui principalement lié à l'innovation qu'il propose en terme de périphérique d'interaction. Le succès de la Wii en 2006 est principalement lié à sa manette de jeu Wii remote qui offrait des degrés de liberté jusqu'alors inédits. Le succès de l'iPhone d'Apple en 2007 est en grande partie lié à la possibilité d'interagir simultanément avec plusieurs doigts. Le succès de la Xbox 360 de Microsoft est fortement lié au périphérique Kinect qui offre de nouvelles façons d'interagir. La recherche académique et industrielle sur les

1. `http://www.billbuxton.com/multitouchOverview.html`, 2007.

périphériques d'interaction n'a jamais été aussi active. Les nombreux articles publiés ces dernières années dans des conférences comme CHI ou UIST sont là pour en témoigner. Cet engouement est certainement entretenu par les facilités de prototypage (à l'aide de cartes comme Arduino) et d'impression de modèles 3D, facilitée par le développement des Fab lab [2]. Le développement d'un nouveau périphérique d'interaction n'est plus réservé aux industriels, il devient à la portée de l'amateur éclairé.

Si la souris et les écrans multipoints ont radicalement changé l'usage des ordinateurs de bureau et des périphériques mobiles, nous pouvons envisager que les prochains changements majeurs dans l'usage des outils informatiques viendront des nouvelles possibilités offertes par les périphériques d'interaction. Il faut cependant être capable d'exploiter ces possibilités en développant des techniques d'interaction adaptées à la tâche, au contenu et au contexte d'interaction. Rappelons qu'une technique d'interaction est la combinaison de périphériques d'entrées et de périphériques de sorties qui comprend les éléments matériels et logiciels donnant un moyen à l'utilisateur d'accomplir une tâche [Tuc04]. Pour être pertinentes, les capacités d'un périphérique doivent permettre de réaliser des tâches auparavant impossibles à réaliser avec d'autres périphériques ou donner la possibilité de réaliser plus simplement ou plus efficacement les tâches existantes. Par exemple, la possibilité de zoomer le contenu affiché à l'écran avec deux doigts sur l'iPhone a permis d'accroître l'utilisabilité de tout un ensemble de tâches sur périphériques mobiles.

L'efficacité d'une technique d'interaction dépend de nombreux paramètres : matériels, logiciels et humains. Des détails qui peuvent sembler au premier abord anodins peuvent se révéler critiques à l'usage. Pour s'en convaincre, il suffit par exemple de comparer l'utilisation de pavés tactiles de différentes marques. Les pavés tactiles sont en apparence similaires et ils permettent tous de contrôler un pointeur, cependant la taille de la surface tactile, le coefficient de frottement de la surface, la résolution de la mesure de position, la qualité de détection des contacts, la latence du système et les fonctions de transfert sont autant de détails qui changent radicalement l'expérience utilisateur. De nombreux utilisateurs renoncent au pavé tactile de certaines marques pour utiliser la souris alors que, pour d'autres marques, la question d'utiliser une souris ne se pose même pas. La qualité du défilement lors de l'utilisation de deux doigts sur des pavés tactiles de différentes marques est également révélatrice de la difficulté de prendre en compte de nombreux paramètres pour réaliser des techniques d'interaction efficaces.

La connaissance et la prise en compte de ces "détails" sont trop souvent négligées par les industriels et les chercheurs. Par exemple, dans le cadre des tâches de pointage sur les machines de bureau, le pointeur système reste le point de jonction entre, d'une part, les parties matérielles et bas niveau logicielles et, d'autre part, les parties haut niveau logicielles. Transférer l'information des périphériques par le pointeur permet de faciliter le travail de chacun. D'un côté les concepteurs de périphériques et systèmes d'exploitation se concentrent sur la gestion des périphériques d'entrée pour contrôler le pointeur, sans se préoccuper des concepteurs de boîtes à outils logicielles et concepteurs d'applications qui, de l'autre côté, se concentrent sur l'affichage d'informations. Et réciproquement.

2. Contraction de l'anglais FABrication LABoratory, traduction : laboratoire de fabrication

Ainsi la majorité des standards et des boîtes à outils logicielles sont centrées sur l'affichage et très peu sur la prise en compte des capacités des périphériques d'entrée. HTML 5 est un exemple typique de standard ou presque tout est conçu pour l'affichage. Pourtant les boîtes à outils gagneraient beaucoup à prendre en compte les caractéristiques des périphériques d'entrée et de sortie, comme nous le verrons à de multiples reprises dans ce document. D'une manière générale, nous verrons que la gestion des périphériques d'interaction par les systèmes d'exploitation, les boîtes à outils et les applications nécessite d'être repensée.

L'augmentation de la puissance des machines n'est plus le facteur majeur qui permettra aux utilisateurs de travailler plus efficacement, compte tenu de la puissance largement sous-exploitée des machines d'aujourd'hui. Les interfaces homme-machines sont devenues le facteur essentiel d'amélioration des outils informatiques, et ce sont les périphériques d'entrée qui permettront cette amélioration. Nous avons aujourd'hui des périphériques très performants. Jamais nous n'avions disposé de périphériques capables de mesurer autant de degrés de liberté. Ces capacités sont pourtant sous-exploitées par manque de connaissance des capacités motrices et cognitives humaines, manque d'outils logiciels adaptés à leur prise en charge et manque de techniques d'interactions adaptées aux contextes et usages d'interaction. Nous verrons que même des périphériques comme la souris méritent une reconsidération en profondeur de leur prise en charge par les systèmes.

Mes travaux sur les périphériques d'interaction ont commencé durant ma thèse avec le développement du périphérique à retour d'effort DigiHaptic [**Cas04**][3]. Ce périphérique n'a pas révolutionné l'usage de l'informatique mais il m'a permis de dégager de nombreuses problématiques qui ont alimenté par la suite mes travaux de recherche. En développant ce périphérique, j'avais en effet travaillé sur les questions de séparation et d'intégration des degrés de liberté, de filtrage de données, de retours isotonique/élastique/isométrique et de leur combinaison ainsi que la question des fonctions de transfert. Et les travaux présentés dans ce mémoire sont dans la continuité de ces questions. Je les ai menés en postdoc à Toronto sous la direction de Ravin Balakrishnan de février à août 2005, poursuivis dans l'équipe projet Inria Alcove de 2005 à 2009 puis l'équipe projet Inria Mint de 2009 à aujourd'hui.

Mes recherches sont d'abord le fruit d'un travail d'équipe. Je tiens à souligner qu'une des particularités des équipes Alcove et Mint est de regrouper des chercheurs en informatique et génie électrique. Ce regroupement de compétences est un atout en dépit de difficultés de dialogue entre des communautés qui ont des objectifs de recherche différents. Premier doctorant en co-tutelle entre le LIFL et le L2EP, j'ai acquis une double culture et compris les enjeux de chacun. Depuis mon recrutement, je me suis attaché à poursuivre et faire fructifier cette collaboration dans des travaux communs. Les différentes publications écrites ces dernières années attestent de la réussite de cette collaboration.

3. Les publications référencées en caractères gras sont celles dont je suis coauteur.

Mon relatif isolement thématique à mon arrivée dans l'équipe m'a incité à poursuivre de manière rapprochée une collaboration avec Daniel Vogel, entâmée lors de mon séjour post-doctoral. Depuis, et parallèlement à cette collaboration, la création de Mint par Laurent Grisoni puis l'arrivée de Nicolas Roussel et Thomas Pietrzak ont permis de développer une équipe centrée sur l'IHM.

Les axes de recherche que j'ai initiés ont été développés dans 4 thèses. Les thèses de Qing Pan [**CVPC07**, Pan08] et Quan Xu [**XC10**, Qua10], co-encadrés avec Christophe Chaillou, et les thèses d'Anthony Martinet [**MCG09**, **MCG10a**, **MCG10b**, **MCG12**, Mar11] et Jean-Philippe Deblonde [**DCG10**, Deb12], co-encadrés avec Laurent Grisoni. Une cinquième thèse en cours, celle de Jérémie Gilliot [**GCR12**], est co-encadée avec Nicolas Roussel. J'ai également pu collaborer dans le cadre des thèses de Mélisande Biet [**BCGLS08**] et Jonathan Aceituno [**RCAV12**]. J'ai par ailleurs co-encadré avec Laurent Grisoni le séjour post-doctoral d'Ali Choumane [**CCG10**], j'ai encadré le séjour doctoral de Bruno De Araùjo [**DACJ12**] et je co-encadre avec Nicolas Roussel le séjour post-doctoral de Ludovic Potier [**PPCR12**]. J'ai enfin collaboré avec Andy Cockburn [**CVBC08**, **QCC$^+$12**] de l'Université de Canterbury en Nouvelle Zélande, Radu Vatavu [**VVCG11**] de l'Université Stefan cel Mare de Suceava en Roumanie, Anatole Lécuyer d'Inria Rennes [**HLCC07**, **HLCC08b**, **HLCC08a**, **SLLC09**] et Martin Hachet [**DACJH12**] d'Inria Bordeaux. Certaines de ces collaborations se sont faites dans le cadre des projets REACTIVE [4] et InSTInCT [5]. J'ai grandement contribué au montage puis à la coordination de ces projets qui ont valorisé certains résultats de nos recherches.

Certains de mes travaux de recherche ont trouvé un "débouché" dans le cadre de mes enseignements. Je suis responsable des unités d'enseignements optionnelles IHM et projet encadré "interface multipoints" de la première année du master informatique. Ces UE, créées avec des collègues, sont aujourd'hui largement suivies et appréciés par les étudiants. J'ai par ailleurs fortement contribué à monter la maquette de la spécialité de master Image Vision Interaction dont je suis responsable pour la mention informatique. Cette formation attire de bons étudiants de toute la France et l'étranger et alimente les équipes de recherche et les entreprises de la région. Convaincu de l'importance des interfaces homme-machines dans l'innovation, j'ai notamment introduit les interfaces multipoints et la reconnaissance de gestes dans mes enseignements.

Ces deux thèmes font partie du chapitre 3 qui détaille les travaux réalisés sur les périphériques d'interaction directs et les interfaces gestuelles. Le prochain chapitre, quant à lui, traite principalement de l'interaction indirecte et des fonctions de transfert en vue de reconsidérer la gestion des périphériques d'interaction indirects dans les systèmes actuels. Ces deux chapitres ont la même structure : ils présentent tout d'abord la problématique de départ et décrivent ensuite les travaux relatifs à cette problématique auxquels j'ai participé. Le chapitre 4 qui conclut ce mémoire présente quelques perspectives ouvertes par l'ensemble de mes travaux.

4. `http://reactive.berck-handicap.com`
5. `http://anr-instinct.cap-sciences.net`

Chapitre

2 Interaction indirecte, filtrage et fonctions de transfert

Le concept de la *manipulation directe* a été formalisé en 1983 par Ben Schneiderman dans le contexte de la métaphore du bureau. Son objectif est de permettre à l'utilisateur de manipuler directement les objets qui lui sont présentés. La manipulation directe implique une représentation permanente des objets d'intérêt et l'utilisation d'actions incrémentales, rapides et réversibles. Un exemple de manipulation directe est le redimensionnement d'une image en déplaçant ses bords ou ses coins, plutôt que de saisir ses dimensions en utilisant des boîtes de dialogue. La manipulation directe permet de réduire l'apprentissage tout en permettant d'accomplir les tâches plus rapidement. En 1983, la manipulation directe ne pouvait se faire que par le biais d'une souris et d'un pointeur. Depuis sont apparus les écrans tactiles qui permettent d'interagir de manière co-localisée, sans l'intermédiaire d'un pointeur. Pour distinguer ces deux formes d'interaction, nous qualifierons l'interaction co-localisée d'*interaction directe* tandis que les autres interactions seront dites *indirectes*. Les deux types d'interaction permettent, bien entendu, une manipulation directe.

L'interaction indirecte introduit une séparation physique entre l'espace moteur, lieu physique où l'utilisateur interagit, et l'espace visuel où le pointeur, et plus généralement l'objet d'intérêt, est représenté. Cette séparation physique permet de modifier la relation entre ce qui est mesuré dans l'espace moteur et ce qui est contrôlé dans l'espace visuel. Cette relation correspond à la notion de fonction de transfert qui décrit les relations entre les espaces moteur et visuel. Les grandeurs mesurées dans l'espace moteur peuvent être des positions ou des vitesses dans le cas de périphériques isotoniques, des forces ou des déplacements dans le cas de périphériques élastiques ou isométriques [Zha95]. Les grandeurs contrôlées dans l'espace visuel sont la position ou la vitesse.

Shumin Zhai a mis en évidence les relations privilégiées qui existent entre, d'une part, périphériques isotoniques et contrôle en position et, d'autre part, périphériques élastiques et isométriques et contrôle en vitesse [Zha95]. Ces travaux ont permis de préciser l'ordre d'une *fonction de transfert* par rapport à la nature du périphérique utilisé. Il existe cependant une infinité de fonctions de transfert pour un ordre donné et à l'exception de quelques travaux aux résultats souvent contradictoires [JC90, AZ01, Gib62, LCF76], l'influence de ces fonctions sur les performances restait mal comprise. C'est principale-

ment pour cette raison que les chercheurs préféraient utiliser des fonctions de transfert constantes dans le cas de tâches de pointage, ignorant les fonctions utilisées par les systèmes d'exploitation, ou alors utilisaient les fonctions de transfert des systèmes d'exploitation sans connaître leur fonctionnement.

Nos travaux partent de ce constat et ont pour objectif de répondre aux problématiques suivantes :

1. Les chercheurs ne connaissent pas ou connaissent mal les fonctions de transfert utilisées par les systèmes d'exploitation modernes, que ce soit pour le pointage ou le défilement. Ce manque de connaissance conduit les chercheurs à utiliser le plus souvent les fonctions les plus simples, c'est-à-dire des fonctions constantes et à traiter ces fonctions dans les plans expérimentaux comme des variables de contrôle, c'est-à-dire qu'elles peuvent influencer une variable dépendante mais ne sont pas étudiées et par conséquent sont maintenues constantes d'une condition expérimentale à une autre. Ces variables de contrôle pourraient en fait se révéler être des variables de confusion et remettre en question les résultats des expériences. Lorsque les chercheurs utilisent les fonctions de transfert des systèmes d'exploitation, ce manque de connaissance entraîne également une mauvaise description des fonctions utilisées, en particulier parce qu'on ne connaît pas les paramètres qu'elles prennent en considération. Nous verrons par exemple que la résolution du périphérique d'entrée n'est le plus souvent pas prise en compte et influence fortement le comportement du pointeur. C'est ainsi un paramètre à décrire pour être en mesure de répliquer une expérience. Nos travaux proposent des outils pour investiguer de manière systématique les fonctions de transfert utilisées par les systèmes ainsi que des librairies pour accéder aux événements bas niveau des périphériques et appliquer les fonctions de transfert. Nous présenterons des travaux effectués sur les tâches de pointage et de défilement.

2. Les fonctions de transfert sont par ailleurs la seule technique disponible sur tous les systèmes qui facilite le pointage ou le défilement. La caractérisation de ces fonctions montrera que les systèmes actuels utilisent par défaut des fonctions dynamiques, que ces fonctions sont différentes entre les systèmes et qu'il existe des différences de performances significatives entre ces fonctions. Nous verrons également que les gains constants souvent utilisés par les chercheurs comme condition de référence pour comparer d'autres techniques présentent de moins bonnes performances que ces fonctions dynamiques. L'étude de ces fonctions et leur optimisation peuvent représenter des gains de performances importants compte tenu de leur utilisation quotidienne par des centaines de millions de personnes. Les propositions de nouvelles fonctions de transfert ou de nouveaux mécanismes pour les gérer pourraient être rapidement déployés à large échelle compte tenu des modifications modérées à opérer sur les systèmes. Enfin, des fonctions de transfert optimisées pourraient réduire l'intérêt de techniques de facilitation du pointage de plus haut niveau, comme celles qui manipulent le gain [WWBH97, CF03, BGBL04, WFL+09], ou alors permettre de développer de nouvelles techniques d'aide au pointage.

3. Les fonctions de transfert pour le pointage ne concernent pas seulement les périphériques isotoniques comme la souris. Les périphériques élastiques et isométriques sont également à prendre en considération. Ils ont l'intérêt de présenter un encombrement minimal et de ne pas nécessiter de débrayage comparé aux périphériques isotoniques. Des travaux antérieurs ont cependant montré que leurs performances sont moindres comparées aux périphériques isotoniques [RS90, KE88, RS90]. Les fonctions de transfert linéaires n'ont cependant jamais été systématiquement étudiées, toute comme l'influence de la raideur du périphérique. Nous montrerons l'influence déterminante des fonctions de transfert pour optimiser les performances de ces périphériques.

4. Les périphériques isotoniques et élastiques ont des propriétés complémentaires : performance plus importante en absence de débrayage pour les périphériques isotoniques, possibilité de parcourir de grandes distances sans débrayage avec les périphériques élastiques. Nous nous sommes intéressés à la façon de combiner ces deux catégories de périphériques en un seul. Nous avons proposé des solutions aux problèmes de combinaison des retours isotoniques et élastiques et proposé des modèles permettant de prédire quand ce type de périphérique devient avantageux.

5. La souris n'a en apparence pas beaucoup évolué depuis 30 ans. L'ergonomie de prise en main est aujourd'hui meilleure, elle possède désormais une molette et des boutons supplémentaires. La propriété qui a le plus évolué est sa résolution de mesure de position qui a été multipliée par un facteur 100 en 30 ans : elle est passée d'environ 100 CPI [1] pour le premier Macintosch à 10000 CPI aujourd'hui pour certains modèles. Pourtant nous verrons que cette résolution est très largement sous-exploitée et nous proposerons de réviser le fonctionnement des systèmes actuels pour permettre d'interagir dans des tâches demandant une précision importante. Nous proposerons notamment d'exploiter la précision motrice humaine et d'adapter les fonctions de transfert au degré de précision nécessaire de la tâche.

6. En amont de l'application des fonctions de transfert, nous nous intéressons aux problèmes de filtrage pour les systèmes interactifs. Certains périphériques comme la souris ne nécessitent pas de filtrage supplémentaire à celui qui est réalisé en interne du périphérique. D'autres périphériques comme le Kinect de Microsoft ou la Nintendo Wii fournissent des données bruitées qui ont besoin d'être filtrées avant de pouvoir être utilisées dans des systèmes interactifs. Les filtres classiques, non optimaux pour les systèmes interactifs, nous ont conduit à proposer une nouvelle technique de filtrage.

Mes travaux sur ces questions ont débuté lors de mon postdoc et se poursuivent aujourd'hui. Deux thèses y ont été associées, l'une a été soutenue [Pan08] et l'autre est sur le point de l'être [Deb12]. Ces travaux ont fait l'objet de huit publications dont je suis coauteur :

– ACM UIST 2007 : [**CVPC07**]
– HCI Journal 2008 : [**CVBC08**]

1. Pour réduire les ambiguïtés, nous utilisons CPI (count per inch) pour parler de la résolution de la souris et DPI (dots per inch) pour parler de celle de l'écran.

– ACM CHI 2008 : [**CV08**] - Honorable mention
– ACM IHM 2010 : [**DCG10**] (article court)
– ACM UIST 2011 : [**CR11**]
– ACM CHI 2012 : [**CRV12**] (article court)
– ACM UIST 2012 : [**RCAV12**, **QCC$^+$12**]

La suite de ce chapitre en présente une synthèse.

2.1 De la grandeur physique à sa numérisation : problèmes de filtrage

Un périphérique d'entrée mesure une ou plusieurs grandeurs physiques [2] par le biais de capteurs. Le signal mesuré et numérisé peut subir des perturbations imprédictibles et indésirables que l'on appelle bruit. Ce bruit de mesure peut être causé par la chaleur, des champs magnétiques, des problèmes de résolution du capteur ou encore des calculs numériques instables.

J'ai été confronté aux problèmes de filtrage dès ma thèse lors du développement du DigiHaptic [**Cas04**]. La position des manettes de ce périphérique était mesurée par un potentiomètre rotatif et un convertisseur analogique-numérique mesurait la tension du pont diviseur. La position mesurée présentait un bruit faible mais le calcul de la vitesse correspondante était fortement bruité. Après avoir testé différentes alternatives, j'avais opté pour un filtre passe-bas du premier ordre tel que décrit à la page 51 de la thèse. Ce filtre était efficace pour réduire efficacement le bruit mais au prix de l'introduction d'un retard. Le réglage de la fréquence de coupure du filtre consistait alors à trouver un compromis entre la réduction du bruit et la minimisation du retard.

Lors de mon séjour post-doctoral au DGP de l'Université de Toronto, j'ai rencontré Daniel Vogel qui travaillait alors sur des techniques mains libres de pointage et de sélection à distance sur grands écrans [VB05]. Il utilisait un système Vicon [3] qui permet de suivre un ensemble de marqueurs avec une résolution de position élevée (sub-millimétrique). Le doigt était utilisé pour définir un rayon dont l'intersection avec l'écran définissait la position du pointeur. Même si la résolution de mesure de la position des marqueurs de la main ne présentait pas de bruit apparent, la technique de ray-casting introduisait un facteur d'amplification élevé (jusqu'à 90) qui rendait la technique difficilement utilisable pour sélectionner des cibles de petites tailles. Daniel Vogel avait alors testé différents filtres sans grand succès : les paramètres des filtres testés étaient difficiles à régler et les résultats n'étaient pas satisfaisants. Je lui avais alors proposé le filtre que j'avais utilisé durant ma thèse. Il était nécessaire de diminuer de façon importante la fréquence de coupure pour réduire la quantité de bruit à un niveau acceptable, ce qui introduisait un retard important et préjudiciable pour l'interaction. Nous avons alors eu l'idée de

2. On appelle grandeur une propriété d'un phénomène, d'un corps ou d'une substance, que l'on peut exprimer quantitativement sous forme d'un nombre et d'une référence. [...] La référence peut être une unité de mesure, une procédure de mesure, un matériau de référence, ou une de leurs combinaisons [VIM12].

3. `http://www.vicon.com`

changer dynamiquement la fréquence de coupure du filtre en fonction de la vitesse de la main. Cette idée se basait sur l'observation que les utilisateurs sont plus sensibles au bruit quand la vitesse de déplacement est faible et plus sensible au retard quand la vitesse de déplacement est élevée. Le filtre obtenu possédait quatre paramètres dont le réglage permettait d'obtenir un bon compromis réduction du bruit/retard.

Nous avons diffusé ce filtre auprès de nombreux chercheurs et nous l'avons utilisé dans de nombreux projets, ce qui nous a permis de l'optimiser sur plusieurs années. L'intérêt suscité par le filtre nous a poussé à le publier pour qu'il soit disponible au plus grand nombre [**CRV12**]. Baptisé « 1€ filter », en hommage au $1 recognizer qui a simplifié la reconnaissance de gestes [WWL07], le filtre permet de régler facilement deux paramètres afin d'obtenir un bon compromis entre réduction du bruit et introduction de retard.

Le 1€ filter est un filtre adaptatif passe-bas du premier ordre : sa fréquence de coupure est adaptée à chaque nouvel échantillon, en fonction d'une estimation de la vitesse ou plus généralement de la dérivée d'un signal. En premier lieu, le filtre prend en compte les possibles fluctuations de la fréquence d'arrivée des échantillons. Même si les périphériques échantillonnent généralement à fréquence fixe, les événements transmis au système ne suivent pas la même cadence. La fréquence de la boucle de traitement des événements d'une application peut aussi varier suivant la nature des calculs à effectuer suite à l'arrivée d'un événement.

Un filtre passe bas du premier ordre peut être écrit de manière discrète en prenant en compte la période d'échantillonnage T_e et une constante de temps τ qui peut s'écrire en fonction de la fréquence de coupure f_c (Equation 2.1). Ainsi la valeur filtrée $\hat{X}_i$ à l'instant i est calculée en utilisant la valeur courante non-filtrée X_i et la valeur précédente filtrée $\hat{X}_{i-1}$, selon l'équation 2.2. Comme la période d'échantillonnage T_e peut être automatiquement recalculée en utilisant les étiquettes de temps des échantillons, f_c est le seul paramètre de configuration de l'équation 2.2. Comme pour n'importe quel filtre passe-bas, diminuer f_c a pour conséquence de réduire le bruit mais d'introduire du retard. Trouver un compromis entre les deux est difficile puisque les utilisateurs apparaissent plus sensibles au bruit à faible vitesse et plus sensibles au retard à vitesse importante. C'est pour cette raison qu'une fréquence de coupure adaptative en fonction de la vitesse fonctionne bien : pour réduire le bruit, une faible valeur de f_c est utilisée quand la vitesse est faible et pour réduire le retard, f_c est augmentée quand la vitesse augmente.

$$\tau = \frac{1}{2\,\pi\,f_c} \tag{2.1}$$

$$\hat{X}_i = \left(X_i + \frac{\tau}{T_e}\hat{X}_{i-1}\right)\frac{1}{1+\frac{\tau}{T_e}} \tag{2.2}$$

$$f_c = f_{c_{min}} + \beta\,|\dot{\hat{X}}_i| \tag{2.3}$$

Nous avons trouvé qu'une simple relation linéaire entre la fréquence de coupure et la norme de la vitesse permet d'obtenir de bons résultats (Equation 2.3). La vitesse $\dot{\hat{X}}_i$ est

calculée à partir du signal brut et de la période d'échantillonnage puis filtrée en utilisant un filtre passe-bas choisi pour éviter les variations brusques de la dérivée dues au bruit. En pratique une valeur constante de 1Hz donne de bons résultats, ce qui laisse deux paramètres à configurer : l'ordonnée à l'origine $f_{c_{min}}$ et la pente β (Equation 2.3). Ces deux paramètres peuvent être facilement configurés en deux étapes. Pour la première étape β est fixé à 0 et $f_{c_{min}}$ est réglé à une valeur par défaut de 1 Hz. L'utilisateur maintient alors immobile ou quasi immobile la partie de son corps (typiquement la main ou un doigt) dont la position est mesurée par le système de suivi et règle $f_{c_{min}}$ pour réduire le bruit tout en gardant un retard acceptable durant ces mouvements à faible vitesse. Pour la seconde étape la partie du corps est bougée rapidement dans différentes directions et β est augmenté pour réduire le retard. Il est important de noter que ces deux paramètres β et $f_{c_{min}}$ ont une relation claire pour l'utilisateur : si le retard à vitesse importante est un problème alors, il faut augmenter β ; si le bruit à faible vitesse est un problème, alors il faut diminuer $f_{c_{min}}$. Les rotations peuvent être filtrées en suivant une méthodologie similaire mais en filtrant l'axe et l'angle séparément ou en filtrant directement des quaternions.

Nous avons comparé les performances du 1€ filter à d'autres filtres (moyenne glissante, filtrage exponentiel simple et double, Kalman) afin d'observer si, pour une réduction de bruit similaire, le retard obtenu pour le 1€ filter était effectivement plus faible. Les détails de la comparaison sont décrits dans l'article [**CRV12**]. Pour résumer, un bruit artificiel était introduit sur la position d'un pointeur de souris et la distance entre la position filtrée et non-filtrée pour chacun des filtres était mesurée. Les filtres étaient réglés préalablement pour réduire le bruit à des niveaux similaires. La figure 2.1 illustre les différences mesurées entre les filtres.

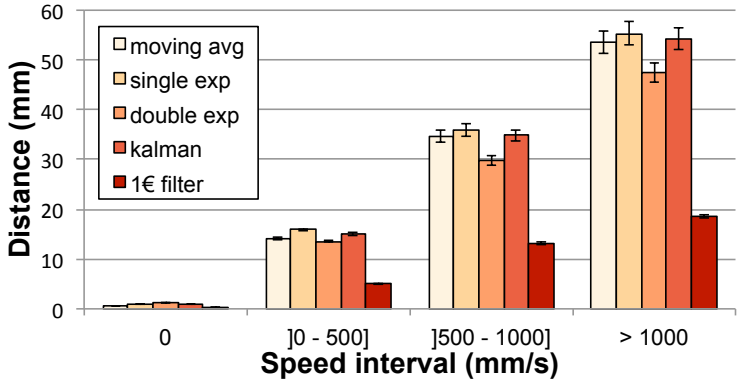


FIGURE 2.1: Distance moyenne entre les positions réelles et filtrées des curseurs pour différents intervalles de vitesses et différents filtres. Les barres d'erreur représentent les intervalles de confiance de la moyenne à 95%.

Le 1€ filter propose une alternative simple et efficace à la moyenne glissante dont l'efficacité est le plus souvent médiocre. Une analyse plus détaillée des filtres reste à réaliser. Idéalement la figure 2.1 devrait représenter en ordonnée le retard de chaque filtre exprimé en secondes. Par ailleurs, les filtres ont été réglés empiriquement pour réduire la quantité de bruit à des niveaux similaires. Une recherche exhaustive des paramètres

des filtres qui réduirait le bruit à des niveaux identiques rendrait la comparaison plus rigoureuse. Enfin, il serait nécessaire de réaliser une évaluation qualitative de la facilité de réglage des filtres dans différents scénarios.

Une fois que le signal d'un périphérique a été filtré, se pose la question de la mise en relation des mouvements dans l'espace physique et ceux dans l'espace virtuel. Cette relation est décrite par une fonction de transfert.

2.2 Fonctions de transfert pour périphériques isotoniques

Les fonctions de transfert prennent tout leur sens en interaction indirecte. J'ai été confronté pour la première fois à la question des fonctions de transfert lors de ma thèse où le problème était de mettre en correspondance le déplacement d'une manette avec celle d'un objet à l'écran. Si le rapport (gain) entre le déplacement à l'écran et celui dans l'espace physique était trop important, il était alors difficile de positionner précisément un objet à l'écran. Si en revanche il était trop faible, il devenait alors nécessaire de mettre en place un mécanisme de débrayage, préjudiciable aux déplacements virtuels de grande ampleur. J'avais alors cherché en vain dans la littérature une méthode pour déterminer une valeur de gain qui permettrait à l'utilisateur d'interagir avec des performances optimales. J'ai eu l'opportunité d'approfondir cette question lors de mon post-doctorat **[CVBC08]**. Les paragraphes qui suivent commencent par présenter un résumé de ces travaux de post-doctorat.

2.2.1 Fonctions constantes et dynamiques

Pour les périphériques isotoniques dans les tâches de pointage, le gain (*Control-Display gain* ou *CD gain* en anglais) est une grandeur sans unité qui met en correspondance le déplacement du périphérique de pointage avec celui du pointeur virtuel qu'il contrôle. Il peut s'exprimer comme le rapport entre la vitesse du pointeur ($V_{pointeur}$) et celle du périphérique ($V_{périphérique}$), toutes deux exprimées dans la même unité.

$$gain = \frac{V_{pointeur}}{V_{périphérique}} \tag{2.4}$$

Des problèmes de quantification peuvent apparaître si la résolution du périphérique d'entrée multipliée par le gain ne permet pas d'atteindre tous les pixels à l'écran. Ce gain maximum peut être calculé comme le rapport entre la résolution du périphérique de pointage et celle de l'écran, exprimées dans la même unité (e.g. DPI).

Nous appelons *espace de travail* l'espace maximal à l'intérieur duquel les mouvements de l'utilisateur pour déplacer le périphérique ne sont pas contraints. Quand le gain est trop faible ou que l'espace physique pour déplacer le périphérique est limité, l'utilisateur peut avoir recours à une opération de débrayage pour déplacer le pointeur sur une grande distance. Le débrayage consiste à repositionner le périphérique dans l'espace moteur sans modifier la position du pointeur. L'espace de travail du périphérique peut être bien

défini, comme les dimensions d'un pavé tactile, ou moins bien défini comme l'amplitude des mouvements confortables des membres ou la place disponible sur un bureau pour manipuler le périphérique.

Comme Jellinek et Card l'ont souligné [JC90], la loi de Fitts ne comprend pas de terme pour le gain. Sachant que cette loi a été établie dans le cadre du pointage direct, où le gain peut être considéré comme égal à 1, ce n'est pas étonnant. Par ailleurs, les manipulations de gain pour un périphérique peuvent être prises en compte indirectement dans la loi de Fitts en considérant chaque valeur de gain comme un périphérique différent. Cela ne veut pas forcément dire que la loi de Fitts est indépendante du gain et l'intégrer comme un paramètre supplémentaire serait utile pour prédire les performances de pointage.

L'état de l'art sur l'influence des gains constants sur les performances utilisateurs est précisément décrit dans [**CVBC08**]. Pour résumer, il n'y a pas de conclusion claire concernant l'effet du gain sur les performances. Accot et Zhai [AZ01] et Jellinek et Card [JC90] ont trouvé que des gains faibles ou importants dégradent les performances, créant un profil du temps de pointage en fonction du gain en forme de U. Gibbs [Gib62] trouva que les performances décroissent avec les gains importants mais Johnsgard [Joh94] observa l'inverse. Langolf et al. [LCF76] ont trouvé que les performances diminuent rapidement, passé une certaine valeur de gain mais Buck [Buc80] n'a trouvé aucun effet. Dans toutes ces études, la plage de gain évaluée était trop faible ou présentait des problèmes de quantification. Par ailleurs, les distances et largeurs considérées étaient réduites.

Les fonctions de gain dynamiques modifient la valeur du gain en fonction de la vitesse du périphérique. Typiquement la valeur du gain augmente avec la vitesse. Ce comportement est motivé par le modèle hybride d'impulsion initiale optimisée pour les mouvements de pointage humains (*hybrid optimized initial impulse motor control model*) [MSK$^+$88, MSK$^+$90]. Ce modèle suppose qu'un premier mouvement, dit balistique, est réalisé en direction de la cible avec une vitesse élevée. Si la phase de mouvement balistique se termine sur la cible, la tâche est terminée ; sinon, une seconde phase, dite de mouvement correctif a lieu. Elle correspond à l'utilisation de vitesses plus faibles en direction de la cible. Plusieurs gestes correctifs peuvent se succéder jusqu'à ce que la cible soit atteinte (Figure 2.2).

Durant la phase balistique, le gain élevé a pour effet de réduire la distance dans l'espace moteur entre la position initiale du pointeur et celle du centre de la cible. Durant la phase corrective, le gain faible a pour effet d'agrandir la taille de la cible dans l'espace moteur. D'autres travaux ont montré l'avantage d'augmenter la taille des cibles dans l'espace moteur [BGBL04] ou dans l'espace visuel [MB02], ou encore d'augmenter la taille des cibles et diminuer la distance dans l'espace visuel [GB05]. Toutes ces techniques nécessitent cependant de connaître la position des cibles, ce qui n'est pas toujours possible. L'adaptation dynamique du gain ne nécessite pas cette connaissance et par conséquent est plus générale.

Les travaux antérieurs sur les fonctions dynamiques sont peu nombreux et leurs conclusions sont une nouvelle fois contradictoires [JC90, Gra96, TD91, MR94], en raison de

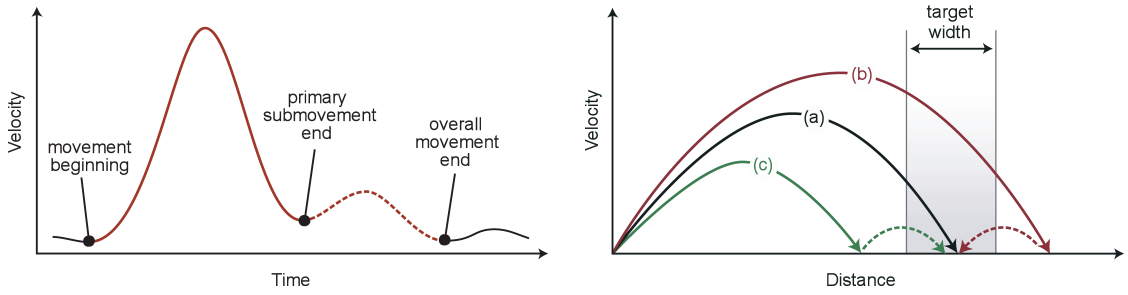


FIGURE 2.2: (gauche) Décomposition du mouvement de pointage en phases balistique et corrective (adapté de Meyer et al., 1988). (droite) (a) correspond à la situation où la cible est atteinte directement à l'issue de la phase balistique. (b) et (c) correspondent aux cas les plus courants où, à l'issue de la phase balistique, la cible n'a pas encore été atteinte (c) ou a été dépassée (b), nécessitant une phase de mouvement correctif.

l'utilisation de fonctions discontinues ou utilisant une plage de gain faible comparée aux fonctions utilisées par les systèmes modernes (Figure 2.3). Les courbes représentées figure 2.3 ont été déterminées à partir de données stockées dans les bases de registres et d'informations publiquement disponibles. Les travaux présentés à la section suivante montreront que les courbes des fonctions de Windows sont correctes alors que les courbes des fonctions de Mac OS X ont la bonne allure mais pas le bon facteur d'échelle.

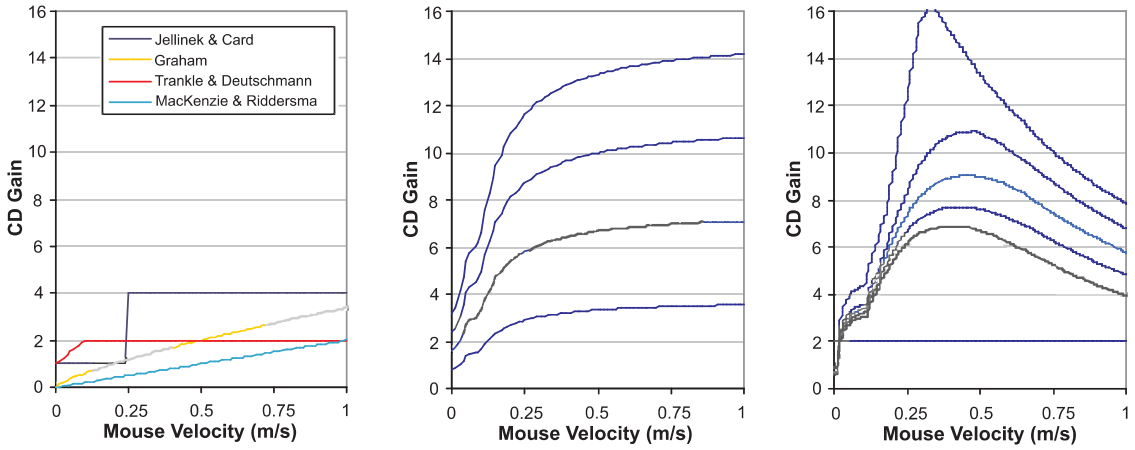


FIGURE 2.3: Représentation de la courbe de gain en fonction de la vitesse du périphérique. Les travaux antérieurs ont utilisé des fonctions discontinues ou restreintes à une plage de gain faible (figure de gauche), en comparaison des fonctions utilisées par les systèmes modernes (au centre plusieurs courbes de Windows XP/Vista et à droite plusieurs courbes de Mac OS X).

D'un point de vue théorique, la réduction de la distance et l'augmentation de la taille de la cible dans l'espace moteur conduisent à une réduction de l'indice de difficulté dans

cet espace. Dans le cas des gains constants, lorsqu'un gain est multiplié par un facteur k, la distance et la largeur de la cible dans l'espace moteur sont divisées par k. Ainsi l'indice de difficulté de la tâche reste identique. Dans le cas d'un gain dynamique, on peut considérer qu'idéalement la fonction va produire un premier gain CD_D durant la phase balistique et un second gain CD_W durant la phase corrective et que la transition entre les phases balistique et corrective s'effectue lors du franchissement du bord de la cible. Dans ce cas, on peut montrer que l'indice de difficulté dans l'espace moteur (ID_{mot}) est réduit par rapport à celui dans l'espace visuel (ID) selon la formule 2.5.

$$ID_{mot} \approx ID + log_2\left(\frac{CD_W}{CD_D}\right) \tag{2.5}$$

Deux expériences ont été réalisées afin d'évaluer l'influence des fonctions de gain constantes et dynamiques sur les performances utilisateurs. La première utilisait un écran d'ordinateur de bureau et la seconde un écran haute résolution de très grandes dimensions. Les expériences ont été conçues pour éviter les problèmes de quantification et permettre de suivre le déplacement des membres des utilisateurs en utilisant un système Vicon. Le détail des protocoles expérimentaux ainsi que ceux des résultats des expériences sont précisés dans l'article **[CVBC08]**. La fonction Windows a été utilisée comme fonction de gain dynamique dans les deux expériences (Figure 2.3).

Les résultats de ces expériences montrent que pour les deux types d'écrans, les fonctions qui présentent des valeurs de gain faibles dégradent de façon prononcée les performances des utilisateurs. Les fonctions qui présentent des valeurs de gain élevées accentuent quant à elles le dépassement de la cible tout en augmentant dans une moindre mesure le temps de pointage. Bien que des travaux antérieurs aient émis l'hypothèse que la bande passante limitée des membres peut expliquer la dégradation des performances pour de faibles valeurs de gain [AZ01], nos résultats indiquent que la vitesse maximale des membres ainsi que le temps de débrayage expliquent mieux cette dégradation. Les gains élevés n'introduisent qu'une légère dégradation des performances. Ainsi le profil représentant le temps de pointage en fonction du gain a plus l'allure d'une forme de L que celle d'une forme de U, évoquée dans des travaux précédents [AZ01, JC90]. D'une manière générale, le gain a peu d'influence sur les performances des utilisateurs, sous réserve que les limites de vitesse et de précision des membres ne soient pas atteintes, tout comme les limites de l'espace de travail des membres ou du périphérique. Ces résultats permettent de dresser plusieurs recommandations pour guider le choix d'un gain constant dans un contexte donné.

Pour acquérir des cibles éloignées sans débrayer, l'utilisateur doit augmenter son espace de travail. Les résultats des deux expériences montrent que l'espace de travail utilisé sans débrayer par les participants est de l'ordre de 36 cm. Il ressort par ailleurs que la vitesse des gestes augmente avec la taille de l'espace de travail, jusqu'à ce qu'une vitesse limite soit atteinte. Ainsi on peut estimer que l'espace de travail maximal (OR_{max}) qui permet à l'utilisateur de travailler confortablement tout en restant en dessous de cette vitesse

limite ne doit pas dépasser 30 cm. En connaissant la plus grande distance à parcourir sur l'écran (D_{max}), on peut en déduire que le gain minimal (CD_{min}) à utiliser est :

$$CD_{min} = \frac{D_{max}}{OR_{max}} \tag{2.6}$$

Le gain maximal utilisable (CD_{max}) correspond à la plus petite valeur entre le gain maximum utilisable compte tenu de la précision des membres humains et la résolution de mesure des périphériques. Le gain maximal utilisable pour une précision donnée des membres (CD_{lmax}) dépend de la taille de la plus petite cible à sélectionner et de la précision des membres. Les expériences menées indiquent que les participants sont capables de contrôler, sans dégradation des performances, des déplacements dans l'espace moteur de l'ordre de 0,2 mm. Dans des travaux plus récents [BWC11], Bérard et al. montrent que ce qu'ils qualifient de résolution humaine, pour une souris, est de l'ordre de 700 à 1400 DPI, soit entre 0.022 mm et 0.011 mm. Le gain maximum pour éviter les problèmes de quantification (CD_{qmax}) s'exprime, quant à lui, comme le rapport de la résolution du périphérique de pointage et celle de l'écran. Pour résumer le gain maximal utilisable peut se calculer comme :

$$CD_{max} = min\left(CD_{qmax} = \frac{Mouse_{res}(DPI)}{Screen_{res}(DPI)}, CD_{lmax} = \frac{W_{min}}{Hand_{res}}\right) \tag{2.7}$$

Une représentation graphique de la plage de gains utilisable est illustrée sur la figure 2.4. Par exemple, en considérant une souris 400 CPI, un écran de 20 pouces avec une résolution de 100 DPI, une distance maximale à parcourir à l'écran de 360 mm, une taille minimale des cibles de 2 mm et un espace de travail de 250 mm, on obtient $CD_{min} = 1,4$ et $CD_{max} = min(CD_{qmax} = 4, CD_{lmax} = 10) = 4$. Si pour un contexte donné, CD_{max} est plus petit que CD_{min}, alors les paramètres de la tâche doivent être modifiés. Par exemple, la taille minimale des cibles peut être augmentée, la distance maximale réduite ou alors la résolution du périphérique de pointage peut être augmentée.

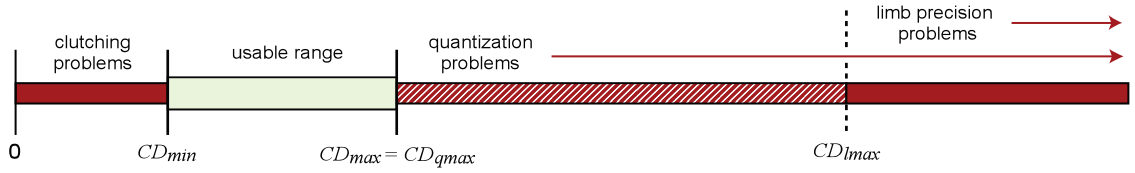


FIGURE 2.4: Représentation de la plage de gain utilisable, dans le cas des fonctions de gain constantes.

Les résultats des expériences ont montré que la fonction de gain dynamique de Windows est en moyenne 3% plus efficace que les gains constants et que l'amélioration des performances est de l'ordre de 6% pour les cibles les plus petites. Ces résultats nous poussent

à encourager les chercheurs à utiliser ce type de fonction comme technique de référence pour comparer d'autres techniques de pointage.

Pour ces travaux, nous avons utilisé la fonction de Windows dont la nature avait été déterminée en utilisant les informations publiques de Microsoft [Poi02] ainsi que les informations de la base de registre. Cette fonction avait été réimplémentée de manière à pouvoir accéder à des valeurs de réglages inaccessibles par l'interface proposée dans le panneau de configuration de la souris. Cependant, le pointeur contrôlé par notre implémentation n'était pas toujours parfaitement en phase avec le pointeur système, révélant qu'il nous manquait des détails d'implémentation non publiés. Nous avions également déterminé l'allure de la fonction OS X en interprétant les valeurs du registre OS X sans avoir eu la possibilité d'expérimenter ces courbes pour vérifier si elles correspondaient effectivement à celles utilisées par le système.

Suite à la publication de cet article, j'ai été contacté en avril 2008 par Mark Cranness, un Néo-Zélandais qui s'intéressait de près aux fonctions de transfert à titre personnel et surtout en tant que joueur. Il avait commencé à écrire un blog [4] et avait découvert mes travaux lors de ses recherches sur le sujet. Il écrit notamment :

> *During all this mouse testing, I spent a lot of time normalizing the pointer speed between the control panel mouse options and the DPI settings in the mouse's hardware. I don't think I realized until now how essential it is to enable mouse pointer acceleration for best pointer "feel" with any mouse. I strongly recommend that you double check to make sure this this feature is enabled. It's available in Control Panel, Mouse, Pointer Options under "Enhance Pointer Precision".*

J'ai eu de nombreux échanges de mails avec Mark qui nous ont conduits à réimplémenter complètement la fonction Windows afin de clairement comprendre son fonctionnement. Ainsi Mark Cranness a constaté une erreur dans les formules utilisées par Microsoft pour rendre la fonction indépendante de la fréquence de mise à jour de l'écran dans les versions de Windows XP et Vista, problème qui a ensuite été corrigé dans Windows 7 [5].

J'ai alors cherché à savoir également de manière précise quels sont les mécanismes utilisés par les systèmes d'exploitation modernes pour transformer les informations envoyées par un périphérique de pointage en déplacement du pointeur à l'écran. Ainsi la fonction de transfert est plus qu'une simple formule explicite dépendant du seul déplacement dans l'espace moteur, c'est à proprement parler une fonction au sens informatique du terme, c'est à dire un algorithme qui va produire un déplacement relatif du pointeur en pixels suite à un déplacement relatif du périphérique de pointage, exprimé en points (*dots* en anglais), et d'autres informations enregistrées lors des précédents appels de la fonction. Si les fonctions d'un système à l'autre peuvent sembler similaires, certaines

4. `http://www.codinghorror.com/blog/2007/10/mouse-ballistics.html`
5. `http://donewmouseaccel.blogspot.com/2009/06/out-of-sync-and-upside-down-windows.html`

personnes éprouvent un réel inconfort avec les fonctions de certains systèmes comme le témoigne l'extrait suivant :

> *So what's wrong with Mac OS X's mouse acceleration curve ? Simply put, it's the wrong shape. For mouse motion to feel natural (at least for most people), the curve has to start by moving upward fairly moderately, then gradually flattening out as the value of X increases. Mac OS X's, curve, however, starts off by being too steep, staying too steep for too long, and then flattening out too abruptly. In practical terms this means that, frequently, as a user tries to use the mouse to move the pointer from point A to point B, the pointer motion feels sluggish. The user then tries to compensate for the sluggishness by moving the mouse faster, and the pointer suddenly goes flying across the screen and overshoots point B. A comfortable and useful curve is actually shaped like a curve. Mac OS X's curve, however, is shaped more like a cliff.* [6]

2.2.2 Caractérisation des fonctions dynamiques

Comme nous allons le montrer par la suite, les fonctions de transfert utilisées par défaut par les systèmes d'exploitation modernes sont dynamiques. Compte tenu de la nature de ces fonctions, elles pourraient interférer avec d'autres techniques de facilitation du pointage, en particulier celles qui manipulent le gain, e.g. [WWBH97, CF03, BGBL04, WFL$^+$09]. Par conséquent, on pourrait s'attendre à ce que les chercheurs travaillant sur la facilitation du pointage essaient de désactiver la fonction de transfert du système pour précisément déterminer son effet, ou alors essaient de systématiquement caractériser son effet.

De nombreux auteurs utilisent un gain constant comme technique de référence mais le plus souvent les détails donnés sont incomplets et ne permettent pas de savoir quelle valeur de gain a été effectivement utilisée. Par exemple, Cockburn et Firth expliquent que le gain *"was set to a constant ratio of approximately 1 :1.6"* dans leurs expériences effectuées sur un système Linux [CF03]. Cependant ils ne détaillent pas comment ils ont réglé le système pour obtenir cette valeur. La technique de pointage prédictive du Delphian Desktop a été évaluée par Asano et al. sur le système Windows XP avec un CD ratio *"set to a constant value of 0.5"* [ASK$^+$05]. Une fois de plus, l'article n'explique pas comment ce réglage a été obtenu. Pour le Bubble Cursor évalué sur le même système, Grossman et Balakrishnan expliquent que *"mouse acceleration was set to 0, with a control-display ratio of 1/2"* [GB05]. La signification exacte de *"set to 0"* est ambiguë, compte tenu de l'interface de configuration de Windows XP (Figure 2.5). Par ailleurs, nous verrons qu'un gain sans unité de 0,5 n'est pas accessible via l'interface. Dans leur étude sur les cibles collantes, Mandryk et Gutwin précisent que *"Windows pointer acceleration was turned off, and the baseline mouse gain was set to the midpoint"* [MG08]. La définition exacte de cette condition de référence est inconnue, ce qui est gênant puisque que les auteurs en font une mise à l'échelle sur 11 valeurs entre 0,05 et 1,0. Dans leur expérience sur les icônes collantes, Worden et al. déclarent que *"normal mouse gain was set at a constant 1 mickey to 3 pixels ratio for all conditions"* [WWBH97]. Cependant, comme les résolutions de l'écran et de la

6. `http://tidbits.com/article/8893`

souris ne sont pas spécifiées, ce gain ne peut pas être exprimé de façon indépendante du matériel, ce qui rend leur étude difficile à comparer à d'autres.

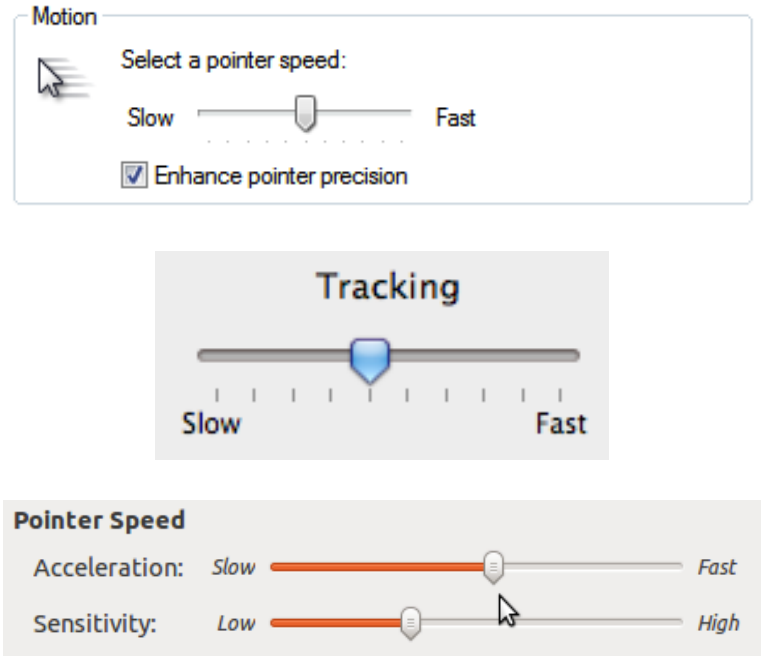


FIGURE 2.5: (haut) Interface de configuration de la fonction de transfert de la souris et du touchpad sous Windows 7. L'interface utilisée sous Windows XP et Vista est identique. Aucune bulle d'aide ou message d'aide n'est associé aux contrôles de l'interface. (milieu) Interface de OS X 10.6.7 pour la souris. Une bulle d'aide est associée au slider disant "Faîtes glisser le curseur pour régler la vitesse de déplacement du pointeur par rapport au mouvement de votre doigt". (bas) Interface de configuration de Ubuntu 10.10. Une page d'aide précise à propos du premier slider : "Use the slider to specify the speed at which your mouse pointer moves on your screen when you move your mouse" et pour le second : "Use the slider to specify how sensitive your mouse pointer is to movements of your mouse".

Définir une fonction de transfert indépendante du matériel, même une fonction constante, est en réalité difficile avec les systèmes actuels. Par exemple, Wobbrock et al. ont eu de grosses difficultés pour désactiver l'accélération du pointeur sous Windows Vista et contrôler dynamiquement le gain pour leur étude sur la Angle Mouse. Ils reconnaissent que *"although some on-line documentation discusses pointer ballistics in Windows, it does not contain sufficient information to establish the slider-to-gain mapping"* [WFL+09]. Un moyen de forcer l'utilisation d'une fonction de transfert est de travailler directement avec les informations envoyées au système par le périphérique de pointage. Par exemple Blanch et al. ont utilisé les coordonnées absolues d'un *puck* sur une tablette graphique comme information d'entrée pour leur technique Semantic Pointing [BGBL04].

D'autres auteurs utilisent les réglages par défaut du système sur lequel ils travaillent. Pour les Ninja Cursors, Kobayashi et Igarashi précisent que *"the mouse speed and accelera-*

tion rate were set to the Windows XP default values (middle speed, no acceleration)" [KI08]. Pour DynaSpot, Chapuis et al. utilisent *"the default X Window acceleration function"* [CLP09]. Cependant, il est difficile de savoir si ces fonctions prennent en compte les caractéristiques physiques des périphériques, les auteurs devraient donner un maximum de détails sur leurs caractéristiques : résolution, fréquence... Un autre problème avec cette approche est qu'il n'est plus possible de répliquer une expérience lorsque le système en question n'est plus disponible.

Dans certains cas, il est simplement impossible de savoir quelle fonction a été utilisée. Par exemple MacKenzie et Isokoski précisent seulement avoir utilisé *"an optical USB Microsoft IntelliMouse with four buttons and a scroll wheel"* et un *"experimental software written in Java"*. Les informations de la souris étaient certainement transmises à l'application Java par le système d'exploitation sous-jacent et sa fonction de transfert mais aucun d'entre eux ne sont explicitement mentionnés dans l'article.

Ces différents exemples illustrent la nécessité de mieux comprendre les fonctions de transfert utilisées par les systèmes modernes.

D'un point de vue système, la plupart, sinon tous les périphériques de pointage, respectent la classe *Human Interface Devices (HID)* du standard USB. Cette classe regroupe une série d'équipements comprenant claviers, souris, touchpads, joysticks, mais aussi téléphones, télécommandes, lecteurs de codes-barres ou voltmètres. Par conséquent les périphériques HID doivent fournir une description détaillée de leurs caractéristiques pour être correctement reconnus et utilisés. Entre autres, un périphérique de pointage précise pour chaque axe le caractère absolu ou relatif, linéaire ou non-linéaire de la valeur transmise (appelée *counts*), sa taille, l'intervalle logique des valeurs, l'intervalle physique correspondant, l'unité système et l'exposant utilisés. La description précise aussi la période d'échantillonnage du périphérique.

La spécification HID définit un *simple boot report format* pour les souris qui permet de les utiliser avant que le système d'exploitation ne soit chargé, afin d'accéder à la configuration bas-niveau du système. Ce format décrit les mouvements suivant chaque axe de manière relative, linéaire, sans unité et codé sur un octet par axe, entre -127 et +127. La plupart des souris supportent ce format et pour beaucoup, c'est aussi ce format qui est utilisé une fois que le système est chargé.

L'intervalle de polling est typiquement de 8 ms, ce qui correspond à une fréquence de mise à jour de 125 Hz. Pour les périphériques qui spécifient leurs intervalles logiques et physiques ainsi que l'unité système et l'exposant, la résolution peut être calculée suivant la formule $Res = LogRange/(PhyRange * 10^{Exp})$. Malheureusement, la plupart des périphériques omettent de préciser ces informations, ce qui rend la formule inapplicable. Pour ces périphériques, les systèmes considèrent généralement que la résolution est de 400 CPI[7]

7. Nous utilisons CPI (*counts per inch*) et PPI (*pixels per inch*) à la place de DPI pour clairement distinguer les résolutions des périphériques d'entrée et de sortie.

Le fait que les systèmes n'ont pas de moyen fiable pour connaître la résolution d'un périphérique de pointage devient de plus en plus problématique puisque les constructeurs proposent des périphériques avec des résolutions non seulement de plus en plus importantes mais en plus ajustables [8]. Une souris de résolution plus importante envoie des valeurs plus grandes à la même fréquence, cela engendre un nombre de counts plus important. Les souris haute résolution utilisent en effet deux octets pour chaque axe et peuvent réduire la période de polling à 1 ms. Une fonction de transfert qui ne prend pas en compte la résolution du périphérique ou la fréquence d'envoi des données interprétera inévitablement de manière erronée les counts et produira des effets indésirés, le plus courant étant une amplification importante des mouvements réalisés dans l'espace moteur. Ce problème est si fréquent que de nombreuses personnes confondent la résolution de la souris avec la vitesse du pointeur, comme l'illustre le panneau de vulgarisation du vocabulaire technique d'une grande chaîne de magasins française, réprésenté figure 2.6

FIGURE 2.6: Panneau de vulgarisation du vocabulaire technique d'une grande chaîne de magasins française, expliquant la signification de la résolution de la souris.

Modifier la résolution d'un périphérique ou utiliser un périphérique avec une résolution différente ne devrait pas changer la mise en correspondance des mouvements entre les espaces moteur et visuel. Modifier la fonction de transfert devrait être le seul moyen d'arriver à cette fin.

Afin d'aider à caractériser des fonctions de transfert, nous avons développé EchoMouse qui est un périphérique matériel qui a été conçu pour mesurer la réponse d'un système lors de gestes de pointage reçus d'un équipement HID. Ce périphérique est basé sur un PIC de Microchip qui a été programmé pour apparaître comme une souris HID classique. Ce périphérique n'a aucun capteur de mouvement. A la place, le descripteur HID a été modifié pour permettre d'envoyer des informations au format HID boot format qui sont aussitôt renvoyées par le périphérique et qui sont indistinguable d'informations envoyées par une réelle souris.

Une première version du périphérique a été développée dans le cadre de la thèse de

8. Par exemple, la résolution de la Logitech Gaming Mouse G500 peut être ajustée entre 200 et 5700 CPI

Jean-Philippe Deblonde [**DCG10**]. L'envoi des informations se faisait en appuyant sur un bouton. A chaque appui sur le bouton, le périphérique incrémentait le nombre de counts envoyés et le déplacement correspondant du pointeur était mesuré. Cette façon de procéder avait l'avantage de ne nécessiter l'écriture d'aucun code pour envoyer les données au périphérique. Bien que fastidieuse, cette méthode a permis de vérifier que les courbes Windows testées dans l'article [**CVBC08**] sont très proches des courbes déterminées avec EchoMouse. En revanche, les courbes qui avaient été calculées à partir des informations du registre OS X ne correspondent pas à celles trouvées avec EchoMouse, même si l'allure reste la même (Figure 2.7).

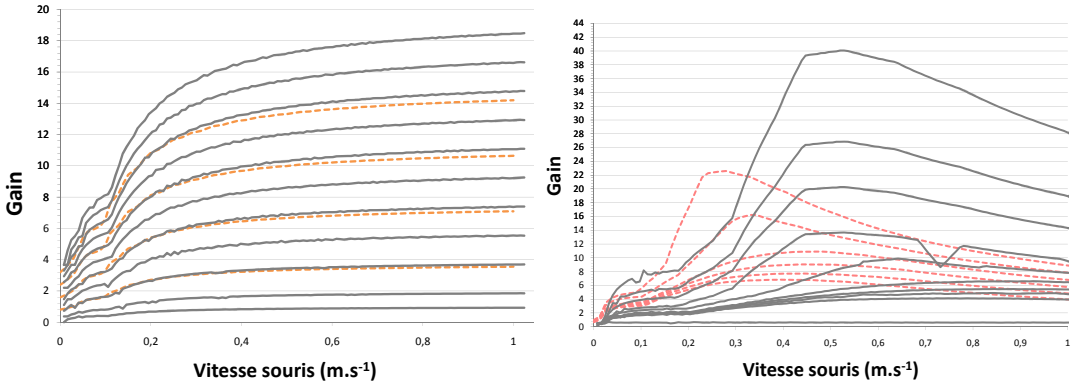


FIGURE 2.7: (gauche) Les courbes grises correspondent aux 11 fonctions de gain de Windows XP quand la case "Améliorer la précision du pointeur" est cochée, de "lente" (curseur tout à gauche, courbe du bas), à "rapide" (curseur tout à droite, courbe du haut). Les courbes en pointillés orange correspondent aux facteurs d'échelle de 0,25 ; 0,5 ; 0,75 et 1,0 testées par Casiez et al. [**CVBC08**]. (droite) Les courbes grises correspondent aux fonctions de gain pour les 10 positions du curseur sur l'interface de configuration de la souris de Mac OS X, de "lente" (courbe du bas) à "rapide" (courbe du haut). Les courbes en rouge pointillés correspondent aux courbes obtenues par Casiez *et al.* [**CVBC08**]. Les vitesses ont été calculées en prenant la résolution effective de l'écran et en supposant une résolution de 400 CPI pour EchoMouse.

EchoMouse permet d'avoir une vue macroscopique des fonctions de transfert d'un système. Elle ne permet cependant pas de comprendre complètement les détails d'implémentation des fonctions ou de savoir si les fonctions sont indépendantes du matériel utilisé. Pour ces raisons, nous avons développé libpointing, une boîte à outils qui permet de répliquer et comparer les fonctions de transfert [**CR11**].

Libpointing a été développée avec plusieurs objectifs en vue. D'abord, nous souhaitions une méthode pour accéder directement aux périphériques de pointage HID et ainsi court-circuiter les fonctions de transfert du système. Ensuite, nous souhaitions répliquer aussi fidèlement que possible les fonctions de Windows, OS X et Xorg. Troisièmement, nous voulions que la boîte à outils fonctionne sur ces trois systèmes pour comparer notre implémentation avec celles des systèmes. Enfin, nous souhaitions faire la comparaison entre les fonctions ainsi répliquées et d'autres.

La boîte à outils comprend environ 10000 lignes de code écrites en C++ et développées sous OS X 10.6, Ubuntu 10.10 et Windows XP, Vista et 7. Un aspect important de libpointing est l'utilisation d'URIs [BLFM05] pour indiquer le périphérique d'entrée, celui de sortie ainsi que la fonction de transfert. Les détails de libpointing et des fonctions de transfert sont donnés dans l'article [**CR11**]. Les fonctions de transfert ont été déterminées en utilisant les documentations publiquement disponibles [9] [Poi02], et les codes sources disponibles pour OS X [10] et Xorg [11]. EchoMouse a été utilisée pour révéler des détails d'implémentation.

Les figures 2.8, 2.9 et 2.10 illustrent les fonctions utilisées par Windows 7, OS X et Xorg. La figure 2.11 montre les fonctions de transfert utilisées par défaut sous Windows, OS X et Xorg. Globalement, et en dépit de quelques différences, les familles de courbes ont beaucoup de points en commun. Les trois systèmes ne prennent que partiellement en compte les caractéristiques des périphériques d'entrée et de sortie. OS X est le seul système qui prend en compte la résolution réelle du périphérique d'entrée (Windows suppose que la résolution est de 400 CPI et Xorg ne l'utilise pas du tout). Xorg est le seul système qui prend en compte les étiquettes de temps associées aux événements en entrée (les deux autres systèmes utilisent des constantes pour les fréquences). Xorg ignore complètement la fréquence d'affichage et sa résolution tandis que OS X utilise des constantes (les différentes versions de Windows divergent sur cette question).

Tous les systèmes utilisent des fonctions non-linéaires par défaut mais Windows et Xorg supportent également l'utilisation de fonctions de gain simples. Comme tous les systèmes ne prennent pas en compte correctement les résolutions et fréquences des périphériques, aucun ne permet de supporter l'application de gain constant sans unité, pourtant utilisée par de nombreux chercheurs.

Les trois systèmes utilisent des valeurs entières de coordonnées de pixels mais sauvegardent des restes pour obtenir une précision dite *subpixel*. Cette information n'est pas utilisable à notre connaissance par les applications, du fait de l'absence d'API permettant d'y accéder. Windows 7, Mac OS X et Xorg ne remettent jamais à zéro ces restes alors qu'ils sont remis à zéro suivant différentes stratégies sous Windows XP et Vista. En comparant nos curseurs personnalisés avec ceux des trois systèmes, nous avons pu valider nos implémentations des fonctions Windows et Xorg mais nous avons constaté de légères différences pour la fonction OS X. On peut expliquer ces dernières par la présence d'un algorithme de prédiction de trajectoire qui demande des informations que nous ne sommes pas en mesure de fournir.

9. `http://xorg.freedesktop.org/wiki/Development/Documentation/PointerAcceleration`
10. `http://opensource.apple.com/source/IOHIDFamily/`
11. `http://cgit.freedesktop.org/xorg/xserver/tree/`

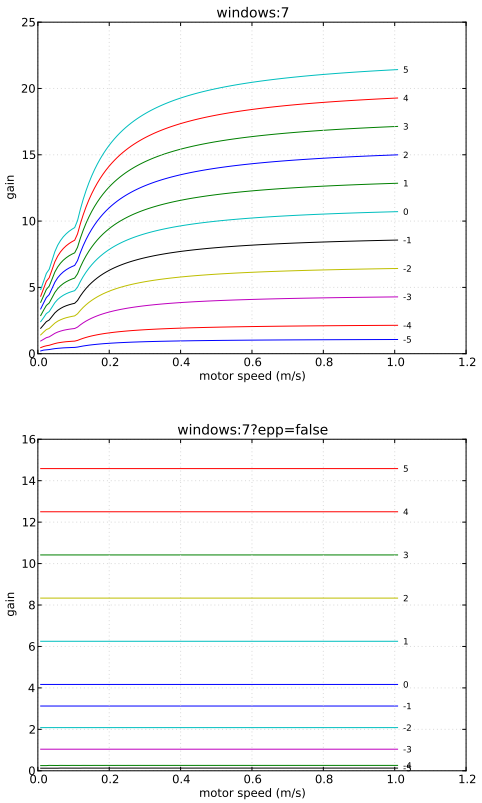


FIGURE 2.8: Fonctions Windows 7 disponibles via l'interface visible Figure 2.5, avec et sans l'amélioration de la précision.

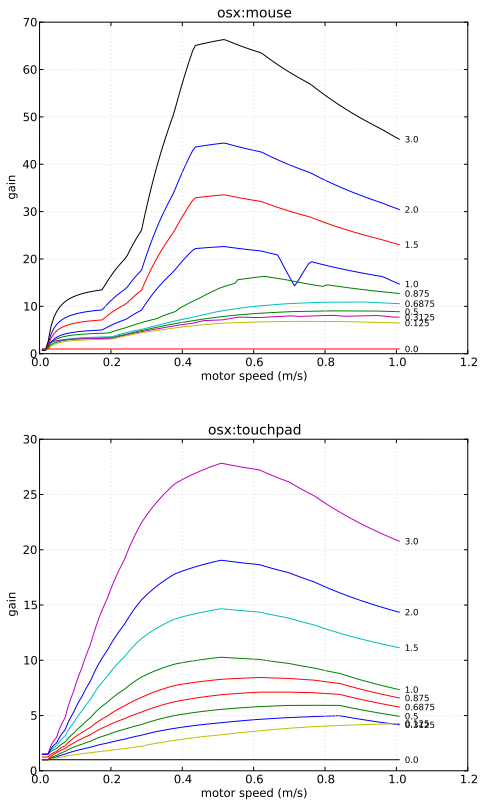


FIGURE 2.10: Fonctions OS X 10.6.7 pour la souris (disponible via l'interface montrée Figure 2.5) et touchpads.

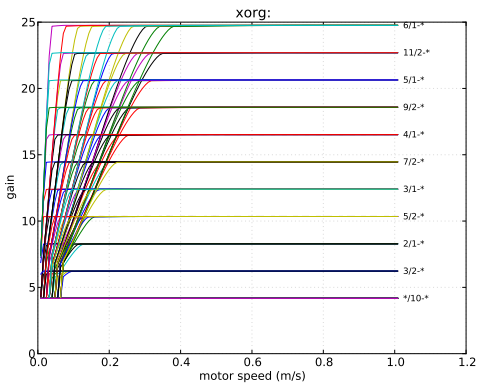


FIGURE 2.9: Fonctions Xorg functions disponibles dans Ubuntu 10.10 via l'interface montrée Figure 2.5.

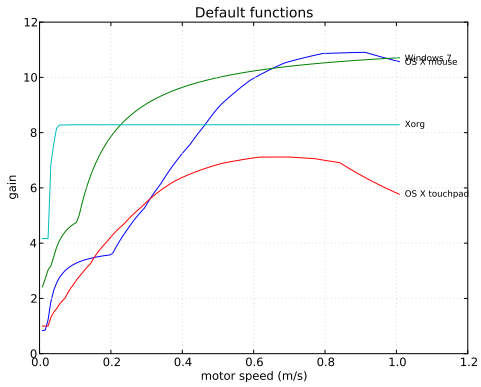


FIGURE 2.11: Fonctions par défaut sous Windows 7, OS X 10.6.7 (souris et touchpad) et Xorg.

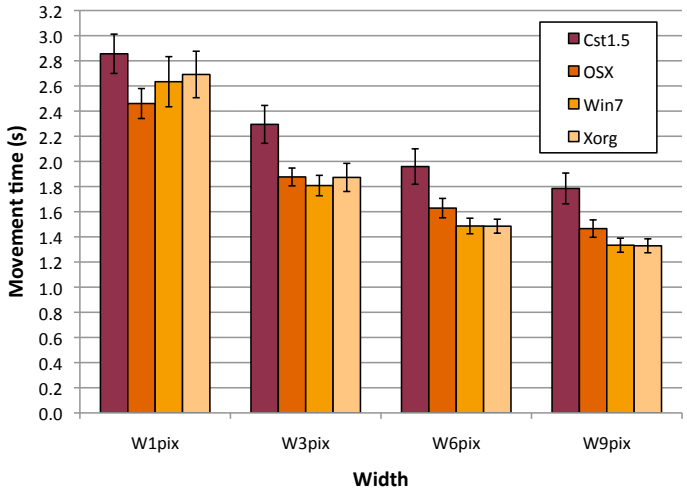


FIGURE 2.12: Temps moyen de pointage pour chaque fonction de transfert et largeur de cible. Les barres d'erreur représentent les intervalles de confiance à 95%.

Nous avons par ailleurs réalisé une expérience de comparaison des fonctions utilisées par défaut sur les trois systèmes. Nous avons également ajouté une fonction de gain constant, puisque ces fonctions sont généralement utilisées comme technique de référence pour la comparaison de techniques de facilitation du pointage. L'expérience consistait en une tâche de pointage réciproque, avec un accent mis sur les petites cibles (1, 3, 6, 9 pixels, soit de 0.26 à 2.32 mm). La distance entre les cibles était constante, égale à une valeur de 300 mm. Les détails des conditions expérimentales et des résultats sont donnés dans l'article [**CR11**].

En résumé, les résultats de cette expérience montrent qu'il existe des différences significatives entres les fonctions mais qu'il n'existe pas de meilleure fonction pour toutes les tailles de cibles (Figure 2.12). Windows 7 et Xorg améliorent le temps de pointage de plus de 9% comparé à OS X et de plus de 24% comparé à un gain constant pour les largeurs de cibles de 6 et 9 pixels. Cependant, les différences pour OS X disparaissent pour les cibles de 1 et 3 pixels. En revanche, OS X améliore le temps de pointage de 13% comparé à un gain constant et 8% par rapport à Xorg, pour des cibles de 1 pixel.

Ces résultats nous amènent à formuler un certain nombre de recommandations à l'égard des chercheurs ayant recourt aux fonctions de transfert. La première est d'éviter d'utiliser des fonctions de gain constantes lors de la comparaison de techniques, à moins que ce choix soit clairement justifié. Aucun des systèmes actuels utilise de telles fonctions et des systèmes comme OS X ne permettent pas d'utiliser des fonctions de gain constantes. Nous recommandons par conséquent d'utiliser de préférence la fonction de transfert par défaut du système considéré.

Pour faciliter la réplication de techniques de pointage et d'expériences, nous recommandons aux chercheurs de fournir le maximum de détails sur les fonctions de transfert

qu'ils utilisent. Si des gains constants sont utilisés, nous recommandons de les décrire sous forme de valeurs sans unité, de façon à les abstraire de considérations matérielles. Détailler comment un gain constant a été obtenu peut aussi aider à identifier des problèmes de méthodologie. Par exemple, il peut être important d'expliquer comment les restes sont gérés puisque leur influence sur les performances des utilisateurs reste à éclaircir. Si un système particulier est utilisé, la configuration de ce système doit être décrite de façon non ambiguë. Une capture d'écran est probablement la façon la moins ambiguë de décrire ces paramètres. Comme nous avons vu que certains systèmes ne les prennent pas en compte, nous recommandons également, pour être exhaustif, de préciser la résolution et la fréquence des périphériques d'entrée et de sortie. Lors de l'utilisation de fonctions non-linéaires personnalisées, nous recommandons de les décrire sous forme de tableaux ou de figures, représentant des données exprimées avec des unités physiques.

En plus des fonctions de transfert, il peut être intéressant de préciser la latence d'un système. En effet, elle peut avoir une influence sur les performances et par conséquent être une variable de confusion. Mesurer la latence n'est pas aisé mais certaines caractéristiques du système, comme le type de communication utilisé entre le périphérique et l'ordinateur ou la nature de la synchronisation verticale de l'écran permettent d'estimer la valeur de la latence.

Comme nous l'avons déjà évoqué, les fonctions de transfert devraient être définies indépendamment du matériel. Dans un monde idéal, les fabricants de périphériques de pointage tireraient pleinement partie de la spécification HID et y mettraient toutes les informations nécessaires sur le matériel. Malheureusement, la réalité est assez différente. Les systèmes sont souvent obligés de faire des choix adaptés sur les caractéristiques matérielles. Nous pensons que ces valeurs devraient être visibles et modifiables dans les interfaces de configuration adéquates.

En considérant le niveau de compréhension des interfaces actuelles utilisées pour régler les fonctions de transfert, même au sein des chercheurs, nous croyons que ces interfaces devraient au moins être correctement documentées, à défaut d'être complètement repensées.

Les périphériques élastiques et isométriques, en tant que périphériques d'interaction indirecte, nécessitent également l'utilisation de fonctions de transfert. Lors de mes travaux de thèse sur le DigiHaptic, je pouvais utiliser les manettes du périphérique en mode isotonique, élastique ou quasi isométrique, via le contrôle des moteurs des manettes. J'avais alors été confronté à la question de définir une fonction de transfert pour mettre en relation les déplacements des manettes avec ceux d'un objet virtuel. Je me posais par ailleurs la question de l'influence de la raideur du ressort simulé sur les manettes sur les performances des utilisateurs.

2.3 Fonctions de transfert pour périphériques élastiques/isométriques

La souris est un périphérique de pointage performant mais il existe des environnements de travail qui ne permettent pas de disposer de l'espace de travail nécessaire et d'autres, tels que les lieux publics, dans lesquels l'utilisation d'un périphérique tel que la souris présente un risque de malveillance. Les fabricants d'ordinateurs portables ont répondu aux problèmes liés à ces contextes d'utilisation en proposant le touchpad. Comme la souris, le touchpad est un périphérique isotonique utilisé pour faire du contrôle en position. Cependant le touchpad a un espace de travail réduit qui peut entraîner une augmentation de la fréquence de débrayage, préjudiciable à l'interaction.

L'utilisation d'un contrôle en vitesse, où les informations en entrée du périphérique sont mises en correspondance avec les vitesse et direction du pointeur, permet de supprimer les problèmes de débrayage tout en réduisant considérablement l'espace de travail. En effet, le contrôle en vitesse est mieux adapté aux périphériques isométriques et élastiques qui disposent naturellement d'un mécanisme de recentrage automatique pour ramener l'effecteur du périphérique à une position neutre quand il est relâché [Zha95]. Les périphériques isométriques, comme le TrackPoint [MD96, RS98], ne bougent pas de façon perceptible mais disposent de capteurs de force pour mesurer la force qui leur est appliquée. Et pourtant, les périphériques isotoniques semblent plus performants pour le pointage que les périphériques isométriques [DM94, Epp86, MSB91, PTB98]. Cette différence pourrait s'expliquer par l'utilisation de paramètres non optimaux de ce type de périphérique. Par exemple, le réglage de la fonction de transfert peut aussi affecter les performances. Ainsi, après un réglage empirique de fonctions de transfert, Zhai n'a trouvé aucune différence entre des périphériques isotoniques et isométriques comprenant 6 degrés de liberté [Zha95]. Un autre problème avec les périphériques isométriques est le manque de retour proprioceptif, qui peut augmenter la fatigue [Zha95].

Les périphériques élastiques, quant-à eux, ont un effecteur qui peut être déplacé dans un espace de travail déterminé, avec un ressort qui applique une force opposée au déplacement pour recentrer l'effecteur (Figure 2.14). Cependant, à l'exception de l'expérience pilote de Zhai [Zha95], peu de choses sont connues sur l'effet de la raideur des ressorts d'un périphérique élastique et les conclusions de l'influence du gain ne sont pas arrêtées [CEB78, Epp86, Gib62, KE88], en dépit de nouveaux périphériques élastiques qui apparaissent dans la littérature [FHSH06, KI98] et dans le commerce (Nintendo Nunchuk). Sans une compréhension de l'effet combiné de la raideur et des fonctions de transfert, le réglage de ces paramètres pour les périphériques isométriques et élastiques continuera d'être empirique.

L'étude des travaux antérieurs montre qu'il n'existe pas de conclusion claire concernant l'effet du gain sur les performances des utilisateurs dans le cadre de tâches de pointage. Gibbs [Gib62] et Zhai [Zha95] ont trouvé que la courbe représentative du temps de réalisation de la tâche en fonction du gain décrit une forme en U ; Rutledge et al. [RS90]

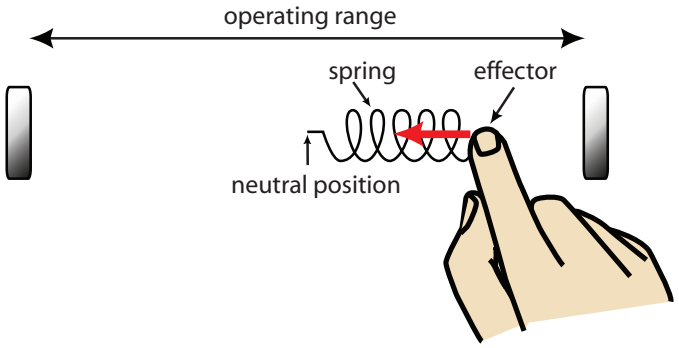


FIGURE 2.13: Périphérique élastique composé d'un ressort (*spring*) attaché à un effecteur (*effector*). La force de résistance est proportionnelle au déplacement de l'effecteur, qui est limité par l'espace de travail (*operating range*)

ont observé que les performances se dégradent avec les gains élevés et Kantowitz et Elvers n'ont trouvé aucun effet [KE88]. Cependant, dans ces expériences, la plage de gain testée était trop petite ou les distances et largeurs de cibles utilisées trop restreintes. De leur côté, Rutledge et al. [RS90] ont comparé des fonctions de transfert linéaires et non-linéaires et ils ont trouvé que la performance décroît avec les valeurs élevées de gains. Par ailleurs, seule l'expérience préliminaire de Zhai avec deux personnes a examiné l'influence de la raideur sur les performances des utilisateurs [Zha95]. Dans son expérience comparant périphériques élastique et isométrique, aucune différence significative de performance n'a été trouvée. Zhai note cependant que la fatigue des participants augmente avec les périphérique isométriques et que le retour proprioceptif est plus riche avec les périphériques élastiques. Enfin, l'interaction entre fonction de transfert et raideur n'a jamais été évaluée.

Pour ces raisons, nous avons mené une expérience pour évaluer l'influence des fonctions de transfert et de la raideur des périphériques élastiques, de retour isotonique à quasi-isométrique. L'expérience s'est restreinte aux fonctions de transfert avec gain constant. Si un effet existe, alors les chercheurs pourront concevoir des fonctions de transfert non-linéaires en se basant sur ces résultats.

L'expérience a été réalisée avec un Phantom Desktop [12] pour simuler les différents périphériques élastiques. La tâche était une tâche de pointage réciproque en une dimension. L'intervalle de gain considéré allait de 0 à 1600 $N.m^{-1}$, la taille des cibles allait de 2 à 8 mm, les distance de 75 à 300 mm et la plage de gain de 0.05 $m.s^{-1}.N^{-1}$ à 0.7 $m.s^{-1}.N^{-1}$. Le protocole expérimental ainsi que les résultats sont détaillés dans l'article correspondant [**CV08**].

Les résultats de l'expérience montrent un effet prononcé du gain sur les performances, à la fois sur le taux d'erreur et le temps de pointage. Ceci montre que le gain est un facteur critique à régler sur les périphérique avec contrôle de vitesse. La représentation du temps

12. `http://www.sensable.com`

de pointage en fonction du gain fait apparaître une courbe en forme de U avec une valeur optimale de gain qui dépend de la raideur de l'effecteur. Ces résultats confirment la forme générale de la courbe observée par Gibbs [Gib62] et Zhai [Zha95]. En complément, nous montrons que la valeur optimale de gain (le bas de la forme en U) se déplace vers les faibles valeurs de gain quand la raideur augmente. Par ailleurs, la plage de valeurs optimales (l'écrasement du bas du U) augmente à mesure que la raideur diminue. Ceci montre que le réglage du gain est plus délicat avec les périphériques isométriques. Une analyse des données avec la loi de Fitts a permis d'expliquer l'origine de cette forme en U. Les faibles valeurs de gain rendent les mouvements correctifs plus faciles et plus rapides mais nécessitent plus de temps pour atteindre la cible. En revanche, des valeurs de gain élevées permettent d'atteindre plus rapidement la cible mais rendent les mouvements correctifs plus difficiles, ce qui augmente globalement le temps de pointage.

Si nous avions obtenu les mêmes valeurs de gains optimaux pour les différentes raideurs, cela aurait signifié que les utilisateurs se servent uniquement de l'information de force pour contrôler la vitesse du pointeur. Nos résultats montrent que les participants se servent également de l'information de déplacement, utilisant une combinaison des informations de force et de déplacement pour contrôler la vitesse de déplacement. En fait, pour différentes raideurs présentant un profil similaire d'utilisation des forces mais des déplacements différents, les performances sont meilleures lorsque les déplacements augmentent, ce qui suggère que les utilisateurs se basent davantage sur le déplacement que sur la force pour contrôler la vitesse.

Avec une sélection appropriée du gain, les différentes raideurs testées présentent des performances équivalentes, sauf lorsque la raideur est nulle. Dans ce cas, on observe une augmentation du taux d'erreur et une augmentation de 15% du temps de pointage. La seule différence entre les raideurs de 0 et 30 $N.m^{-1}$ était le retour élastique. Ainsi notre étude confirme que les périphériques élastiques et isométriques sont mieux adaptés au contrôle en vitesse. Cependant, nous montrons également, que même une faible raideur est suffisante. La différence de performance entre la condition isotonique pure et les conditions élastique/isométrique est plus faible que celle trouvée par Zhai [Zha95], certainement parce que l'expérience était réduite à un seul degré de liberté.

Cette étude permet également d'établir les recommandations suivantes. En considérant l'importance de la proprioception, de la fatigue et la difficulté de régler les gains constants pour les raideurs élevées, nous recommandons aux concepteurs de périphériques élastiques de diminuer la raideur des effecteurs. Cependant, la réduction de la raideur entraîne une augmentation de l'espace de travail. Cela peut être un problème puisque les périphériques utilisant un contrôle en vitesse sont souvent utilisés dans des situations où l'espace physique est restreint. Dans cette situation, nous recommandons d'utiliser la borne supérieure de la plage de gains optimale et de continuer à utiliser la raideur la plus faible possible pour obtenir l'espace de travail désiré.

En tenant compte des différentes éléments, si la raideur est supérieure à 120 $N.m^{-1}$, une valeur de gain de 0.1 $m.s^{-1}.N^{-1}$ donnera les performances optimale, que ce soit au

niveau du temps que du taux d'erreur. Pour des raideurs inférieures à 120 $N.m^{-1}$, cette valeur peut être augmentée jusqu'à 0.7 $m.s^{-1}.N^{-1}$ lorsque la raideur se rapproche de 0.

En utilisant l'équation 2.8, il est possible de vérifier que la valeur de gain choisie (CG) en $m.s^{-1}.N^{-1}$ est compatible avec l'espace de travail désiré (OR) en m, la raideur du périphérique (k) en $N.m^{-1}$ et la taille de l'écran (D) en m. Si l'équation 2.8 aboutit à un espace de travail plus grand que celui du périphérique alors le gain peut être augmenté mais cela peut se faire au prix d'une dégradation des performances.

$$OR = 2 + \frac{5 \cdot D}{k} + \frac{146}{k \cdot CG} \tag{2.8}$$

Comme exemple d'application, on peut chercher le gain à utiliser avec le Nunchuk de la Nintendo Wii. Le Nunchuk possède un joystick élastique à deux degrés de liberté, avec un espace de travail de 20 mm et une raideur de 60 $N.m^{-1}$. En considérant une distance maximale de cible de 300 mm et un gain de 0.7 $m.s^{-1}.N^{-1}$, l'équation 2.8 donne un espace de travail de 30 mm. En considérant la large plage de gain optimaux pour cette raideur, on peut raisonnablement augmenter le gain pour que l'espace de travail soit compris dans les 20 mm du Nunchuck sans dégradation des performances.

Ces travaux ont évalué les fonctions de gain constantes pour les périphériques élastiques/isométriques dans le cadre de tâches de pointage. D'autres études similaires seraient à mener afin de mieux comprendre l'influence des fonctions de transfert sur les performances des utilisateurs. En particulier, l'étude de fonctions non-linéaires reste à réaliser. D'autres tâches seraient également à prendre en considération, comme les tâches de navigation en environnement 2,5D, tâche très fréquente avec ce type de dispositif dans les jeux vidéo.

Pour des tâches de pointage sans débrayage, les performances des périphériques élastiques/isométriques restent cependant bien moindres comparées à celles des périphériques isotoniques. Les périphériques élastiques quant à eux sont intéressants pour parcourir de grandes distances, là où il serait nécessaire de débrayer avec un périphérique isotonique. L'idée de combiner les deux retours permettrait donc de tirer partie des avantages de chaque catégorie de périphérique. Cette idée avait germée durant ma thèse [**Cas04**], p.73. J'avais alors décliné différentes solutions pour réaliser ce mode hybride en programmant les manettes du périphérique. Lors de mon post-doctorat, j'avais travaillé durant le premier mois sur la continuité de ces travaux mais la lecture d'un article alors fraîchement publié [DLB+05] sur des travaux similaires m'avait fait changer de problématique.

Ces travaux ont ensuite été repris dans le cadre de la thèse de Qing Pan, démarrée en octobre 2005, au moment de ma prise de fonctions. L'idée était alors de quitter le monde de la 3D pour voir comment ce type de technique pouvait être intéressante dans un environnement de bureautique. La dégradation importante de performances constatée

en présence de débrayage lors des études menées lors de mon postdoc m'a incité à creuser cette idée.

2.4 Combiner retours isotonique et élastique : RubberEdge

Le débrayage, s'il est limité avec des périphériques comme la souris, devient prédominant avec des périphériques de type touchpad. D'une part, les espaces de travail des périphériques sont différents : plusieurs dizaines de centimètres pour une souris et quelques centimètres pour un touchpad (typiquement entre 4 et 10 cm) ; d'autre part, les membres impliqués dans l'utilisation de chaque périphérique ne sont pas les mêmes : la main pour la souris, les doigts pour le touchpad. Les surfaces d'affichage quant à elles augmentent en résolution et en taille. Par exemple, un écran de MacBook Pro a une taille de 17" et possède une résolution de 1920×1200, ce qui correspond à une densité de pixels de 130 DPI. Avec les écrans de la taille d'un mur, les différences sont encore plus importantes.

Comme évoqué précédemment, le débrayage d'un périphérique peut être réduit en augmentant le gain, cependant nous avons vu que des gains trop importants peuvent dégrader les performances des utilisateurs. L'utilisation de fonctions de gain non-linéaires permet de minimiser ce problème. Il reste néanmoins qu'un faible espace de travail réduit la plage de vitesses accessibles et par conséquent rend la conception de telles fonctions non linéaires d'autant plus complexe. Par ailleurs, il existera toujours des situations dans lesquelles l'utilisateur sera contraint de débrayer. L'utilisation d'un contrôle en vitesse avec des périphériques comme le TrackPoint[13] peut permettre d'améliorer les performances sur de grandes distances, quand l'utilisation d'un périphérique isotonique nécessiterait de débrayer.

Pour conserver les bénéfices du contrôle en position pour les cibles de distance moyenne tout en permettant de réaliser des mouvements sur de longues distances sans débrayer, des techniques de contrôle hybride position-vitesse ont été proposées [BH97, Syn]. Dans des environnements virtuels, Bowman et Hodges ont développé la technique Stretch Go-Go [BH97]. Une main virtuelle est allongée ou raccourcie à vitesse constante lorsqu'elle se rapproche d'une des extrémités de l'espace de travail. Avec la technique EdgeMotion utilisée sur les touchpad de la marque Synaptics, un contrôle en vitesse est utilisé en appuyant verticallement sur la surface quand le doigt est aux limites de l'espace de travail. En pratique, passer d'un contrôle en position à un contrôle en vitesse en changeant la direction du mouvement peut être délicat. Par ailleurs, compte tenu de la forme rectangulaire des touchpads, les directions de déplacement sont limitées à l'horizontale et la verticale. Ces différentes techniques ne fournissent aucun retour d'information au niveau haptique, si bien que la transition entre les contrôles en position et vitesse sont difficiles à distinguer et la vitesse est difficile à contrôler. Zhai a montré que le retour élastique est mieux adapté à un contrôle en vitesse [Zha95]. Ainsi Dominjon et al. ont utilisé un retour élastique pour leur technique de contrôle hybride nommée Bubble [DLB+05]. Cependant, leur technique présente des discontinuités de trajectoire et de vitesse lors de

13. `http://www.pc.ibm.com/ww/healthycomputing/trkpnt.html`

la transition entre contrôle de position et contrôle de vitesse, ce qui met davantage en évidence les difficultés de rendre utilisable un contrôle hybride.

Nous avons proposé **[CVPC07]** un périphérique hybride bi-dimensionnel appelé RubberEdge (Figure 2.14). Contrairement aux travaux précédemment mentionnés, nous avons conçu des fonctions qui permettent de réaliser une transition continue entre contrôle en position et contrôle en vitesse. Pour évaluer notre technique, nous avons réalisé des expériences de comparaison entre notre technique et une technique de contrôle en position uniquement. Elles révèlent des améliorations de performances de l'ordre de 20% avec un espace de travail d'une taille similaire à celle d'un touchpad. Nous avons également proposé deux modèles permettant de prédire les performances de pointage en présence de débrayage avec des périphériques isotoniques et avec des périphériques hybrides.

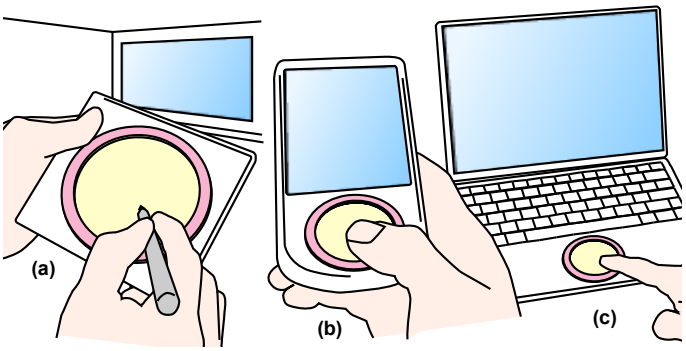


FIGURE 2.14: Schémas de principe de périphériques hybrides basés sur RubberEdge : (a) une tablette utilisée avec un grand écran ; (b) un smartphone avec un touchpad ; (c) un ordinateur portable avec un touchpad

Dominjon et al. [DLB+05] utilisaient les fonctions décrites par les équations 2.9 et 2.10 pour calculer le retour de force associé au retour élastique et la vitesse correspondante du pointeur. Dans l'équation 2.9, le retour de force F est proportionnel à la distance entre la position P de l'effecteur et sa projection à la frontière de la zone élastique selon la direction définie par le centre de la zone de travail et la position de l'effecteur. La raideur k est associée à la zone élastique. La direction de la force est par conséquent tout le temps radiale, selon $\vec{r}$. L'équation 2.10 détermine la vitesse du pointeur en fonction de la norme de la force selon un polynôme de degré trois.

$$\vec{F} = -k \cdot (P - N) \cdot \vec{r} \tag{2.9}$$

$$\vec{V} = K \cdot F^3 \cdot \vec{r} \tag{2.10}$$

Cette formulation introduit une discontinuité de trajectoire lorsque celle-ci n'est pas radiale avant d'atteindre la frontière de la zone isotonique. Le pointeur va alors décrire une trajectoire radiale selon l'équation 2.10, dès l'instant où l'effecteur va entrer dans la zone

élastique (Figure 2.15). Une discontinuité de vitesse va également avoir lieu puisque, selon l'équation 2.9, la force initiale dans la zone élastique est nulle et par conséquent la vitesse est également nulle.

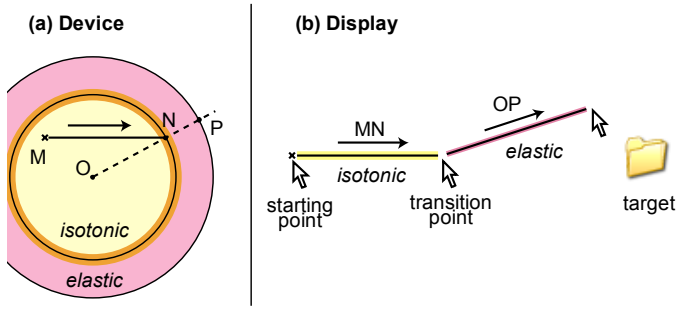


FIGURE 2.15: Discontinuité de trajectoire avec l'utilisation de fonctions élémentaires : (a) déplacement de l'effecteur dans la zone isotonique pour atteindre une cible distante ; l'utilisateur bouge l'effecteur entre les points M et N dans la zone isotonique avant d'entrer dans la zone élastique ; (b) à l'écran, le pointeur va alors dévier de sa trajectoire initiale au point de transition suivant la direction définie par $\vec{OP}$.

Les fonctions utilisées pour RubberEdge permettent d'adoucir la trajectoire en faisant tourner et translater la frontière de la zone isotonique/élastique après la transition. La vitesse est également lissée en combinant les vitesses avant et après la transition. En donnant une masse et une inertie à la zone isotonique, nous alignons la trajectoire dans la zone élastique avec celle de l'effecteur dans la zone isotonique, tout en permettant à l'utilisateur de changer de direction dans la zone élastique. L'idée sous-jacente est de considérer le mouvement de translation et de rotation d'un objet circulaire, comme une assiette de cuisine, quand il est tirée par une ficelle attachée à sa périphérie (Figure 2.16). Quand l'utilisateur sort de la zone, le point de sorti N est sauvegardé. Dans la zone élastique, le vecteur défini par N et la position de l'effecteur P donne la direction de la force à appliquer sur l'assiette. En appliquant le moment cinétique, N pivote vers N' et la direction de la force devient radiale à O, le centre de la zone isotonique.

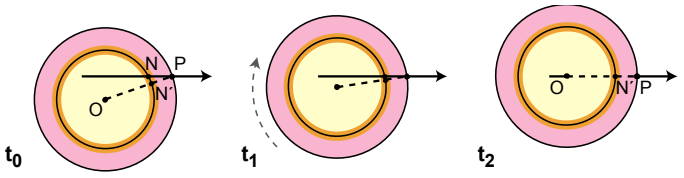


FIGURE 2.16: Décomposition de la rotation et de la translation avec application du moment cinétique.

La vitesse angulaire de la zone isotonique est calculée en utilisant le théorème du moment cinétique. Pour lisser la vitesse de transition, nous combinons la vitesse avant la sortie de la zone isotonique avec celle de zone élastique pendant un tiers de seconde.

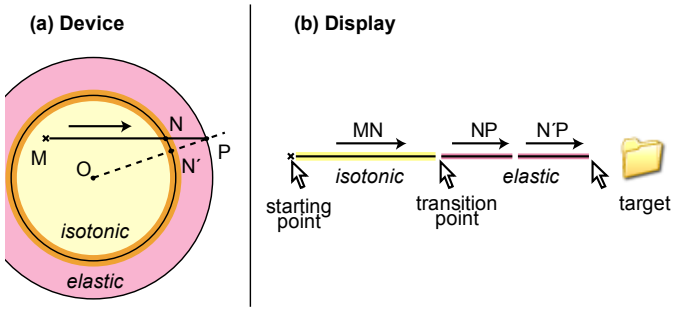


FIGURE 2.17: Continuité de la trajectoire avec RubberEdge : (a) l'utilisateur bouge du point M au point N dans la zone isotonique, puis arrive dans la zone élastique ; (b) sur l'écran, la trajectoire initiale NP est modifiée progressivement pour obtenir N'P.

RubberEdge a été évalué dans une expérience contrôlée. Le but était de comparer les performances de RubberEdge à celles d'une technique de contrôle en position en présence de débrayage. L'objectif était également d'étudier s'il existait une interaction entre ces techniques et les fonctions de transfert (gain constant et gain dynamique). Il est à noter que RubberEdge a été simulé avec un Phantom Omni et que toutes les techniques ont été évaluées avec ce périphérique. L'utilisation d'un périphérique à retour de force a permis dans ce cas de figure de réaliser facilement un prototype de notre périphérique hybride et de limiter les possibles interaction de facteurs de formes. Le plan expérimental ainsi que les résultats détaillés sont décrits dans l'article associé [**CVPC07**]. Les résultats montrent qu'un contrôle hybride position/vitesse améliore les performances par rapport à un contrôle isotonique quand la quantité de débrayage est significative. Les résultats montrent également qu'une fonction de transfert dynamique permet de réduire le débrayage par rapport à un gain constant mais que le contrôle hybride permet d'améliorer les performances indépendamment de la fonction de transfert.

Nous avons par ailleurs développé un modèle de débrayage qui permet de prédire les performances en présence de débrayage. Celui-ci est basé sur une estimation du nombre de débrayages requis pour atteindre une cible. Nous avons développé un modèle similaire pour RubberEdge. Ces modèles permettent de s'abstraire des conditions expérimentales, en particulier de la taille de la zone isotonique. Une comparaison des modèles aux résultats empiriques montre la bonne qualité de prédiction de ces modèles. Ces derniers permettent en particulier de prédire quand le contrôle hybride devient avantageux. Par exemple, les modèles montrent qu'avec un ordinateur portable possédant un écran de 38 cm et un touchpad de 4 cm, le contrôle hybride est meilleur pour atteindre des cibles à une distance de plus de 30 cm. Pour un PDA HP iPAQ hx4700 possédant un touchpad de 1cm et un écran de 10 cm montre, nos modèles montrent que le contrôle hybride devient avantageux pour atteindre des cibles éloignées de plus de 5 cm.

Ces travaux montrent qu'il est possible de combiner des périphériques avec des propriétés différentes afin de tirer le meilleur parti de chacun. Nous allons maintenant voir

comment exploiter des propriétés intrinsèques des périphériques, comme leur résolution, pour cette fois-ci interagir dans des tâches demandant une grande précision.

2.5 Exploiter la richesse de la résolution en entrée

Nos travaux sur les fonctions de transfert ont mis en évidence que la résolution des périphériques de pointage de type souris et pavés tactiles n'est pas prise en compte par les systèmes d'exploitation modernes. S'il est possible de modifier les fonctions de transfert pour qu'elles prennent en compte la résolution du périphérique d'entrée, il n'est en revanche pas possible de tirer partie de cette finesse de mesure de position pour interagir avec plus de précision. Les fonctions de transfert pour le pointage manipulent en effet la position du pointeur en coordonnées entières exprimées en pixels. Le système génère des événements à chaque changement de position du pointeur. Ces événements sont utilisés par les boîtes à outils pour mettre à jour les éléments de l'interface. Bien que la résolution soit un paramètre essentiel de la souris, elle reste à l'heure actuelle inexploitée ou exploitée à mauvais escient, cela 30 ans après l'utilisation à grand échelle des souris informatiques et en dépit d'une augmentation impressionnante de leur résolution ces dernières années. Nos travaux sur ce sujet ont été fortement inspirés par les travaux sur la détermination des fonctions de transfert des systèmes actuels ainsi que des travaux récents de Bérard et al. [BWC11].

Lorsqu'on déplace un périphérique de pointage, les informations qu'il mesure sont traduites par la fonction de transfert en un déplacement du pointeur en pixels. Le système déplace alors le pointeur qui génère des événements de déplacement qui sont envoyés aux boîtes à outils des interfaces graphiques qui les transmettent à leur tour aux widgets (Figure 2.18). Il est curieux et révélateur de constater que ces événements sont communément appelés « *événements souris* » alors qu'ils correspondent en réalité à des « *événements pointeur* ». En fait, un périphérique de pointage ne permet pas d'interagir directement avec des données : c'est un périphérique qui permet de contrôler un pointeur, qui lui, permet de contrôler des données. Il y a 30 ans, la sensibilité des souris était comparable à la densité de pixels des écrans, ainsi cette simplification de la réalité était raisonnable (la souris d'origine du Macintosch avait 90 CPI, assez proche de la densité de pixels de l'écran qui était de 72 PPI). Le problème est que cette simplification de la réalité perdure aujourd'hui.

Il y a 15 ans, dans son article intitulé *"The Eyes Have It (...)"* [Shn96], Schneiderman a mis en évidence le fait que nos remarquables capacités perceptuelles sont souvent sous-exploitées. Compte tenu des faits énoncés précédemment, il est raisonnable de dire que nos remarquables capacités motrices sont actuellement sous-exploitées, en particulier lorsque l'on considère la sensibilité des souris actuelles qui est typiquement de plus de 800 CPI et qui peut atteindre 10000 CPI.

Nos travaux [**RCAV12**] proposent d'exploiter la résolution des périphérique d'entrée permettant d'atteindre une précision de manipulation qui dépasse celle de l'écran. Ce que

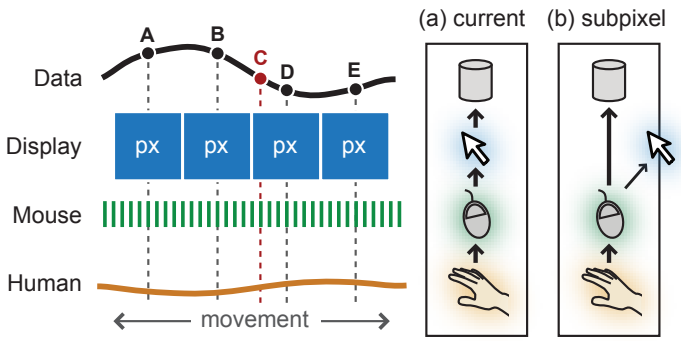


FIGURE 2.18: Systèmes de mise en correspondance : (a) actuellement, les mouvements humains sont discrétisés une première fois selon la résolution de la souris puis une seconde fois selon la densité de pixels de l'écran : les données situées "entre" les pixels, comme 'C' sont ainsi inatteignables ; (b) une mise en correspondance subpixel permet de discrétiser les mouvements humains, uniquement selon la résolution de la souris, ce qui permet une manipulation précise des données.

nous proposons est essentiellement une modification subtile mais substantielle de l'architecture des systèmes actuels où les déplacements des périphériques de pointage sont directement mis en correspondance avec les données manipulées. Cela permet une *interaction subpixel* ainsi qu'une augmentation de la précision des techniques d'interaction classiques (manipulation de widgets) et une compatibilité avec les techniques complémentaires, comme les lentilles.

Pour illustrer le problème plus concrètement, on peut considérer le cas d'une application de lecture de vidéos où la précision de sélection d'une image est limitée par la largeur en pixels de l'écran. Par exemple, l'application QuickTime a un potentiomètre de largeur fixe de 315 pixels. Bouger le potentiomètre d'un pixel correspond à un déplacement de 2,8s dans une vidéo de 15 minutes et 22.8 secondes dans une vidéo de 2 heures. Cela peut être suffisant pour un positionnement grossier mais pour d'autres tâches, comme passer uniquement les publicités peut devenir difficile, voire impossible quand la longueur de la vidéo augmente. Un exemple encore plus extrême est la sélection d'une image de référence pour une vidéo (poster frame) en utilisant le potentiomètre de largeur fixe de l'application Keynote. Il existe bien d'autres exemples où les pixels ne permettent pas d'obtenir une précision suffisante : choisir un album dans iTunes CoverFlow, faire une découpe au pixel près dans une image de plusieurs millions de pixels, ajuster un événement dans un calendrier à la minute près, définir des dimensions de murs au centimètre près dans un logiciel de CAO...

Une solution pour résoudre ces problèmes de pécision pourrait être d'ajouter simplement plus de pixels. Pour revenir à notre exemple de vidéos, considérons la possibilité d'obtenir une précision de manipulation d'une seconde dans une vidéo de 2 heures, en augmentant la largeur du potentiomètre. Dans ce cas, il faudrait un potentiomètre de 7200 pixels de large, la largeur de 4 écrans HD de 1080p. Ce n'est clairement pas envisageable. Une solution courante est donc d'introduire un autre mode de navigation,

comme des touches mais celles-ci peuvent être lentes et entraîner des erreurs, en partie parce que ce paradigme ne suit plus les principes de la manipulation directe.

Pour rendre la manipulation directe plus précise, les travaux antérieurs ont introduit des changements de modes explicites, comme l'utilisation de zoom. En plus des problèmes inhérents aux modes, ces techniques ne remettent pas en cause l'interaction avec des pixels entiers et ignorent totalement la résolution des périphériques de pointage ainsi que les capacités motrices humaines. Des travaux antérieurs ont montré que la précision motrice humaine est bien supérieure à la densité de pixels de l'écran. Ainsi, Guiard et al. [GBLM99] ont montré que les utilisateurs peuvent facilement atteindre des cibles de 0.06 mm dans l'espace moteur (423 CPI), mais que ce n'est pas une limite supérieure puisque cela correspondait à la plus petite largeur évaluée. Plus récemment Bérard et al. [BWC11] ont défini la notion de *Device's Human Resolution* (DHR) comme la "plus petite cible que les utilisateurs peuvent acquérir en utilisant un effort ordinaire avec un périphérique particulier". Pour la souris, ils ont obtenu des valeurs de DHR allant de 700 à 1400 CPI. Cela signifie que les humains peuvent facilement exploiter des résolutions de 7 à 14 fois supérieures à un écran classique de 100 PPI. Le défi est de mettre à profit ces capacités humaines afin d'obtenir un contrôle effectif des interfaces de manipulation directes.

Pour exploiter cette précision, le système doit dans un premier temps connaître les caractéristiques du périphérique, en particulier sa résolution, exprimée en CPI. Cela est important pour que le système interprète correctement ce qui est mesuré : une résolution plus importante doit correspondre à une mesure plus précise, pas à une amplification des mouvements du pointeur. Malheureusement, la plupart du temps, les fabricants ne fournissent pas cette information et, probablement pour cette raison, les fonctions de Microsoft Windows et X.Org ignorent complètement la résolution des périphériques d'entrée [**CR11**].

Comme nous avons déjà expliqué, les fonctions de transfert appliquent un gain dynamique sur les déplacements du périphérique, déplacent le pointeur en utilisant la partie entière du résultat et enregistrent la partie fractionnaire. Nous souhaitons que cette partie fractionnaire soit aisément accessible au code en charge de l'interaction. Nous proposons de généraliser ce que certains systèmes font pour les périphériques absolus : fournir la partie fractionnaire en utilisant des coordonnées flottantes pour les coordonnées du pointeur. Pour résumer, nous proposons d'abord que la fonction de transfert prenne en compte la résolution du périphérique de pointage et ensuite que les valeurs flottantes résultantes des calculs soient transférées directement au code de plus haut niveau.

Cette solution permet de tirer intégralement partie de la résolution du périphérique d'entrée : les mouvements dans l'espace moteur ne sont plus mis à l'échelle par la résolution en entrée et contraints par la résolution d'affichage. La quantité d'information disponible peut ainsi être multipliée par un facteur 100, en considérant la plus haute résolution de souris disponible (10000 CPI) et la résolution courante des écrans (100 DPI). Par ailleurs

le comportement du pointeur n'est pas modifié : il continue de bouger exactement à la même vitesse qu'il le fait habituellement.

Prendre en compte la résolution du périphérique ne signifie pas que les utilisateurs puissent l'exploiter. Il est par conséquent important de prendre en considération la DHR définie par Bérard et al. [BWC11]. Des périphériques avec une résolution inférieure à la DHR ne permettent pas de mesurer des mouvements fins dans l'espace moteur. En même temps, les utilisateurs risquent de ne pas tirer partie de résolutions supérieures.

L'article [**RCAV12**] détaille les équations qui permettent de calculer le nombre de subpixels associés à la fonction de transfert courante, en fonction de la résolution du périphérique, celle de l'écran et la DHR. Lorsque le nombre de subpixels n'est pas suffisant pour manipuler des données, nous proposons également une méthode pour modifier la fonction de transfert courante d'un système afin d'obtenir le nombre de subpixels souhaité. Nous détaillons par ailleurs dans quelles conditions les coordonnées flottantes deviennent utiles et jusqu'à quel point elles peuvent être utilisées.

Pour exploiter les coordonnées flottantes de manière universelle, les systèmes d'exploitation et les boîtes à outils graphiques nécessitent un certain nombre de changements mineurs mais fondamentaux. Tout d'abord, les fonctions de transfert doivent prendre en compte la résolution des périphériques d'entrée. Les systèmes d'exploitation doivent ensuite utiliser les restes enregistrés entre deux événements successifs pour créer des coordonnées flottantes. Les coordonnées subpixels seraient transmises par le système de fenêtrage et finalement reçues par les boîtes à outils graphiques. Les boucles d'événements, méthodes, fonctions de rappel doivent être mises à jour pour manipuler des coordonnées flottantes.

Les boîtes à outils doivent également fournir aux développeurs une façon de prendre en considération la résolution humaine directement ou indirectement. Idéalement, un développeur spécifierait le nombre d'éléments de son modèle et le système modifierait automatiquement la fonction de transfert lorsqu'un widget supportant les subpixels serait manipulé. Cela nécessite également que les fonctions de transfert du système puissent être modifiées dynamiquement. Les gestionnaires de placement pourraient également ajuster la taille des widgets, par exemple en diminuant la largeur d'un potentiomètre subpixel, sans sacrifier la précision de contrôle des données sous-jacentes.

2.6 Caractérisation des fonctions de transfert pour le défilement

Nous nous sommes pour l'instant concentré sur l'étude des fonctions de transfert pour les tâches de pointage. Une autre tâche essentielle des interfaces actuelles est le défilement. Il peut être réalisé en utilisant des molettes de souris, des gestes sur un pavé tactile ou encore des périphériques isométriques. Un élément fondamental que tous ces périphériques doivent prendre en considération est la fonction de transfert qui met en relation les actions physiques sur le périphérique (degrés de rotation, millimètres de déplacement ou force en Newtons) avec le défilement à l'écran : typiquement pixels, lignes

ou pages. Compte tenu de l'importance du défilement, il est surprenant de constater qu'il y a eu peu de recherche publique sur les fonctions de transfert de défilement. Comme exceptions, on peut citer Hinckley et al. [HCBM02] et Cockburn et al. [CQGF12], mais ces études sont ambiguës sur les fonctions qui ont été exactement testées ou comparées. Par exemple, Hinckley et al. déclarent "Nous avons testé le périphérique en utilisant ses paramètres par défaut". Retrouver quelle était cette fonction de transfert est aujourd'hui quasiment impossible et Cockburn et al. reconnaissent explicitement la nécessité de déterminer quelles sont les fonctions de transfert utilisées par les systèmes actuels.

La méconnaissance des fonctions de transfert pour le défilement pose plusieurs problèmes pour les chercheurs. Tout d'abord les fonctions actuelles sont inconnues puisqu'elles sont enfouies le plus souvent dans les drivers des périphériques. Cela rend l'observation de différences de performances entre des périphériques difficiles à interpréter et pénalise également toute amélioration itérative de l'état de l'art. Ensuite, les chercheurs manquent d'outils pour examiner, décrire et répliquer les fonctions de transfert. Enfin, les chercheurs peuvent introduire de façon non volontaire des biais expérimentaux, liés à des interactions inconnues entre les fonctions de transfert et les paramètres expérimentaux.

Comparées aux fonctions de transfert pour le pointage qui ne prennent en compte que la vitesse du périphérique, les fonctions de transfert pour le défilement peuvent prendre en considération l'accélération, le temps entre deux débrayages ou encore la direction de défilement. On ne sait pas aujourd'hui si ces paramètres sont pris en considération et comment ils le sont.

Tout comme la détermination des fonctions de transfert pour le pointage, nous avons utilisé EchoMouse pour simuler un périphérique particulier, nous avons ensuite envoyé des profils d'interaction dans l'espace moteur et analysé les événements de défilement reçus par une application pour en déduire la fonction de transfert. Nous avons analysé les fonctions de transfert des molettes de souris de Mac OS X 10.7.3, Microsoft Windows 7, Microsoft IntelliPoint (8.20.468 sur Windows) et Control Center de Logitech (version 3.5.1-23 sur Mac OS X), qui représentent les systèmes d'exploitation et les fabricants de périphériques les plus populaires. Tous les résultats sont détaillés dans [**QCC$^+$12**] et résumés Figure 2.19.

Nous avons observé que ni Windows 7 (avec la fonction de transfert par défaut), ni les drivers Logitech SetPoint ne prennent en compte la vitesse, la direction, la durée d'interaction ou le débrayage. Leurs fonctions de transfert appliquent simplement un gain constant. Pour les autres systèmes, la vitesse est le facteur le plus souvent pris en compte, suivi par la direction de défilement et le débrayage. Le détail des fonctions de transfert est donné dans [**QCC$^+$12**].

Il est intéressant de constater que les différences entre les fonctions de transfert analysées sont importantes, tant dans le choix des paramètres utilisés que dans la façon dont ils

	Velocity	Direction	Duration	Clutching
Apple Mac OS X	●	◐	○	◐
Microsoft Windows 7	○	○	○	○
Microsoft IntelliPoint	●	●	○	●
Logitech SetPoint	○	○	○	○
Logitech Control Center	●	◐	●	○

FIGURE 2.19: Synthèse des paramètres pris en compte par les fonctions de transfert de différents systèmes ou logiciels. ○ : aucune prise en compte du paramètre, ● : prise en compte, ◐ : partiellement pris en compte.

sont exploités. En effet, deux fonctions de transfert qui prennent compte les mêmes paramètres peuvent présenter des courbes avec des différences notables.

Les recherches sur le défilement se concentrent sur le développement de nouveaux périphériques d'interaction [ZSS97], de nouvelles fonctions de transfert [CQGF12, HCBM02] ou de nouvelles techniques d'interaction [KI06]. Les techniques de défilement sont aussi utilisées dans des expériences où le défilement n'est pas le principal centre d'intérêt mais un moyen d'accomplir une tâche. Dans tous les cas, les évaluations comparatives sont normalement conçues pour évaluer le gain de performance par rapport à l'état de l'art. Les choix faits dans l'implémentation et l'administration des conditions expérimentales doivent être établis avec la connaissance des fonctions de transfert associées, pour s'assurer que des différences observées sont d'abord dues aux techniques considérées plutôt qu'à une interaction avec des fonctions de transfert.

Nous faisons trois recommandations principales : tout d'abord, pour faciliter la réplication d'expérience, un gain constant devrait être utilisé comme condition de base dans toutes les expériences où le gain n'est pas attendu comme un facteur influençant les performances. En effet, compte tenu de la nature complexe des fonctions observées, de leurs variations suivant les drivers et les fabricants, il peut être extrêmement difficile de répliquer des fonctions dynamiques entre plusieurs expériences. Ensuite, les interfaces utilisateurs pour désactiver l'accélération doivent être considérées avec méfiance et les chercheurs doivent d'assurer que l'accélération est bien désactivée. Enfin, les fonctions de transfert utilisées doivent être décrites avec un maximum de détails.

2.7 Conclusion

Ce chapitre résume un travail conséquent sur les fonctions de transfert pour les systèmes interactifs. Nous avons vu qu'elles sont un élément intrinsèque de l'interaction indirecte et qu'elles restaient très mal connues des chercheurs, compte tenu du faible nombre de travaux sur ce sujet. Nos travaux sur les tâches de pointage ont porté sur la caractéri-

sation des fonctions de transfert utilisées par les systèmes modernes, leur évaluation et comparaison aux fonctions utilisant des gains constants. Ils ont mis en évidence l'intérêt des fonctions dynamiques comme technique de facilitation du pointage. Nous avons également proposé une librairie pour acquérir les informations bas niveau des périphériques, répliquer et appliquer les fonctions de transfert des systèmes. Ce travail permet de faciliter la réplication d'expérience, en particulier lorsque les systèmes que nous utilisons aujourd'hui auront disparu.

La caractérisation des fonctions a mis en évidence les lacunes des systèmes, d'une part pour supporter des fonctions indépendantes des caractéristiques matérielles, et d'autre part, pour tirer partie de la résolution des périphériques d'entrée. Nous avons également montré l'importance de ces fonctions pour les périphériques élastiques et isométriques. Ces études nous ont amené à proposer un périphérique hybride isotonique-élastique pour combiner les avantages de chaque type de retour.

D'une manière générale ces travaux montrent qu'il existe un riche potentiel inexploité dans les périphériques d'entrée classiques. Beaucoup des travaux présentés dans ce chapitre restent préliminaires et ouvrent la voie vers de nombreuses pistes de recherche : toute interaction indirecte fait intervenir des fonctions de transfert qui peuvent être optimisées.

Un des objectifs à court terme est d'obtenir une méthode d'optimisation de la courbe des fonctions de transfert dynamiques pour les tâches de pointage, en prenant en considération les paramètres de la tâche, les capacités des périphériques et celles des utilisateurs. D'autres tâches sont également à considérer comme les rotations 3D avec des souris ou la navigation dans des environnements 2,5D sur tablettes.

Nous avons vu à plusieurs reprises que l'utilisation de la vitesse du mouvement se révèle efficace pour ajuster dynamiquement des paramètres, comme cela a été fait pour le 1€ filter et les fonctions de transfert pour les périphériques isotoniques. Ce paramètre du mouvement mériterait d'être examiné de manière plus systématique pour voir dans quelles autres situations la prise en compte de vitesse pourrait être utile.

D'une manière générale, nous manquons de modèles pour formaliser les fonctions de transfert. Nous avons essayer, à chaque fois que cela était possible, de nous abstraire des conditions expérimentales. Chaque nouvelle tâche nécessite cependant de nouvelles expériences pour tirer de nouvelles conclusions. Nous espérons à terme réussir à dégager des règles universelles sur les fonctions de transfert.

Nous avons dans ce chapitre étudié des interfaces classiques pour l'interaction indirecte. Nous allons dans le prochain étudier des interfaces émergentes que sont les écrans multipoints. Nous allons une nouvelle fois constater qu'il existe de nombreuses perspectives d'amélioration de ces interfaces.

Chapitre

3 Interaction tactile et gestuelle

Parallèlement au premier axe de recherche sur l'interaction indirecte, présenté dans le précédent chapitre, nous avons développé un second axe qui porte sur l'interaction directe avec des périphériques tactiles et sur l'interaction gestuelle.

Les écrans tactiles sont restés plus 30 ans dans les laboratoires avant d'être accessible au grand public. Bill Buxton appelle ce laps de temps *"The long nose of innovation"*. Dès 1972, le PLATO IV [1] utilisait un écran tactile comprenant une grille de 16 lignes et 16 colonnes pour déterminer la position d'un point de contact. Le premier écran multi-points a été développé en 1984 par les laboratoires Bell Labs [2] et il a fallu attendre la sortie de l'iPhone d'Apple en 2007 et celle de l'iPad en 2010 pour démocratiser ce paradigme d'interaction. Ces écrans multipoints sont désormais incontournables pour les périphériques mobiles.

La confusion et l'ambiguïté qui règnent dans les utilisations du terme tactile méritent que l'on reprenne la terminologie. Sans doute le terme d'interaction tactile a-t-il été choisi pour se démarquer de l'interaction avec la souris qui permet d'interagir principalement avec la paume de la main. Peut-être eut-il été plus judicieux de parler d'interaction digitale puisque l'interaction tactile s'entend dans un sens restreint : elle désigne une interaction directe entre les doigts et le contenu affiché sur un écran. Cette confusion entre digitale et tactile se retrouve dans le terme "pavé tactile" (*touchpad* en anglais). Il s'agit ici aussi d'une interaction digitale mais indirecte. Puisque cette terminologie s'est imposée, nous l'utiliserons également en précisant si l'interaction est mono-point ou multipoints, selon qu'elle utilise un ou plusieurs doigts. Le caractère ambigu du terme tactile réside, quant à lui, dans l'utilisation du mot tactile dans le domaine haptique pour parler de retour cutané. Il est en effet possible, comme nous le verrons, de créer un retour tactile sur des surfaces tactiles. Dans ce contexte, l'interaction tactile correspond à la production d'un retour haptique et permet d'obtenir une sensation tactile à l'extrémité des doigts.

Nos travaux sur l'interaction gestuelle sont un prolongement de ceux sur l'interaction tactile. Celle-ci peut en effet être enrichie par l'exploitation d'un historique de données

1. http://en.wikipedia.org/wiki/Plato_computer
2. Bill Buxton a écrit un historique de l'interaction multi-points : http://www.billbuxton.com/multitouchOverview.html

enregistrées lors du geste qui permet l'interaction tactile. Selon Kurtenbach et Hulteen "*A gesture is a motion of the body that contains information. Waving goodbye is a gesture. Pressing a key on a keyboard is not a gesture because the motion of a finger on its way to hitting a key is neither observed nor significant. All that matters is which key was pressed.*" [KH90]. Dans cet exemple du clavier, la façon d'appuyer sur une touche n'est pas prise en compte par le système. Qu'une touche soit pressée avec douceur ou force n'a pas d'importance pour le système, ce qui compte est d'identifier la touche pressée. Les gestes les plus simples sont ceux effectués avec un pointeur de souris ou un doigt. Reconnaître des gestes plus complexes nécessite d'utiliser des périphériques avec un nombre plus important de degrés de liberté.

L'ensemble des problématiques abordées dans ce chapitre est détaillé ci-dessous :

1. Le fait d'avoir une interaction directe introduit une occultation par la main et le bras d'une partie de l'écran. Cette occultation peut cacher tout ou partie de l'objet sur lequel l'utilisateur interagit ou alors cacher des messages affichés par le système. Compte tenu de l'importance de ce problème, il était surprenant de constater qu'il n'existait pas d'étude systématique de ce sujet, que ce soit dans le cadre d'une interaction avec un stylo ou alors d'une interaction multipoints. Nous avons réalisé des études sur ces problèmes et proposé un modèle permettant de prédire la partie de l'écran occultée.

2. Dans la vie quotidienne, le retour d'informations haptiques est important pour manipuler des outils de manière efficace. Par exemple le retour tactile lié à la forme d'un bouton sur un appareil permet de facilement le localiser et l'identifier sans utiliser un retour visuel. Ces informations sont absentes des surfaces tactiles existantes. Nous présenterons une technologie développée au sein de l'équipe MINT qui permet d'obtenir un retour tactile sur surfaces tactiles. Ce retour tactile peut être utilisé pour synthétiser un retour réaliste. On peut également envisager son utilisation dans le but de créer un retour d'information sans chercher à être réaliste. Nous verrons comment utiliser ce retour comme technique d'interaction pour faciliter les tâches de pointage.

3. De plus en plus de contenus sont aujourd'hui proposés en trois-dimensions (3D). Les écrans multipoints représentent un défi pour le développement de nouvelles techniques d'interaction, en particulier pour la manipulation 3D. Les écrans multipoints offrent la possibilité de facilement intégrer ou séparer les degrés de liberté d'une tâche. Nous montrons par exemple qu'intégrer tous les degrés de liberté pour la manipulation 3D ne permet pas nécessairement d'améliorer les performances et nous proposons des stratégies pour séparer de manière efficace les degrés de liberté.

4. Les crayons restent des outils de choix pour un certain nombre de tâches comme l'écriture. Cependant, sans l'utilisation coordonnée de la main non-dominante ou l'utilisation de boutons graphiques, les crayons supportent peu de modes d'interaction. Le plus souvent deux modes sont proposés : un pour écrire et l'autre pour effacer. Nous avons proposé un stylo permettant une transition fluide et élégante entre différents modes d'interaction et la combinaison de l'utilisation d'un stylo et des doigts.

5. Les surfaces tactiles sont propices à l'exécution de gestes de commandes. Les concepteurs d'interfaces sont alors confrontés à la difficulté de choisir des gestes faciles à mémoriser, avec une sémantique adaptée à la commande et faciles à réaliser. Peu de travaux ont cependant étudié la difficulté perçue d'exécution de gestes. Nous évaluons un ensemble de critères permettant de prédire la difficulté d'exécution d'un geste.

6. Le développement récent des interfaces sans contact, de type Microsoft Kinect, permettent de suivre les mains et d'autres parties du corps mais ne disposent naturellement pas de boutons, ce qui pose des problèmes pour la génération d'événements discrets. Nous proposons une technique, basée sur l'analyse de la dynamique et de la trajectoire du geste, pour distinguer les tâches de sélection de celles de saisie d'objets en 3D.

7. L'interaction sur une surface offre des avantages différents de ceux en environnement libre. Nous avons cherché comment combiner interaction sur une surface et interaction au dessus d'une surface pour exploiter au mieux chaque espace d'interaction. Le contexte d'application est ici l'interaction 3D dans un scénario de modélisation 3D.

Mes travaux sur ces questions s'inscrivent dans le cadre des projets ANR InSTInCT [3] et Reactive [4] ainsi que du projet Interreg SHIVA [5]. Deux thèses y ont été associées, l'une a été soutenue [Mar11] et l'autre est en cours (Jérémie Gilliot). Ces travaux ont fait l'objet de treize publications dont je suis coauteur :

– IEEE Haptics Symposium 2008 [**BCGLS08**]
– ACM CHI 2009 [**VCC$^+$09**]
– IEEE VR 2010 [**CCG10**] (article court)
– ACM VRST 2010 [**MCG10b**] (article court)
– IEEE 3DUI 2010 [**MCG10a**] (article court)
– ACM IHM 2010 [**CRVG10**] (prix du meilleur article)
– INTERACT 2011 [**VVCG11**]
– ACM UIST 2011 [**VC11**]
– ACM CHI 2011 [**CRVG11**] (Honorable mention)
– IEEE Haptics Symposium 2012 [**GALSC12**]
– GI 2012 [**DACJ12**]
– ACM CHI 2012 [**VC12**]
– IEEE TVCG 2012 [**MCG12**]

La suite de ce chapitre en présente un résumé.

3.1 Interaction tactile

Nous décrivons dans cette section les travaux réalisés sur l'interaction tactile, en partant de l'étude des problèmes d'occultation sur des tablettes avec stylos et surfaces multipoints ; nous continuons avec la présentation des travaux réalisés dans l'équipe pour

3. http://anr-instinct.cap-sciences.net
4. http://reactive.berck-handicap.com
5. http://www.shiva-project.eu

ajouter du retour tactile aux interfaces tactiles et nous présentons une technique d'interaction qui utilise ce retour pour faciliter les tâches de pointage ; nous continuons avec les problèmes d'intégration et de séparation des degrés de liberté sur les interfaces multi-points dans le cadre de tâches de manipulation 3D ; enfin nous terminons cette section par la présentation d'un crayon multi-modal permettant de passer de manière élégante entre différents modes.

3.1.1 Problèmes d'occultation

Nous utilisons des stylos depuis l'école maternelle. Compte tenu de l'aisance de manipulation de cet outil, acquise au fil des années, on pourrait s'attendre à ce qu'écrire, dessiner et interagir avec des objets à l'aide d'un stylo numérique sur un écran soit aussi facile que sur support papier. La pratique montre qu'il n'en est rien. La difficulté accrue d'utilisation d'un stylo avec support numérique pourrait s'expliquer d'une part, par l'aspect dynamique et interactif du contenu, et, d'autre part, par l'occultation d'une partie de l'écran par la main de l'utilisateur (Figure 3.1a). Ainsi des retours visuels liés à une interaction ou des événements affichés par le système peuvent être occultés. Les tâches sont également plus diversifiées sur support numérique que sur support papier.

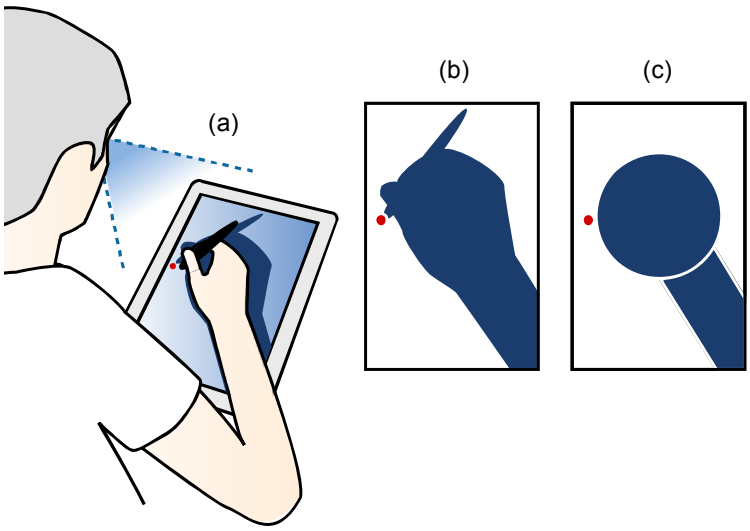


FIGURE 3.1: (a) Résultat de l'occultation engendrée par la main manipulant un stylo en interaction directe. (b) silhouette d'occultation correspondante, du point de vue de l'utilisateur ; (c) modèle géométrique simplifié de l'occultation, basé sur un rectangle et un disque. Le point rouge correspond à l'extrémité du stylo.

Des travaux antérieurs ont suggéré que l'occultation détériore les performances [HB04, IDA+06]. D'autres travaux ont proposé des techniques d'interaction pour minimiser les problèmes d'occultation [AG04, RB03, ZM06]. Aucune étude systématique de ce problème n'a cependant été réalisée et il n'existe aucun modèle permettant de prédire la partie de l'écran occultée à partir des informations de position du stylo. Certains concepteurs d'applications prennent certainement en compte les problèmes d'occultation de

manière empirique (utilisation d'une boîte englobante, voir plus loin), mais cela ne permet pas de proposer des règles de conception universelles. Pour étudier ce problème de manière rigoureuse, nous avons réalisé une expérience contrôlée qui utilise une combinaison inédite de capture vidéo, associée à des marqueurs de réalité augmentée et des techniques de traitement d'image pour isoler les parties de la main et du bras qui occultent l'écran, et cela du point de vue de l'utilisateur [**VCC$^+$09**]. Ces images sont appelées silhouettes d'occultation (Figure 3.1b). L'analyse de ces silhouettes a révélé que la main et le bras peuvent occulter jusqu'à 47% d'un écran de 12" et que la forme de la zone occultée varie entre les participants. A partir de ces observations, nous avons créé un modèle géométrique basé sur 5 paramètres pour décrire la forme générale de la zone occultée, comprenant un cercle et un rectangle ajustables (Figure 3.1c). En utilisant des algorithmes d'optimisation non-linéaires, nous avons ajusté ce modèle géométrique aux silhouettes capturées dans l'expérience. Nous avons obtenu un score de précision/rappel[6] F_1 de 0,81, meilleur que celui d'une boîte englobante (0,40), que les concepteurs d'interfaces utilisent souvent implicitement pour prendre en compte l'occultation. Enfin, nous avons introduit une version prédictive du modèle géométrique pour permettre le développement de nouvelles techniques d'interaction exploitant cette estimation de la zone occultée. Nous développons certains de ces aspects ci-dessous et renvoyons à l'article correspondant pour de plus amples détails [**VCC$^+$09**].

Nous avons commencé par réaliser une étude préliminaire avec 12 participants droitiers afin de mieux cerner les problèmes liés à l'occultation lors de l'utilisation d'une tablette et d'un stylo. Un modérateur guidait les participants pour qu'ils réalisent une tâche consistant à terminer une présentation sur une tablette. Les applications utilisées étaient un navigateur internet, un tableur et un outil de présentations. Les tâches considérées ne nécessitaient pas d'entrer du texte. L'expérience était filmée et les participants utilisaient la méthode de pensée à voix haute. Nous avons mis en évidence que beaucoup de participants n'ont pas vu des messages de notification du système ou des pré-visualisations, par exemple pour le formatage de texte. Nous avons par ailleurs observé la réalisation de mouvements inefficaces, par exemple, lorsque les participants surlignent du texte de gauche à droite (ils surlignent fréquemment davantage de mots que nécessaire) ; des mouvements de contorsion du bras pour réussir à pré-visualiser un résultat ; l'adoption d'une position de repos en bas à droite de l'écran (tous les participants étaient droitiers).

Nous avons également mené une expérience formelle pour mesurer la taille et la forme des zones occultées sur une tablette. Nous avons réalisé cela en enregistrant le point de vue de chaque participant à l'aide d'une caméra installée sur la tête (3.2a). Nous avons considéré une tâche de sélection et une autre consistant à tracer un cercle selon des conditions pré-définies. Les traitements d'images effectués pour obtenir les silhouettes d'occultation, représentées sur la figure 3.5, sont détaillées dans [**VCC$^+$09**].

6. En reconnaissance de formes, la précision correspond à la proportion d'éléments extraits par une méthode qui sont corrects et le rappel correspond à la proportion d'éléments extraits qui sont corrects par rapport à l'ensemble des éléments corrects existants.

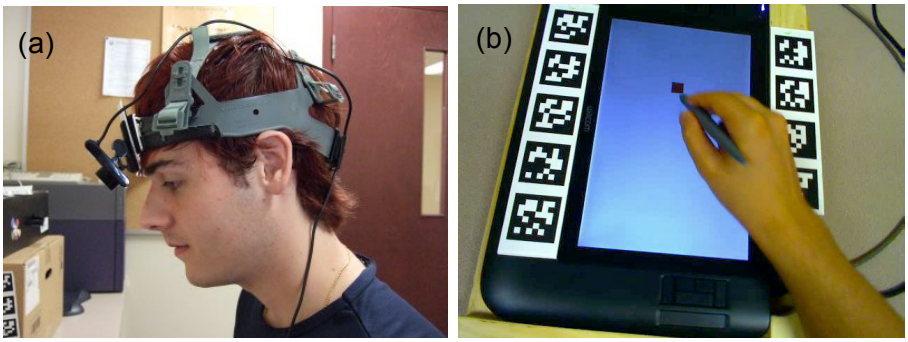


FIGURE 3.2: Matériel utilisé dans l'expérience : (a) caméra positionnée sur la tête pour capturer le point de vue de l'utilisateur ; (b) marqueurs attachés aux bords de l'écran (image prise par la caméra installée sur la tête).

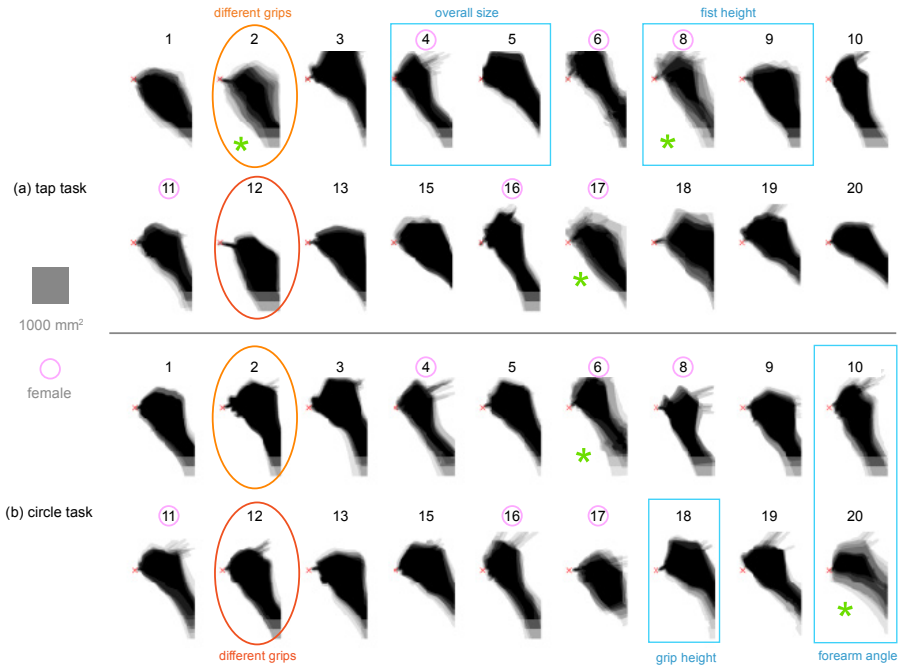


FIGURE 3.3: Silhouette d'occultation moyenne de chaque participant pour : (a) tâche de sélection et (b) tâche de tracé de cercle.

Puisque l'occultation varie en fonction de la position du stylo sur l'écran, nous avons également calculé la proportion de l'écran occulté pour chaque case d'une grille de 7 × 11 cases. Comme nous pouvions nous y attendre, nous observons que l'écran est d'autant plus occulté que le sylo se trouve en haut à gauche (Figure 3.4).

Les analyses de cette expérience révèlent quatre résultats principaux. Tout d'abord, une partie importante de l'écran peut être occultée suivant la position du stylo. Dans l'expérience, il y avait typiquement jusqu'à 38% de l'écran occulté. Le second point est que

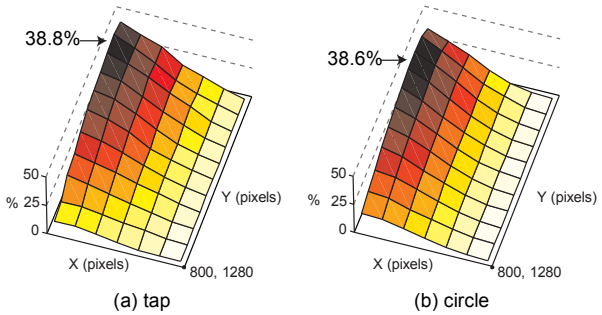


FIGURE 3.4: Proportion de l'écran occulté suivant la position du sylo sur l'écran et la tâche considérée.

les pixels qui se situent autour de l'extrémité du stylo ne sont pas occultés autant que prévu. Troisièmement, nous avons observé que les participants semblent avoir des signatures de silhouettes d'occultation personnelles mais qu'on observe de grandes similarités entre elles. Comme dernier point, il semble qu'il n'existe pas de relation simple entre la taille de la zone occultée et la taille des membres. Les différences entre les silhouettes d'occultation peuvent s'expliquer principalement par différentes façons de tenir le stylo.

Pour modéliser les silhouettes d'occultation, nous avons envisagé différents modèles de formes d'occultation (Figure 3.5). Si, en ajustant les modèles aux silhouettes d'occultation, nous obtenons des valeurs qui sont constantes à travers les différentes conditions expérimentales, nous pourrons alors utiliser ces informations pour développer un modèle prédictif d'occultation en fonction de la position du stylo. Nous sommes partis de deux modèles opposés : un premier modèle utilisant un rectangle englobant qui suppose que les pixels situés à droite et en dessous de la position du stylo sont occultés, et un second modèle qui utilise une forme flexible faite de courbes de Bézier. Bien que ce dernier soit très précis, son nombre important de paramètres rend difficile l'ajustement du modèle et l'interprétation des résultats, ce qui complique sa mise en application. Notre objectif était de créer un modèle avec un faible nombre de paramètres permettant néanmoins d'obtenir de bons résultats. Pour cette raison nous avons remplacé le second modèle par un cercle et un rectangle, illustré figure 3.5b et dont les détails se trouvent dans [**VCC$^+$09**]. Un ajustement des modèles aux données de l'expérience montre que le modèle géométrique donne de bien meilleurs résultats que le modèle de boîte englobante (Figure 3.5). Les valeurs obtenues pour chaque paramètre du modèle et pour chaque participant sont détaillées dans l'article. Celles-ci varient peu entre les participants pour le rayon du cercle et la largeur du rectangle. D'autres paramètres liés à la façon de saisir le stylo sont sensiblement les mêmes pour chaque participant. Par contre, l'angle lié à l'orientation du bras présente une variance plus importante qu'un modèle prédictif devrait ajuster. Les bases d'un modèle cinématique inverse ont été proposées pour estimer la configuration du bras à partir de la connaissance de la position de l'extrémité du stylo. Celui-ci a été approfondi par Daniel Vogel afin de proposer un système qui permet de révéler des informations normalement cachées [VB10].

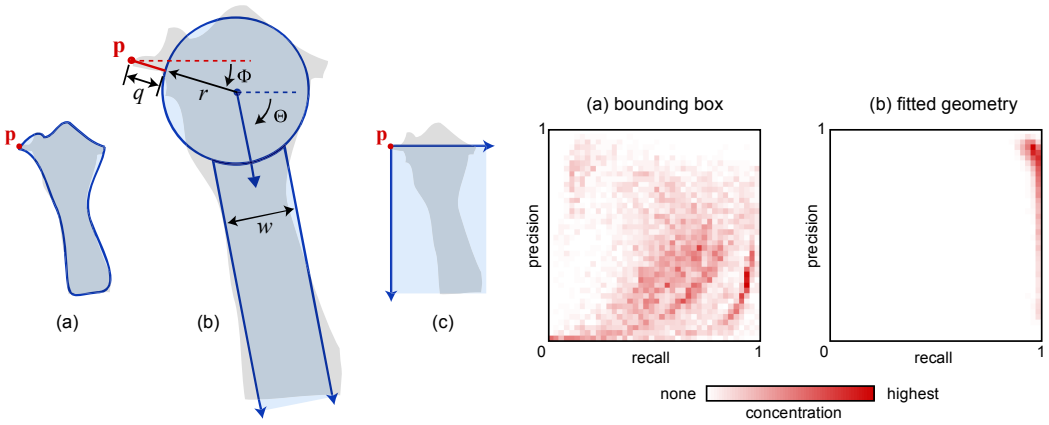


FIGURE 3.5: A gauche : Trois modèles de formes d'occultation : (a) courbes de Bézier ; (b) cercle et rectangle ; (c) rectangle englobant. p est la position du stylo. A droite : performance précision/rappel pour la boîte englobante (a) et le modèle géométrique (b).

L'interaction tactile multipoints pose bien entendu des problèmes similaires d'occultation, qui sont cependant plus complexes. L'étude de ce problème est plus difficile que dans le cas de l'interaction avec un stylo, compte tenu du vocabulaire important reposant sur l'utilisation de différentes combinaisons des doigts des deux mains. Nous avons étudié 18 conditions couvrant 9 types de contacts avec 3 tâches : actions de toucher l'écran (*tapping* en anglais), glisser-déposer (*dragging* en anglais) et transformer (Figure 3.6) [**VC12**]. Nous avons réalisé l'expérience sur une table Microsoft Surface 1, utilisant l'illumination diffuse infra-rouge et une configuration très proche de celle de l'expérience précédente. La méthodologie pour obtenir les formes d'occultation est la même.

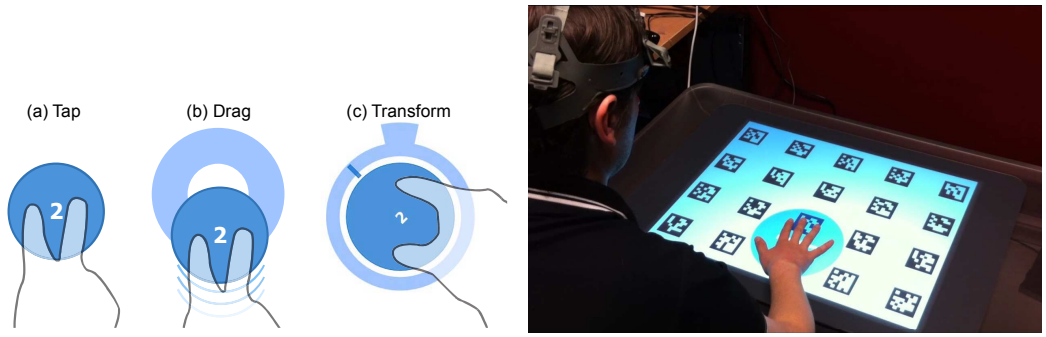


FIGURE 3.6: Gauche : Tâches expérimentales (exemple avec 2 doigts) : (a) toucher l'écran ; (b) glisser-déposer un objet ; (c) tâche de transformation comprenant rotation et changement d'échelle. Droite : configuration expérimentale. La zone occultée est capturée par une caméra fixée sur la tête et rectifiée en utilisant des marqueurs de réalité augmentée affichés sur l'écran.

L'observation des différentes silhouettes d'occultation a montré que les participants ont des postures de mains similaires. Bien qu'il existe des différences entre les individus, il

y a suffisamment de points communs pour utiliser les formes moyennes d'occultation afin de déterminer la partie de l'écran occultée dans un contexte interactif (Figure 3.7). Pour guider les concepteurs d'interfaces à prendre en compte l'occultation, nous avons ainsi proposé des gabarits de formes d'occultation calibrés. Nous avons ensuite proposé deux modèles permettant de prendre en compte l'occultation en temps réel. Comme nous avons conduit notre expérience sur une surface basée sur une technologie optique, nous proposons un premier modèle d'occultation prenant en compte les informations infra-rouges avec un rectangle représentant l'avant bras (Figure 3.7). Pour obtenir un modèle fonctionnant également pour des technologies capacitives, qui ne fournissent que la position des doigts, nous avons créé un second modèle. Celui-ci est basé sur le modèle d'occultation précédemment présenté pour le stylo, en ajoutant des ellipses pour les doigts, sans augmenter le nombre de paramètres du modèle. L'évaluation des deux modèles a montré que leur performance est bonne (score F1 > 0.8), ce qui montre que la proposition d'interfaces prenant en compte l'occultation en temps réel est possible, indépendamment de la technologie utilisée.

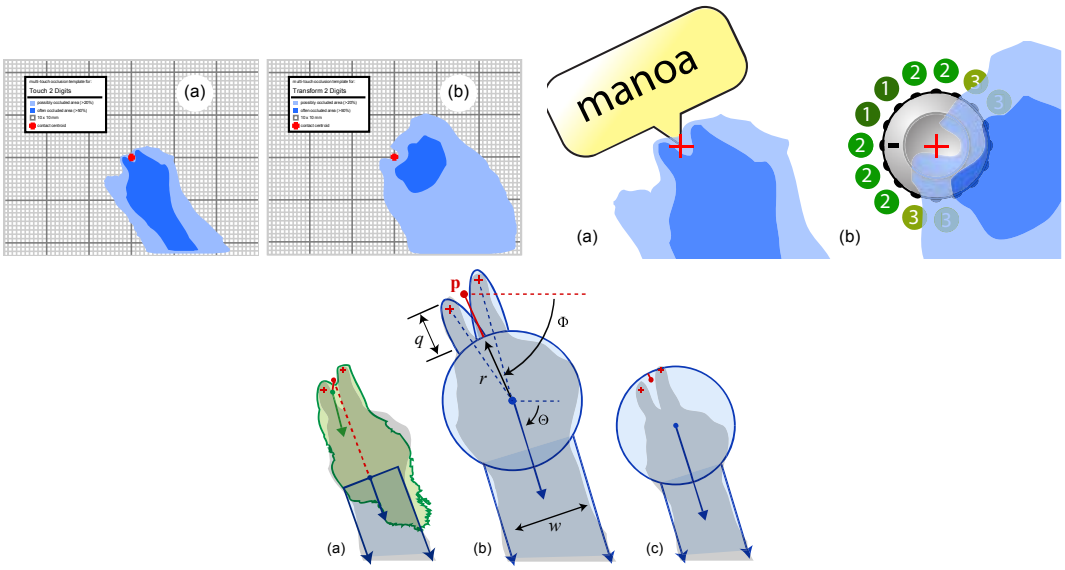


FIGURE 3.7: Haut gauche : gabarits d'occultation pour : (a) tâche de toucher avec deux doigts ; (b) transformation avec deux doigts. Le bleu foncé représente la zone souvent occultée (> 50%) et le bleu ciel, la zone peu occultée (> 10%). La croix rouge représente le centroïde de contact. Haut droite : Exemple d'application des gabarits d'occultation : (a) optimisation du placement d'une bulle d'information activée avec 2 doigts ; (b) Options de placement d'un retour visuel contrôlé par un potentiomètre rotatif contrôlé avec deux doigts : les chiffres représentent la pertinence de placement de l'information. En bas : trois modèles d'occultation : (a) modèle avec les informations infra-rouge (en vert) et un rectangle ; (b) modèle avec cercle, rectangle et ellipses ; (c) modèle utilisé pour l'occultation avec un stylo.

Les différents travaux présentés ont étudié le problème d'occultation dans des contextes d'interaction avec un stylo et d'interaction multipoints. Nous avons proposé différents

modèles qui permettent de prendre en compte ces occultations en temps réel dans les interfaces. Ces travaux permettent de bénéficier des fonctionnalités développées pour l'interaction avec une souris, quand on utilise une interface tactile. En plus de ne pas perdre "l'existant", nous avons voulu développer une technologie inédite qui permet d'ajouter du retour tactile sur les surfaces tactiles. Cette technologie permettra peut-être d'utiliser un retour tactile pour palier les problèmes d'occultation, en informant par exemple l'utilisateur qu'un objet occulté vient de changer d'état.

3.1.2 STIMTAC : ajouter du retour tactile sur les surfaces tactile

La production de retour tactile sur des surfaces a fait l'objet de nombreux travaux de recherche. La principale technologie utilisée pour créer ce retour est celle des matrices de picots, où chaque picot peut être déplacé indépendamment selon une direction normale à la surface de simulation [WF95, NMYT98, TFK+06, YNYH06]. Les matrices de picots permettent d'obtenir des sensations différentes simultanément sous un doigt. Cependant, la densité des picots est souvent faible et leur dynamique est aussi limitée, ce qui réduit la qualité du rendu. Cette technologie est par ailleurs encombrante et incompatible avec l'interaction co-localisée.

L'idée d'utiliser des picots est de reproduire la géométrie d'une surface. Une autre approche est de modifier dynamiquement les propriétés de cette surface et ainsi produire une information qui peut être interprétée par les utilisateurs comme un changement local de géométrie. Deux technologies permettent actuellement de modifier le frottement d'une surface : l'électro-vibration et l'effet mécanique de film d'air comprimé. L'électro-vibration permet d'augmenter le frottement d'une surface en attirant plus ou moins le doigt d'une surface, par le contrôle d'une tension appliquée au corps de l'utilisateur [BPIH10]. L'effet mécanique de film d'air comprimé permet, quant à lui, de réduire le frottement d'une surface [BGLS07]. Le film d'air comprimé est généré par la mise en vibration d'une surface à une fréquence ultrasonique et une amplitude de quelques micromètres. Comme la fréquence de vibration est en dehors de la bande-passante des mécano-récepteurs de la peau, on ne sent pas la vibration elle-même mais son effet sur la tribologie du toucher : la surface devient plus glissante quand l'amplitude augmente. Contrairement aux matrices de picots, les interfaces à frottement programmable permettent d'obtenir une finesse de retour tactile seulement limitée par la résolution du capteur de mesure de position du doigt. Leur faible encombrement, faible consommation, et relative facilité de mise en oeuvre suscite beaucoup d'intérêt pour les périphériques mobiles mais également pour les pavés tactiles.

Le STIMTAC, développé à Lille au sein de l'équipe projet ALCOVE puis MINT, a été l'un des premiers dispositifs à utiliser le film d'air comprimé pour créer un retour tactile. Le premier prototype en une dimension a été développé dès 2004 et il a fallu 4 ans de travail pour obtenir une surface en deux dimensions permettant de produire un retour tactile et en même temps capable de mesurer la position du doigt, comme le fait un pavé tactile. Deux années supplémentaires ont été nécessaires pour le miniaturiser et réduire

sa consommation énergétique (Figure 3.8). Un historique des différentes versions est détaillé dans cet article [**AGS+11**].

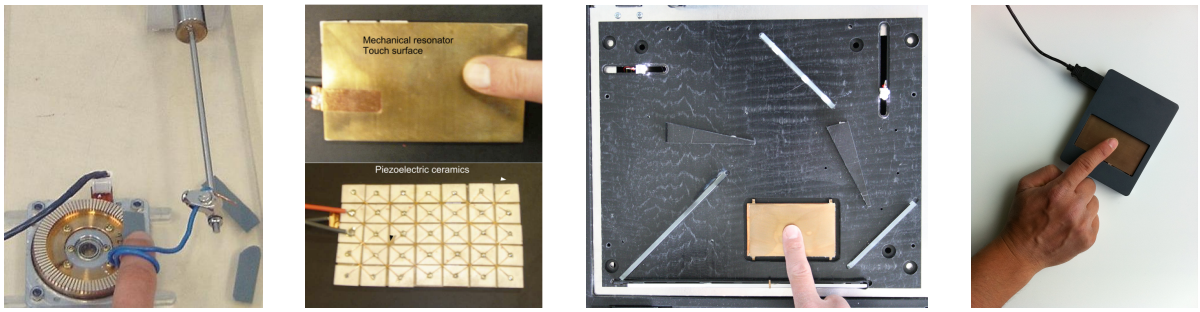


FIGURE 3.8: Les différentes versions de STIMTAC. Préfiguration en 2004, version produisant uniquement un retour tactile en 2007, version produisant un retour tactile et permettant de mesurer la position du doigt en 2008 et prototype de version miniaturisée USB en 2010.

Ces différentes versions de prototypes étaient opaques : elles utilisaient une surface métallique en cuivre-béryllium dont la face arrière était recouverte de céramiques piézoélectriques. Nous avons récemment développé une version qui permet de faire vibrer un support transparent tout en plaçant les céramiques aux extrémités [**GALSC12**]. Cette version laisse envisager l'utilisation de cette technologie pour des périphériques mobiles et des tablettes.

Le frottement programmable peut être utilisé pour synthétiser un retour réaliste correspondant à la texture d'un objet. Pour évaluer cette capacité de retour et le degré de finesse du rendu, nous avons simulé des textures dont la forme de base est un signal carré. Le choix de cette forme de signal a été réalisé après avoir observé que les variations franches de frottement sont plus faciles à percevoir que des variations continues, comme c'est le cas pour des formes sinusoïdales. Nous avons ensuite fait varié la période spatiale de ce signal et réalisé une expérience de détection de seuils différentiels (ou JND pour *Just Noticeable Difference*) [**BCGLS08**]. L'expérience consistait à simuler tour à tour deux périodes spatiales. Celles-ci étaient simulées sur le même périphérique et les utilisateurs pouvaient passer d'une période à l'autre à l'aide des touches de clavier. Ils devaient indiquer quelle était la période spatiale la plus grande. Différentes périodes spatiales de référence ont été évaluées (0.25, 0.35, 0.5 et 1.0 cm) et pour chacune d'entre elles des comparaisons étaient réalisées, allant de -20% à 20% avec un pas de 5%. La fraction de Weber obtenue est de l'ordre de 9%, pour toutes les périodes spatiales de référence. Ce résultat montre que les périodes spatiales sont perçues avec la même acuité sur l'intervalle de valeurs considérées. Par ailleurs, ces résultats sont proches de ceux obtenus par d'autres chercheurs pour des périodes spatiales réelles [MGDS84, NKK01].

Cette expérience montre qu'il est possible de produire un retour tactile permettant de percevoir des textures avec la même finesse que dans la réalité. D'autres expériences de

FIGURE 3.9: Version transparente de STIMTAC. Les céramiques piézo-électriques placées de chaque côté de la surface sont faciles à distinguer.

psychophysique sont à réaliser afin de mieux comprendre la relation entre les paramètres de contrôle du STIMTAC et ce que les gens perçoivent. Nous allons voir dans la section suivante comment utiliser STIMTAC dans une tâche de facilitation du pointage. L'analyse des résultats des expériences permettra de mieux comprendre la nature de ce que les utilisateurs perçoivent.

3.1.3 Surfpad : Amélioration du pointage en utilisant le retour tactile

Il parait difficile d'envisager utiliser un jour la version actuelle de STIMTAC pour simuler une large gamme de textures réelles. En revanche, le périphérique est capable de produire un retour d'information riche qui peut être utilisé pour enrichir les interfaces et développer de nouvelles techniques d'interaction homme-machine. L'idée d'utiliser le STIMTAC pour la facilitation du pointage a été inspirée par les techniques d'interaction qui modifient le gain sur les cibles pour en faciliter l'acquisition [WWBH97, CF03, BGBL04], en particulier SemanticPointing de Renaud Blanch et al. [BGBL04]. Ces techniques réduisent le gain quand le pointeur est sur une cible dans le but d'en agrandir la taille dans l'espace moteur et ainsi réduire l'indice de difficulté de la tâche. Indirectement, l'effet obtenu est de freiner le pointeur sur les cibles. Comme le STIMTAC est capable de moduler le coefficient de frottement de sa surface, l'idée est d'augmenter la quantité de frottement sur les cibles pour freiner le doigt. Il s'arrêterait plus rapidement et la cible pourrait ainsi être acquise plus vite.

Des travaux antérieurs ont déjà investigué l'utilisation de retour haptique pour faciliter

l'acquisition de cibles. Les tâches de pointage discret 2D sont celles qui ont reçu le plus d'attention [AMH95, AM96, Key97, OMBG00, DY01, HKLC03]. Le pointage réciproque 1D [CB05], le pointage réciproque 2D et le franchissement [FB08], la traversée de tunnels [CZMM99, DMH00] ainsi que des tâches de nature écologique [OMBG00, DMH00, CB05] ont également été étudiés. Le dispositif le plus utilisé pour ce type d'étude est une souris, les autres dispositifs incluant un *trackpoint* [CZMM99], un stylo [FB08], un *trackball* [Key97], un *joystick* [HKLC03] et un dispositif à six degrés de liberté [OMBG00]. Le retour haptique est fourni en exerçant une force sur le dispositif pour contraindre son mouvement [AM96, Key97, OMBG00, DMH00, DY01, HKLC03] ou en déplaçant ou faisant vibrer un ou plusieurs de ses éléments [AMH95, AM96, CZMM99, CB05, FB08].

Le pointage avec retour haptique a été principalement comparé au pointage sans retour d'information. Toutes les études ont montré un effet positif, le retour haptique réduisant le temps de pointage [AM96, CZMM99, DMH00, DY01, HKLC03, CB05, FB08], le temps pour s'arrêter sur la cible [AMH95] ou le taux d'erreur [OMBG00]. Le déplacement ou la vibration de parties mécaniques pouvant générer des sons audibles, des précautions ont souvent été prises pour les masquer, le retour auditif et le retour haptique produisant des effets similairement positifs [AMH95, CB05]. Le retour haptique a également été comparé au retour visuel [AMH95, CZMM99] et à l'adaptation de gain [CB05], et des chercheurs se sont intéressés à la combinaisons des différentes modalités [AMH95, AM96, CZMM99, CB05].

Les dispositifs haptiques cités sont tous basés sur des technologies électromagnétiques simples et éprouvées (e.g. solénoïdes, moteurs vibrants et masselottes). La gamme de sensations haptiques permises est en contrepartie très limitée. Contrairement aux dispositifs vibrotactiles classiques, ceux basés sur l'effet de film comprimé fournissent un retour passif : ils ne transfèrent pas d'énergie mais modifient la manière dont elle se dissipe au niveau de la zone de contact lors d'un processus de friction initié par l'utilisateur.

Les techniques de pointage sensibles aux cibles sont *a priori* plus efficaces lorsque celles-ci sont espacées [Bal04]. Dans la réalité cependant, il n'est pas rare d'observer des groupes de cibles relativement denses. Bien que ce problème soit clairement identifié dans la littérature, peu de recherches ont été menées pour évaluer l'impact exact des distracteurs sur les techniques existantes, ou pour en concevoir de nouvelles les prenant spécifiquement en compte.

Certaines conditions des expériences décrites dans [Key97] et [WWBH97] impliquaient un distracteur incontournable sur le chemin de la cible. Une condition de l'expérience de [HKLC03] impliquait un distracteur placé à 0, 90, 180 ou 270° autour de la cible par rapport à la direction du pointage. Dans [DY01], l'utilisateur devait pointer alternativement une cible parmi 13 disposées en croix, les autres jouant le rôle de distracteurs contournables. Néanmoins, Cockburn & Brewster [CB05] sont *a priori* les seuls à avoir étudié l'impact de distracteurs multiples sur un retour haptique et une technique agissant dans l'espace moteur (*sticky targets*) pour une tâche écologique (sélection dans un menu). Les résultats de toutes ces expériences suggèrent un impact négatif des distracteurs sur le

temps de pointage, le taux d'erreur ou la satisfaction des utilisateurs, tous les auteurs recommandant une étude plus poussée de cet impact.

La méconnaissance de l'impact des distracteurs sur les techniques de facilitation du pointage et les perspectives ouvertes par le STIMTAC sont également à l'origine de notre travail. L'idée est toujours de créer une différence de frottement. Puisque le STIMTAC ne peut que réduire le frottement et non l'augmenter, nous proposons une technique qui repose sur l'inversion figure/fond : au lieu de ralentir le pointeur à proximité des cibles, pourquoi ne pas faciliter son déplacement là où il n'y en a pas ? Notre technique, appelée Surfpad [**CRVG10**, **CRVG11**] utilise ainsi l'effet de film comprimé de STIMTAC pour réduire le coefficient de friction de la plaque lorsque le pointeur est en dehors des cibles et le laisser à son niveau normal dans le cas contraire. Différentes fonctions de modulation de la friction peuvent être envisagées : une fonction en escalier qui augmente instantanément le frottement au franchissement de la cible et une courbe en cloche, similaire à celle utilisée par Blanch et al. [BGBL04] qui modifie progressivement le frottement suivant la distance au centre de la cible.

Nous avons réalisé différentes expériences pour comparer les performances de Surfpad à celles d'une technique de référence : utilisation d'un gain constant de 2 avec le frottement par défaut de la surface (*Control*) et le frottement minimal (*Control-*), correspondant au STIMTAC toujours activé ; Semantic Pointing testé avec une fonction en marche d'escalier (*SemPoint*Π) et une courbe en cloche (*SemPoint*Ω) ; Surfpad testé avec une fonction qui augmente le frottement et dont la courbe à la forme d'une marche d'escalier (*Surfpad*Π) et avec une autre fonction dont la courbe est une cloche identique à utilisé dans Semantic Pointing (*Surfpad*Ω). Les résultats de la comparaison des 6 techniques entre elles, testées avec la même fonction de gain, n'ont pas montré de différences significatives entre les deux conditions de contrôle. *Surfpad*Π et les 2 variantes de Semantic Pointing améliorent les performances de 9 et 18% respectivement, comparé aux conditions de contrôle (Figure 3.10). *Surfpad*Ω n'a pas montré de différence significative avec les conditions de contrôle. Les résultats montrent également que ces différences peuvent être expliquées, d'une part, par une réduction significative du temps d'approche pour *Surfpad*Π et les deux conditions de Semantic Pointing, et d'autre part, par une réduction significative du temps de sélection pour *SemPoint*Π et *SemPoint*Ω, comparé aux autres techniques.

L'amélioration des performances observées pour *SemPoint*Π peuvent s'expliquer par un effet mécanique lié à l'augmentation du frottement ou alors un retour d'information lié au changement instantané de frottement au franchissement de la frontière de la cible. L'analyse détaillée des résultats nous conduit à privilégier la seconde hypothèse : un retour d'information serait prépondérant à l'effet mécanique et permettrait de diminuer les temps de réponse. Pour valider cette hypothèse, nous avons testé la condition Anti-Surfpad, qui consiste à diminuer le frottement sur les cibles. Nous avons cependant observé une dégradation significative des performances dans cette condition. Cela peut être expliqué soit par un effet mécanique négatif plus important que le retour d'informa-

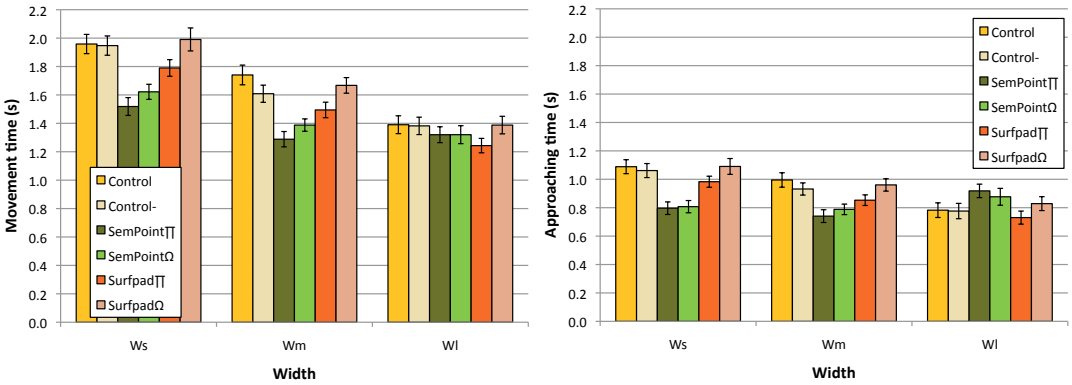


FIGURE 3.10: Temps moyen de pointage pour chaque technique et chaque largeur (gauche). Temps d'approche pour chaque technique et chaque largeur (droite). Les barres d'erreur représentent les intervalles de confiance à 95% de la moyenne.

tion ou un retour d'information contre-intuitif. D'autres expériences seraient nécessaires pour conclure sur ce point.

Nous avons enfin réalisé une expérience visant à évaluer l'influence des distracteurs pour chacune des techniques. Nous avons pour cela fait varier la densité de distrateurs. Les résultats montrent que Surfpad permet de conserver un gain de performance de l'ordre de 9% comparé à la condition de contrôle, cela indépendamment du nombre de distrateurs. Semantic Pointing montre une dégradation des performances qui peut atteindre 100% pour la plus grande densité de distracteurs (Figure 3.11). Cela peut s'expliquer par l'augmentation du débrayage dans l'espace moteur lié aux plus grandes distances à parcourir. Ces résultats renforcent notre conviction que Surfpad fournit principalement un retour d'information plutôt qu'un effet mécanique.

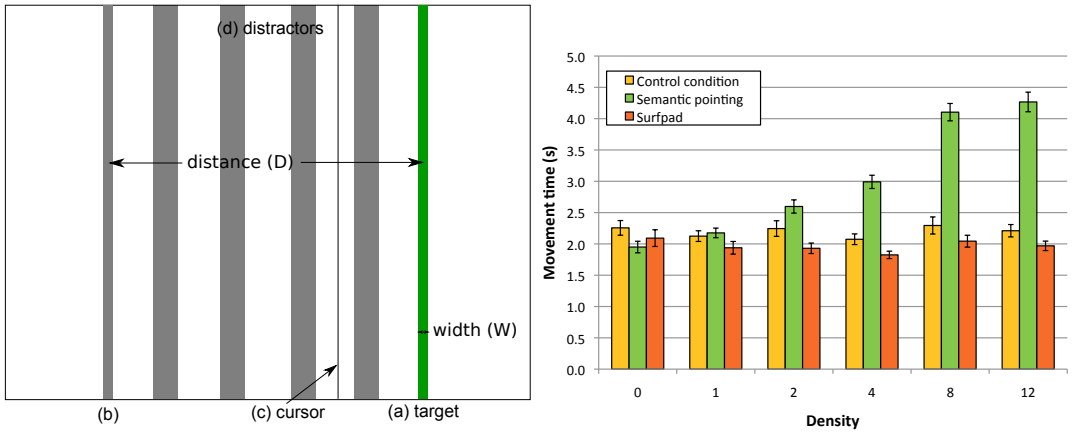


FIGURE 3.11: Représentation des cibles et des distrateurs à l'écran (gauche). Temps moyen de pointage pour chaque technique et densité de distracteurs (droite). Les barres d'erreur représentent les intervalles de confiance à 95% de la moyenne.

La robustesse de Surfpad aux distracteurs est particulièrement novatrice dans la catégorie des techniques de facilitation du pointage sensibles aux cibles. Cela permet de ne plus avoir à déterminer avec exactitude quelles cibles vont être sensibles.

Le cadre des tâches de pointage considéré pour Surfpad ne permet pas de mettre à profit le contrôle simultané de plusieurs points de contact rendu possible par les surfaces multipoints. Les exemples les plus fréquents en deux dimensions sont la mise à l'échelle d'un document et la translation, la rotation et le changement d'échelle d'objets. Ces opérations permettent un contrôle combiné des degrés de liberté. D'autres tâches nécessitent de contrôler davantage de degrés de liberté, comme les tâches de manipulation 3D. Ces tâches posent la question de la façon de mettre en relation les degrés de liberté contrôlés sur la surface tactile avec ceux manipulés dans la tâche, ce que nous allons étudier dans la section suivante.

3.1.4 Manipulation 3D sur surfaces multipoints

La manipulation 3D reste un défi pour les concepteurs d'interfaces 3D puisqu'elle implique le contrôle de 6 degrés de liberté : 3 pour la position et 3 pour la rotation. Avec une souris et les interfaces de bureau actuelles, la rotation d'un objet en 3D peut prendre de 10 à 30 secondes [HTP+97], bien plus lente que la rotation d'un objet réel qui s'effectue en 1 à 2 secondes [WMSB98].

Comparé à la souris, les interfaces multipoints fournissent une bande passante supplémentaire grâce à la manipulation simultanée de plusieurs points de contacts. Le geste de translation/rotation/changement d'échelle (TRS) pour la manipulation de données 2D est un exemple typique de ce paradigme d'interaction [HCV+06]. La manipulation 3D avec des interfaces multipoints reste peu étudiée alors que la manipulation multipoints de données en 2D a été largement explorée. Ceci peut s'expliquer par la difficulté de mettre en relation la position des points de contact avec les attributs 3D à contrôler.

Jacob et al. [JSMM94] ont étudié l'influence sur les performances de la relation entre la structure de contrôle d'un périphérique et la structure perceptuelle de la tâche. Les attributs peuvent être séparables ou intégrables : ils sont intégrables s'ils sont perçus comme un tout indivisible et séparables si les attributs peuvent être perçus comme distincts selon des dimensions identifiables. Ils ont mis en évidence que les performances sont meilleures quand les deux structures sont en correspondance. Selon cette définition, l'orientation et la position d'un objet 3D sont deux triplets d'attributs intégrables, ce qui fait de la manipulation 3D une tâche intégrable [Gar74]. Il a été montré que les doigts ont des degrés de liberté séparables [IKHW08]. Le pouce, l'index et le majeur peuvent bouger séparément les uns des autres alors que les utilisateurs perçoivent les attributs de position et d'orientation comme un tout. Ce décalage entre la nature séparable de la structure d'entrée des périphériques multipoints et la nature intégrable des tâches de manipulation 3D pose la question de la façon d'optimiser la mise en correspondance de ces deux structures.

Ce thème de la séparation et de l'intégration des degrés de liberté était au centre de ma

thèse. Les travaux de thèse d'Anthony Martinet dans le cadre du projet InSTInCT ont permis d'approfondir cette question dans le contexte des interfaces multipoints pour les tâches de manipulation 3D.

Peu de techniques ont été proposées dans la littérature pour la manipulation 3D sur interfaces multipoints. Hancock et al. ont présenté des techniques de manipulation 3D en utilisant un à trois doigts [HCC07]. Avec trois doigts, les utilisateurs peuvent à la fois translater et faire tourner les objets 3D. Cette technique, appelée Shallow-Depth, ne permet cependant de manipuler que 5 degrés de liberté. Pour manipuler les 6 degrés de liberté nécessaires à la manipulation 3D, les auteurs ont introduits Sticky Tools [HCC09], où chaque doigt contrôle séparément des degrés de liberté qui sont intégrés. Cependant le choix des degrés de liberté manipulés par chaque doigt n'est pas motivé et l'efficacité de Sticky Tools n'a pas été évaluée. Reisman et al. [RDH09] ont introduit, quant à eux, une méthode pour manipuler de façon intégrée les six degrés de liberté de la manipulation. Leur méthode utilise un solveur de contraintes qui minimise entre deux pas de temps l'erreur entre la projection à l'écran des points de contact sur l'objet 3D et la position des doigts sur l'écran, de manière à obtenir un équivalent de la technique TRS en 3D. La technique n'a jamais été évaluée non plus.

Les techniques de manipulation 3D listées précédemment permettent de contrôler différents sous-ensembles de degrés de liberté suivant le nombre de doigts en contact avec l'écran. Pour en faciliter la comparaison, nous avons choisi d'adapter la taxonomie de Card et al. [CMR91]. Nous voulions un moyen de représenter la relation entre le nombre de doigts, leur utilisation de manière directe ou indirecte et les degrés de liberté contrôlés dans la tâche. Nous voulions aussi représenter si ces degrés de liberté sont contrôlés de manière intégrale ou séparable. Pour illustrer la taxonomie, les techniques Sticky Tools et Screen-Space sont représentées sur la figure 3.12. Les détails de la taxonomie sont présentés dans l'article associé [**MCG12**].

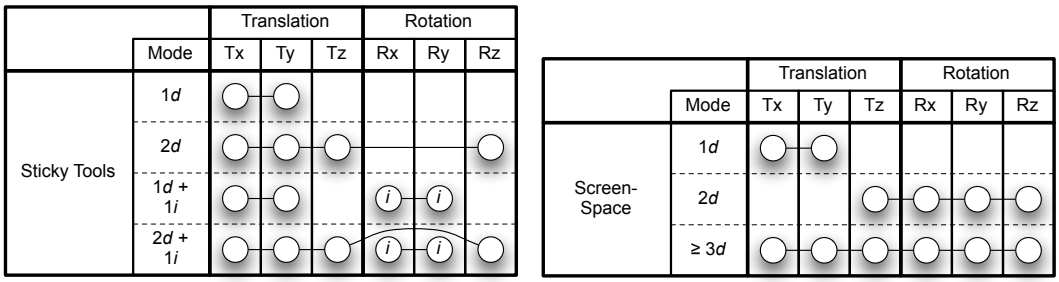


FIGURE 3.12: Représentation de Sticky Tools et Screen-Space en utilisant la taxonomie proposée.

En considérant la nature séparable des degrés de liberté de la main, il apparaît impossible de faire correspondre exactement la structure perceptuelle de tâche avec la structure de contrôle des écrans multipoints. Les techniques d'interaction précédemment présentées montrent qu'il n'existe pas de réponse claire à ce problème. D'un côté, Screen-Space pro-

pose de contrôler les 6 degrés de liberté de façon intégrale et d'un autre côté Sticky Tools propose une séparation entre les degrés de liberté. Comme ces deux techniques n'ont été ni évaluées ni comparées, il est difficile de savoir quelle approche est la meilleure. Si la séparation des degrés de liberté apparaît comme étant la meilleure, cela pose également la question de la meilleure façon de les séparer.

Durant des évaluations informelles de Sticky Tools, nous avons observé que le contrôle intégré des translations et rotations est difficile à utiliser. Nous avons émis l'hypothèse qu'une séparation claire des degrés de liberté des rotations et translations améliore la performance des utilisateurs quand les degrés de liberté de la manipulation 3D doivent être contrôlés de manière séparés. Partant de ce principe, nous avons proposé une technique, appelée DS3. Cette technique combine la Z-technique [**MCG10a**] pour le contrôle des translations et le solveur de contrainte de Reisman et al. [RDH09] pour contrôler l'orientation.

Nous avons comparé cette technique à Sticky Tools et Screen-Space dans une tâche d'insertion d'un cylindre dans un trou (Figure 3.13). Les détails de l'expérience sont décrits dans [**MCG10b**]. Les résultats montrent que DS3 améliore le temps de la tâche de 36% comparé à Screen-Space et 34% comparé à Sticky Tools. Ces résultats peuvent être expliqués par une analyse de l'efficacité de coordination des degrés de liberté de la tâche. Screen-Space, qui couple fortement les translations et rotations, montre la plus faible coordination pour les translations et les rotations. Il apparaît que, pour la tâche de manipulation 3D que nous avons considérée, l'utilisation les degrés de liberté séparés d'un écran multipoints de façon intégrée dégrade les performances. En revanche, séparer les degrés de liberté de la tâche pour se rapprocher de la structure séparée du périphérique d'entrée permet d'obtenir de meilleures performances. Ainsi, suivre les conclusions de Jacob et al. dans le cas de périphériques 2D pour manipuler des objets 3D n'est pas possible. Quand on doit réaliser un choix entre séparation et intégration des degrés de liberté, notre étude suggère que les performances sont meilleures si la technique d'interaction suit la structure du périphérique d'entrée plutôt que la structure de la tâche.

Les résultats de l'expérience montrent également que la stratégie de séparation des degrés de liberté a de l'importance : les performances de DS3 sont 34% meilleures que Sticky Tools. L'analyse détaillée des données suggère que la séparation claire des translation et rotations permet d'expliquer ce résultat.

Les techniques de manipulation 3D dont nous venons de parler définissent des modes d'interaction suivant le nombre de doigts en contact avec la surface. Nous allons voir un autre moyen de spécifier des modes d'interaction, en utilisant un stylo multi-modal.

3.1.5 Conté : Stylo multimodal

La souris informatique peut fournir différents modes d'interaction en fonction des boutons qui sont pressés. Compte tenu du faible nombre de boutons, les concepteurs d'interfaces ont davantage recours à l'utilisation d'icônes, boutons, menus et palettes d'outils

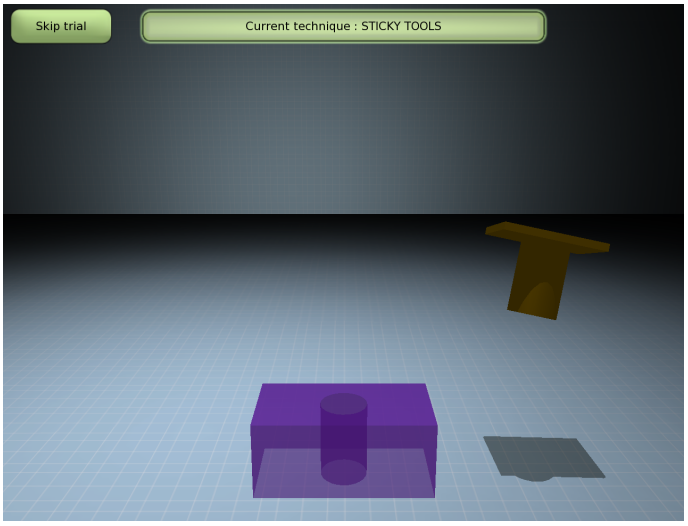


FIGURE 3.13: Tâche d'insertion d'un cylindre dans un trou.

pour sélectionner des modes d'interaction. Les logiciels de dessin sont des exemples typiques d'interfaces qui offrent un nombre important de modes. Comparé à l'utilisation de boutons, les palettes d'outils offrent l'avantage de sélectionner sans ambiguïté un mode d'interaction et de fournir un retour visuel pour connaître le mode d'interaction en cours d'utilisation. Ces palettes nécessitent cependant de nombreux aller-retours entre le lieu d'interaction et la palette, en particulier quand les changements de modes sont fréquents. D'une manière générale, basculer de façon élégante entre différents modes a toujours été un défi depuis les premiers systèmes interactifs. Les lecteurs se rappelleront certainement de la démonstration de Sketchpad [7] en 1964 où l'utilisateur était constamment en train d'appuyer sur des boutons physiques.

Nous avons vu dans la section précédente que l'interaction multipoints peut offrir différents modes d'interaction en fonction du nombre de doigts en contact avec la surface. Cette façon de désigner des modes est cependant limitée à certains types de tâches (e.g. manipulation 3D) ou alors requiert l'utilisation de la seconde main. Les stylos offrent également la possibilité de définir des modes d'interaction avec un vocabulaire plus riche que la souris : utilisation de boutons comme la souris, mais également de la pression [RBB04], de l'orientation [TXW+08] ou la forme de la saisie [SBG+11]. Ces méthodes de spécification de modes peuvent être ambiguës et sujettes à erreurs. Un moyen simple d'ajouter un second mode est d'utiliser la seconde extrémité du stylo, comme dans la réalité pour les crayons qui possèdent une gomme. L'utilisation de différents points de contact permet alors aux utilisateurs d'obtenir un choix explicite du mode d'interaction utilisé. Nos travaux étendent cette idée de changer de modes en se basant sur les points de contact, ou plus généralement les caractéristiques géométriques en contact. Nous nous sommes également inspirés de la façon dont les artistes utilisent des pastels Conté, qui se présentent sous la forme de parallélépipèdes rectangles. Les facettes de ce

7. http://www.youtube.com/watch?v=mOZqRJzE8xg

crayon permettent de créer une diversité de traces suivant la partie en contact avec une feuille de papier (Figure 3.14). Une version digitale de ce crayon permettrait de détecter également quel sommet, arête ou face est en contact avec une surface. De ce point de vue, Conté s'inspire de ToolStone [RS00] mais avec plusieurs différences notables : tout d'abord ToolStone est un cube avec des dimensions bien plus importantes, ensuite seules les faces sont utilisées, enfin ToolStone est conçu pour être utilisé avec la main non-dominante en utilisation indirecte avec une tablette. Conté, quant à lui, peut fonctionner avec une seule main tout en permettant d'utiliser les doigts pour effectuer des actions complémentaires. De façon similaire à Manual Deskterity [HYP+10], il est également possible de saisir Conté dans la paume de la main pour combiner des actions avec les doigts.

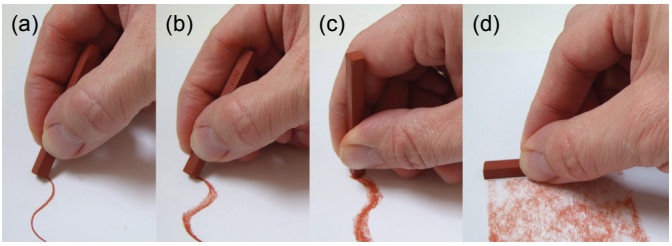


FIGURE 3.14: Différentes formes de lignes dessinées en utilisant un crayon Conté avec : (a) un coin ; (b) un côté ; (c) une extrémité ; et (d) une face.

Conté est inspiré de son homonyme pour les artistes. Nous avons cependant modifié ses dimensions suite à une série de tests préliminaires. Par exemple, la base de la forme originale est un carré que nous avons transformé en rectangle pour permettre aux utilisateurs de facilement discriminer des arêtes de dimensions différentes. Le résultat est un facteur de forme simple mais qui permet de passer rapidement entre différents modes tout en fournissant des paramètres supplémentaires comme l'angle de rotation du crayon lui-même ou l'angle qu'il forme avec l'horizontale ou encore la pression. Le facteur de forme supporte potentiellement 26 contacts différents, classés en 7 types : 8 sommets, 4 côtés courts, 4 côtés moyens, 4 côtés longs, 2 faces aux extrémités, 2 faces larges, et 2 faces étroites (Figure 3.15).

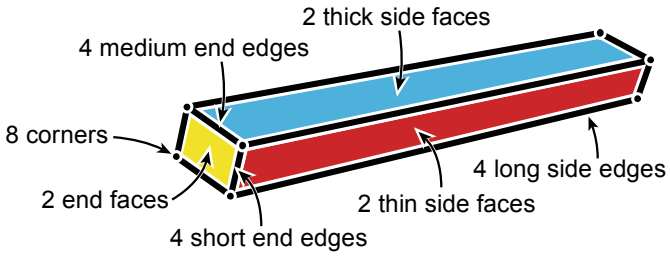


FIGURE 3.15: Les 7 types de contacts.

Pour chaque type de contact, la combinaison de la prise en main et de la caractéristique géométrique en contact détermine la disponibilité de paramètres d'entrée additionnels et des possibilités de manipulation différentes (Figure 3.16).

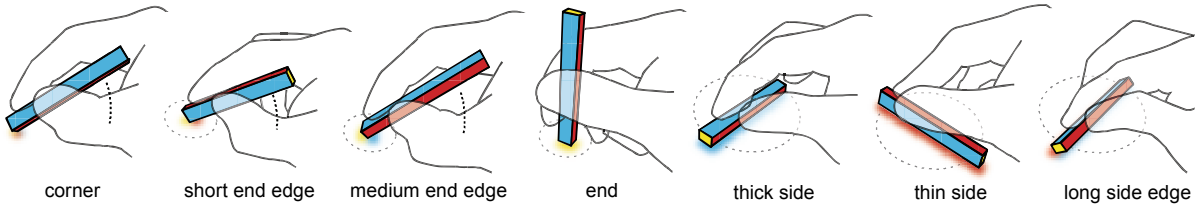


FIGURE 3.16: Prises en main typiques utilisées pour différents contacts.

L'espace de conception fourni par Conté regroupe l'ensemble des espaces créés par les tablettes commerciales avec crayons, de ToolStone [RS00] et Manual Deskterity [HYP+10]. A la base, Conté peut fonctionner comme un stylo, mais il utilise aussi les affordances et les caractéristiques des différents contacts pour changer de mode (e.g. un coin pour dessiner et une extrémité pour mettre en évidence). L'utilisation des doigts peut être aussi combinée ou coordonnée avec Conté. Par exemple, Conté peut être utilisé comme ToolStone par la main non-dominante pour définir un mode pour la main dominante. De façon sans doute plus intéressante, la même main peut à la fois tenir Conté dans la paume pour définir un mode et régler finement ce mode avec les doigts ainsi libérés. Conté peut aussi autoriser le concept "pen + touch = new tools" de Manual Deskterity. En plus de la prise en compte des objets touchés par les doigts pour interpréter le mode d'utilisation du crayon, Conté peut être utilisé pour définir un mode pour les doigts, compte tenu du nombre important de modes qu'il supporte. Nous définissons ci-dessous quelques unes des caractéristiques qui déterminent l'espace de conception de Conté :

- Association de modes : l'association d'un mode à une caractéristique géométrique doit prendre en considération à la fois l'affordance physique et la précision d'interaction avec cette caractéristique géométrique (e.g. un côté court pour surligner, un côté long utilisé comme règle, une extrémité pour estampiller).
- Groupes de modes : garder les points de contact des modes associés à des tâches connexes proches les uns des autres (e.g. création et édition à chaque extrémité).
- Utilisation de "phrases d'adjacence" : des transitions rapides entre des caractéristiques géométriques proches peuvent permettre d'activer des modes spéciaux (e.g. une rotation d'une arête à un sommet).
- Paramètres pour la manipulation directe : les positions X-Y et les angles peuvent être utilisés pour la manipulation directe (e.g. positionner et orienter une ligne de référence), ajuster un mode (e.g. choisir l'option par défaut), ou alors permettre plusieurs modes par face comme ToolStone.
- Paramètres à l'appel d'un mode : la position X-Y et l'angle peuvent servir à régler précisément des modes (e.g. l'objet en contact peut être utilisé pour créer le contexte et l'angle de contact utilisé pour choisir un sous mode).
- Maintien de contact : pour les positions stables du crayon, Conté peut être utilisé

pour maintenir un mode tout en libérant la main pour interagir. Faire apparaître des widgets à proximité du crayon lors de l'appel d'un mode peut permettre d'exploiter cette propriété.
– Utilisation du toucher et de la saisie dans la paume de la main : en relâchant la saisie de Conté, le pouce ou un autre doigt peuvent manipuler des objets à proximité.

Créer un prototype de Conté a été un défi pour plusieurs raisons. Tout d'abord, il était essentiel que le périphérique soit petit, ce qui écarte un certain nombre de solutions. Ensuite, nous voulions qu'il fonctionne sur une table multipoints non modifiée. Enfin, il devait rester suffisamment simple pour permettre aux chercheurs et amateurs de créer leur propre périphérique.

De nombreux prototypes ont permis d'aboutir à une version simple et fonctionnelle (Figure 3.17). Conté émet de la lumière infra-rouge qui est détectée par les caméras de surfaces multipoints et convertie par traitement d'images en événements Conté. Emettre de la lumière infra-rouge nécessite peu d'électronique et n'importe quelle surface multipoints utilisant l'illumination diffuse infra-rouge, comme la Surface 1 de Microsoft, peut détecter les motifs infra-rouges. Les détails de réalisation du prototype sont décrits dans l'article correspondant [**VC11**]. Cette simplicité réclame plusieurs concessions : nous ne sommes capables de détecter et identifier que 10 des 26 contacts possibles et nous ne pouvons pas mesurer l'angle entre le crayon et la surface. Nous détectons le contact d'un sommet mais nous ne sommes pas capable d'identifier lequel ; de même pour les côtés courts, moyens et longs.

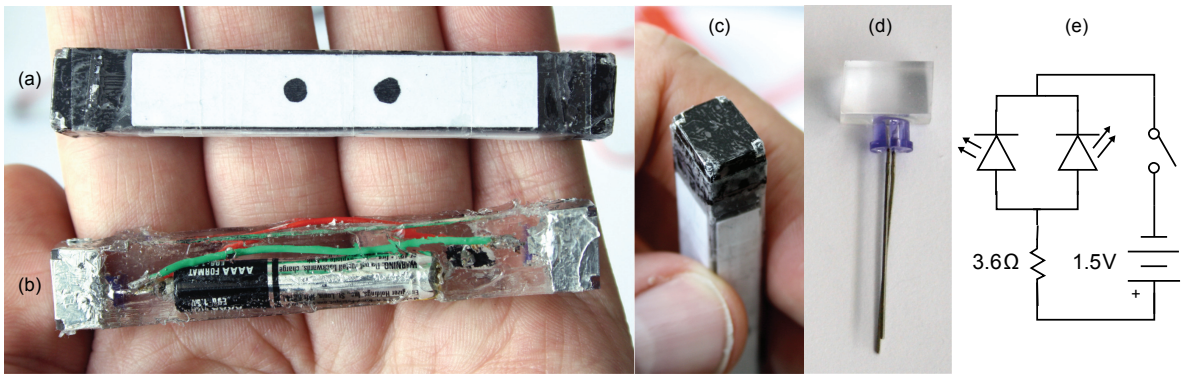


FIGURE 3.17: Prototypes infra-rouge de Conté. (a) périphérique complet ; (b) après inclusion, montrant la batterie et le circuit électronique ; (c) extrémité montrant les "fenêtres" infra-rouges ; (d) LED infra-rouge incluse dans un bloc d'acrylique ; (e) schéma du circuit électronique.

Les grandes lignes du traitement d'images sont illustrées figure 3.18 et détaillées dans [**VC11**].

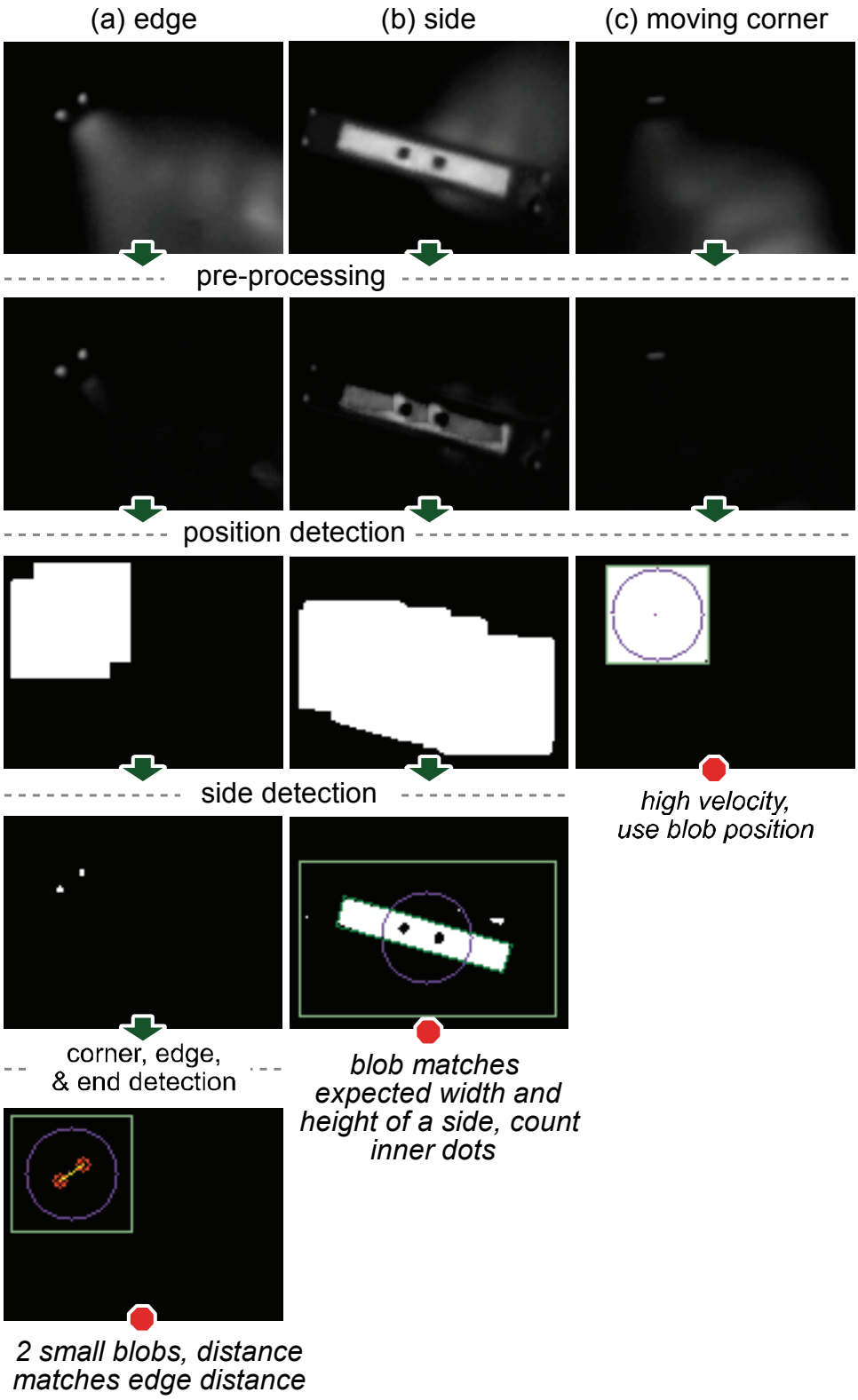


FIGURE 3.18: Traitement d'image : (a) arête ; (b) face ; (c) sommet en cours de déplacement. Chaque ligne montre une étape de traitement : préparation des données ; détermination de la position ; détection des caractéristiques géométriques ; détection des sommets, arêtes et faces.

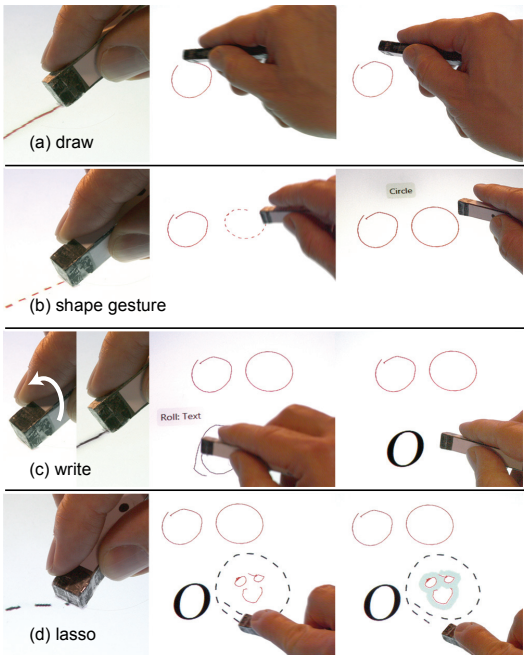


FIGURE 3.19: Utilisation des sommets et des arêtes de Conté avec une seule main. (a) dessin et annotation avec un sommet ; (b) réalisation de formes à reconnaître avec une arête courte ; (c) rotation d'un côté court à un sommet pour changer le mode du sommet en reconnaissance de texte ; (d) utilisation d'une arête moyenne pour la sélection.

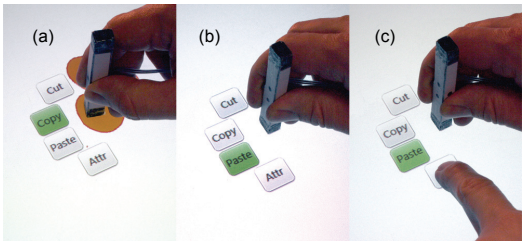


FIGURE 3.20: Menu contextuel en utilisant une extrémité : (a) comme le côté large est aligné suivant la verticale, le menu apparaît avec l'item "copier" sélectionné par défaut ; (b) "coller" est sélectionné par défaut quand le côté large est aligné suivant l'horizontale ; (c) le doigt sélectionne une autre commande qui devient la nouvelle commande par défaut pour l'orientation courante du côté large.

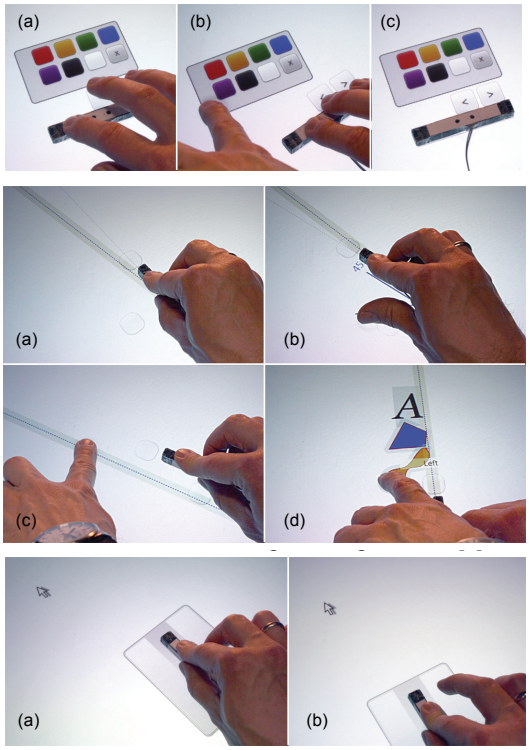


FIGURE 3.21: Haut : (a) utilisation d'une main pour passer en revue les palettes ; (b) détacher une palette pour la garder visible ; (c) Conté posé sur une face stable qui permet de conserver le mode palette quand il est relâché. Deuxième exemple en partant du haut : (a) faire apparaître et positionner un guide avec Conté ; (b) ajuster l'angle ; (c) détacher la ligne de guidage pour la faire apparaître en permanence ; (d) alignement des objets à gauche en utilisant un geste vers la gauche. Dernier exemple : utilisation de Conté comme une souris : (a) pointage ; (b) sélection.

Enfin, nous avons illustré l'utilisation des modes de Conté dans une application de dessin. Ainsi, suivant la caractéristique géométrique en contact avec la surface, le geste peut être interprété comme une trace dessinée, une forme à reconnaître, de l'écriture ou une sélection d'objets (Figure 3.19). Conté a également été utilisé pour faire apparaître des menus contextuels (Figure 3.20), des palettes d'attributs, des guides d'alignements et un mode souris (Figure 3.21).

En partant de la simple idée de reconnaître différents contacts avec un crayon parallélépipédique, nous avons été amenés à relever des défis matériels et logiciels. Nous avons illustré quelques techniques d'interaction qui démontrent les points clés de l'espace de conception de Conté. Ces techniques illustrent le potentiel de Conté. Les historiens de l'art ont argumenté que l'invention d'un crayon aussi polyvalent que Conté a influencé un nouveau style d'art [Sch41]. Au risque de paraître trop enthousiaste, peut-être qu'une version digitale peut avoir des effets similaires en interaction homme-machine.

3.2 Interaction gestuelle

Dans cette section, nous détaillons les travaux réalisés en interaction gestuelle, en partant de l'évaluation de la difficulté perçue d'exécution de gestes ; puis en présentant une technique d'interprétation de gestes pour la sélection et la saisie d'objets et enfin nous présentons l'utilisation de gestes dans un contexte de modélisation 3D.

3.2.1 Evaluation de la difficulté perçue d'exécution de gestes

L'exécution de gestes de commandes est un moyen simple d'éviter de recourir à des menus ou des boutons physiques ou virtuels. L'exécution d'un geste permet de définir à la fois l'objet et l'action qu'il va subir, quand l'utilisation de menus nécessite dans un premier temps de sélectionner l'objet d'intérêt et dans un second temps d'atteindre et de naviguer dans un menu pour sélectionner la commande désirée. Les gestes sont particulièrement adaptés aux interfaces tactiles pour lesquelles la navigation dans un menu demande un déplacement physique qui peut être important puis une navigation qui peut être difficile. D'une manière générale, trois critères principaux déterminent le succès d'une interface basée sur des gestes : la technologie d'acquisition, les algorithmes de reconnaissance de gestes et enfin l'ensemble de gestes. Les technologies pour acquérir les gestes [GFB04, KR10, MWW10, RB10, WMW09] et les algorithmes de reconnaissance [KZ04, Li10, WWL07] sont maintenant robustes et facilement disponibles. Cependant, développer des techniques et des critères pour aider les concepteurs à créer un ensemble de gestes intuitifs et simples à réaliser reste une aire de recherche active. Pour qu'un geste soit intégré avec succès dans une application, il doit remplir de multiples critères : il doit être reconnu sans ambiguïté [AS10, LLR02, LLR99] ; bien correspondre à la fonction à laquelle il est associé [MWW10, NSMG03, WMW09] ; être facile à mémoriser [NSMG03] ; et être facile à réaliser [MWW10, WMW09, NSMG03].

Les chercheurs ont offert deux stratégies pour aider les concepteurs. La première est d'utiliser des modèles prédictifs pour analyser les gestes candidats. Ceux-ci ont été uti-

lisés avec succès pour évaluer l'ambiguïté de reconnaissance [AS10, LLR99] et ont fait des progrès pour prédire le temps d'exécution [CZ07, Iso01]. Malheureusement, créer des modèles prédictifs pour des critères comme le temps d'exécution est difficile à cause de la complexité et de la variabilité des gestes humains ; facteurs qui sont aussi influencés par les capacités cognitives et motrices individuelles, ainsi que le contexte culturel. Pour ces raisons, les chercheurs ont proposé une seconde stratégie qui repose sur l'usage d'études utilisateurs formelles pour la conception participative et l'évaluation d'ensembles de gestes [AS10, MWW10, NSMG03, RB10, WMW09]. Impliquer des utilisateurs dans tout processus de conception est une bonne idée mais l'énergie nécessaire pour concevoir, réaliser et analyser ce type d'expériences est importante comparée à l'utilisation de modèles prédictifs.

Nous proposons une solution pratique entre un modèle et une étude utilisateurs [**VVCG11**]. La notion de difficulté recouvre de multiples critères qui incluent la facilité avec laquelle un geste peut être appris et exécuté. La notion de difficulté a été mentionnée dans des travaux antérieurs [MWW10, NSMG03, WMW09], mais il n'y a eu jusqu'alors aucune tentative pour l'évaluer en détails ou l'estimer. Pour cela, nous avons réalisé une première expérience où nous avons demandé aux participants d'exécuter les gestes représentés sur la figure 3.22 puis de noter le niveau de difficulté absolu de chaque geste sur une échelle de 1 à 5 (1 étant très facile et 5 très difficile à exécuter) et enfin de classer tous les gestes par ordre croissant de difficulté relative (Figure 3.23). Les résultats montrent un niveau de cohérence important entre les participants sur l'évaluation du niveau de difficulté d'exécution. Compte tenu de cette cohérence, nous pouvons chercher à estimer cette difficulté en l'absence d'expérience. En substance, s'il existe une corrélation avec un ou plusieurs descripteurs alors ces descripteurs peuvent être utilisés pour estimer la difficulté d'exécution d'un geste.

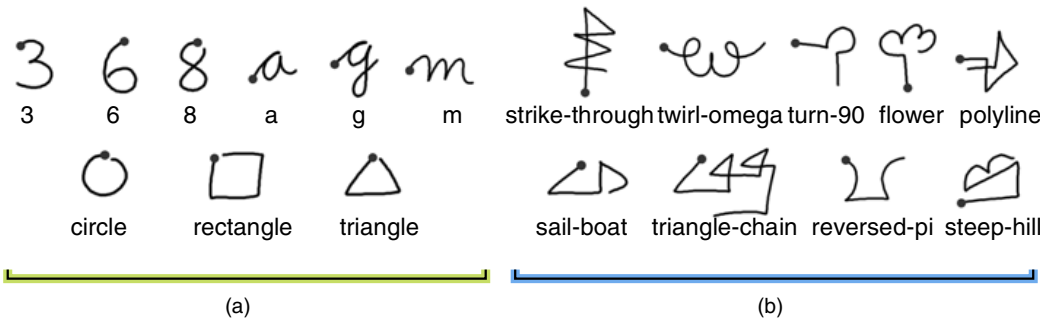


FIGURE 3.22: Les 18 gestes utilisés dans la première expérience : (a) 9 gestes considérés comme familiers ; (b) 9 gestes considérés comme non-familiers.

Nous avons évalué de nombreux descripteurs : les 13 descripteurs de Rubine [Rub91], les 7 descripteurs additionnels évalués par Long et al. [LLRM00], les 7 invariants spatiaux de Hu habituellement utilisés en traitement d'image pour analyser les contours et les formes [Pra01] (p. 606), la mesure de complexité Isokoski [Iso01] et le temps d'exécution

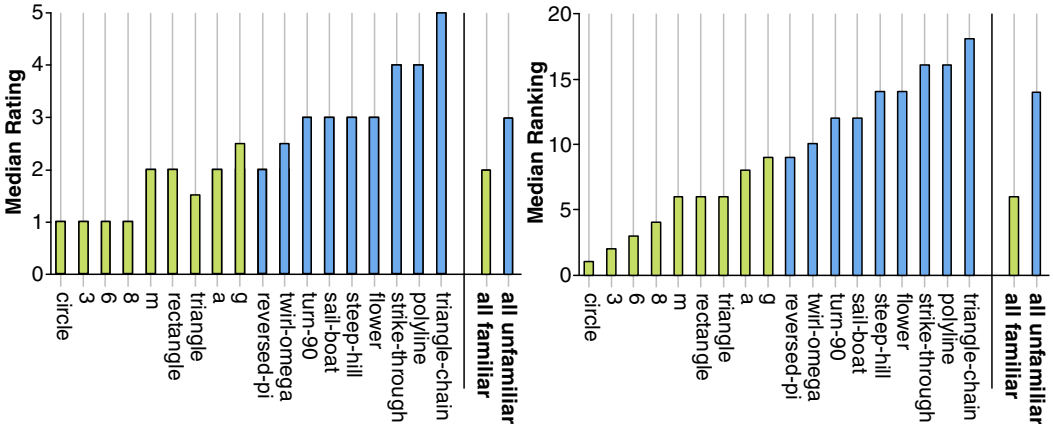


FIGURE 3.23: Gauche : valeur médiane de difficulté (les scores plus importants correspondent à des gestes jugés plus difficiles à exécuter). Droite : classification médiane des gestes (une valeur plus importante indique un geste plus difficile à exécuter)

prédit par le modèle de Cao et Zhai [CZ07]. Les résultats montrent que le temps d'exécution permet d'obtenir la meilleure corrélation. Nous avons également observé que le modèle CLC [CZ07] ne permettait pas de prédire correctement les temps d'exécution.

Nous avons réalisé une seconde expérience pour valider les résultats de la première, avec un nouvel ensemble de gestes. Nos résultats montrent que l'estimation de la difficulté relative peut être prédite avec plus de 93% de précision en utilisant une mesure du temps d'exécution et 87% en utilisant le modèle de production du temps du premier ordre d'Isokoski [Iso01]. En utilisant un classificateur Baysien et le temps d'exécution, nous pouvons mesurer la difficulté absolue avec 83% de réussite. Puisque les temps prédits par le modèle CLC [CZ07] réduisent la précision de notre classification à 25%, nous avons proposé une approche alternative. Nous avons trouvé que le temps d'exécution peut être raisonnablement estimé en utilisant plusieurs échantillons de temps. Avec 3 utilisateurs fournissant 3 gestes exemples, notre règle de classification permet d'obtenir 75% de précision en moyenne.

Réduire la difficulté d'exécution de gestes est un des objectifs lors de la conception d'un ensemble de gestes. Nos travaux montrent que les utilisateurs ont une perception similaire de la difficulté d'exécution et qu'elle est fortement corrélée avec le temps d'exécution. Par ailleurs, cette difficulté peut être estimée en utilisant deux règles simples pour les classifications relative et absolue. Parce que les modèles existants ne permettent pas de prédire le temps d'exécution nécessaire pour nos règles de classification, nous montrons qu'une estimation du temps d'exécution peut être obtenue en utilisant seulement quelques échantillons de quelques utilisateurs. De plus cet ensemble peut déjà exister quand les concepteurs entraînent le système de reconnaissance.

Pour l'exécution de gestes de commandes, seule la forme géométrique du geste est prise en considération. La segmentation du geste est par ailleurs réalisée, soit à l'aide d'un

bouton ou alors en effectuant un contact avec une surface. Nous allons voir dans le travail suivant que d'autres paramètres du geste peuvent être pris en considération, comme la vitesse ou l'accélération, afin d'interpréter l'action de l'utilisateur, comme ici dans des tâches de sélection et saisie d'objets 3D.

3.2.2 Buttonless clicking : interprétation du geste

Les interfaces sans contact, comme le Microsoft Kinect, permettent de suivre les mains et d'autres parties du corps. Elles ne disposent cependant pas de boutons et ne sont pas suffisamment précises pour suivre les doigts des utilisateurs. Ainsi il n'est pas possible d'utiliser le geste de pincer le pouce et l'index pour, par exemple, saisir un objet et le déplacer dans l'espace. D'autres modalités peuvent être utilisées pour déclencher des événements discrets comme la réalisation de gestes particuliers ou la reconnaissance vocale. Les utilisateurs sont cependant peu enclins à accomplir les différentes étapes de calibration ou d'entraînement nécessaires pour la reconnaissance gestuelle et vocale. Ces méthodes sont également sujettes à erreurs et prennent du temps à réaliser.

Nous avons proposé une nouvelle approche basée sur une analyse de la trajectoire et de la dynamique du geste qui permet de discriminer la sélection, la saisie et le dépôt d'objets en 3D [**CCG10**]. La technique que nous proposons ne nécessite aucun entraînement ou calibration et peut être utilisée avec n'importe quel périphérique 3D.

Des projets comme *dontclick*[8] permettent d'expérimenter différentes techniques pour se passer des boutons de la souris : technique de franchissement de cible qui est rapide mais peut introduire des faux positifs ; réalisation d'un geste particulier (e.g. queue de cochon) sur la cible qui réduit les faux positifs mais prend du temps ; utilisation d'une temporisation qui peut entraîner des faux positifs et prend également du temps. Shapewriter est un autre exemple d'interface qui introduit un moyen efficace de déterminer les caractères à sélectionner en se basant sur une analyse de la trajectoire [KZ07]. Tous ces exemples se focalisent sur la sélection et ne prennent pas en compte le glisser-déposer.

Nous avons réalisé une première expérience avec deux personnes en leur demandant de réaliser des tâches de sélection et de saisie et déposé d'objets 3D en utilisant un bouton. Nous avons ensuite déterminé les profils d'accélération et de vitesse et nous avons remarqué un profil particulier pour la saisie d'objets. Pour cette tâche, l'accélération du périphérique lors de l'entrée dans la cible décroît jusqu'à atteindre un minimum puis augmente à nouveau avant que l'utilisateur presse le bouton. Ce profil peut s'expliquer par une anticipation de la trajectoire dans la direction où l'utilisation prévoit de déplacer l'objet. Cela correspond à un phénomène de co-articulation du geste également observé dans l'écriture manuscrite. Pour la sélection, le profil le plus caractéristique était celui de la vitesse. La vitesse diminuait dès l'entrée du pointeur dans la cible et jusqu'à ce que le bouton soit pressé.

A partir de ces observations, nous avons proposé un algorithme qui permet de détecter ces caractéristiques (Figure 3.24). Quand le pointeur entre dans une cible, les profils de

8. `http://www.dontclick.it`

trajectoire, vitesse et accélération sont analysés. La saisie est déterminée d'abord par la détection d'un minimum local dans le profil d'accélération puis une augmentation de la distance entre le pointeur et le centre de la cible. Pour la sélection, nous examinons si la vitesse du périphérique reste en dessous d'un seuil pendant un certain temps. Le dépôt d'objets intervient quand la vitesse du périphérique passe en dessous d'un autre seuil de vitesse.

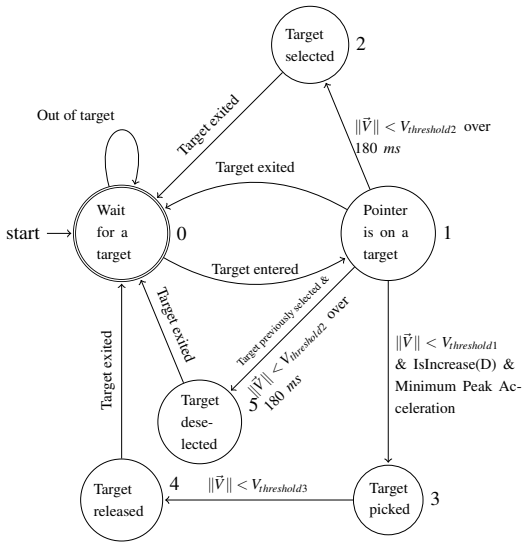


FIGURE 3.24: Diagramme d'états-transitions représentant l'algorithme pour prédire sélection, dé-sélection, saisie et lâcher. D est la distance entre la position du pointeur et le centre de la cible.

Nous avons réalisé une expérience contrôlée pour évaluer la performance de notre algorithme. La tâche consistait à sélectionner ou saisir et déposer des cibles. Les résultats montrent un taux de succès global de 90% à travers toutes les conditions pour notre algorithme, plus faible que le taux de 96% obtenu avec un bouton. Nous avons été heureux de constater que notre algorithme ne demande quasiment aucun entraînement et qu'il est robuste pour l'ensemble des participants, avec les paramètres définis lors de l'étude initiale. Les résultats expérimentaux montrent que l'orientation n'a aucune influence sur les résultats mais que la taille de la cible affecte les résultats avec une dégradation des performances pour les plus petites cibles, en particulier pour la tâche de sélection.

Ces résultats montrent que notre approche représente une alternative crédible à l'utilisation de boutons pour la sélection et la saisie-dépôt d'objets, pour des cibles suffisamment larges. La prise en compte de la taille des cibles permettrait sans doute d'améliorer les résultats.

3.2.3 Mockup Builder : combiner l'interaction sur et au-dessus de la surface

Une surface tactile et le volume qui se trouve au dessus forment deux espaces qui présentent chacun des qualités spécifiques pour l'interaction. Chaque espace est ainsi mieux

adapté à certaines tâches que d'autres. Ecrire ou dessiner une forme en deux dimensions sera toujours plus efficace en utilisant une surface comme support alors que le déplacement d'un objet en trois dimensions reste plus facile en espace libre. La majorité des recherches se limitent à l'utilisation de l'un des deux espaces. Nous proposons d'explorer la combinaison des deux espaces dans un scénario de modélisation 3D [**DACJ12**].

Müller-Tomfelde et al. ont proposé différentes méthodes pour exploiter l'espace au dessus de la surface de façon à interagir avec le contenu d'une façon plus proche de la réalité [MTHB+10]. Ainsi la position de la main au dessus de la surface peut être utilisée pour positionner des objets 3D. Wilson et al. ont par ailleurs proposé des techniques pour interagir avec différents écrans en exploitant la posture des utilisateurs dans l'espace [WB10]. Les utilisateurs peuvent interagir avec un contenu 2D sur ou au dessus de la surface ou également entre des surfaces. Le corps est alors utilisé pour transférer du contenu d'une surface à une autre. Nos travaux explorent le *continuous interaction space* introduit par Marquardt et al. [MJGJ11] où l'idée est d'utiliser les deux espaces d'interaction conjointement.

Nous proposons une approche de modélisation directe pour créer, éditer et manipuler des modèles 3D en utilisant un petit ensemble d'opérations. Notre système est constitué d'un environnement semi-immersif basé sur un écran multipoints stéréoscopique, combiné à une caméra de profondeur Kinect et deux Gametrak [9] qui sont utilisés pour identifier et suivre le pouce et l'index de chaque main au dessus de la surface (Figure 3.25). Nous avons ajouté un bouton sur l'index, qui est utilisé pour identifier le geste de saisie sans aucune ambiguïté.

La surface multipoints est utilisée principalement pour réaliser des esquisses. En utilisant la main dominante, les utilisateurs peuvent créer des formes planaires en utilisant des lignes et des courbes tracées avec un ou plusieurs traits. Les traits dont les extrémités sont proches les unes des autres sont fusionnés en un seul trait et les formes dessinées sont modélisées par des segments et des courbes. Un geste de gribouillage permet d'effacer un objet. Un geste démarré avec la main non-dominante est interprété soit comme une transformation d'objet s'il est effectué sur celui-ci, soit comme une transformation du monde. Les translations sont opérées avec un doigt et les combinaisons de translations, rotations et changement d'échelles sont effectuées avec plusieurs doigts. Un geste démarré avec la main non-dominante peut être complété par la main dominante. De plus la main non dominante peut être utilisée pour définir des contraintes pour la main dominante. Par exemple, un utilisateur peut esquisser une ligne qui va définir un plan de symétrie. Il va alors commencer par sélectionner la ligne avec la main non-dominante puis tracer une esquisse avec la main dominante. Cette utilisation des deux mains est inspirée du modèle asymétrique de Guiard [Gui87], où la main dominante réalise des mouvements fins et manipule des outils et la main non-dominante sert d'espace de référence et permet de réaliser des mouvements grossiers.

9. `http://en.wikipedia.org/wiki/Gametrak`

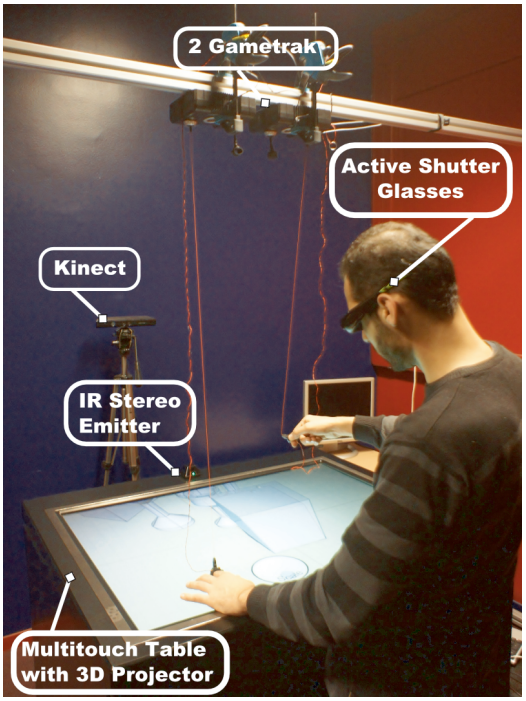


FIGURE 3.25: Vue globale de la configuration matérielle.

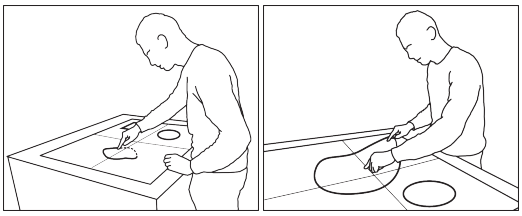


FIGURE 3.26: Interaction bi-manuelle sur la surface : réalisation d'une esquisse avec la main dominante (gauche) et re-dimensionnement avec deux mains en utilisant la main non-dominante (droite).

Les gestes effectués au dessus de la surface sont interprétés comme création ou édition d'objets 3D. La création consiste à extruder des formes planaires précédemment esquissées sur la surface. L'utilisateur sélectionne la forme avec son index de la main dominante puis réalise un geste de pince pour extruder la forme selon la normale à la surface (Figure 3.27). La hauteur est alors mise à jour en continu et co-localisée avec la position du doigt jusqu'à ce que le bouton soit relâché. Les formes peuvent être aussi extrudées selon une trajectoire définie dans l'espace après que l'utilisateur a sélectionné cette opération dans un menu affiché sous la main non-dominante (Figure 3.27). Quand l'utilisateur définit la trajectoire, le chemin est continuellement ré-évalué et ajusté en utilisant des segments et des courbes pour créer des formes régulières. Sans cette opération, les formes créées seraient inexploitables.

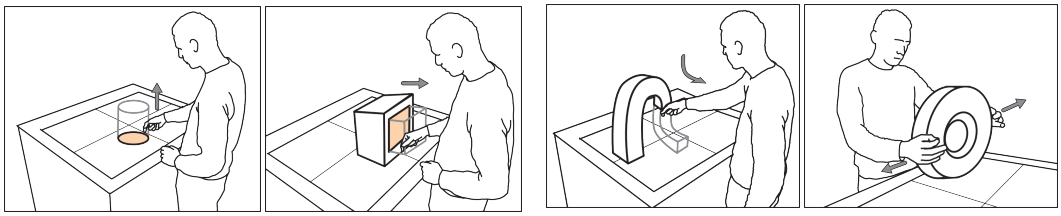


FIGURE 3.27: Extrusion d'une face suivant sa direction normale ; extrusion suivant une trajectoire ; mise à l'échelle en utilisant deux mains.

L'édition suit la métaphore du pousser/tirer où les caractéristiques topologiques d'un objet (sommets, arêtes et faces) sont déplacées dans l'espace suivant la direction normale à la face à laquelle ils appartiennent. L'utilisateur active d'abord la caractéristique géométrique en plaçant l'index de la main dominante à proximité et le sélectionne avec un geste de pince. La position de la caractéristique géométrique est ensuite mise à jour selon la position de l'index jusqu'à ce que le geste de pince soit relâché. Les faces peuvent aussi être extrudées suivant une trajectoire définie par l'utilisateur une fois que l'opération correspondante a été sélectionnée dans un menu affiché sous la main non-dominante. Si aucune caractéristique géométrique n'est sélectionnée lors du geste de pince, les utilisateurs peuvent réaliser des esquisses en 3D.

L'interaction bi-manuelle utilisée sur la surface est également possible au dessus de la surface, permettant de faire tourner, déplacer et mettre à l'échelle en utilisant les deux mains. Comme sur la surface, la main non-dominante commence l'interaction en utilisant un geste de pince. La main non-dominante définit les translations alors que la main dominante ajoute les rotations et changements d'échelle, en utilisant une méthode proposée par Wang et al. [WPP11]. Ces interactions directes apparaissent bien plus efficaces comparées aux interactions indirectes en utilisant uniquement une surface multipoints (e.g. changer la profondeur d'un objet tout en le déplaçant dans le plan de l'écran).

Les techniques présentées précédemment utilisent des opérations manuelles asymétriques pour implicitement transiter entre la réalisation d'esquisses, la transformation d'objets et la manipulation du monde. La main non-dominante peut être aussi utilisée pour compléter des opérations réalisées avec la main dominante. Tout d'abord, la main non-dominante peut-être utilisée pour sélectionner le mode utilisé par la main dominante. Les modes sont présentés dans un menu contextuel positionné sous la main non-dominante (Figure 3.28). Les modes présentés dans le menu contextuel correspondent à ceux disponibles par rapport à l'action en train d'être effectuée par la main dominante. La transparence du menu est ajustée en fonction de la distance de la main à la surface : au dessus de 15 cm, le menu est totalement transparent et devient progressivement opaque quand la main non-dominante s'approche de la surface. Pour améliorer l'accessibilité, le menu contextuel suit la main non-dominante mais sa position est progressivement figée quand elle approche de la surface, afin de réduire les instabilités spatiales et les erreurs lors de la sélection d'un item. Cela est simplement réalisé grâce au 1€ filter pour lequel la fréquence de coupure est ajustée en fonction de la distance.

Les choix de modes incluent la sélection du type d'extrusion (normal à une face ou selon une trajectoire), l'opération de clonage et l'opération de basculement de vue. Le basculement est disponible quand une face est sélectionnée. L'opération consiste à faire une rotation du monde pour aligner la face avec la surface. La main non dominante peut être aussi utilisée pour sélectionner un élément géométrique qui sert de contrainte à la main dominante. Nous utilisons les contraintes de plans et d'arêtes pour les opérations d'extrusion et de positionnement. Par exemple, la main non-dominante peut sélectionner une face d'un objet pour définir la hauteur maximale ou minimale d'un objet en train d'être extrudé avec la main dominante (Figure 3.28). Lors de la translation, une contrainte planaire définit la limite au delà de laquelle un objet ne peut plus être déplacé. Le profil d'un objet peut également être mis à l'échelle avec la main non-dominante tout en étant extrudé par la main dominante.

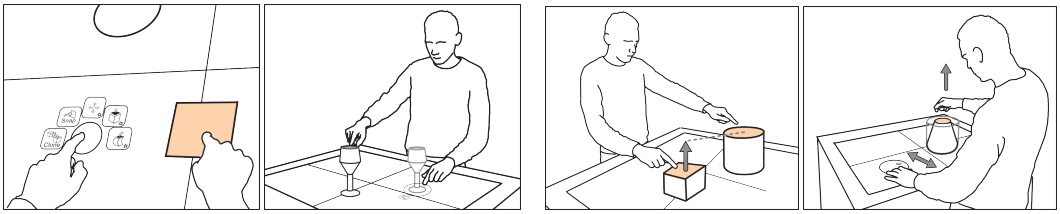


FIGURE 3.28: Exemple de menu présenté sous la main non-dominante ; opération de clonage ; définition d'une contrainte de hauteur ; mise à l'échelle avec la main non-dominante tout en extrudant avec la main dominante.

Nous avons comparé de façon informelle notre système avec deux experts : le premier maîtrisait parfaitement notre système et le second était un architecte expert de Rhino3D. Les temps de modélisation en utilisant chacun des systèmes étaient sensiblement équivalents, et cela pour différents types de modèles. Ces premiers résultats sont encourageants et donnent une première validation de notre approche.

3.3 Conclusion

Ce chapitre a synthétisé plusieurs contributions importantes pour mieux comprendre et enrichir l'interaction tactile et gestuelle. Partant de la problématique de l'occultation en interaction directe, nous avons mené des études pour mieux comprendre son influence, ce qui nous a conduit à proposer des modèles qui peuvent servir par la suite au développement d'interfaces qui adaptent la présentation de leur contenu à l'occultation.

Nous avons ensuite présenté le STIMTAC qui permet de produire un film d'air comprimé sous le doigt afin de modifier le frottement d'une surface. Nous avons présenté les résultats d'une étude montrant que le périphérique est en mesure de reproduire des textures de forme de base carrée avec la même acuité que dans la réalité. Nous avons ensuite exploré l'utilisation du retour tactile pour développer une technique facilitation du pointage sensible aux cibles, appelée Surfpad, qui permet d'améliorer les performances de pointage même en présence de distracteurs.

Nous nous sommes ensuite intéressés au problème de l'intégration et de la séparation des degrés de liberté sur surfaces tactiles multipoints pour les tâches de manipulation 3D. Nous avons vu que pour ces tâches, l'intégration des degrés de liberté dégrade les performances et nous avons montré que la stratégie de séparation des degrés de liberté de la tâche est importante pour optimiser les performances.

En considérant l'importance des modes et des outils dans l'interaction, nous avons également proposé un crayon multimodal qui permet de passer de manière élégante d'un mode à un autre tout en facilitant l'utilisation combinée des doigts.

Concernant l'interaction gestuelle, nous nous sommes intéressés à la perception de la difficulté d'exécution de gestes et nous avons montré qu'elle est perçue de manière homogène par les utilisateurs et qu'elle peut être facilement estimée par le temps d'exécution. Nous nous sommes par ailleurs pencher sur la discrimination entre la sélection et la saisie d'objets 3D sans utiliser de bouton et nous avons proposé un algorithme basé sur l'analyse de la dynamique et de la trajectoire du geste qui montre de très bons résultats.

En dernier lieu, nous avons présenté un système de modélisation qui combine interaction sur surfaces tactiles et interaction au dessus pour tirer le meilleur partie des deux espaces d'interaction.

Comme perspectives, nous envisageons de poursuivre nos travaux sur le retour tactile, afin de développer des icônes tactiles qui enrichiraient les interfaces graphiques. Cela nécessite la réalisation d'expériences de psychophysique que nous menons. Nous nous intéressons également au problèmes de latence sur les interfaces multipoints qui pénalisent la manipulation directe. Nous travaillons par ailleurs à l'exploitation d'informations supplémentaires, comme l'identification des doigts pour créer de nouvelles techniques d'interaction. Enfin, nous travaillons à l'ajout de contrôle indirect sur les écrans multipoints pour les situations où l'interaction directe est pénalisante.

Chapitre

4 Conclusions et perspectives

Aquoi ressembleront les prochaines générations d'interfaces homme-machines ? Il est difficile de répondre précisément à cette question. Nous pensons qu'elles émergeront de possibilités offertes par de nouveaux périphériques, adaptées à la réalisation de certaines tâches, dans des contextes d'usages particuliers. La souris a permis l'avènement des interfaces graphiques de bureau, les écrans multipoints ont révolutionné l'usage des smart phones et des tablettes numériques. Cependant, beaucoup de tâches restent trop fastidieuses à réaliser avec ces interfaces. J'aurais difficilement pu écrire ce mémoire sur une tablette multipoints. Il m'est en revanche plus facile de le lire sur une tablette. Comme le rappel justement un des axiomes de Bill Buxton *"Everything is best for something and worst for something else"* [1]. Chaque interface est adaptée à une gamme d'usages particulière. Tout comme la télévision n'a pas remplacé la radio et internet n'a pas remplacé la télévision, les tablettes ne remplaceront pas les ordinateurs portables et la souris informatique ne disparaîtra pas. Les interfaces de demain remplaceront ou compléteront certaines tâches aujourd'hui réalisées avec d'autres interfaces, dans des contextes d'usages donnés. Il existe un large champ de perspectives d'améliorations. Deux pistes sont à envisager : améliorer les interfaces existantes ou proposer de nouvelles interfaces. Les travaux que nous avons présentés dans ce document s'intéressent aux deux alternatives.

Nous avons vu que l'amélioration des interfaces existantes passe par une reconsidération de la gestion des périphériques d'interaction dans les systèmes actuels. Le chapitre 2 a mis en évidence l'importance des fonctions de transfert pour les périphériques d'interaction indirects et a montré que les systèmes actuels ne permettent pas d'exploiter les capacités motrices des utilisateurs puisqu'ils ne prennent pas en compte les caractéristiques physiques des périphériques (e.g. leur résolution). Cela conduit, par exemple, à des comportements différents du pointeur souris d'un système à un autre et d'une configuration matérielle à une autre, pour des gestes identiques effectués dans l'espace moteur. Nous avons mis en évidence qu'il existe une marge d'optimisation des fonctions de transfert et que l'exploitation des capacités des utilisateurs permet d'accomplir des tâches qui étaient impossibles à réaliser. Dans le troisième chapitre, nous avons mon-

1. `http://www.billbuxton.com/multitouchOverview.html`, 2007.

tré que la prise en compte de l'occultation sur les tables tactiles est un nouveau facteur d'amélioration des interfaces, tout comme une meilleure compréhension des capacités de séparation et d'intégration des degrés de liberté pour les tâches de manipulation 3D sur les écrans multipoints.

Le développement de nouvelles interfaces homme-machine est également au cœur de nos travaux. La compréhension des limites des périphériques existants nous a conduit à développer le périphérique RubberEdge présenté dans le chapitre 2. Dans le chapitre 3, nous avons vu que le retour tactile est un facteur important d'enrichissement des périphériques et nous avons montré comment développer de nouvelles techniques d'interaction permettant de faciliter les tâches existantes. Les développements du périphérique Conté et du système de modélisation Mockup Builder ont montré comment exploiter des capacités motrices humaines pour développer de nouvelles interfaces homme-machine et faciliter les tâches d'interaction.

Les possibilités de création de nouvelles interfaces homme-machine dépendent du développement de périphériques et de techniques d'interaction qui mettent en relation les capacités motrices et cognitives des utilisateurs avec les tâches, pour des contextes d'interaction déterminés. Les capacités des périphériques doivent permettre d'exploiter au maximum les capacités motrices des utilisateurs (Figure 4.1). La tâche détermine quelles sont les capacités des utilisateurs à exploiter et la technique d'interaction permet d'optimiser leur utilisation, en exploitant au maximum la région en intersection sur la figure 4.1. Le développement de nouvelles interfaces homme-machine doit permettre de déplacer le cercle représentant les capacités de la machine vers le cercle représentant celles de l'homme de façon en mieux en tirer partie.

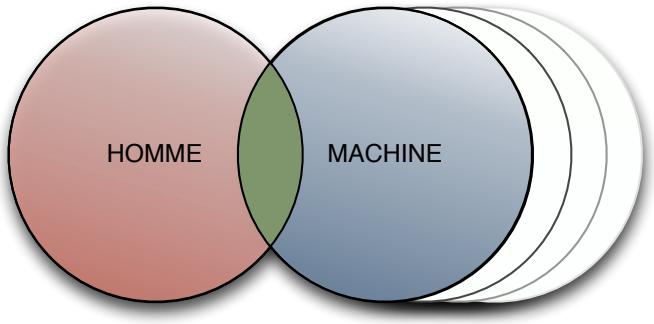


FIGURE 4.1: Ce schéma représente l'interaction entre les capacités de l'homme et celles de la machine, qui correspond aux capacités de la machine utilisables par l'homme. Les techniques d'interaction doivent permettre d'exploiter au maximum cet espace en intersection. Il est également possible de l'augmenter, en développant par exemple des périphériques permettant d'exploiter davantage les capacités humaines.

Nos futurs travaux porteront à la fois sur l'étude des capacités motrices humaines dans les systèmes interactifs et le développement de nouveaux périphériques et nouvelles techniques d'interaction. Notre objectif est de faire émerger de nouvelles interfaces homme-

machines qui s'imposeront demain comme se sont imposés la souris et les écrans multipoints. Cela passe également par le développement d'outils logiciels pour faciliter la capitalisation des connaissances acquises et la valorisation des travaux de recherche.

Notre approche passe par une considération systématique de toutes les étapes de traitement des informations d'un système interactif : de la numérisation d'une grandeur physique par le biais d'un capteur à son interprétation dans une application pour la mise à jour informations, en passant par toutes les étapes de traitements intermédiaires effectués par les périphériques et les systèmes informatiques. Alan Kay disait en 1982 *"People who are really serious about software should make their own hardware."* [2]. De façon similaire, développer ses propres périphériques d'interaction, comme nous l'avons fait pour le STIMTAC, oblige à résoudre de nombreux problèmes qui ont été résolus il y a 30 ans et qui ont été très peu, voire jamais, revisités depuis.

Les recherches dans le domaine de la physiologie de la posture et du mouvement peuvent nous donner des informations précieuses sur les capacités motrices humaines. Cependant, le fait de travailler dans le cadre de systèmes interactifs introduit une adaptation des gestes moteurs aux stimuli visuels. Ainsi, ce couplage perception-action doit être étudié à part entière dans le cadre les systèmes interactifs. Les fonctions de transfert sont un des facteurs qui modifient ce couplage perception-action. Nous avons mis en évidence des différences significatives entre les fonctions par défaut utilisées par les systèmes d'exploitation modernes. Une première perspective est de comprendre comment la forme de la courbe d'une fonction de transfert influence les performances, de manière à être capable de l'optimiser. Ceci s'applique dans le cadre des tâches de pointage mais également pour les tâches de défilement que nous avons commencées à étudier. Nous envisageons également d'étudier l'influence d'autres paramètres que la vitesse pour les fonctions de pointage. Par exemple, le débrayage pourrait être un facteur pris en compte par la fonction de transfert, comme il l'est dans certains cas pour le défilement.

Nous avons vu que la résolution est un facteur important à prendre en considération pour les fonctions de transfert, cependant peu de périphériques fournissent cette information. Nous envisageons de développer une méthode automatique de détermination de la résolution des périphérique de pointage. Nous envisageons également d'étudier en détails la résolution des membres humains en fonction du périphérique utilisé et proposer une méthode qui permettrait de la déterminer rapidement.

Les fonctions de transfert pour le glisser-déposer sur des cibles en dehors de l'écran fait également partie des perspectives. Cette tâche est en effet relativement fréquente : déplacement d'un objet dans une liste dont une partie n'est pas affichée, sélection de plusieurs cellules dans un tableur dont certaines ne sont pas affichées... Cette tâche implique le passage à un contrôle de vitesse quand le pointeur arrive aux bords de l'écran. Le contrôle de la vitesse est en général difficile à réaliser et mériterait une étude approfondie. Le contrôle en vitesse avec des périphériques isotonique est également présent dans la navigation en environnements 3D. De plus en plus de jeux 3D sont disponibles sur tablettes

2. Creative Think Seminar, le 20 juillet 1982.

et téléphones mobiles. Ces jeux utilisent une navigation 2D et demie en utilisant deux doigts pour contrôler les vitesses de déplacement du personnage et de rotation de la vue. Une nouvelle fois, le contrôle en vitesse avec un périphérique isotonique pose problème et demande une étude systématique des fonctions de transfert.

Au début du second chapitre, nous avons mis en évidence les problèmes de filtrage et de latence dans les systèmes interactifs. Les écrans tactiles présentent naturellement une latence qui est perceptible par les utilisateurs. L'influence de cette latence sur les performances utilisateurs est aujourd'hui inconnue et mérite d'être étudiée. Les causes de cette latence sont également peu connues et nous envisageons de proposer des techniques pour la compenser.

Nous allons poursuivre nos travaux sur les interfaces à retour tactile. Nous sommes entrain de réaliser des études psychophysiques qui nous aideront à mieux comprendre les relations entre les variables de contrôle du STIMTAC et les perceptions des utilisateurs. Nous espérons que ces études permettront de développer des icônes tactiles faciles à discriminer et identifier. Celles-ci permettraient d'enrichir les interfaces graphiques ou serviraient de mécanisme de notification passif. Nous travaillons par ailleurs sur d'autres facteurs de forme pour le STIMTAC, telles que des formes en anneaux pour la navigation dans des listes par exemple. Nous travaillons également à un contrôle du périphérique permettant de faire bouger le doigt de l'utilisateur.

Nous continuons nos travaux sur les écrans multipoints : nous cherchons à ajouter du contrôle indirect pour, par exemple, atteindre des cibles distantes ou interagir avec de petits objets. Compte tenu de la nature absolue indirecte de ce type de tâche, nous étudions les capacités de précision de positionnement d'un doigt sans retour visuel. Nous étudions par ailleurs d'autres façons de définir des modes sur ces interfaces. Pour cela, nous travaillons à l'identification des doigts en contact, ce qui permettrait ensuite d'envisager tout un panel de techniques d'interaction inédites.

Pour conclure, j'aimerais revenir sur une déclaration de Michel Serres à l'occasion des 40 ans de l'Inria : *"les nouvelles technologies nous ont condamnés à devenir intelligents"* [3], puisqu'elles permettent un accès immédiat à la connaissance et délèguent les travaux répétitifs aux machines. Je me permettrais de compléter que cette intelligence ne pourra s'exprimer qu'avec des interfaces homme-machines suffisamment riches pour débrider notre pouvoir créatif.

3. Les nouvelles technologies : révolution culturelle et cognitive. 11 décembre 2007. Lille. http://interstices.info/m-serres-lille

Sélection d'articles de recherche

Les articles les plus pertinents sont identifiés par une étoile (⋆).

A-1 Géry Casiez, Nicolas Roussel, and Daniel Vogel. 1€ filter: A simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 2527–2530, New York, NY, USA, 2012. ACM → Video

A-2 Géry Casiez and Nicolas Roussel. No more bricolage! methods and tools to characterize, replicate and compare pointing transfer functions. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 603–614, New York, NY, USA, 2011. ACM → Video ⋆

A-3 Philip Quinn, Andy Cockburn, Géry Casiez, Nicolas Roussel, and Carl Gutwin. Exposing and understanding scrolling transfer functions. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, New York, NY, USA, 2012. ACM

A-4 Géry Casiez, Daniel Vogel, Ravin Balakrishnan, and Andy Cockburn. The impact of control-display gain on user performance in pointing tasks. *Human-Computer Interaction*, 23(3):215–250, 2008 ⋆

A-5 Géry Casiez and Daniel Vogel. The effect of spring stiffness and control gain with an elastic rate control pointing device. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 1709–1718, New York, NY, USA, 2008. ACM ⋆

A-6 Géry Casiez, Daniel Vogel, Qing Pan, and Christophe Chaillou. RubberEdge: reducing clutching by combining position and rate control with elastic feedback. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 129–138, New York, NY, USA, 2007. ACM → Video ⋆

A-7 Nicolas Roussel, Géry Casiez, Jonathan Aceituno, and Daniel Vogel. Giving a hand to the eyes: Leveraging input accuracy for subpixel interaction. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, New York, NY, USA, 2012. ACM → Video ⋆

A-8 Daniel Vogel, Matthew Cudmore, Géry Casiez, Ravin Balakrishnan, and Liam Keliher. Hand occlusion with tablet-sized direct pen input. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 557–566, New York, NY, USA, 2009. ACM

A-9 Daniel Vogel and Géry Casiez. Hand occlusion on a multi-touch tabletop. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 2307–2316, New York, NY, USA, 2012. ACM ★

A-10 Géry Casiez, Nicolas Roussel, Romuald Vanbelleghem, and Frédéric Giraud. Surfpad: riding towards targets on a squeeze film effect. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 2491–2500, New York, NY, USA, 2011. ACM ★

A-11 Anthony Martinet, Gery Casiez, and Laurent Grisoni. Integrality and separability of multitouch interaction techniques in 3d manipulation tasks. *Visualization and Computer Graphics, IEEE Transactions on*, 18(3):369 –380, march 2012

A-12 Daniel Vogel and Géry Casiez. Conté: multimodal input inspired by an artist's crayon. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 357–366, New York, NY, USA, 2011. ACM → Video ★

A-13 Radu-Daniel Vatavu, Daniel Vogel, Géry Casiez, and Laurent Grisoni. Estimating the perceived difficulty of pen gestures. In Pedro Campos, Nicholas Graham, Joaquim Jorge, Nuno Nunes, Philippe Palanque, and Marco Winckler, editors, *Human-Computer Interaction - INTERACT 2011*, volume 6947 of *Lecture Notes in Computer Science*, pages 89–106. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-23771-3_9

A-14 A. Choumane, G. Casiez, and L. Grisoni. Buttonless clicking: Intuitive select and pick-release through gesture analysis. In *Virtual Reality Conference (VR), 2010 IEEE*, pages 67 –70, march 2010

A-15 Bruno R. De Araùjo, Géry Casiez, and Joaquim A. Jorge. Mockup builder: direct 3d modeling on and above the surface in a continuous interaction space. In *Proceedings of the 2012 Graphics Interace Conference*, GI '12, pages 173–180, Toronto, Ont., Canada, Canada, 2012. Canadian Information Processing Society → Video ★

1€ Filter: A Simple Speed-based Low-pass Filter for Noisy Input in Interactive Systems

Géry Casiez[1,2,3], **Nicolas Roussel**[3] **& Daniel Vogel**[4]

[1]LIFL, [2]University of Lille & [3]Inria Lille, France
[4]Cheriton School of Computer Science, University of Waterloo, Canada
gery.casiez@lifl.fr, nicolas.roussel@inria.fr, dvogel@uwaterloo.ca

ABSTRACT

The 1€ filter ("one Euro filter") is a simple algorithm to filter noisy signals for high precision and responsiveness. It uses a first order low-pass filter with an adaptive cutoff frequency: at low speeds, a low cutoff stabilizes the signal by reducing jitter, but as speed increases, the cutoff is increased to reduce lag. The algorithm is easy to implement, uses very few resources, and with two easily understood parameters, it is easy to tune. In a comparison with other filters, the 1€ filter has less lag using a reference amount of jitter reduction.

ACM Classification Keywords

H.5.2 [Information interfaces and presentation]: User interfaces - Input devices and strategies.

General Terms

Algorithms, Performance, Human Factors

Author Keywords

Signal; noise; jitter; precision; lag; responsiveness; filtering

INTRODUCTION

Noisy signals occur when an original time varying value undergoes undesirable and unpredictable perturbations. These may be caused by things like heat and magnetic fields affecting hardware circuitry, the limits of sensor resolution, or even unstable numerical computation. Noisy signals are a common problem when tracking human motion, particularly with custom sensing hardware and inexpensive input devices like the Kinect or Wiimote. In addition, even signals from established high-end sensing systems can become noisy when interaction techniques use large scaling effects. A common example is using a Vicon tracking system to implement ray casting with a wall display [6]: calibration problems and hand tremor add further perturbations to the ones amplified by the pointing technique.

Noise affects the quality of a signal in two primary ways [9]. It can reduce *accuracy*, by adding an *offset* between the observed values and the true ones. More often, it reduces *precision*, where repeated observations of values exhibit *jitter* – many different values are observed for a single true one. Jitter has a large effect on the way people perceive and act. For example noisy values are harder to read and unstable cursors hinder target acquisition [3, 7, 5]. One usually wants to filter noisy signals to reduce, and possibly remove, the unwanted parts of the signal. However, filtering inherently introduces time latency – commonly called *lag* – which reduces system *responsiveness*. Lag may not be an issue in domains like artificial perception and decision making, but with interactive feedback, it is very important. In fact, it is the combination of precision and responsiveness that are crucial: people can point accurately in spite of an offset, but only with minimal lag and jitter. The difficulty is that implementing and tuning a filter to minimize both jitter and lag is challenging, especially with little or no background in signal processing.

In this paper we describe the 1€ filter ("one Euro filter"), a tool to improve noisy signal quality with a tunable jitter and lag balance. It uses a low-pass filter, but the cutoff frequency changes according to speed: at low speeds, a low cutoff reduces jitter at the expense of lag, but at high speeds, the cutoff is increased to reduce lag rather than jitter. The intuition is that people are very sensitive to jitter and not latency when moving slowly, but as movement speed increases, people become very sensitive to latency and not jitter. We compare the 1€ filter to alternative techniques and show how it can reduce that same amount of jitter with less lag. It is also efficient and easy to understand, implement, and tune: the algorithm can be expressed in a few lines; it uses only basic arithmetic; and it has only two independent parameters that relate directly to jitter and lag. Other researchers and ourselves have already used variations of it in many projects. In fact, the "dynamic recursive low-pass filter" used by the third author in [6] established the basic principle, but it required four parameters and a fixed sample rate. The '1€' name is an homage to the $1 recognizer [10]: we believe that the 1€ filter can make filtering input signals simpler and better, much like the $1 recognizer did for gestures.

After a review of the jitter, lag, and alternative filtering techniques, we describe the 1€ filter in detail with an implementation, discuss tuning with different applications, and conclude with an illustrative comparison.

JITTER, LAG, AND FILTERING

Several studies show jitter and lag have a negative impact on performance. MacKenzie et al. found mouse movement times increased 16% with 75 ms lag, and up to 64% with 225

CHI'12, May 5–10, 2012, Austin, Texas, USA.

ms lag [3]. With 3D hand tracking, Ware and Balakrishnan found that only 50 ms lag reduced performance by more than 8% [7]. Pavlovych and Stuerzlinger found no performance degradation below 58 ms lag using a mouse or Wiimote, but increasing jitter from 4 to 8 pixels doubled error rates for small targets [5]. Assuming a 100 PPI screen, 4 pixels corresponds to 1mm of jitter mean-to-peak: close to the 0.4 mm of jitter they found with the established Optitrack system.

Although the precision of an input device may be very good, it does not take into account scaling effects introduced by interaction techniques. Device input is often scaled up, so people can cover more display distance with less device movement. For example, default operating system mouse transfer functions can be scaled up $12\times$ [1] and factors as high as $90\times$ have been used when ray casting on wall sized displays [6]. Regardless of native device precision, scaling amplifies even small sensing perturbations, increasing jitter.

These results highlight the importance of balancing jitter and lag. Jitter should be less than 1mm mean-to-peak, but lag should be below 60 ms. As we shall see, any filter introduces some lag and considering 40-50 ms of inherent system lag [5], that leaves less than 10-20 ms for the filter.

Moving average

By the Central Limit Theorem and reasonable assumptions, averaging enough successive values of a noisy signal should produce a better estimate of the true one [9]. As a result, a *moving average* of the last n data values is commonly used by computer scientists as a kind of filter. For example, Myers et al. [4] used one for laser pointers and reduced hand tremor jitter from ± 8 pixels to between ± 2 and ± 4 pixels using a 0.5s window ($n = 10$). Since all n values are weighted equally, this creates a lag up to n times the sampling period.

Low-pass filters and exponential smoothing

With human movements, noise typically forms high frequencies in the signal while actual limb movements have lower frequencies. A low-pass filter is designed to let these desired low frequency portions pass through, while attenuating high frequency signals above a fixed cutoff frequency. The *order* of a low-pass filter relates to how aggressively it attenuates each frequency: first order filters reduce the signal amplitude by half every time the frequency doubles, while higher order variants reduce the signal amplitude at a greater rate. A discrete time realization of a first order low-pass filter is given by Equation 1 where X_i and $\hat{X}_i$ denote the raw and filtered data at time i and α is a smoothing factor in $]0, 1]$:

$$\hat{X}_i = \alpha X_i + (1-\alpha)\,\hat{X}_{i-1} \quad (1)$$

The first term of the equation is the contribution of new input data value, and the second term adds inertia from previous values. As α decreases, jitter is reduced, but lag increases since the output responds more slowly to changes in input. Since the contribution of older values exponentially decreases, a low-pass filter will have less lag than a high n moving average filter.

Smoothing techniques used in business and economic forecasts are similar in approach to a low-pass filter. The equation for *single exponential smoothing* is very similar to Equation 1. As the name suggests, *double exponential smoothing* uses two of these equations to handle trends in the signal. Although not formally documented, the Microsoft Kinect skeleton filters appear to be a variant of this type of smoothing[1]. LaViola extended double exponential smoothing for predictive tracking [2], building on Equations 1 and 2 to predict positions τ time steps in the future (Equation 3):

$$\hat{X}_i^{[2]} = \alpha\,\hat{X}_i + (1-\alpha)\,\hat{X}_{i-1}^{[2]} \quad (2)$$

$$P_{t+\tau} = \left(2 + \frac{\alpha\,\tau}{1-\alpha}\right)\hat{X}_i - \left(1 + \frac{\alpha\,\tau}{1-\alpha}\right)\hat{X}_i^{[2]} \quad (3)$$

Kalman filters

Unlike the techniques above, Kalman filters make assumptions about the system generating the signal. Typically used for navigation and tracking, they work well when combining data from different sensors (e.g. a GPS and a speedometer) or when the system can be modeled by equations (e.g. determining vehicle acceleration from accelerator pedal position). Kalman filters rely on a *process model* and a *measurement model*. The standard Kalman filter uses a discrete-time linear stochastic difference equation for the process model and assumes that process and measurement noise are independent of each other, white, and are normally distributed [8]. When estimating the true position of a moving object, the process model is typically a linear function of the speed and the previous estimated position. With additional complexity, Extended and Unscented variants of Kalman filters can also model non-linear processes and observations [8].

In the frequent case where the process and measurement noise covariances are not known, one must determine them empirically. This task can be challenging, and an improperly tuned filter can increase and even degrade the signal [9], by creating artificial "overshooting" movements for example. Moreover, understanding Kalman filters requires mathematical knowledge beyond basic linear algebra such as statistics, random signals, and stochastic methods. Implementing them requires a language or library with matrix operations. And, as demonstrated by LaViola for predictive tracking, they can be considerably slower to compute than double exponential smoothing predictors (approximately $135\times$) with similar jitter and lag performance [2].

THE 1€ FILTER

The 1€ filter is an adaptive first-order low-pass filter: it adapts the cutoff frequency of a low-pass filter for each new sample according to an estimate of the signal's speed, or more generally, its derivative value. Even though noisy signals are often sampled at a fixed frequency, filtering can not always follow the same pace, especially in event-driven systems. To accommodate possible fluctuations, we rewrite equation 1 to take into account the actual time interval between samples. Using a direct analogy with an electrical circuit, where a resistor in series with a capacitor defines a first order low-pass filter, α can be computed as a function of the sampling period T_e and a time constant τ, both expressed

[1] `http://cm-bloggers.blogspot.com/2011/07/kinect-sdk-smoothing-skeleton-data.html`

in seconds (Equation 4). The resistor and capacitor values define the time constant ($\tau = RC$) and the corresponding cutoff frequency f_c, in Hertz, of the circuit (Equation 5).

$$\alpha = \frac{1}{1 + \frac{\tau}{T_e}} \quad (4)$$

$$\tau = \frac{1}{2\,\pi\,f_c} \quad (5)$$

$$\hat{X}_i = \left(X_i + \frac{\tau}{T_e}\hat{X}_{i-1}\right)\frac{1}{1 + \frac{\tau}{T_e}} \quad (6)$$

$$f_c = f_{c_{min}} + \beta\,|\dot{\hat{X}}_i| \quad (7)$$

The sampling period T_e (or its inverse, the sampling rate) can be automatically computed from timestamps, so the cutoff frequency f_c is the only configurable parameter in equation 6. As with any low-pass filter, decreasing f_c reduces jitter, but increases lag. Finding a good trade-off between the two is difficult since people are more sensitive to jitter at low speeds, and more sensitive to lag at high speeds. This is why an adaptive cutoff frequency works well. To reduce jitter, a low f_c is used at low signal speeds, and to reduce lag, f_c is increased as speed increases. We found that a straightforward linear relationship between cutoff frequency f_c and the absolute speed works well (Equation 7). The speed (i.e the derivative $\dot{\hat{X}}_i$) is computed from raw signal values using the sampling rate and then low-pass filtered with a cutoff frequency chosen to avoid high derivative bursts caused by jitter. Our implementation uses a fixed value of 1 Hz, leaving only two configurable parameters: the intercept $f_{c_{min}}$ and the slope β shown in Equation 7. Details of the algorithm are provided in the Appendix.

Tuning and Applications

To minimize jitter and lag when tracking human motion, the two parameters can be set using a simple two-step procedure. First β is set to 0 and $f_{c_{min}}$ to a reasonable middle-ground value such as 1 Hz. Then the body part is held steady or moved at a very low speed while $f_{c_{min}}$ is adjusted to remove jitter and preserve an acceptable lag during these slow movements. Next, the body part is moved quickly in different directions while β is increased with a focus on minimizing lag. Note that parameters $f_{c_{min}}$ and β have clear conceptual relationships: if high speed lag is a problem, increase β; if slow speed jitter is a problem, decrease $f_{c_{min}}$. Rotational input uses a similar tuning process, but rotation axis and angle are filtered separately.

Another application of the 1€ filter is displaying noisy numerical values, such as an unsteady frame rate used to monitor graphical application performance. The goal is to reduce jitter to make the numerical output legible while minimizing lag so the value remains timely. Tuning is similar to above: adjust $f_{c_{min}}$ until the text becomes stable, then increase β until just before the text become unstable.

COMPARISON WITH OTHER FILTERS

To compare the 1€ filter with other techniques, we created a Python application that periodically samples the XY position of the system cursor, adds noise, and displays filtered cursor positions. Each filter can be tuned interactively and all filters can be shown simultaneously making it possible to visually compare jitter reduction and lag across parameter settings and filters. Once tuned, timestamped positions can be logged for the system cursor (with and without noise) and filtered positions of all filters. We used a MacBook Pro with a 1440 × 900 pixel display (109 PPI).

In our comparison, we used independent Gaussian white noises for X and Y with a 50 dB SNR[2], a public implementation of the Kalman filter[3], and custom implementations of a moving average, single exponential, and LaViola's double exponential smoothing. We tuned moving average first and used its performance as a baseline. We found that averaging more than 14 data values did not reduce jitter further and only increased lag, so we used n=14. Then we interactively tuned the other filters to primarily match the jitter reduction of moving average, and secondarily attempting to reduce lag.

Tuning single exponential smoothing to match the reference jitter requires a low alpha value (α=0.11) which introduces lag. This highlights the difficulty of tuning with only a single parameter. For LaViola's double exponential smoothing filter, the reference jitter is obtained with a lower alpha value (α=0.06) and with lower lag. However, this causes overshooting when the pointer abruptly decelerates. For the Kalman filter, we set the measurement noise covariance to the variance of the introduced noise (18.06) as in [2], and adjusted the process noise covariance until we obtained the reference jitter reduction (at a value of 0.3). The amount of lag for this setting was comparable to the moving average and single-exponential. For the 1€ filter, we matched the reference jitter and optimized lag using the tuning procedure described above. In the first tuning step, setting $f_{c_{min}}$ = 1 Hz and β = 0 matched the reference jitter and lag was similar to single exponential smoothing. In the second tuning step, increasing β to 0.007 made the lag almost imperceivable yet maintained the reference jitter when stationary or moving slowly. A supplementary video demonstrates this tuning process and visualizes filter performance.

For a quantitative comparison, we logged the system cursor at 60 Hz for about 1 hour during regular desktop use, then added white noise and applied the filters using the settings above. Figure 1 shows the distance from each filtered cursor position to the true one, binned into four speed intervals. Note that since we tuned the filters to match a reference jitter when not moving, the error between filtered position and noiseless position is primarily due to lag when moving. With higher speeds, the filtered position lags farther and farther behind, increasing this distance (the small distances in the 0 mm/s interval are likely due to offset or overshooting). All filters introduce a similar amount of lag except for the 1€ filter which has less lag across all speed intervals.

As an overall comparison, we computed the Standard Error of the Mean (SEM) in mm for each filter for this data

[2]This signal-to-noise ratio was estimated from Gametrak data using a zero phase shift filter and is consistent with numbers in [2]

[3]`http://greg.czerniak.info/node/5`

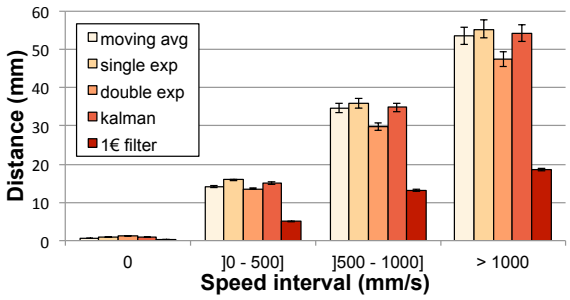


Figure 1. Mean distance between filtered and true cursor position for each speed interval and filter. Error bars represent 95% CI.

set. The 1€ filter has the smallest SEM (0.004) followed by LaViola's double exponential smoothing (0.013), the moving average and the Kalman filter (0.015), and single exponential smoothing (0.016). Our intention for this evaluation is to illustrate the performance of the 1€ filter in an intuitive way under realistic conditions. We are exploring alternative comparisons with user experiments, synthetic reference movements, different noise configurations, and examples of "noisy" hardware.

CONCLUSION

Human-Computer Interaction researchers and practitioners should stop filtering noisy input with a moving average. In most cases, they do not need to wrestle with low-level signal processing issues or with more complex techniques like Kalman filtering – which can be difficult to understand, tune, and implement. The 1€ filter is an intuitive and practical alternative since it is easy to understand, implement, and tune for low jitter and lag. Best of all, it produces better results.

REFERENCES

1. Casiez, G., and Roussel, N. No more bricolage! Methods and tools to characterize, replicate and compare pointing transfer functions. In *Proc. of UIST'11*, ACM (2011), 603–614.
2. LaViola, J. J. Double exponential smoothing: an alternative to kalman filter-based predictive tracking. In *Proc. of EGVE '03*, ACM (2003), 199–206.
3. MacKenzie, I. S., and Ware, C. Lag as a determinant of human performance in interactive systems. In *Proc. of CHI '93*, CHI '93, ACM (1993), 488–493.
4. Myers, B. A., Bhatnagar, R., Nichols, J., Peck, C. H., Kong, D., Miller, R., and Long, A. C. Interacting at a distance: measuring the performance of laser pointers and other devices. In *Proc. of CHI '02*, 33–40.
5. Pavlovych, A., and Stuerzlinger, W. The tradeoff between spatial jitter and latency in pointing tasks. In *Proc. of EICS '09*, ACM (2009), 187–196.
6. Vogel, D., and Balakrishnan, R. Distant freehand pointing and clicking on very large, high resolution displays. In *Proc. of UIST '05*, ACM (2005), 33–42.
7. Ware, C., and Balakrishnan, R. Reaching for objects in vr displays: lag and frame rate. *ACM ToCHI 1*, 4 (Dec. 1994), 331–356.
8. Welch, G., and Bishop, G. An introduction to the Kalman filter. SIGGRAPH course, ACM, Aug. 2001.
9. Wilson, A. D. Sensor- and recognition-based input for interaction. In *The Human Computer Interaction handbook*. CRC Press, 2007, 177–199.
10. Wobbrock, J. O., Wilson, A. D., and Li, Y. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proc. of UIST '07*, ACM (2007), 159–168.

APPENDIX A - 1€ FILTER

Algorithm 1: 1€ filter

EXT: First time flag: *firstTime* set to *true*
Data update rate: *rate*
Minimum cutoff frequency: *mincutoff*
Cutoff slope: *beta*
Low-pass filter: *xfilt*
Cutoff frequency for derivate: *dcutoff*
Low-pass filter for derivate: *dxfilt*
IN : Noisy sample value: *x*
OUT: Filtered sample value

if *firstTime* **then**
  firstTime ← *false*
  dx ← *0*
else
  dx ← (*x* - *xfilt*.hatxprev()) * *rate*
end
edx ← *dxfilt*.filter(*dx*, alpha(*rate*, *dcutoff*))
cutoff ← *mincutoff* + *beta* * |*edx*|
return *xfilt*.filter(*x*, alpha(*rate*, *cutoff*))

Algorithm 2: *Filter* method of Low-pass filter

EXT: First time flag: *firstTime* set to *true*
IN : Noisy sample value : *x*
Alpha value : *alpha*
OUT: Filtered value

if *firstTime* **then**
  firstTime ← *false*
  hatxprev ← x
end
hatx ← *alpha* * *x* + (1 - *alpha*) * *hatxprev*
hatxprev ← *hatx*
return *hatx*

Algorithm 3: Alpha computation

IN : Data update rate in Hz: *rate*
Cutoff frequency in Hz: *cutoff*
OUT: Alpha value for low-pass filter

tau ← 1.0 / (2*π**cutoff*)
te ← 1.0 / *rate*
return 1.0 / (1.0 + *tau*/*te*)

No more Bricolage! Methods and Tools to Characterize, Replicate and Compare Pointing Transfer Functions

Géry Casiez[1,2,3] *and Nicolas Roussel*[2]
[1]LIFL, [2]INRIA Lille & [3]University of Lille
Villeneuve d'Ascq, France
gery.casiez@lifl.fr, nicolas.roussel@inria.fr

ABSTRACT
Transfer functions are the only pointing facilitation technique actually used in modern graphical interfaces involving the indirect control of an on-screen cursor. But despite their general use, very little is known about them. We present EchoMouse, a device we created to characterize the transfer functions of any system, and libpointing, a toolkit that we developed to replicate and compare the ones used by Windows, OS X and Xorg. We describe these functions and report on an experiment that compared the default one of the three systems. Our results show that these default functions improve performance up to 24% compared to a unitless constant CD gain. We also found significant differences between them, with the one from OS X improving performance for small target widths but reducing its performance up to 9% for larger ones compared to Windows and Xorg. These results notably suggest replacing the constant CD gain function commonly used by HCI researchers by the default function of the considered systems.

ACM Classification: H.5.2 [Information interfaces and presentation]: User interfaces - Graphical user interfaces.

General terms: Documentation, Experimentation, Human Factors, Measurement, Performance, Standardization

Keywords: Pointing, control-display gain functions, CD gain, pointer acceleration, transfer functions, toolkit

INTRODUCTION
Indirect control of an on-screen cursor with a separate device has been the prevalent way of pointing in graphical interfaces for many years. The mouse is undoubtedly the most popular pointing device in this context. As explained by Moggridge [20], it was not chosen simply because Engelbart invented it, but because it turned out to be the device that performed best for pointing and clicking on a display, outperforming everything else that was tried in early tests with users. More than forty years later, the mouse still provides a good match between human performance and the demands of desktop graphical interfaces [13], the touchpad offering a similar match on laptop configurations. Despite the current trend for tactile screens, indirect pointing will thus probably remain the prevalent paradigm for some time.

In indirect pointing configurations, movements in the control space can be mapped to different ones in the display space. We use the term *transfer function* to refer to the relationship between movements in the two spaces. It is very important to note that in order for this relationship to be meaningful, it has to be hardware-independent. All movements must thus be described using standard length and time units (e.g. meters) and not device-specific ones (e.g. mickeys and pixels). In the case of simple linear relations, the term *CD gain*[1] is commonly used to refer to the scale factor between the two spaces, e.g. $CDgain = V_{\text{display}}/V_{\text{control}}$ [8]. To be meaningful, this coefficient must be unitless, which will only be the case if the two factors involved in its computation are expressed using the same length and time units. The CD gain can be constant or dynamically adjusted over time based on control space kinetics or extrinsic information, for example. Whether static or dynamic, CD gain settings involve a trade-off between gross and fine positioning [16]. High gains reduce the time it takes to approach a distant target but make it hard to precisely position the cursor on it. Conversely, low gains support precise positioning but increase the time to cover large distances.

It is generally assumed that in accordance with Meyer et al.'s *optimized initial impulse* model [19], the so-called "pointer acceleration" mechanisms implemented in modern desktops increase the CD gain as the user's hand or finger velocity increases. The transfer functions of Microsoft Windows, Apple OS X and Xorg (the X.Org Foundation server) are actually the only pointing facilitation mechanisms available to all users of these systems. But despite their general use, very little is known about them. Current knowledge on velocity-based transfer functions relies on evaluations of basic ones adapting the CD gain in discrete or continuous ways using low-order polynomials, e.g. [11, 14, 21]. The internal details and design rationales of the functions that we all use are mostly unknown. And with the notable exception of Casiez et al.'s work on Windows XP and Vista functions [8], their impact has never been studied.

This paper reports on efforts we made with the long-term goals of advancing this state of knowledge and bringing it

[1]The term *CD ratio* is also used and corresponds to the inverse of the gain.

UIST'11, October 16–19, 2011, Santa Barbara, CA, USA.

in line with current practices. The paper is organized as follows. We first briefly review the related work, explaining how pointing facilitation research deals with transfer functions and what is known about them. We then explain what is actually involved from a system perspective when pointing on a desktop. We present `EchoMouse`, an HID device we created to characterize the transfer functions of any system, and `libpointing`, a toolkit that we developed to replicate and compare the ones used by Windows, OS X and Xorg[2]. We describe these functions and report on an experiment that compared the default one of the three systems. We conclude with a discussion and implications for future work.

RELATED WORK

Pointing transfer functions can be considered as low-level and general-purpose mechanisms for pointing facilitation. Before explaining what is known about them, it is interesting to look at how research on other pointing facilitation techniques takes them into consideration.

Transfer functions in other pointing facilitation research

As we explained, it is usually assumed that desktop systems dynamically adjust the CD gain based on movement speed. This behavior might well interfere with other pointing facilitation mechanisms, especially those that manipulate the CD gain, e.g. [23, 10, 5, 22]. As a consequence, one would expect researchers working on pointing facilitation to try to disable the system's transfer function, to precisely characterize it, or to systematically investigate its effect.

Transfer functions are usually treated as control variables, meaning they might influence a dependent variable but are not under investigation and are thus held constant from one test condition to another. One would expect a clear description of these control variables to ease the replication of a technique or experiment with different hardware configurations or operating systems, in case the control variable would in fact be a confounding one. However a review of the recent literature on pointing facilitation shows that the level of details provided is often incomplete or unclear.

Numerous authors report using a constant CD gain or ratio as a baseline condition or as a basis for their technique but fail to describe it with sufficient details. In [10], for example, Cockburn & Firth explain that the CD gain "*was set to a constant ratio of approximately 1:1.6*" in experiments running on a Linux system, but do not explain why they chose this particular value nor how they enforced it. The predictive pointing technique of the Delphian Desktop was evaluated by Asano et al. on Windows XP with a CD ratio "*set to a constant value of 0.5*" [2]. Again, the paper does not explain how this ratio was enforced. For the Bubble Cursor on the same system, Grossman & Balakrishnan say that "*mouse acceleration was set to 0, with a control-display ratio of 1/2*" [12]. The exact meaning of "*set to 0*" is unclear considering the Windows XP configuration interface (Figure 4), and we will see that a unitless constant ratio of 0.5 is not achievable through it (Figure 6). In their study of sticky targets, Mandryk & Gutwin said "*Windows pointer acceleration was turned off, and the baseline mouse gain was set to the midpoint*" [18]. The exact definition of this baseline is unknown which is unfortunate since the authors scaled it by 11 values between 0.05 and 1.0, CD gain being one of the experiment factors. In their own study of sticky icons, Worden et al. say that "*normal mouse gain was set at a constant 1 mickey to 3 pixels ratio for all conditions*" [23]. But as the mouse and display resolutions are not specified, this gain can not be expressed in a unitless hardware-independent way, which makes their results difficult to compare with those from other studies.

Enforcing a hardware-independent transfer function, even a constant gain, is actually quite difficult with current systems. Wobbrock et al. had to go to great lengths to disable Windows Vista's pointer acceleration and dynamically control the CD gain for their Angle Mouse study, for example. They acknowledge that "*although some on-line documentation discusses pointer ballistics in Windows, it does not contain sufficient information to establish the slider-to-gain mapping.*" [22]. A good way of enforcing a transfer function is to use a device not attached to the system cursor and an API that provides access to its raw data. As an example, Blanch et al. used the absolute coordinates of a puck on a Wacom tablet as input for their Semantic Pointing technique [5].

An alternative to enforcing a particular baseline is to use the default transfer function of the system. For the Ninja Cursors, Kobayashi & Igarashi say "*the mouse speed and acceleration rate were set to the Windows XP default values (middle speed, no acceleration)*" [15]. For DynaSpot, Chapuis et al. used "*the default X Window acceleration function*" [9]. As it is unclear whether these functions take into account specific characteristics of the devices, extensive details should be provided about them including their resolution (per length unit) and frequency. A problem with this approach is that it will be possible to replicate or reproduce the experiment only as long as the original system can be used.

In some cases, there is simply no way of knowing which transfer function was used. In [17], for example, MacKenzie & Isokoski only report using "*an optical USB Microsoft IntelliMouse with four buttons and a scroll wheel*" and an "*experimental software written in Java*". Mouse data was presumably provided to the Java application by the underlying operating system through its operative transfer function, but none of them is explicitly mentioned in the paper.

What is known about pointing transfer functions

An extensive review of the literature on transfer functions has been recently conducted by Casiez et al. [8]. Prior to their work, research on the effects of these functions on pointing performance had been largely inconclusive. In all constant CD gain studies, the range of gain evaluated was either small or had quantization problems. And the few dynamic transfer functions evaluated poorly resembled the ones used in modern systems, most of them involving a few discrete steps or simplistic low-level polynomials, e.g. [11, 14, 21].

Casiez et al. [8] showed there exists a wide range of constant CD gains for which performance is constant and provided a way to compute that range knowing the hardware and target widths and distances used in a particular context. They

[2]The source code for `EchoMouse` and `libpointing` is available from `http://libpointing.org/`.

also showed that the acceleration mechanisms of Windows XP resulted in faster pointing than constant gain functions: they found an average 3.3% improvement and up to 5.6% for small targets or long distances. In a different study, Casiez and Vogel showed that the impact of transfer functions on performance can be severe in the case of force input [7].

Windows XP mechanisms were re-implemented for the study described in [8] based on information extracted from the Windows registry and documentation publicly available from Microsoft [1]. However, the cursor controlled by this implementation was not in perfect sync with the system one due to missing details in the documentation. Figure 3 of Casiez et al.'s paper shows a plot of four Windows XP functions based on their custom implementation. It also shows a plot of six OS X functions that were estimated from the analysis of publicly available Apple source code, but not re-implemented.

POINTING: A SYSTEM PERSPECTIVE

Most if not all modern pointing devices conform to the *Human Interface Devices* (HID) class of the USB standard. This class covers a variety of equipments including keyboards, mice, touchpads and joysticks but also telephones, remote controls, barcode readers and voltmeters. HID devices are thus required to provide extensive descriptions of their characteristics to be properly recognized and used. Among other things, a pointing device description specifies for each axis whether transmitted values (called *counts*) are absolute or relative, linear or nonlinear, their byte size, their logical range, the corresponding physical range and the unit system and exponent used. The description also specifies the time interval that should be used when polling for data transfers.

The HID specification defines a simple *boot report format* for mice that allows to use them before the operating system is loaded, for low-level system configuration [3, p. 61]. This format describes movements along two axis with relative, linear, unitless counts encoded with one byte per axis, between -127 and +127. Most mice support this format and for many, it is also the one they use to report to the system once it is loaded. This format is also supported by touchpads, although they could provide an absolute location, so they can be used in place of a mouse even at boot time.

The polling interval is typically set to 8 ms for mice, leading to an update frequency of 125 Hz. For devices that specify their logical and physical range and their unit system and exponent, the resolution can be computed in counts per unit with: $Res = LogRange/(PhysRange * 10^{Exp})$. Unfortunately, many report formats including the boot one specify unitless values with no physical range, which makes this formula inapplicable. For these devices, systems usually assume a resolution of 400 CPI[3].

The fact that systems have no reliable way of knowing a pointing device's resolution is becoming more and more problematic as manufacturers not only propose high-resolution ones but also some where it is adjustable[4]. A higher resolution results in more counts reported for the same distance traveled by the device, by sending bigger values at the same rate or by reporting more often. High-resolution mice indeed use two-bytes values for each axis and can set a polling interval as low as 1 ms. A transfer function not aware of the resolution change or not taking time properly into account will inevitably misinterpret the reported counts and produce undesirable effects, the most common one being considerably amplified movements. This problem is so frequent that a lot of people equate the resolution of the device with the cursor speed, as illustrated by this text displayed in the mouse section of a consumer electronics retailer:

> "*Specified in DPI, the resolution corresponds to the speed of the mouse cursor on your screen. The higher it is, the less you will have to move the mouse for the same on-screen distance, though you will be less accurate.*"

Changing the resolution of a pointing device or switching to one with a different resolution should not alter the mapping between movements in the control and display spaces. Changing or reconfiguring the transfer function should be the only way of doing that.

ECHOMOUSE

`EchoMouse` is an electronic device that we designed to measure a system's response to pointing movements received from an HID equipment. Based on a Microchip PIC (18LF14K50) programmed in C using the PICkit 2 development environment, it includes a switch and two LEDs for debugging purposes (Figure 1). Its program uses a mouse firmware provided by Microchip, so it appears as an ordinary HID mouse to the system. It has no motion sensor, though. We instead added a USB endpoint to the firmware to which a program can send an HID report to be echoed by the device on its mouse endpoint. Reports are expected and echoed in HID boot format. They are thus indistinguishable from genuine mouse reports and handled as such by the system.

We have used our `EchoMouse` to look into the transfer functions of Windows, OS X and Xorg with a specific program implementing the following procedure. After placing the

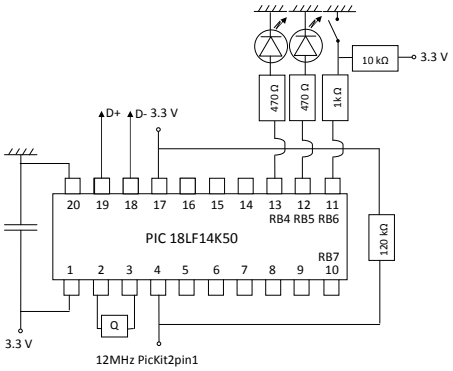


Figure 1: Electronic diagram of EchoMouse. The 3.3V can be easily generated from the 5V of the USB port using a voltage divider or a Zener diode.

[3] We will use CPI (*counts per inch*) and PPI (*pixels per inch*) instead of DPI to make a clear distinction between input and output resolutions.

[4] The resolution of the Logitech Gaming Mouse G500 can be adjusted between 200 and 5700 CPI, for example.

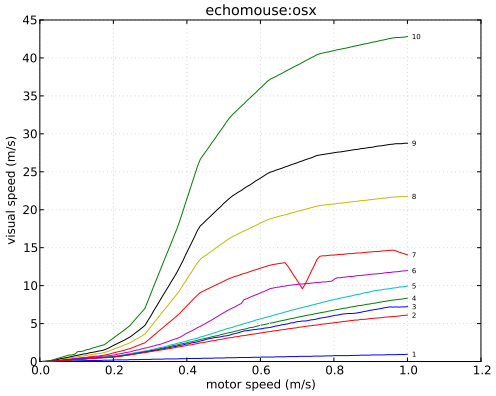


Figure 2: Speed plot for OS X 10.5.6 transfer functions ($n = 10$). Count and pixel values were converted to speed values considering the actual resolution of the monitor used and assuming a 400 CPI resolution for EchoMouse.

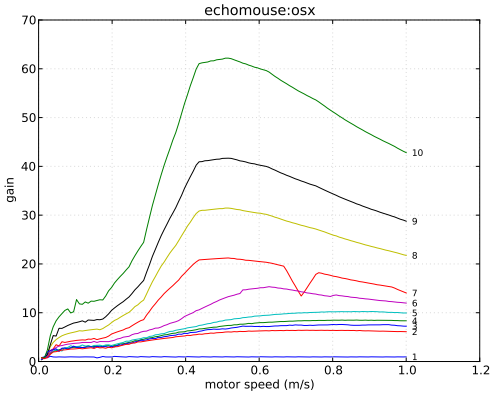


Figure 3: Gain plot of the data shown in Figure 2.

system cursor on the left edge of the screen, the program prepares an HID report describing a $(dx, 0)$ translation. It sends it n times to `EchoMouse` at 125 Hz, waits for a few milliseconds and polls the system for the new cursor position. It then divides the horizontal pixel distance traveled by the cursor by n and stores this number along with dx in a table. This procedure accounts for potential subpixel precision. When repeated for all dx between 1 and 127, it provides an extensional description of the transfer function used by the system. Repeating this for every pointer acceleration setting provides the descriptions of all the functions supported by the system.

Figures 2 and 3 illustrate the 10 transfer functions available in OS X 10.5.6 through the *Tracking* slider of the mouse preference pane shown on Figure 5. Figure 2 shows the speed in display space as a function of speed in control space. One can easily see with this plot that the functions used by OS X are not linear, can not be approximated by a single low-order polynomial and are likely defined in a piecewise fashion. The singularity observed in function 7 is inexplicable, however. And the differences between functions are not easy to perceive, especially at low speeds.

Figure 3 shows the CD gain as a function of speed in control space. This plot makes it easier to compare the functions at low speeds and to relate them to the movements in motor space, a CD gain of 1 corresponding to a horizontal line. The fact that some of the functions strongly decrease after a certain speed is hard to explain. We hypothesize that the functions were designed in a spatial space like the one of Figure 2 and that the designers were actually not aware of this slope change. The singularity observed in function 7 remains inexplicable. At this point, one can only hypothesize that the designers of the functions never tried to plot them.

`EchoMouse` allowed us to investigate the transfer functions used by three systems without spending too much time on their internals. Replicating the device is relatively easy. A similar but pure software approach is also probably achievable by developing fake mouse drivers, e.g. using `uinput` on Linux, or synthesizing low-level mouse motions, e.g. using the `SendInput` function on Windows. The `EchoMouse` approach is however restricted to observation, it can not alter the system functions. One can never be sure that the pseudo motions injected in the system have actually been processed when reading the cursor location. And one can hardly know if and which hardware or movement characteristics are taken into account by the system when applying the functions.

Although valuable for preliminary studies, an outside observation point is not enough to fully understand and compare the transfer functions used by modern desktop interfaces. In addition to `EchoMouse`, we thus developed `libpointing`, a toolkit that allows to replicate and compare them.

LIBPOINTING

The `libpointing` toolkit was designed with several goals in mind. First, we wanted a way of directly accessing HID pointing devices to bypass the system's transfer functions. Second, we wanted to replicate as faithfully as possible the transfer functions of Windows, OS X and Xorg. Third, we wanted the toolkit to run on these platforms to be able to compare our implementations to the genuine ones. And fourth, we wanted to support comparisons between the replicated functions and other ones.

The toolkit consists of about 10,000 lines of C++ developed on OS X 10.6, Ubuntu 10.10 and Windows XP, Vista and 7. Although parts of it use the Qt framework, care has been taken so that its essential components can be used with other GUI frameworks. A key aspect of `libpointing` is that it supports the use of URIs [4] to specify input and display devices as well as transfer functions. Combined with object factories, this makes it possible to (re)define at runtime the instances used by a program and contributes remarkably to the flexibility of the whole. The following summarizes the main other features of the toolkit.

Pointing devices

`PointingDevice` instances are created from URIs using the static `create` method of that class. Other methods allow to

check whether a device is active, to obtain its resolution (in counts per inch), update frequency and URI, and to associate a callback to it. The callback will be executed every time the device has a motion or button event to report, passing it a timestamp, dx and dy values (in counts) and an integer coding the buttons states.

The toolkit provides direct access to any connected HID pointing device through platform-specific subclasses and URIs such as `osxhid:/USB/4600000/AppleUSBTCButtons`. The special URI `any:` matches any supported device and will list the available ones in the console if a debug option is passed on the query string. All HID `PointingDevice` objects support hot (re)plugging of the corresponding device. The toolkit also includes two pseudo-device subclasses for debugging and testing that can be instantiated with URIs such as `noisy:?cpi=400&hz=125` and `dummy:?cpi=800&hz=125`. The first one will execute the callback at the specified frequency to report movements synthesized by a 2D Perlin noise generator. The second one will never execute the callback but will return the specified values when queried for its resolution and update frequency.

Display devices

`DisplayDevice` instances are also created from URIs using a static `create` method. Other methods allow to obtain the horizontal and vertical bounds (in pixels), sizes (in inches or millimeters) and resolutions (in pixels per inch) as well as the refresh rate and the URI of a particular display.

URIs such as `osxdisplay:/69676098` and platform-specific subclasses provide access to the displays connected to the computer. A pseudo-device subclass is also available that will simply store the configuration values passed on the query string, e.g. `dummy:?ppi=96&hz=60`, and return them as expected when requested by the above methods.

Transfer functions

`TransferFunction` instances are created using a static `create` method from a URI, a `PointingDevice` and a `DisplayDevice`. Other methods allow to obtain the URI of a function, to clear its internal state and to apply it to dx_{in} and dy_{in} values (in counts) with a timestamp to produce dx_{out} and dy_{out} values (in pixels). The toolkit provides subclasses that correspond to different transfer functions. Care has been taken so that all implementations are platform-independent, i.e. all the transfer functions proposed by `libpointing` can be used on all the supported platforms. Although it imposes some constraints, we believe that having cross-platform implementations is important: a long-term goal for `libpointing` could be to serve as a living archive of the functions tried and used in research and commercial systems.

Three subclasses replicate the functions used by Windows (`windows:`), OS X (`osx:`) and Xorg (`xorg:`). The next section of the paper will describe these functions with extensive details. The special URI `system:` can be used to create the single appropriate instance of these subclasses that corresponds to the function used by the system. Configuration settings passed on the optional query string are applied to both the created instance and the system function.

Two other subclasses implement constant CD gain in both the naive and the right way. The first one simply multiplies the dx_{in} and dy_{in} values by a specified factor, e.g. `naive:?gain=2`, and returns the nearest integers as dx_{out} and dy_{out}. As this ignores the resolution of the input and output devices and multiplies counts to produce pixels, the effective unitless gain will most probably not be the one requested (it is usually higher, input devices having higher resolutions than displays). The second implementation (`constant:`) takes the resolutions into account to effectively produce a hardware-independent constant gain. It converts counts into distances, multiplies these distances by the specified factor and returns their pixel equivalent.

The toolkit provides various other subclasses, including a `sigmoid:` function and a `composition:` one, the latter allowing to compose an arbitrary number of functions. Adding a new function is simply a matter of creating a new subclass, implementing its `getURI`, `clearState` and `apply` methods and modifying the `TransferFunction::create` method.

Utilities

`libpointing` includes some test and debugging programs that allow to list the available devices and their characteristics, for example. The toolkit also includes a transfer function plotting tool written in Python using matplotlib. This tool proved quite useful as it provided some visual confirmation that our implementations of the Windows, OS X, and Xorg functions matched the data collected using `EchoMouse`. It was also used to plot all the curves shown in this paper.

The toolkit also includes an application that allows to test an arbitrary number of transfer functions at the same time specified by their URI as command-line arguments. The program creates an on-screen cursor (a small square) for each function, a single pointing device being used to control all of them. In addition to supporting informal comparisons between functions or between different settings of the same functions, this application proved again quite useful to compare our implementation of the Windows, OS X and Xorg functions with the system ones.

WINDOWS:, OSX: AND XORG: TRANSFER FUNCTIONS

As explained, one of our goals with `libpointing` was to replicate as faithfully as possible the transfer functions of Windows, OS X and Xorg. We not only wanted cursors controlled by our implementations to follow the system ones as closely as possible, but we also wanted to replicate the controls on the functions available to users from the relevant configuration interfaces. Our work was based on the documentation and source code publicly available from Microsoft, Apple and the Freedesktop community. This section presents the key findings that emerged from it.

The curves shown in the figures below have been plotted assuming the following pointing and display devices: `dummy:?cpi=400&hz=125`, `dummy:?ppi=96&hz=60`.

windows:

The transfer functions used in Microsoft Windows were redesigned for Windows XP, released in 2001. The rationales

for this redesign and its general principles are described in a public document. Together with the configuration interface found in the "*Pointer Options*" tab of the "*Mouse Properties*" dialog (Figure 4), this document served as a starting point for our work. Note however that our experience was similar to that of Wobbrock et al. [22]: the information available was not sufficient to replicate the functions.

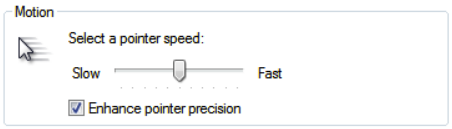


Figure 4: Windows 7 configuration interface with default settings. The same are used by Windows XP and Vista. No tooltip or help text is associated to these controls.

The transfer function code runs in an execution space where floating-point arithmetic is not available. The following equations illustrate how a (dx, dy) displacement in counts is transformed into pixel values when "*Enhance pointer precision*" is checked (Figure 4). Two important scale factors are used to convert count values to input speeds ($inConv$) and output speeds to pixel values ($outConv$). Computations for the y direction are omitted for brevity:

$$mag = max(|dx|, |dy|) + min(|dx|, |dy|)/2 \quad (1)$$
$$v_{in} = mag \times inConv \quad (2)$$
$$gain = lookup(v_{in}) \times pSpeed/10 \quad (3)$$
$$vx_{in} = dx_{in} \times inConv \quad (4)$$
$$vx_{out} = vx_{in} \times gain \quad (5)$$
$$px = vx_{out} \times outConv \quad (6)$$
$$dx_{out} = \lfloor px + rx \rfloor \quad (7)$$

The system computes a fixed-point approximation of the displacement vector magnitude (1). The magnitude is converted into an input velocity (2). A lookup table provides a base CD gain value for that velocity that is scaled by a factor ($pSpeed$) related to the "*pointer speed*" slider (Figure 4) to obtain the actual gain to apply (3). Each direction is then treated separately the following way. The directional input speed is computed (4) and multiplied by the gain to produce the output speed (5), which is then converted to a pixel displacement (6). The function returns the integral part (floor) of the sum of this displacement and the remainder of previous computations (7).

Although $inConv$ should be the quotient of the pointing device update frequency by its CPI resolution, empirical tests showed that it is always $1/3.5$. This constant value might be an approximation of 125/400, these numbers being common for mice. Empirical tests also showed that although $outConv$ should be the quotient of the display resolution by the pointing device update frequency, it is not the case either. Its actual value depends on the display resolution and frequency but also involves hardwired constants (96 PPI, 60 Hz, 150) and varies between system versions, XP and Vista differing from 7. Each of these three versions also uses a different algorithm to handle the x and y remainders: XP clears them when the pointer stops or changes direction, Vista clears them only when the pointer changes direction and 7 never clears them.

The lookup table that returns a base CD gain for a given device speed is stored in the Windows registry, so it should be possible to modify it. The position of the "*pointer speed*" slider (Figure 4) determines the value of the scale factor ($pSpeed$) applied on this base gain. Available values are: (slow) 1, 2, 4, 6, 8, 10 (default), 12, 14, 16, 18 and 20 (fast).

When "*Enhance pointer precision*" is unchecked, a naive constant CD gain is used. Based on the slider position, the available values for this gain are: (slow) 0.03125, 0.0625, 0.25, 0.5, 0.75, 1.0 (default, one pixel for one count), 1.5, 2.0, 2.5, 3.0 and 3.5 (fast). In this mode, no matter the system version, the remainders are never cleared.

The Windows transfer functions are available in `libpointing` through URIs such as `windows:<version>?slider=0&epp=true` where `<version>` can be one of `xp`, `vista` or `7`. The `slider` parameter encodes the slider position between -5 and +5 where 0 corresponds to the default position. The `epp` parameter indicates whether "*Enhance pointer precision*" is checked or not. Figure 6 shows the curves associated to each slider position, with and without enhanced pointer precision. Our implementation of these functions consists of about 200 lines of code. On XP, Vista and 7, a cursor controlled by it remains superimposed with the genuine one whatever movements are made. Our cursor can even respond to pointing device movement before the system cursor when the vertical synchronization of the display is disabled.

osx:

The source code for the internal parts of OS X that deal with pointing transfer functions is publicly available as part of the IOHIDFamily[5] project, the main concerned files being IOHIDSystem/IOHIPointing.cpp and IOHIDSystem/IOHIDSystem.cpp. From the archived versions of this project, it seems that the current pointer acceleration mechanisms first appeared in OS X 10.2, released in 2002. However, although the source code is available, the design rationales and principles of operation of these mechanisms are unknown. Figure 5 shows the related configuration interface, located in the "*Mouse*" pane of the system preferences. Note that from a user-perspective, the acceleration mechanisms are also badly documented, the tooltip associated to the slider being potentially misleading.

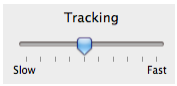


Figure 5: OS X 10.6.7 configuration interface for the mouse. A tooltip associated to the slider says "*Drag to adjust how fast you want the pointer to follow the movement of your mouse*".

[5] `http://opensource.apple.com/source/IOHIDFamily/`

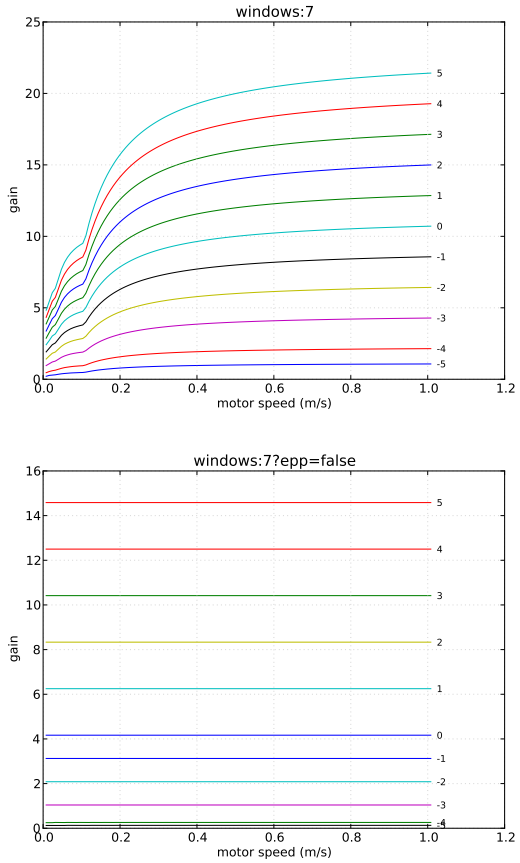


Figure 6: Windows 7 functions available through the interface shown in Figure 4 with and without enhanced pointer precision.

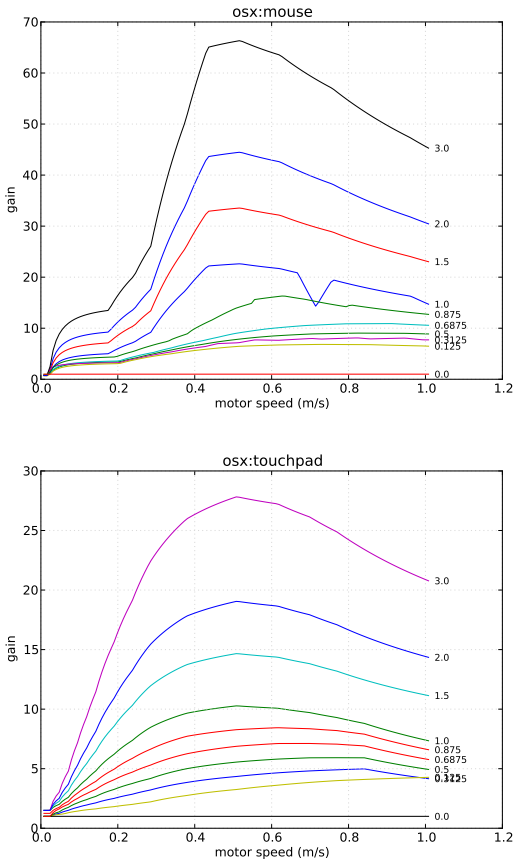


Figure 8: OS X 10.6.7 functions for mice (available through the interface shown in Figure 5) and touchpads.

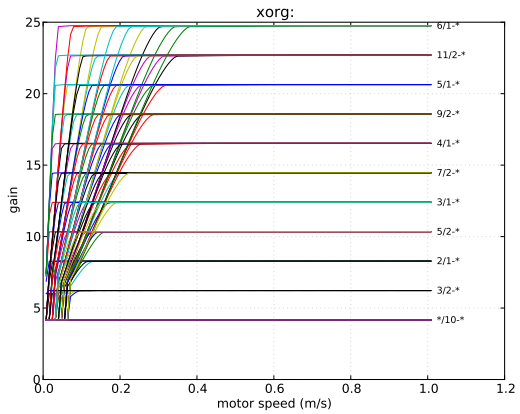


Figure 7: Xorg functions available in Ubuntu 10.10 through the interface shown in Figure 10.

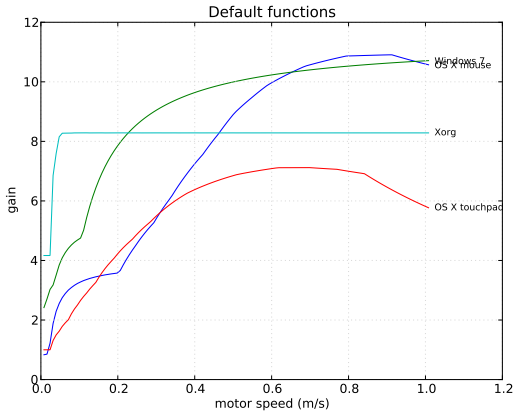


Figure 9: Default functions used by Windows 7, OS X 10.6.7 (mouse and touchpad) and Xorg.

It was relatively easy for us to isolate the portions of code from IOHIDFamily responsible for pointer acceleration (about 500 lines of code, mainly in IOHIPointing.cpp) and to add the necessary wrappers and definitions to make them compile on Windows and Linux. The basic operating principles followed by this code are also relatively simple and somewhat similar to those of Windows.

Each pointing device has an associated acceleration table provided by its driver (some also define a separate table for scrolling). This table specifies one or more curves defined by a series of segments and a *scale* level. The slider shown in Figure 5 allows users to specify a desired scale among the following: (slow) 0, 0.125, 0.3125, 0.5, 0.6875 (default), 0.875, 1, 1.5, 2, and 3 (fast). The system interpolates between the curves provided by the driver to create one that matches the desired scale and maps a vector magnitude to a CD gain value. Then for each (dx, dy) displacement, it computes an approximation of the vector magnitude using the same equation as Windows[6], uses the created curve to find the right CD gain, applies it to dx and dy, adds the previous remainders and returns the integral part of the result after updating the remainders. These remainders are never cleared.

The acceleration curves stored in device drivers are hardware-independent. When it interpolates between them, the system takes the resolution of the input device into account. However, it uses hardwired constants for the resolution of the display (96 PPI) and the frequency of the input and output devices (67 Hz). A detailed inspection of the drivers available on OS X 10.6.7 showed that several of them use the same tables. We identified two particular tables, one for mice and the other for touchpads, that seem to be used by all such devices that rely on Apple's drivers.

OS X transfer functions are available in `libpointing` through URIs such as `osx:<name or path>?scale=0.5`. The names `mouse` and `touchpad` can be used to load one of the generic acceleration tables that we found. A specific table can also be loaded from a file by specifying its path. The `scale` parameter encodes the desired scale. Figure 8 shows the curves associated to each slider position for generic mice and touchpads. Note that contrary to what it may seem in both cases, the unitless gain obtained for a scale of 0 is not always 1 but fluctuates between 0.9 and 1. From what we know, a constant CD gain is not achievable using these generic tables, even a naive one.

Although it can stay close in some situations, a cursor controlled by our `osx:` implementation does not remain superimposed with the genuine one. The reason appears to be some additional control mechanisms implemented in IOHIDSystem.cpp. Our current understanding is that these mechanisms feed the output of the function we just described into a trajectory prediction algorithm that schedules on-screen cursor updates synchronized with display refresh. No correction seems to be implemented in case a prediction was wrong, though. In effect, the whole acts as a low-pass filter on cursor movements. We hypothesize this explains the upward shift between the curves shown in Figure 3 and those in Figure 8 (above plot). Our present inability to schedule calls synchronized with display refresh prevents us from replicating these mechanisms, but we are currently investigating ways to circumvent this problem.

[6] All computations are also made using fixed-point arithmetic.

xorg:
The pointer acceleration mechanisms currently used by Xorg were introduced in 2008. The source code for these mechanisms is publicly available as part of the Xorg source tree[7]. It was again relatively easy for us to isolate the relevant portions of code (about 1500 lines of code, mainly in dix/ptrveloc.c) and to add the necessary wrappers and definitions to make them compile on Windows and OS X. This time, documentation for the design rationales and operating principles was also available[8], although a bit cryptic.

The changes introduced in Xorg in 2008 notably aimed at facilitating the exploration of transfer functions. The current architecture of the code supports 9 different *profiles* implemented within the new "*predictable*" scheme and the older "*lightweight*" scheme "*retained mostly for embedded scenarios*". Profiles can be considered as different transfer functions, although they share some common mechanisms and code. Numerous configuration settings are associated to them. But genericity and flexibility have a price: not only is the Xorg code for pointer acceleration much larger than the one used on other systems, but it is also far less readable.

The "*predictable*" scheme computes the euclidean distance corresponding to each displacement reported by the device and divides it by the time elapsed since the previous one. This instantaneous velocity is stored in a short history list ($n = 16$ by default) that is used to maintain a better estimation of the real pointing device velocity. Two adjustable settings also play an important part: *acceleration*, given as a fraction, and *threshold*. The first one defines a high value for the (naive) CD gain to be applied to displacements, considering a default low value of 1. The second one defines the minimum velocity that needs to be achieved to switch from the low gain to the high one. The active profile specifies how the estimated velocity will be used to determine the actual CD gain within these constraints. All computations are made with floating-point arithmetic. Remainders are preserved and never cleared.

The Xorg "*predictable*" transfer functions are available in `libpointing` through URIs such as `xorg:<profile>?accnum=2&accden=1&thr=4` where `<profile>` names one of the 9 available profiles, `accnum` and `accden` define the acceleration fraction and `thr` the threshold. On Ubuntu 10.10, a cursor controlled by our implementation remains superimposed with the genuine one.

It should be noted that a wide variety of command-line and graphical interfaces exists to configure the different profiles and their settings. Figure 10 shows the configuration interface available in the "*Pointer speed*" section of the "*Mouse preferences*" application of Ubuntu 10.10. Although the code that we use is not functionally limited to it, we will now fo-

[7] `http://cgit.freedesktop.org/xorg/xserver/tree/`
[8] `http://xorg.freedesktop.org/wiki/Development/Documentation/PointerAcceleration`

cus on the default profile used by Ubuntu ("*classic*") and the relevant settings that can be adjusted through this particular interface.

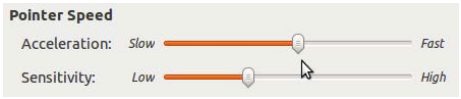


Figure 10: Ubuntu 10.10 configuration interface. A help page says about the first slider: "*Use the slider to specify the speed at which your mouse pointer moves on your screen when you move your mouse*". About the second: "*Use the slider to specify how sensitive your mouse pointer is to movements of your mouse*".

When the threshold is non-null, the "*classic*" profile implements a smooth transition between the low and high gain values. The sliders shown in Figure 10 only allow such configurations. As the label indicates, the upper slider controls the acceleration setting. When dragged, it feels like a continuous control but actually supports only a predefined set of values: (slow) 3/10, 4/10, 5/10, 6/10, 7/10, 8/10, 9/10, 10/10, 1/1, 3/2, 2/1 (default), 5/2, 3/1, 7/2, 4/1, 9/2, 5/1, 11/2, and 6/1 (fast). The bottom slider controls the threshold and actually feels like a discrete control. The available values are: (low) 1, 2, 3, 4 (default), 5, 6, 7, 8, 9, and 10 (high). In total, the interface shown in Figure 10 thus gives access to $19 \times 10 = 190$ configurations of the "*classic*" profile.

Figure 7 shows a plot of these 190 functions. As one would expect, the 90 functions with an acceleration setting lesser or equal than 1, those labeled */10-*, correspond to a naive constant gain of 1 (considering the 400 CPI and 96 PPI used for plotting the curves). Note that this is the only naive constant gain achievable through the interface shown in Figure 10 and that this interface does not allow to achieve a unitless constant gain.

Summary

Figure 9 shows the default transfer functions used by Windows, OS X and Xorg. Overall, despite a few differences, the different families have a lot in common.

The three systems take only partially into account the characteristics of the input and output devices. OS X is the only system that uses the real resolution of the input device (Windows assumes a 400 CPI resolution and Xorg does not use it). Xorg is the only system that takes input event times into account (the two others use harwired constant frequencies). Xorg completely ignores the display frequency and resolution while OS X uses hardwired constants for them (Windows varies on that topic).

All systems use a non linear function by default, but Windows and Xorg also support the use of naive constant gain functions. As the systems fail to properly take into account the resolution and frequency of the devices, none actually supports a unitless constant gain.

Windows and OS X both use fixed-point arithmetic. The three systems work with integer pixel coordinates but preserve the remainders to achieve subpixel precision when pointing. Windows 7, Mac OS X and Xorg never clear these remainders while they are cleared using different strategies on Windows XP and Vista.

The comparison of our custom cursors with the three system ones validated our Windows and Xorg implementations but revealed a slight difference for OS X presumably due to a trajectory prediction algorithm requiring information we are not yet able to provide.

EXPERIMENT

Our initial motivation for this experiment was to compare the performance of real-world transfer functions. Assuming they were probably used by many people and somewhat representative of theses systems, we decided to compare the default functions used by Windows, OS X and Xorg. We also added a constant CD gain function, as they are often used as a baseline for comparing pointing facilitation techniques.

Apparatus

A 400 CPI USB corded Logitech mouse was used as input device. A low-end model was preferred to a high-resolution one as 400 CPI is the default resolution considered by all systems. We used a 23" LCD display at a 1920×1200 resolution (98.5 PPI). The experiment was coded in C++ with the QT framework on a Windows 7 Professional machine with a NVidia GeForce GTX 460 graphics card. Our *libpointing* toolkit was used to get raw input from the mouse and apply the different transfer functions. Vertical synchronization of the display was disabled in order to be able to update our cursor's position at the mouse frequency (125 Hz). In this configuration, our controlled cursor was slightly in advance compared to the system one, which prevented any confounding effect of lag in the experiment.

Task

We used a reciprocal one dimensional pointing task (Figure 11). Each trial began after the previous target was successfully selected and ended with the selection of the current target. After a target was successfully selected, it turned grey and the next one (on the other side of the screen) turned green. If a participant missed a target, a sound was heard and an error was logged. Participants had to successfully select the current target before moving to the next one, even if it required multiple clicks. Participants used the left mouse button to select targets. After each block of trials, a cumulative error rate was displayed and a message encouraged participants to conform to an approximately 4% error rate by speeding up or slowing down.

Participants

Sixteen unpaid participants with a mean age of 30.6 (SD = 7.8, min = 23, max = 46) served in the experiment (15 male and 1 female, 13 right-handed and 3 left-handed). All participants worked most of their time with a computer. Three participants used exclusively OS X, four Windows 7, three Ubuntu 10.10. Two used Ubuntu 10.10 and Windows 7, two OS X and Windows 7, one Ubuntu 10.10 and OS X, and one Windows 7 and Ubuntu 10.10. Four participants used the mouse exclusively, two the touchpad and the remainder both devices. Among the sixteen participants, twelve kept the default settings for the mouse or touchpad while four slightly

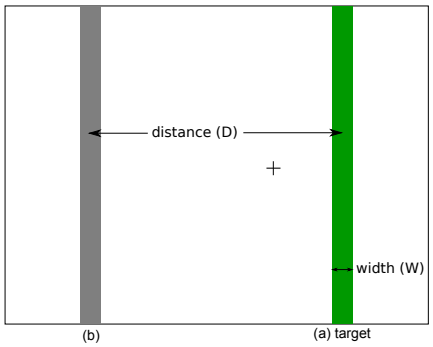


Figure 11: Experimental display. Targets were rendered as solid vertical bars equidistant from the center of the display in opposite directions along the horizontal axis. The target to be selected was colored green (a), and the previous one gray (b). The cursor was represented by a one-pixel-thick black cross 10 pixels wide.

increased the speed of their cursor by moving the slider in their configuration panel one or two ticks to the right.

Design

A repeated measures within-subjects design was used. The independent variables were the transfer function used (TF) and the target width (WIDTH).

Target distance was kept constant at 299.9 mm = 1,163 pixels. We decided for this moderate single distance because Casiez et al. had found stronger differences between constant CD gain and the Windows XP functions for small targets and long distances[9]. This decision was taken to reduce the duration of the experiment and to highlight the effect of WIDTH. The rationale was also that if no effect of TF was found with these settings, it would be likely that no such effect exists.

WIDTH was evaluated with four levels: W_{9pix} = 2.32 mm = 9 pixels, W_{6pix} = 1.55 mm = 6 pixels, W_{3pix} = 0.77 mm = 3 pixels, W_{1pix} = 0.26 mm = 1 pixel. Targets three pixels wide are common when resizing a window or clicking between two letters to position a text cursor. One pixel targets are less frequent but occur for example when selecting adjacent vertices or edges without zooming in vector drawing applications. The index of difficulty ranged from 7.0 to 10.2.

The transfer functions evaluated were constant CD gain of 1.5[10] (*Cst1.5*), the default Windows 7 function (*Win7*), the default OS X 10.6.7 function for mice (*OSX*) and the default Xorg function (*Xorg*). According to Casiez et al.'s method [8] and considering our experimental settings, the minimum gain value to prevent clutching was 30/30 = 1 and the maximum value that could be chosen given quantization problems and human limbs precision was equal to min(400/98.5, 0.26/0.2) = 1.3. The chosen CD gain value of 1.5 represents a good trade-off between these bounds and the CD gain value of 2 often used as a baseline in pointing experiments.

[9]The largest distance in [8] for a similar desktop configuration was 36 cm.
[10]Remainders were handled the same way as the other functions: they were never reset.

Participants were introduced to the task and had about 30 seconds to get used to it. They then completed three successive BLOCKS for each TF. Each BLOCK consisted of 24 trials: 6 repetitions of the 4 WIDTHS. WIDTHS were presented in decreasing order. The presentation order for TF was counterbalanced across participants using a balanced Latin Square design. Participants were encouraged to take a break after every 6 trials. They had to press the spacebar once they felt ready to start a new block. The desk was empty except for the keyboard and screen, and participants were instructed to use as much space as they wished to move the mouse. The experiment lasted approximately 15 minutes.

In summary, the experimental design was: 16 participants × 4 TF × 3 BLOCKS × 4 WIDTH × 6 trials = 4,608 total trials.

RESULTS

The dependent variables were the error rate and the movement time.

Error Rate

Targets that were not selected on the first attempt were marked as errors. Participants followed the instructions with an overall error rate of 4.1%. A repeated measures ANOVA showed a significant effect of WIDTH on error rate, the latter increasing as target width decreases ($F_{3,45}$=14.5, $p<0.001$). Pairwise comparisons showed significant differences between the smallest width and the three other widths ($p=0.001$; W_{1pix}: 9.7%, W_{3pix}: 2.8%, W_{6pix}: 1.7%, W_{9pix}: 2.1%).

Movement Time

Movement time is the main dependent measure and is defined as the time taken to move from a target to the next one and click on it. Targets marked as errors were removed from the timing analysis. We also considered trials at least three standard deviations away from the mean for each TF×WIDTH condition as outliers and removed them from the data analysis (1.6% of the trials).

A repeated measures ANOVA showed that the presentation order of TF had no significant effect or interaction on movement time, indicating that a within-participant design was appropriate. Repeated measures ANOVA showed a significant effect of BLOCK ($F_{2,30}$=14.2, $p<0.001$) on movement time. Pairwise comparisons showed a significant decrease in the movement time between the first block and the two remaining ($p<0.001$; Block 1: 2.05 s, Block 2: 1.93 s, Block 3: 1.94 s). The first block was thus removed from subsequent analysis.

Repeated measures ANOVA showed a significant main effect of TF ($F_{3,45}$=20.7, $p<0.001$)), WIDTH ($F_{3,45}$=244.4, $p<0.001$)) and a significant TF × WIDTH interaction ($F_{9,135}$=3.2, $p=0.023$)) on movement time (Figure 12). Post-hoc analysis showed significant differences between *Cst1.5* and the three other transfer functions ($p<0.001$, *Cst1.5*: 2.22 s, *OSX*: 1.86 s, *Win7*: 1.81 s, *Xorg*: 1.84 s). This shows that *Cst1.5* is more than 20% slower compared to the three default transfer functions. We did not control for clutching but according to the experimenter observation, it was infrequent for all conditions. When it occurred, it was at the beginning of the first block which was removed from the time analysis.

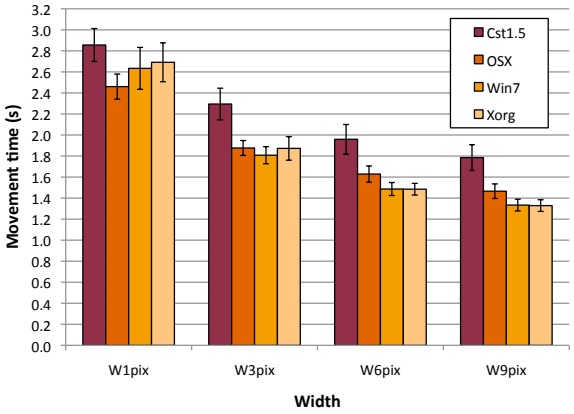


Figure 12: Mean movement time for TF and WIDTH, error bars representing 95% confidence interval.

For W_{9pix}, pairwise comparisons[11] showed significant differences ($p<0.001$) between *Cst1.5* and the three other transfer functions. We also observed significant difference ($p=0.013$) between *OSX* and *Win7* (*Cst1.5*: 1.78 s, *OSX*: 1.47 s, *Win7*: 1.33 s, *Xorg*: 1.33 s). For W_{6pix} pairwise comparisons showed significant differences ($p<0.002$) between *Cst1.5* and the other functions. *OSX* showed again significant difference ($p=0.001$) with *Win7* (*Cst1.5*: 1.96 s, *OSX*: 1.63 s, *Win7*: 1.48 s, *Xorg*: 1.48 s). For W_{3pix}, we again observed significant differences ($p<0.002$) between *Cst1.5* and the other functions. However, the significant difference between *OSX* and *Win7* disappears (*Cst1.5*: 2.29 s, *OSX*: 1.88 s, *Win7*: 1.81 s, *Xorg*: 1.87 s). For W_{1pix}, we observed significant differences ($p<0.048$) between *OSX* and *Cst1.5*, *Xorg* (*Cst1.5*: 2.85 s, *OSX*: 2.46 s, *Win7*: 2.63 s, *Xorg*: 2.69 s).

Our results show there is no single function that is best for all target widths. *Win7* and *Xorg* improve movement time by more than 9% compared to *OSX* and more than 24% compared to *Cst1.5* for widths 6 and 9 pixels. However, the difference with *OSX* disappears for 3 and 1 pixels targets. In contrast, *OSX* improves movement time by 13% compared to *Cst1.5* and 8% compared to *Xorg* for the 1 pixel target when the other functions do not show significant differences.

Qualitative Feedback

At the end of the experiment, participants were asked if they found some differences in the control of the cursor. All noticed there was a condition (*Cst1.5*) where they had to move the mouse over greater distances to reach targets and complained it was less comfortable than the other conditions. None of the participants was able to notice a difference between the three other conditions (*Win7*, *OSX* and *Xorg*).

DISCUSSION

The knowledge acquired by studying, replicating and comparing the transfer functions used by three different systems brings us to the following suggestions.

Choosing a Baseline Transfer Function

The use of a constant CD gain function as a baseline to compare with other techniques should be prohibited unless clearly justified. None of the prevalent systems uses such a function by default and it might not even be obtainable on some, like OS X. As we explained in the previous section, most participants of our experiment declared using the default settings of their system, and results from the experiment show that these default functions outperform a constant CD gain. We thus recommend to use the default transfer function of the considered system as a baseline condition.

Reporting Transfer Functions

To facilitate the replication or reproduction of pointing techniques and experiments, we recommend that researchers provide extensive details concerning the transfer function(s) they used.

If constant CD gains were used, we recommend to report them using unitless values in order to abstract them from hardware specifics. Detailing how a constant gain was achieved might also help detecting potential flaws in the methodology. As an example, it might be important to explain how remainders were handled as it remains unclear if they can affect performance. If a system function was used, the system and its particular configuration settings should be unambiguously described. A screen-shot is probably the most unambiguous way of reporting these settings. For completeness and as we have shown that some systems do not take them into account, we also recommend to report the resolution and frequency of both the input and output devices. For custom non linear functions, we recommend describing them using figures or tables with physical units mapping the device speed to the cursor speed or CD gain.

An alternative for describing transfer functions would be to use a notation based on URIs, similar to what we have started to do in `libpointing`. URIs are interesting because they allow to combine a class description, an instance description and optional parameters, e.g. `windows:vista?setting=2&epp=false`. They can be given fully expanded, with all possible parameters, or in a condensed form specifying only the ones differing from default values. If `libpointing` indeed turns into a living archive for transfer functions, we will certainly need some registry to standardize and officialize these notations.

In addition to the transfer function(s) used, the latency of the system might also be worthwhile to report as it can impact performance and might be a confounding variable. We acknowledge that measuring it is quite difficult. But some of the parameters that affect it can probably be described, such as the characteristics of the communication link between the input device and the computer, or the synchronization characteristics of the display.

Configuration Interfaces and Documentation

As we already stated, transfer functions should be defined in hardware-independent ways. Their implementation should thus be given either hardware-independent data, or the information to do the required conversions. In an ideal world, pointing device manufacturers would take full advantage of the HID specification to put all the necessary information in their device descriptions. Unfortunately, the reality is quite different... As systems are often forced to make educated

[11] Using Bonferroni correction for all post-hoc analysis.

guesses about device characteristics, we believe these should be visible and modifiable in the relevant configuration interfaces. Exposing wrongly estimated values should help raise the level of consciousness of the public about the difference between input resolution and cursor speed, for example.

Considering the different understandings of the current interfaces used for tuning the system transfer functions, even among researchers, we believe these interfaces should at least be properly documented if not completely redesigned.

CONCLUSION AND FUTURE WORK

In this paper, we presented a custom device and a toolkit that helped us characterize, replicate and compare the pointing transfer functions used on a daily basis by millions of people around the world. We showed in a controlled experiment that the default transfer functions used in Windows, OS X and Xorg outperform a constant CD gain similar to those used by most researchers. Our results also show a significant interaction between transfer function and target width suggesting that more work needs to be done to understand how these functions affect performance.

This work represents an important step in the understanding and study of pointing transfer functions. A long term goal is to improve the design of transfer functions by taking more into account the hardware characteristics (i.e. mouse vs. touchpad, desktop display vs. wall size display) and the motor capabilities of the users. This includes the study of management strategies for remainders which we hypothesize can be important for the selection of small targets. We are also interested in the study of the impact of the transfer function in relation with the task: a function performing well for pointing could degrade performance in other tasks like drawing, steering or executing command gestures, for example.

In the short term, we plan to study the interaction of the default transfer functions with pointing facilitation techniques manipulating CD gain, e.g. [5, 10, 23]. To our knowledge, these technique were only implemented on top of a constant CD gain and were also only evaluated against constant CD gains. We also plan to investigate the use of indirect mappings on multitouch interfaces. On this topic, Buxton recently said: "*one of the things that I see most neglected is any consideration of when to use relative vs absolute control and varying, including when and how to effectively and dynamically switch from one to the other, and when and how to dynamically adjust C:D ratio*" [6]. We could not agree more.

ACKNOWLEDGMENTS

We thank Damien Marchal and Mark Cranness for their help in the understanding of transfer functions and their contribution to `libpointing`.

REFERENCES

1. Pointer ballistics for Windows XP. Archived white paper, Windows Hardware Developer Center, Oct. 2002.
2. T. Asano, E. Sharlin, Y. Kitamura, K. Takashima, and F. Kishino. Predictive interaction using the Delphian Desktop. *Proceedings of UIST'05*, 133–141. ACM, 2005.
3. M. Bergman, T. Peurach, T. Schmidt, S. McGowan, J. Crowe, R. Dezmelyk, R. Zimmermann, M. Van Flandern, B. Nathan, M. Davis, and J. Rayhawk. Device class definition for human interface devices (HID). Version 1.11, USB Implementers' Forum, June 2001.
4. T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): generic syntax. RFC 3986, IETF, Jan. 2005.
5. R. Blanch, Y. Guiard, and M. Beaudouin-Lafon. Semantic pointing: improving target acquisition with control-display ratio adaptation. *Proceedings of CHI'04*, 519–526. ACM, 2004.
6. W. Buxton, M. Billinghurst, Y. Guiard, A. Sellen, and S. Zhai. *Human input to computer systems: theories, techniques and technology*. 2011. Working draft, `http://www.billbuxton.com/inputManuscript.html`.
7. G. Casiez and D. Vogel. The effect of spring stiffness and control gain with an elastic rate control pointing device. *Proceeding of CHI'08*, 1709–1718. ACM, 2008.
8. G. Casiez, D. Vogel, R. Balakrishnan, and A. Cockburn. The impact of control-display gain on user performance in pointing tasks. *Human-Computer Interaction*, 23(3):215–250, 2008.
9. O. Chapuis, J.-B. Labrune, and E. Pietriga. DynaSpot: speed-dependent area cursor. *Proceedings of CHI'09*, 1391–1400. ACM, 2009.
10. A. Cockburn and A. Firth. Improving the acquisition of small targets. *Proceedings of HCI'03*, 77–80. BCS, 2003.
11. E. D. Graham. Virtual pointing on a computer display: non-linear control-display mappings. *Proceedings of GI'96*, 39–46. Canadian Information Processing Society, 1996.
12. T. Grossman and R. Balakrishnan. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. *Proceedings of CHI'05*, 281–290. ACM, 2005.
13. K. Hinckley. Input technologies and techniques. A. Sears and J. A. Jacko, editors, *Human Computer Interaction Handbook (2nd edition)*. CRC Press, Sept. 2007.
14. H. D. Jellinek and S. K. Card. Powermice and user performance. *Proceedings of CHI'90*, 213–220. ACM, 1990.
15. M. Kobayashi and T. Igarashi. Ninja cursors: using multiple cursors to assist target acquisition on large screens. *Proceeding of CHI'08*, 949–958. ACM, 2008.
16. I. S. MacKenzie. Input devices and interaction techniques for advanced computing. W. Barfield and T. A. F. III, editors, *Virtual environments and advanced interface design*, 437–470. 1995.
17. I. S. MacKenzie and P. Isokoski. Fitts' throughput and the speed-accuracy tradeoff. *Proceeding of CHI'08*, 1633–1636. ACM, 2008.
18. R. L. Mandryk and C. Gutwin. Perceptibility and utility of sticky targets. *Proceedings of graphics interface 2008*, GI '08, 65–72. Canadian Information Processing Society, 2008.
19. D. E. Meyer, R. A. Abrams, S. Kornblum, C. E. Wright, and J. E. K. Smith. Optimality in human motor performance: Ideal control of rapid aimed movements. *Psychological Review*, 95(3):340–370, 1988.
20. B. Moggridge. *Designing interactions*. The MIT Press, Oct. 2006.
21. U. Tränkle and D. Deutschmann. Factors influencing speed and precision of cursor positioning using a mouse. *Ergonomics*, 34(2):161–174, 1991.
22. J. O. Wobbrock, J. Fogarty, S.-Y. S. Liu, S. Kimuro, and S. Harada. The Angle Mouse: target-agnostic dynamic gain adjustment based on angular deviation. *Proceedings of CHI'09*, 1401–1410. ACM, 2009.
23. A. Worden, N. Walker, K. Bharat, and S. Hudson. Making computers easier for older adults to use: area cursors and sticky icons. *Proceedings of CHI'97*, 266–271. ACM, 1997.

Exposing and Understanding Scrolling Transfer Functions

Philip Quinn[1] *Andy Cockburn*[1] *Géry Casiez*[2,3,4] *Nicolas Roussel*[3] *Carl Gutwin*[5]

[1]University of Canterbury
Christchurch, New Zealand
philip.quinn@canterbury.ac.nz
andy@cosc.canterbury.ac.nz

[2]LIFL, [3]INRIA Lille &
[4]University of Lille
Villeneuve d'Ascq, France
gery.casiez@lifl.fr
nicolas.roussel@inria.fr

[5]University of Saskatchewan
Saskatoon, Canada
gutwin@cs.usask.ca

ABSTRACT
Scrolling is controlled through many forms of input devices, such as mouse wheels, trackpad gestures, arrow keys, and joysticks. Performance with these devices can be adjusted by introducing variable transfer functions to alter the range of expressible speed, precision, and sensitivity. However, existing transfer functions are typically "black boxes" bundled into proprietary operating systems and drivers. This presents three problems for researchers: (1) a lack of knowledge about the current state of the field; (2) a difficulty in replicating research that uses scrolling devices; and (3) a potential experimental confound when evaluating scrolling devices and techniques. These three problems are caused by gaps in researchers' knowledge about what device and movement factors are important for scrolling transfer functions, and about how existing devices and drivers use these factors. We fill these knowledge gaps with a framework of transfer function factors for scrolling, and a method for analysing proprietary transfer functions—demonstrating how state of the art commercial devices accommodate some of the human control phenomena observed in prior studies.

ACM Classification: H.5.2 [Information interfaces and presentation]: User interfaces – Input devices and strategies.

Keywords: Control-display gain, scrolling, scroll acceleration, transfer functions.

General terms: Documentation, Design, Measurement.

INTRODUCTION
Scrolling is an essential task in modern computing, and scrolling devices such as mouse wheels and trackpad gestures are ubiquitous. A fundamental element of scroll control that all techniques must address is the *transfer function* that maps the user's actions with the input device (for example, degrees of rotation, millimetres of displacement, or newtons of force) into scrolling movement of the display (typically either pixels, lines, or pages). However, there has been surprisingly little public research on scrolling transfer functions. Notable exceptions include Hinckley et al. [15] and Cockburn et al. [11], but even these studies are ambiguous about the exact functions used or tested against—for example, Hinckley et al. stated "We tested the device using the manufacturer's default settings", but precisely determining the corresponding transfer function is now near impossible; Cockburn et al. explicitly acknowledged the need for further research to understand the role that the system transfer function may have played in their experiment.

The poor understanding of scrolling transfer functions creates several problems for researchers. First, existing methods are unknown because they are embedded in 'black box' driver code, making it difficult for researchers understand the cause of performance differences between devices, or to iteratively improve on the state of the art. Second, replication of scrolling studies is frustrated by ambiguities in experimental settings—researchers lack the tools to examine, report, and replicate transfer functions. Third, researchers may inadvertently introduce confounds into experiments stemming from unknown interactions between particular transfer functions and their experimental treatment.

Casiez and Roussel [8] recently observed similar problems of "bricolage" in research treatment of pointing transfer functions. To address the problem, they developed an electronic device called *EchoMouse* to probe and inspect transfer functions. They also created a software library called *libpointing* that implemented these functions for experimental replication. They used these components to simulate human mouse control at a variety of physical input movement speeds, and to inspect the resultant system response.

Although *EchoMouse* and *libpointing* provide critical hints on how to examine scrolling transfer functions, the mapping from input actions to output effects is more complex for scrolling. While pointing transfer functions attend to two parameters (mouse velocity and user setting), scrolling functions are likely to attend to many more. Multiple parameters are necessary (or advisable) because of the paucity of scrolling input mechanics. For pointing, a modern mouse will register thousands of points per inch, and it can be moved across a two-dimensional area of several inches without clutching (physically disengaging input control in or-

UIST'12, October 7–10, 2012, Cambridge, MA, USA.

der to reposition a limb to repeat the action); in contrast, a typical scroll wheel can only be moved through five or six detents/notches across ~40° of one-dimensional rotation between clutching actions; and the muscle groups used to control wheel rotation (finger extensors and contractors) are likely to induce different control capabilities across scroll directions. Scrolling transfer functions, therefore, are likely to attend to input parameters that include the rate of device movement, time between clutched repetitions, scroll direction, and more. But whether they do this, and how they do it, is currently unknown.

These physical and operational characteristics of devices, and the human capabilities when operating them, have clear implications for the design of scrolling transfer functions. To help understand these issues, the following section presents a framework of the factors influencing scroll control. We then reverse engineer the scrolling transfer functions in state of the art commercial scroll drivers, and confirm that some drivers attend to many input parameters, while others are based purely on the velocity of input control.

FACTORS INFLUENCING SCROLL CONTROL

There has been extensive prior work on taxonomies that aid in understanding the design space of input devices, which we draw on to organise the physical characteristics of scrolling devices. Buxton's [4] early taxonomy organised devices by the physical properties—position, motion, or pressure—and by the number of dimensions along those properties that are sensed. Mackinlay et al. [20] and Card et al. [6, 7] expanded this into a *morphological* analysis, placing devices as points in a parametrically described design space that included the eight combinations of linear/rotary, absolute/relative, and position/force across six linear and rotational dimensions. They also composed chains of connections between the physical parameters and the semantics of an application. Buxton [5] and Hinckley and Sinclair [16] expanded this classification to include devices that operate by touch (rather than a mechanical control), and Lipscomb and Pique [19] added several dimensions of physical device characteristics (including the behaviour of the movement axes, bounds of movement, and self-zeroing behaviour).

These taxonomies can be used to classify the physical sensing properties of scrolling devices; for example, that mouse wheels are single-axis rotary controls that sense discretised changes in rotation, or that trackpads sense absolute one or two-dimensional position. They can also classify the features of the physical controls used to input these properties; for example, mouse wheels can rotate in rigid, discrete detents, or the detents can be soft and the wheel can be inertial (supplying sensor data without active user interaction). These design choices promote different methods of interacting with the device in different scrolling scenarios (for instance, rapid clutching on a discrete wheel vs. flicking and inertial one), and consequently influence the range and type of inputs that are likely to be received. However, the input parameters that are derived from these different methods of interaction are not captured by the above taxonomies.

This section presents a framework for the factors influencing scrolling behaviour and prior scrolling research. The framework is organised across considerations of input parameters, a review of the system-oriented view of scrolling, and prior studies of scrolling gain.

Input Parameters

While the reviewed taxonomies organise the physical characteristics of scrolling devices, they do not focus on how features of the manipulation can be translated into task-specific semantics.

Despite the apparent simplicity of scrolling as unidimensional translation, there exists a broad variety of devices to support it, each of which may use a multitude of input parameters for inferring the user's scrolling intention. For example, a mouse wheel senses discretised rotary motion, but the user's intention may be inferred from any combination of the following: (1) the degrees of rotation; (2) the speed of rotation; (3) the rate of change in the speed of rotation; (4) the duration of interaction; (5) the direction of interaction; and (6) the period of interaction.

In general, actions performed on a device need to be mapped from a physical manipulation to an interface command. In doing so, several *input parameters* can be considered, increasing expressivity. For example, keyboard arrow buttons are a one-dimensional discrete control, but their use may be interpreted through continuous parameters such as duration of activation or the rate of repetition. These additional input channels can be classified into three types: measures of *instantaneous action*, measures of *action duration*, and *cumulative/relative measures*, described below.

Instantaneous measures. Measures of instantaneous action are the physical properties that are sensed by a device or their derivatives from samples over time. For example, spinning a detented mouse wheel produces discrete events of rotational movement; however, sampling several events produces new input parameters of angular velocity, acceleration, and higher-order derivatives.

These measures may alter the interpretation or mapping of the original property. For example, increasing mouse wheel velocity or acceleration may be used to increase the magnitude of scrolling events generated from the input of each wheel event, or may be used as a signal to switch between scrolling modes (such as between line and page-scrolling).

Action duration. The duration that an input is maintained (or is absent) can also serve as an input to a transfer function. For example, if a key or button is held down for more than a certain duration, it may start issuing repeating events at an increasing rate. Similarly, if scrolling velocity is maintained above a certain level, then gain might increase with the assumption that the user wants to travel a long distance.

Cumulative and relative inputs. Input measures may also have a memory of prior actions to determine the input effects. For example, Hinckley and Cutrell [17] described a scrolling transfer function that cumulatively adds gain across rapidly repeated wheel rotations in the same direction; the cumulative effect is cancelled if the user pauses too long or reverses direction. Similarly, rate-based scrolling controls scroll ve-

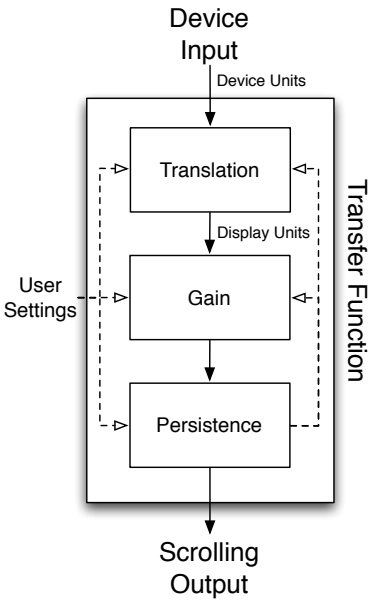


Figure 1: Depiction of the conceptual transformations occurring in a scrolling transfer function.

locity with the relative position of the input device with respect to an anchor point set at the action's initiation.

The System's Perspective

Figure 1 depicts the conceptual transformations that form a transfer function's behaviour in converting human action at the device into resultant display modification in scrolling. Not all of these transformations may be present in any transfer function: some may be absent, some may be combined, and some may be applied multiple times in different components (for example, gain applied by a driver and again by a UI toolkit or application).

Translation converts the device's physically registered events (degrees of rotation, newtons of force, millimetres of displacement, etc.) into units that are comprehensible to the system, such as pixels, lines, or pages. Users may be able to adjust this translation, either through controls on the device, or via a user interface on the system.

A *gain* function may then amplify or attenuate the control signal: for example, to allow slow precise control when the device is manipulated slowly, as well as accelerated scrolling when it is manipulated more aggressively. When gain is supported, users are commonly able to configure its setting.

Finally, a *persistence* component allows for a history of input and calculated parameters (such as input velocity and acceleration) to be preserved, or to allow for effects that are applied across time (such as cumulative effects, inertia, and simulated friction). Data from the persistence component can be used as input into the translation (e.g., allowing a switch from pixel to line scrolling if manipulation is continued for a threshold time), or into the gain (e.g., applying cumulative gain across rapidly repeated scroll wheel clutches). Finally, the user may be able to configure parameters of the persistence component: for example, altering the degree of inertia or friction.

Most scrolling devices conform to the *Human Interface Devices* (HID) class of the USB standard [3]. The HID class provides a common, vendor-independent method for communicating interaction data from common types of devices to a computer system. Devices that implement the appropriate HID usage tables (for example, mice, keyboards, phones, and digitisers [2]) can operate without vendor-specific drivers, allowing a high degree of device/application interoperability.

HID devices report extensive descriptions of their sensing and reporting characteristics to the operating system/driver via HID *descriptors*. Of particular interest to scrolling are the wheel report range (typically 8-bit values interpreted to be between -127 and $+127$, but any size or range may be chosen by a manufacturer), the characteristics of the report (absolute/relative, wrapping/non-wrapping, linear/non-linear, etc.), and the rate at which reports are sent. The resolution and units of these reports can also be specified, but none of the devices we examined did so. System-specific extensions may also exist. For example, starting with Windows Vista, Microsoft allows devices to support horizontal scrolling and high-resolution scrolling by reporting a resolution multiplier and responding to queries from the operating system to configure it [21].

While most of the scrolling devices we examined supplied a 'Wheel' HID usage, notable exceptions to this were trackpads that used configurable gestures to enable a scrolling mode (for example, Apple's laptop trackpads and Magic Trackpad[1]). These devices transmit information about the gestures through proprietary data fields in the HID report, and rely upon manufacturer-specific drivers to interpret them and report scroll events to the operating system.

Despite the vendor-independent nature of the HID specification, drivers from device manufacturers may still play a significant role in defining the device's scrolling behaviour by attending to the input parameters discussed previously (and may be necessary to make exotic hardware that has not been anticipated by the HID usage tables useful at all—for example, trackpad scrolling gestures). Some of these features may include scrolling horizontally, independent transfer functions for each direction to match human capabilities, configurable buttons or gestures to augment or change scrolling behaviour/resolution, or different scrolling modes for each application. For example, the Logitech MX Revolution[2] features a weighted, low-fiction wheel that can have ratchets automatically engaged by the drivers as a user's scrolling behaviour changes.

Prior Studies of Scrolling Gain

In an early scroll wheel description, Gillick et al. [12, 13] described a potential transfer function that treats initial wheel events as line-scrolling, and advanced to page-scrolling once

[1] `http://apple.com/magictrackpad/`
[2] `http://logitech.com/428/130`

events passed a certain threshold-rate. A similar technique was recently described by Montalcini [22].

Hinckley et al. [15] describe a scroll transfer function that operates on the calculated interval between events received from the scrolling device/driver:

$$\Delta y = K_1(1 + K_2\Delta t)^{\alpha}$$

Where K_1, K_2, and α are constants, Δt is the interval between subsequent events, and Δy is the resulting scale factor to apply to reported magnitudes. Hinckley et al. evaluated their function when applied to a driver reporting three lines of scrolling per physical detent, one line per detent, a "standard" three lines per detent without application of the function, and an IBM ScrollPoint (isometric joystick; the configuration parameters of which are not reported) in a repeated tapping task. They found comparable or significantly better performance when using the accelerated functions.

Two further enhancements are detailed in related patents [17, and related continuity data]. One is a feature that detects changes in the scroll direction and temporarily inhibits the application of Δy—aiming to prevent amplification of overshooting errors. The other identifies rapidly repeated clutching of the wheel (in an attempt to travel a long distance) and applies cumulative gain according to the number of successive wheel flicks (N_{flicks}):

$$Z_{\text{scroll}} = \Delta y \cdot G_0 \cdot G_F \cdot N_{\text{flicks}}$$

Where G_0 is the baseline number of lines to scroll per detent, G_F is the additional amount of gain to apply per flick, and Z_{scroll} is the number of lines to scroll.

Kobayashi and Igarashi [18] explored the use of the cursor position as an input parameter to a dynamic transfer function. Their *MoreWheel* technique combines absolute and relative scrolling into the scroll wheel: dragging the mouse with the wheel depressed simulates grabbing the scroll thumb and enables absolute scrolling, while spinning the wheel produces either line or page-scrolling depending on the position of the cursor within the window (for example, line scrolling when the cursor is in the middle of the window, transitioning to page scrolling near the top or bottom edges).

Cockburn et al. [11] describe a method where two transfer functions are transitioned between based on the velocity of the scroll input to enable slow scrolling at a rate akin to Hinckley et al. [15], and rapid scrolling based on a function that utilises information about the length of the document being scrolled. The velocity of incoming scroll events is calculated, smoothed, and used to determine the proportion of each transfer function to apply:

$$g = \left[p \cdot \left(k_s - k_s\alpha^{-v}\right)\right] + \left[(1-p) \cdot \left(k_f \frac{\text{document length}}{\text{viewport size}}\right)\right]$$

Where k_s, k_f, and α are constants, v is the reported input velocity, p is the proportion of the "slow" function to apply (determined by examining the relationship of v to the user's maximum velocity), and g is the resulting scale factor to apply to reported magnitudes. An evaluation of this function using two wheel-based devices and an isometric joystick against the "additive flicking" technique of Hinckley and Cutrell [17] found it to perform significantly faster for long documents.

The above studies have explicitly examined the impact of scrolling transfer functions, but there are many more studies that have examined scrolling with imprecise and non-replicable transfer functions. This is not a criticism of the studies, but rather an unfortunate state of affairs—there has been a lack of tools supporting rigour around scrolling transfer functions. Some studies evaluate scrolling systems without mentioning the gain levels or transfer function used [e.g., 10, 14, 23]; some explicitly state the absence of acceleration, but do not state the constant translation used [e.g., 9]; and others rely on the default settings without stipulating what behaviour results [e.g., 1, 24].

REVERSE ENGINEERING CURRENT SCROLLING TRANSFER FUNCTIONS

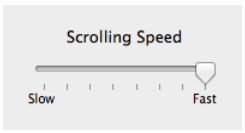
To examine commercial scrolling transfer functions, we used a modified version of the *EchoMouse* [8]: a programmable microcontroller that allowed us to transmit scrolling events to the system as if they originated from an ordinary scrolling device, but in a highly controlled and systematic fashion. To inspect the functions inside a particular device driver, the EchoMouse was modified to present itself as a compatible device from the appropriate manufacturer by manipulating its reported HID vendor and product identifiers. Therefore, by triggering the EchoMouse to emit scrolling events in a pre-defined pattern, and inspecting the scrolling events received by a user application, we can examine how the original events have been transformed.

We tested the scrolling drivers found in Apple Mac OS X 10.7.3, Microsoft Windows 7 (SP1), Microsoft IntelliPoint (8.20.468 on Windows), Logitech SetPoint (driver 5.33.14 on Windows), and Logitech Control Center (3.5.1-23 on Mac OS X)—these represent some of the most popular operating systems and device manufacturers. With each driver, we impersonated the characteristics of several representative devices that they supported to gather data (testing low and high-resolution devices, although no differences between devices was found). The Mac OS X and Windows 7 drivers represent generic drivers that are used by the operating system when no vendor-specific drivers are available. We did not test the Mac OS X version of Microsoft's IntelliPoint driver as it conflicted with our EchoMouse control software, nor did we test an X11 environment as pilot testing showed that it (xorg 1.11.4-2; Fedora 16) does not implement scroll acceleration (the interpretation of each count is left to individual UI toolkits or applications).

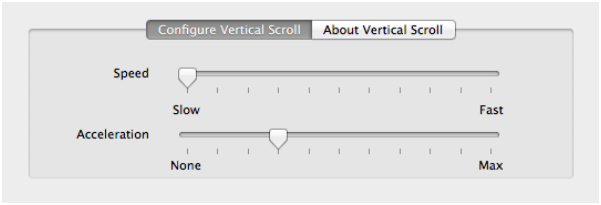
This section presents an analysis of the publicly available information about these functions, followed by the testing methodology that we used to gather data about their embedded transfer functions.

Analysis of Existing Transfer Functions

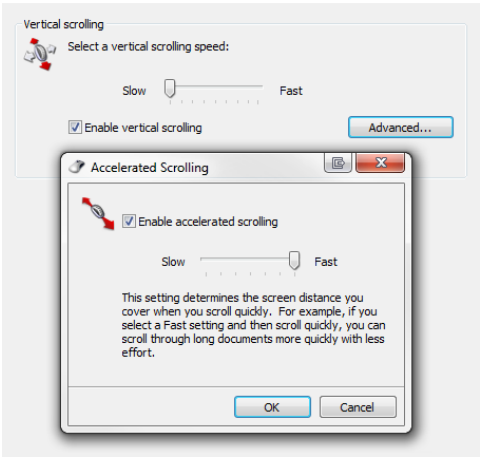
Apple Mac OS X. Apple release several of their low-level input processing frameworks under an open source licence, including those for HID devices. Within the `IO-`



(a) Mac OS X.



(c) Logitech Control Center (Mac OS X).



(b) Microsoft IntelliPoint (Windows).

Figure 2: User interfaces for configuring scroll control.

`HIDFamily` framework[3] (version 368.20, corresponding to Mac OS X 10.7.3 was examined for this study), the `IOHIDFamily/IOHIDPointing.cpp` and `IOHIDSystem/IOHIPointing.cpp` source files contain much of the code pertinent to pointing (and by extension, scrolling) devices.

Drivers can supply an encoded table of acceleration lines (slopes m and intercepts b) to be applied at different input magnitudes; these lines are scaled based on the user's scrolling speed setting (detailed below). When a scroll event is received with magnitude y, it is added to a smoothing window of the last eight events to avoid rapid changes in gain. The average time delta between events in the smoothing window $\overline{\Delta t}$ and average unaccelerated magnitudes $\overline{y}$ is then used to calculate a threshold:

$$l = \left[(K_a \cdot \overline{\Delta t}^2) - (K_b \cdot \overline{\Delta t}) + K_c \right] \cdot r \cdot \overline{y}$$

Where K_a, K_b, and K_c are constants, and r is an input rate multiplier (1 by default).[4] An acceleration line appropriate for an input magnitude larger than l is selected from the table, and applied:

$$y' = y \cdot \frac{b + (l \cdot m)}{|y|}$$

The control exposed to users for this function is a slider in the system preferences to manipulate "Scrolling Speed" with eight intervals from "Slow" to "Fast" (shown in Figure 2(a)), and a corresponding API (`IOHID[Get/Set]ScrollAcceleration()`, where the notches on the slider are mapped to the API values $\{0, 0.12, 0.31, 0.5, 0.69, 0.88, 1, 1.7\}$). An interesting feature of this control is that a negative value (which can only be selected via the API) completely disables scroll acceleration for generic devices, or engages a page-scrolling mode for an Apple trackpad.

[3] `http://opensource.apple.com/source/IOHIDFamily/`

[4] Code also exists for scaling these functions with the screen resolution, but the calculations are currently fixed.

Drivers can report scrolling events in units of either lines or pixels, with an automatic conversion by the system between them of 10 pixels per line. We report output in pixels to match the reports given by the system to user applications.

It should be stressed that the behaviour described above is the *default* that is applied should no better drivers match a connected device. Manufacturers are free to use and adjust the described behaviour in part, or as a whole. For example, Apple's closed-source driver for their laptop trackpads supplies a scrolling acceleration table that can be decoded with the source code provided, but that alone does not guarantee that it will be applied in the manner described above.

Microsoft Windows. Windows provides scrolling information to applications via `WM_MOUSEWHEEL` messages with a parameter indicating the distance the wheel has been rotated in units of `WHEEL_DELTA`. These values are intended to be scaled by the user setting `SPI_GETWHEELSCROLLLINES`, indicating how many lines to scroll per unit of `WHEEL_DELTA` (or, a special value indicating that each unit should be interpreted as a page scroll). The interpretation of "lines" or "pages" is left to the application receiving the message. On all current systems, `WHEEL_DELTA` is set to 120, which allows high-resolution devices to indicate scrolling of fractional lines.[5] We report the output from drivers running under Windows in "lines" (i.e. units of `WHEEL_DELTA`).

Microsoft IntelliPoint. Microsoft's IntelliPoint drivers for their branded devices presents two controls for the user to configure the scrolling transfer function (shown in Figure 2(b)). The first controls `SPI_GETWHEELSCROLLLINES` in the range [1, 40]; the second is a seven-interval slider to control accelerated scrolling from "slow" to "fast" (with an option to disable it entirely).

[5] `http://msdn.microsoft.com/library/ms997498`

Logitech. Logitech produces two driver packages for their devices: *SetPoint* for Windows, and *Control Center* for Mac OS X. The configuration options and range of supported devices differs between these packages; in particular, *SetPoint* provides options to configure `SPI_GETWHEELSCROLLLINES` for line or page scrolling but with no options for acceleration, while *Control Center* allows customisation of the scrolling "speed" (from "slow" to "fast") and "acceleration" (from "none" to "max"), as shown in Figure 2(c) (both of these are continuous sliders, but were tested at the marked intervals).

Testing Methodology

Scrolling events from the EchoMouse have values in the range -127 to $+127$. We observed that reports from devices were typically either -1 (scroll down) or $+1$ (scroll up) with wider values used when the physical manipulation of the device exceeded its HID input report rate (typically 100~125Hz, but high-end devices may report at rates up to 1000Hz); we emulated this behaviour.

A potential issue when impersonating other devices is matching their input resolution. While the USB HID specification allows devices to specify the resolution and physical units of their input, none of the device we tested did so. For instance, a Microsoft Wheel Mouse Optical[6] sends 18 scroll counts per complete revolution of its wheel (20° per detent), while a Logitech MX500[7] sends 24 (15° per detent), but their reports are indistinguishable to a generic driver (similar issues exist for trackpads that transmit events corresponding to millimetres of displacement, or other types of physical control). Because we tested a range of devices with different input resolutions, we report our input velocity in "counts" per unit of time, where one count corresponds to one scroll event of magnitude -1 or $+1$ (issues surrounding device resolution are discussed later).

As we are interested in the various input parameters that transfer functions may attend to (and not only how they operate under levels of velocity), we performed four mechanised tests of each possible configuration of driver and device:

- *Constant velocity:* emulating a constant speed of device operation for five seconds, and measuring the resultant output scrolling velocity as an average over that period.
- *Maintained velocity:* emulating a constant speed of device operation for five seconds, and measuring the resultant output scrolling velocity for each event.
- *Clutching:* we emulated clutching actions, manipulating the speed of device operation, the duration of clutches, and the time between successive clutches.
- *Direction changes:* we emulated direction changes (alternating between scrolling up and scrolling down) while maintaining a constant speed of device operation.

These tests were repeated for each configuration option presented to users (as described above), and across a range of possible user control input rates. Custom software monitored the system's response using the low-level event reporting APIs provided by each operating system (free from potential manipulation by higher-level frameworks or toolkits).

[6] `http://microsoft.com/hardware/en-nz/d/wheel-mouse-optical`
[7] `http://logitech.com/428/910`

	Velocity	Direction	Duration	Clutching
Apple Mac OS X	●	◐	○	◐
Microsoft Windows 7	○	○	○	○
Microsoft IntelliPoint	●	●	○	●
Logitech SetPoint	○	○	○	○
Logitech Control Center	●	◐	●	○

Table 1: Summary of the tested drivers' attendance to tested input features—○: no attendance, ●: attendance, ◐: partial attendance (details in text).

RESULTS AND ANALYSIS

The results of our analyses are summarised in Table 1. We found that neither Microsoft Windows 7 (with a generic device), nor Logitech's SetPoint drivers provide any scroll acceleration (the gain is always constant). Due to the large number of configurations tested and the many commonalities discovered between them, the following subsections present a survey of the most salient and interesting behaviour characteristics and parameters attended to (a complete spreadsheet of the acceleration tables collected is also available[8]). Following the main results, we summarise device-specific issues in the analysis.

Gain with Respect to Velocity

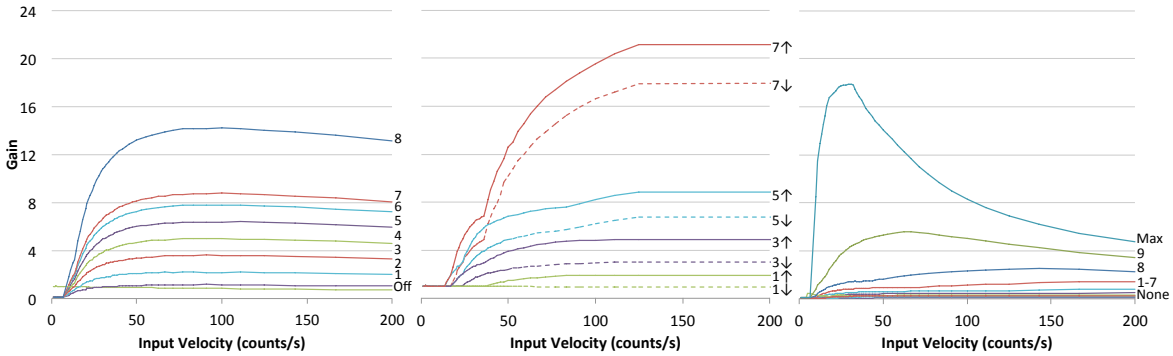
How the different systems alter gain across input velocity is shown in Figures 3(a), (b), and (c) for Mac OS X, Microsoft IntelliPoint, and Logitech Control Center, respectively. The multiple lines in each figure show different levels of user setting for scrolling "speed" (Mac OS X, Figure 2(a)) or scroll "acceleration" (Microsoft IntelliPoint and Logitech Control Center, Figures 2(b) and (c)). The solid and dashed lines in Figure 3(b) differentiate between scrolling direction (up and down); the other drivers respond to both directions equally.

The maximum gain scale factors attainable range from ~14 with Mac OS X, to ~18 with Logitech, and ~21 with Microsoft IntelliPoint. The key differences between the three curve shapes is that Logitech's curves for high acceleration show a dramatic drop in gain after peaking at 18× at ~28 counts/s. This is a result of a falling (but still positive) gradient in the output velocity curve; however the rationale for this design choice is unknown. Both Mac OS X and Logitech allow input to be attenuated (with a gain of less than 1) at low input speeds, increasing the expressivity of devices with poor input resolution.

Direction as an Input

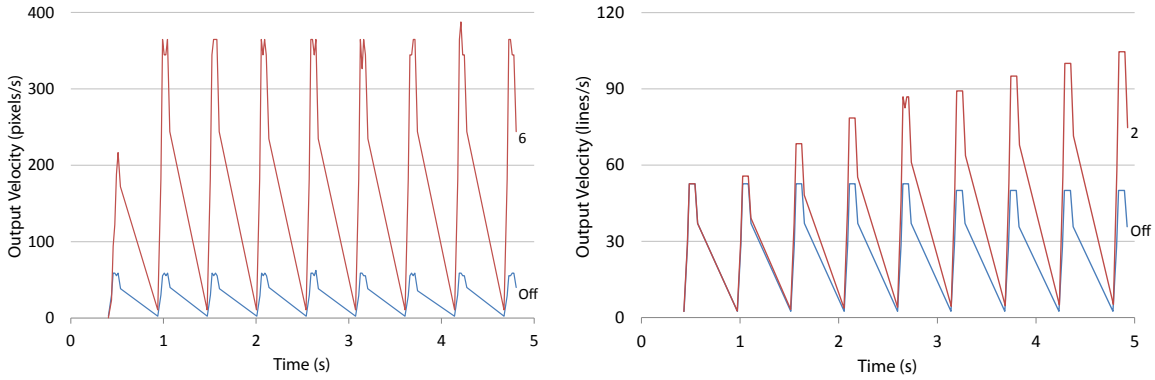
Figure 3(b) shows that Microsoft IntelliPoint drivers apply differing gain levels across each scrolling direction. This is probably applied to compensate for the differing maximum input velocities attainable in the two directions (Cockburn

[8] `http://cortex.p.gen.nz/research/scrolling/`



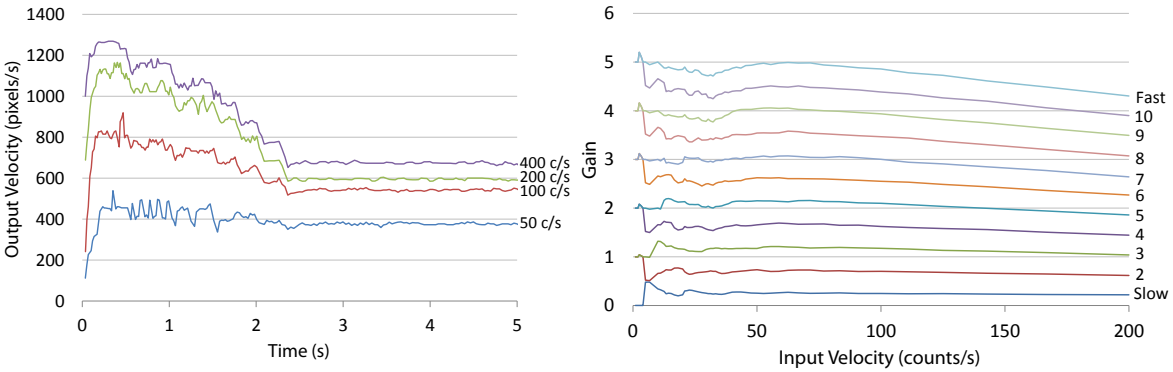
(a) Mac OS X (Generic Device). (b) Microsoft IntelliPoint: up (solid) and down (dashed). (c) Logitech Control Center.

Figure 3: Gain scale factors across input velocity (counts per second) with Mac OS X, Microsoft IntelliPoint (under Windows 7), and Logitech drivers under Mac OS X. Gain is measured as the level of amplification in the system's base unit (pixels per count for Mac OS X and Logitech; lines per count for Microsoft IntelliPoint), and is plotted at varying levels of each driver's respective UI sliders for acceleration.



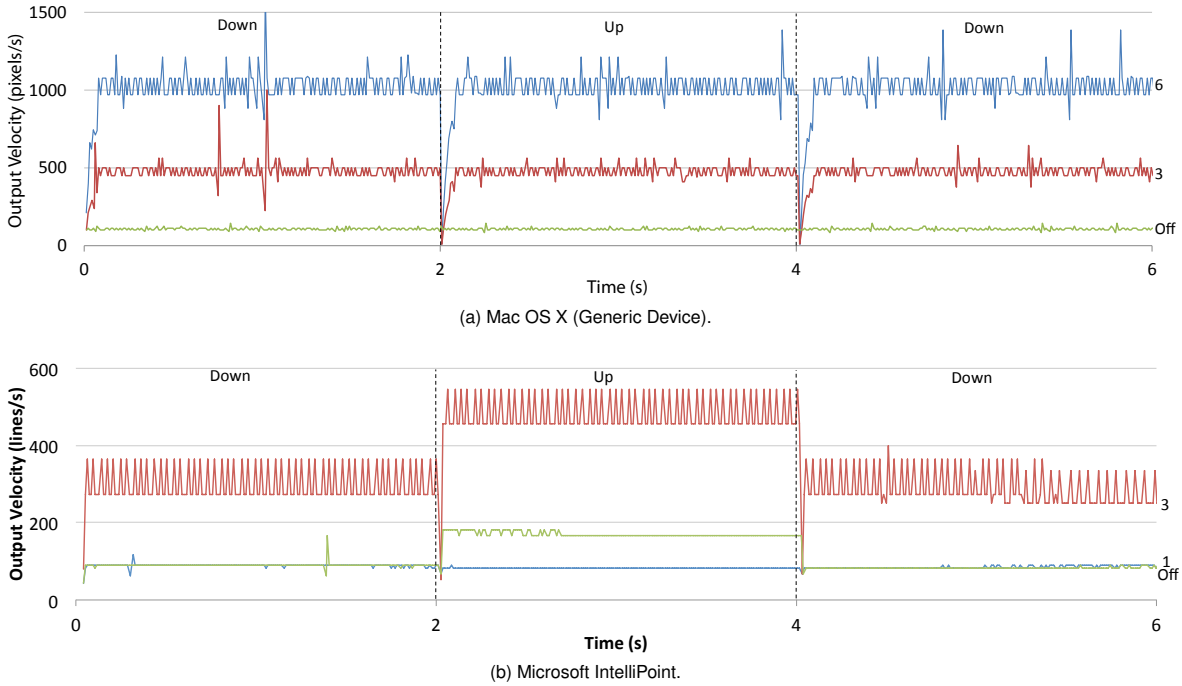
(a) Mac OS X (Generic Device). (b) Microsoft IntelliPoint.

Figure 4: Output velocity response to repeated clutching.



(a) Output velocity over time as a constant input velocity is maintained. (b) Gain across input velocity at levels of the "speed" slider shown in Figure 2(c).

Figure 5: Logitech's Control Center: Attendance to duration and the control of the "speed" slider.



(a) Mac OS X (Generic Device).

(b) Microsoft IntelliPoint.

Figure 6: Output velocity over time as a constant velocity is maintained but input direction is switched at the dashed lines (2 and 4s). Three curves—two levels of acceleration and one with acceleration disabled—are shown.

et al. [11] showed marked differences between maximum scroll wheel rotation speeds upwards and downwards).

Mac OS X and Logitech drivers do not vary gain across direction, but directional changes do momentarily "reset" gain, as shown for Mac OS X in Figure 6(a). This is due to a reset of the smoothing window upon direction change, resulting in reduced gain until the window is re-filled. The same task with Microsoft's IntelliPoint drivers is shown in Figure 6(b), where we observed a very brief drop in gain and the application of different levels of gain for each direction.

Clutching as an Input

Figure 4 shows how Mac OS X and Microsoft IntelliPoint drivers respond to clutching of the scroll wheel (Logitech's response is not shown as it is similar to Mac OS X). The main finding here is that IntelliPoint cumulatively adds gain across successive clutches when acceleration is turned on (similar to the technique described by Hinckley and Cutrell [17]). Mac OS X (Figure 4(a)) and Logitech drivers do not vary their response (the reduced gain for the first impulse is due to an empty smoothing window), however, comments in the Mac OS X driver source code[9] indicate the timeout value for resetting the smoothing window was chosen specifically to avoid doing so between clutches.

Duration as an Input

Figure 5(a) shows that Logitech's driver attends to scroll duration while Mac OS X and Microsoft IntelliPoint do not (not shown). When stimulated with a constant velocity scroll rate, Logitech's output velocity diminishes over approximately 2.5s. This time-based fall-off is particularly marked at high input velocities, and it is therefore likely designed to enhance user performance with Logitech's free-spinning inertial scroll wheels that readily allow high input speeds.

Other Features

It is interesting and potentially important that the speed and acceleration parameters in Logitech's Control Center interact with one another, and that setting acceleration to 'None' does not disable acceleration: Figure 5(b) shows the gain observed at various settings of the speed slider with the acceleration slider set to "None". The noise at low speeds is puzzling, and the lack of constant gain suggests that Logitech Control Center should be used with caution in research experiments.

DISCUSSION

We have presented a framework for understanding the factors influencing the transformation of human action with scrolling devices (particularly scroll wheels) into resultant scrolling output. We have also reverse engineered the scrolling transfer functions from the drivers of popular manufacturers, with results demonstrating substantial variation in both the factors attended to and the manner in which they do so. Key observations include the fact that Microsoft's IntelliPoint drivers apply different levels of gain to different scrolling directions (presumably to accommodate differences in human mechanics), that they also apply cumulative gain across repeatedly performing clutching actions,

[9] `IOHIDSystem/IOHIPointing.cpp`, lines 555–559

and that Logitech's drivers on Mac OS X apply variable gain even when user settings stipulate that acceleration should be turned off.

The framework and findings have several implications for research that aims to develop new transfer functions or use scrolling devices in experimental conditions, and there are many avenues for further work.

Facilitating Rigour in Scrolling Studies

Scrolling researchers are typically interested in either new input devices [e.g., 24], transfer functions [e.g., 11, 15], or new interactive techniques [e.g., 18]. Scrolling devices are also used in experiments where scrolling is not the focus of an investigation, but as an interaction tool.

In all cases, comparative evaluations are normally conducted to measure performance over the state of the art, and the choices made in the implementation and administration of experimental treatments must be made with the awareness and knowledge of the underlying transfer functions. As experimental software typically operates on-top of existing drivers (rather than replacing them), understanding the interaction between the transfer function of the driver and that of the experimental condition is critical in answering the question of whether the treatment is causing any observed difference, or whether it may be attributed to the interacting transfer functions.

What should researchers do to maximise rigour and facilitate replication? We make three recommendations. First, a constant level of gain (i.e. scroll acceleration is disabled) should be used as a baseline in experiments where gain is not intended to be a factor. Given the complex nature of the gain functions observed in our results, their variation across drivers, and their potential volatility across different versions from the same manufacturer, replicating non-constant gain settings across experiments may be extremely difficult. Note that using constant gain also means that there should be no adaptive *translation*, such as a transition from line to page-based scrolling units reported by Gillick et al. [12, 13] (we are unaware of commercial drivers that do so, however some Logitech mice have a mechanical switch that the driver can activate to transition the wheel from detented to free-spinning when a threshold scroll velocity is exceeded—altering the possible range and behaviour of a user's input).

Second, user settings for disabling acceleration should be treated with suspicion, and researchers should check carefully whether acceleration is actually disabled. Ideally, an inspection similar to that described in this paper should be conducted, but otherwise, researchers should avoid drivers that are known to exhibit non-constant velocity scale factors (e.g., Logitech's drivers under Mac OS X, as reported here). Consequently, when gain is disabled, how this was achieved (user settings, API calls, etc.) should be reported.

The third recommendation is to report details of the transfer function, described next.

Reporting Scrolling Transfer Functions

In reporting a transfer function, there are two components that deserve attention: the *translation* and the *gain*.

The translation concerns the device and display resolutions, the level of action required to generate a scroll event on the device, and the magnitude of those events (in display units). The device resolution considers the number of events reported per unit of physical action: for example, Logitech's MX500 mouse reports 24 events per complete wheel rotation, while Microsoft's Wheel Mouse Optical reports 18 (i.e. a complete revolution of the MX500 is equivalent to 1.33 revolutions of the Wheel Mouse Optical). The level of action required to generate a scroll event considers the lower-bounds of physical action generating scroll events: in a scroll wheel, this would be the minimum wheel rotation and resistance to generate a scroll event, but for trackpad scroll gestures it could concern the minimum velocity of movement, or the minimum total displacement. Finally, the event magnitude considers the number of pixels, lines, or pages that the minimum scroll event moves (which may be fractional with high resolution devices [e.g., 21]).

Where a constant gain is used, the scale factor between the event magnitudes from the translation component should be reported. Where non-constant gain is used, the gain function(s) should be described using formulas, figures, and/or tables with the mapping between the output of the translation component to the final scroll behaviour. In both cases, the mechanics of the device and input/output resolutions should be reported.

Limitations and Further Work

Most of the gain functions analysed in this paper were reverse engineered without source code. It is therefore possible that our descriptions of the functions are incomplete because we failed to probe a salient input parameter: for example, we did not probe for attendance to acceleration or jerk (the derivative of acceleration). Similarly, we analysed the data that is sent from the operating system to user applications, and did not consider possible manipulation of that data by applications or the frameworks/libraries they are built upon. These higher-level systems have access to information about the information space being navigated (for example, the document length [11]), and may use that information to further augment scrolling behaviour. We have, however, presented a framework for understanding that such parameters could be attended to and such manipulations may be present, and a method for inspecting their impact if required.

Our attempt to reverse engineer the behaviour of Microsoft's IntelliPoint driver under Mac OS X failed because the driver prevented us from communicating with the EchoMouse. There are two hardware solutions to this problem: either EchoMouse could be engineered to store a pre-programmed set of signals to be emulated, or it could be designed to support a second USB input (one for sending signals to EchoMouse, and the other for sending messages to the driver).

None of the devices we tested supplied information about their physical units or resolution of input. Our analysis therefore used "counts" rather than physical units (such as degrees). Similarly, to our knowledge, drivers currently do not consider the display resolution when calculating gain (e.g., pixel size, pixel density, and/or scaling factors); but as higher resolution devices and displays become available, there are

opportunities for investigating how transfer functions can be better designed to adapt to different input and output resolutions—for example, the interaction between user performance and input resolution (both device resolution, and human capabilities), and similarly for the output resolution (adapting to different display configurations).

There are also other types of scrolling hardware that have not been examined here, most notably trackpads and touch mice (devices that feature a touch-sensitive surface). Some of these devices feature transfer functions that enable features such as simulated momentum and friction when scrolling.

CONCLUSIONS

Scrolling is an elemental interface control, and system transfer functions are fundamental in determining their behaviour. Yet despite their importance, scrolling transfer functions have received little research attention. This paper examined how scrolling transfer functions work and the input parameters they attend to. We described a method to reverse engineer the state of the art in scrolling transfer functions, and we used the method to expose how systems vary with the input parameters they attend to and in their processing of these parameters. As well as providing a firmer foundation for research into improving scrolling transfer functions, the paper's findings also suggest that when evaluating scrolling techniques, researchers should be cautious about potential interactions between the system transfer function and experimental treatment. The method proposed allows system transfer functions to be precisely recorded, aiding experimental replication. In further work, we will examine gesture-based transfer functions on touchscreens and trackpads, and compare user performance with different functions.

ACKNOWLEDGEMENTS

This work was supported by a Royal Society of New Zealand Marsden Grant 10-UOC-020.

REFERENCES

1. Arthur, K. W., Matic, N., and Ausbeck, P. Evaluating touch gestures for scrolling on notebook computers. In *CHI EA '08*, ACM (New York, NY, USA, 2008), 2943–2948.
2. Bates, B. M., Dezmelyk, R., Ingman, R., Lieb, R., McGowan, S., Ray, K., Schumacher, S., Sherman, N. C., Stern, D., van Flander, M., and Zimmerman, R. HID usage tables, Version 1.12. USB Implementer's Forum (2004).
3. Bergman, M., Peuranch, T., Schmidt, T., McGowan, S., Crowe, J., Dezmelyk, R., Zimmerman, R., van Flandern, M., Nathan, B., Davis, M., and Rayhawk, J. Device class definition for human interface devices (HID), Version 1.11. USB Implementer's Forum (June 2001).
4. Buxton, W. Lexical and pragmatic considerations of input structures. *SIGGRAPH Computer Graphics 17*, 1 (January 1983), 31–37.
5. Buxton, W. Touch, gesture, and marking. In *Human-Computer Interaction: Toward the Year 2000*, R. M. Baecker, J. Grudin, W. Buxton, and S. Greenberg, Eds. Morgan Kaufmann Publishers, San Francisco, 1995, ch. 7, 469–482.
6. Card, S. K., Mackinlay, J. D., and Robertson, G. G. The design space of input devices. In *CHI '90*, ACM (New York, NY, USA, 1990), 117–124.
7. Card, S. K., Mackinlay, J. D., and Robertson, G. G. A morphological analysis of the design space of input devices. *ACM Transactions on Information Systems 9*, 2 (Apr. 1991), 99–122.
8. Casiez, G., and Roussel, N. No more bricolage!: methods and tools to characterize, replicate and compare pointing transfer functions. In *UIST '11*, ACM (New York, NY, USA, 2011), 603–614.
9. Chipman, L. E., Bederson, B. B., and Golbeck, J. A. Slidebar: analysis of a linear input device. *Behaviour & Information Technology 23*, 1 (2004), 1–9.
10. Cockburn, A., and Gutwin, C. A predictive model of human performance with scrolling and hierarchial lists. *Human-Computer Interaction 24*, 3 (2009), 273–314.
11. Cockburn, A., Quinn, P., Gutwin, C., and Fitchett, S. Improving scrolling devices with document length dependent gain. In *CHI '12*, ACM (New York, NY, USA, 2012), 267–276.
12. Gillick, W. G., and Lam, C. C., U.S. Patent No. 5530455. U.S. Patent and Trademark Office (Washington, DC, 1996).
13. Gillick, W. G., and Rosenberg, R. A., U.S. Patent No. 5446481. U.S. Patent and Trademark Office (Washington, DC, August 1995).
14. Gutwin, C., and Cockburn, A. Improving list revisitation with listmaps. In *AVI '06*, ACM Press (New York, NY, USA, 2006), 396–403.
15. Hinckley, K., Cutrell, E., Bathiche, S., and Muss, T. Quantitative analysis of scrolling techniques. In *CHI '02*, ACM (New York, NY, USA, 2002), 65–72.
16. Hinckley, K., and Sinclair, M. Touch-sensing input devices. In *CHI '99*, ACM (New York, NY, USA, 1999), 223–230.
17. Hinckley, K. P., and Cutrell, E. B., U.S. Patent No. 7173637. U.S. Patent and Trademark Office (Washington, DC, 2007).
18. Kobayashi, M., and Igarashi, T. MoreWheel: Multimode scroll-wheeling depending on the cursor location. In *UIST 2006 Adjunct Proceedings: Demonstrations* (2006), 57–58.
19. Lipscomb, J. S., and Pique, M. E. Analog input device physical characteristics. *SIGCHI Bulletin 25*, 3 (July 1993), 40–45.
20. Mackinlay, J., Card, S. K., and Robertson, G. G. A semantic analysis of the design space of input devices. *Human-Computer Interaction 5*, 2–3 (1990), 145–190.
21. Microsoft Corporation. Enhanced wheel support in windows. Tech. rep., 2010.
22. Montalcini, A. L., U.S. Patent No. 7661072. U.S. Patent and Trademark Office (Washington, DC, 2010).
23. Wherry, E. Scroll ring performance evaluation. In *CHI EA '03*, ACM (New York, NY, USA, 2003), 758–759.
24. Zhai, S., Smith, B. A., and Selker, T. Improving browsing performance: A study of four input devices for scrolling and pointing tasks. In *INTERACT '97*, Chapman & Hall, Ltd. (London, UK, 1997), 286–293.

HUMAN–COMPUTER INTERACTION, 2008, Volume 23, pp. 215–250

ISSN: 0737-0024 print / 1532-7051 online
DOI: 10.1080/07370020802278163

Taylor & Francis
Taylor & Francis Group

The Impact of Control-Display Gain on User Performance in Pointing Tasks

Géry Casiez,[1] **Daniel Vogel,**[2] **Ravin Balakrishnan,**[2] and **Andy Cockburn**[3]
[1]*University of Lille & INRIA Lille-Nord Europe*
[2]*University of Toronto*
[3]*University of Canterbury*

ABSTRACT

We theoretically and empirically examine the impact of control display (CD) gain on mouse pointing performance. Two techniques for modifying CD gain are considered: constant gain (CG) where CD gain is uniformly adjusted by a constant multiplier, and pointer acceleration (PA) where CD gain is adjusted using a nonuniform function depending on movement characteristics.

Géry Casiez is a computer scientist with interests in empirical evaluation of user interfaces including associated metrics and predictive models of human performance; he is an Assistant Professor in the Department of Computer Science of the University of Lille, France. This work was conducted while he was a postdoctoral fellow at the University of Toronto. **Daniel Vogel** is a graduate student in the Department of Computer Science, University of Toronto. **Ravin Balakrishnan** is an Associate Professor of Computer Science and Canada Research Chair in Human-Centred Interfaces at the Department of Computer Science, University of Toronto. **Andy Cockburn** is a computer scientist with interests in modeling and empirically measuring human performance with interactive systems; he is an Associate Professor in the Department of Computer Science of the University of Canterbury, Christchurch, New Zealand.

CONTENTS

Both CG and PA are evaluated at various levels of relationship between mouse and cursor movement: from low levels, which have a near one-to-one mapping, through to high levels that aggressively amplify mouse movement. We further derive a model predicting the modification in motor-space caused by pointer acceleration. Experiments are then conducted on a standard desk-

top display and on a very large high-resolution display, allowing us to measure performance in high index of difficulty tasks where the effect of clutching may be pronounced. The evaluation apparatus was designed to minimize device quantization effects and used accurate 3D motion tracking equipment to analyze users' limb movements.

On both displays, and in both gain techniques, we found that low levels of CD gain had a marked negative effect on performance, largely because of increased clutching and maximum limb speeds. High gain levels had relatively little impact on performance, with only a slight increase in time when selecting very small targets at high levels of constant gain. On the standard desktop display, pointer acceleration resulted in 3.3% faster pointing than constant gain and up to 5.6% faster with small targets. This supported the theoretical prediction of motor-space modification but fell short of the theoretical potential, possibly because PA caused an increase in target overshooting. Both techniques were accurately modeled by Fitts' law in all gain settings except for when there was a significant amount of clutching. From our results, we derive a usable range of CD gain settings between thresholds of speed and accuracy given the capabilities of a pointing device, display, and the expected range of target widths and distances.

1. INTRODUCTION

Pointing at a target is a fundamental and frequent task in graphical user interfaces (GUIs), so even a marginal improvement in pointing performance can have a large effect on a user's productivity. Commensurate with its importance, pointing and methods to improve it are among the most mature areas of research in human–computer interaction. It is therefore somewhat surprising that pointer acceleration—the widely deployed and simple technique that governs a dynamic relationship between mouse and pointer movement—has not been thoroughly studied.

Pointer acceleration (PA) is the default behavior on the Microsoft Windows XP/Vista and Apple Mac OS X operating systems. It dynamically manipulates the *Control-Display* (CD) gain between the input device and the display pointer as a function of the device velocity: when the velocity of the control device is high, CD gain is high (typically well above 1) and when the control device moves slowly, the CD gain is low (in some cases less than 1). The assumption is that fast device movement implies a great distance must be covered to reach the intended target, so pointer movement can be amplified to quickly cover that distance. Conversely, slow device movement implies that the target is close, so pointer movement should be slow to support accurate adjustments. *Constant gain* (CG) is the simpler method for manipulating CD gain via a constant multiplier regardless of device movement characteristics.

Previous research comparing pointer acceleration with constant CD gain has found that it either harms performance (Graham, 1996; Trankle & Deutschmann, 1991) or it makes little difference (Jellinek & Card, 1990). Unfortunately, these studies evaluated mapping functions that were noncontinuous or conservative, and consequently dissimilar to the continuous and more aggressive functions that are widely used today. Prior research on the effect of constant CD gain is more extensive, but there are no definitive results. Some researchers have shown experimental evidence that performance follows a U-shaped curve with optimal performance at moderate levels (Gibbs, 1962; Zhai, Milgram, & Buxton, 1996), whereas others have found no effect at all (Accot & Zhai, 2001; Arnaut & Greenstein, 1990; Buck, 1980; Jellinek & Card, 1990; Johnsgard, 1994; Langolf, Chaffin, & Foulke, 1976). Jellinek and Card even suggest that CD gain can have no effect or it would violate Fitts' law. Yet intuitively there would seem to be performance barriers such as increased clutching and muscle coordination at low levels of CD gain and limits of fine muscle control at very high levels.

This article investigates and compares the effects of constant CD gain and PA in two experiments. Unlike previous work, we use an aggressive and continuous PA function taken from a modern operating system, evaluate a broad range of constant gain levels, essentially eliminate mouse quantization problems, accurately motion track users' limb movements, and evaluate pointing at very high index of difficulty targets on both a normal desktop display and a 5-m wide, high-resolution display. We also propose a model for target acquisition with pointer acceleration, which adapts the Fitts' law index of difficulty to accommodate the effective motor-space changes created by the PA function. Finally, to aid future pointing research, we propose a model identifying boundary constant CD gain levels to account for quantization effects.

Our empirical results show that the performance of constant CD gain follows an "L-shape" with performance decreasing rapidly at low gain levels but with little degradation at very high levels of gain. We attribute the decrease in performance at low gain levels to increased device clutching and limitations of user limb velocity. We also found that pointing acceleration has a small performance advantage over constant CD gain when selecting small targets or covering long distances.

2. Related Work

2.1. Fitts' Law

Fitts' law (Fitts, 1954) is a highly successful model for predicting the movement time of a pointing task. Originally used to model direct pointing where the hand taps physical objects, Fitts' law is also robust for indirect pointing

where the control device and display pointer are decoupled (Card, English, & Burr, 1978; MacKenzie, 1992). The decoupling of control and display creates two different *spaces*: the *display space,* where we view a representation of the pointing action, and the *motor space,* where we manipulate the control device. Given the intended target's width W and distance D, the total movement time T is predicted with the following equation using MacKenzie's (1992) Shannon formulation:

$$T = a + b \log_2 \left(\frac{D}{W} + 1 \right) \quad (1)$$

The constants a and b are empirically determined for the pointing technique and/or device being used. The logarithmic term is the pointing task's index of difficulty (ID) measured in bits. Intuitively it shows that tasks become more difficult as a target moves farther away, or as a target becomes smaller.

2.2. Constant CD Gain

CD gain (Gibbs, 1962) is a unit free coefficient that maps the movement of the pointing device to the movement of the display pointer (the reciprocal is called the CD ratio; McCormik, 1976). If CD gain is 1, the display pointer moves at exactly the same distance and speed as the control device; when CD gain is greater than 1, the display pointer moves proportionality farther and faster than the control device; and when CD gain is less than 1, the display pointer moves slower, covering less distance than the control device. The CD gain can be computed by taking the ratio of the pointer velocity to device velocity (see equation 2)

$$CDgain = \frac{V_{pointer}}{V_{device}} \quad (2)$$

Quantization can become a problem if the maximum resolution of the control device together with a high CD gain prevents every pixel from being addressable on the display. The maximum CD gain that can be used without quantization problems is calculated by dividing the resolution of the pointing device by the resolution of the display using the same unit of measurement (e.g., DPI).

When CD gain is very low and/or the physical device movement area is constrained, the device may need to be *clutched* to move the display pointer over a long distance. Clutching is when a device is repositioned in motor space without affecting the display pointer. The device movement area constraint may be a well-defined characteristic of the device, such as the limited input area on laptop track pads, or less defined, such as the comfortable range of arm movement or unobstructed surface space. We call the maximum area

of unconstrained physical movement the *operating range* of the device. Note that Jellinek and Card (1990) use the term *device footprint*, but this can be confusing because it also refers to the static area occupied by the physical device.

As noted by Jellinek and Card (1990), there is no term for CD gain in Fitts' law. Considering that it was originally formulated for direct pointing, where CD gain is always equal to 1, this is not surprising. Also, manipulations of CD gain on one device can be accommodated by considering each CD gain level as a different device with different values for *a* and *b* constants. However, this does not mean that Fitts' law is necessarily independent of CD gain, and an enhanced formulation of Fitts' law that included CD gain as a parameter would be useful. Furthermore, including a term related to CD gain may become necessary if the CD gain levels are dynamically manipulated during a pointing task, as is the case in pointer acceleration.

2.3. Prior Studies

CD gain has been studied extensively in the context of physical and virtual control devices, but unfortunately these studies do not provide a definitive picture of the impact of CD gain on user performance.

Gibbs (1962) found that high CD gains improved pointing performance with position and rate control systems. For experimental stimuli, he used a one-dimensional pointing task with a single target distance (22.5 mm) and single target width (3 mm) resulting in a single ID of 3. Error rate was not a reported factor because each pointing task had to be completed successfully (i.e., any errors had to be corrected and this correction cost is included in the trial time). He tested six CD gain (G) levels ranging from 0.15 to 0.90 and five artificial exponential lag (L) times ranging from 0 to 2 sec. His results for position control systems are summarized in the following empirically derived equation:

$$T = 0.91 - \frac{0.02}{G} + 1.212L - \frac{0.106L}{G} - 0.4L^2 + \frac{0.032L}{G^2} - \frac{0.003L^2}{G} \qquad (3)$$

It is evident from this model that higher lag times decrease performance, however if we set L equal to an ideal amount of zero, Gibbs's equation simplifies to $T = 0.91 - 0.02/CD$. This predicts movement time based solely on CD gain, with movement times increasing with CD gain level.

Buck (1980) studied the effect of CD gain using a joystick for input. He used a one-dimensional pointing task with a range of target distance and width combinations selected to produce a consistent ID of 4.2. Like Gibbs's experiment, error rates were not reported because each pointing task had to be completed successfully. Target widths ranged from 0.85 mm to 1.7 mm and distances from 15 mm to 30 mm. He tested rather low CD gain levels of 0.5,

1.0, and 2.0 and found that varying CD gain had no effect on movement time, although he noted that acquisition time (time at which the pointer first crosses the edge of the target) increased as the motor space target width decreased and were independent of the display target width.

Arnaut and Greenstein (1990) evaluated the effect of CD gain on performance using a trackball and touch tablet. Their experimental design used pointing tasks with a single ID of 3.16 and five different CD gain levels ranging from 1 to 3, but results showed no significant effect across CD gain levels (no error rates were reported). In spite of this, they argue that a combination of CD gain and ID should be used to predict movement time, but they do not provide an equation or model.

Johnsgard (1994) found that higher CD gains decreased selection time when using a mouse and a virtual reality glove with mean error rates of 6.5% for the mouse. However, this experiment used low IDs (1 to 4) and low CD gain levels (1, 2, and 3) so any conclusion should be taken within this context. He proposed an equation to model the result and demonstrated that it explained 81% of the variance of his data:

$$T = a + b \log\left(\frac{D}{W}\frac{1}{G} + 1\right) \tag{4}$$

The equation reduces to Fitts' law when CD gain equals 1 (G = 1). However, changing the CD gain divides both the distance and width of the target in motor space and thus should not change the motor space ID. This equation predicts that movement time decreases as CD gain is increased.

Jellinek and Card (1990) found that plotting mean selection times against CD gain resulted in a U-shape, with the best performance when CD gain was near 2 (no error rates were reported). They attributed the effect to increased clutching at low CD gain and to quantization at high gain. They tested CD gain levels of 1, 2, 4, 8, 16, 32, and their IDs ranged from 1.6 to 5.0, with a maximum target distance of 223 mm and a minimum target width of 7 mm. Considering the resolution of their equipment—a 73 DPI display and 200 DPI mouse—any CD gain above 2.7 would cause quantization preventing the user from selecting every pixel. They also observed that a CD gain of 1 required frequent clutching. So in effect, their experiment only allowed accurate testing of a single gain factor (2).

Accot and Zhai (2001) studied a graphics tablet steering task at four CD gain levels from 1 to 16. Like Jellinek and Card (1990), they also found a U-shape relationship between performance and CD gain with error rates increasing with higher CD gain. They observed that different muscle groups were used at different CD gain levels and suggested that this may be the reason for the performance degradation. However, these are qualitative observations with no empirical measurements, so it is difficult to draw definitive conclusions.

Bohan, Thompson, and Samuelson (2003) found that a CD gain of 1 was significantly slower than CD gains of 2, 4, or 8 in a mouse pointing task (no error rates were reported). Like Accot and Zhai (2001), they also attributed this effect to the difference in muscle groups used at a CD gain of 1 compared to the other levels, but this is also based on qualitative observation rather than empirical measurement.

Langolf et al. (1976) studied movement amplitude with varying levels of microscope magnifications, giving a similar effect to CD gain between motor control and visual display through the microscope eyepiece. Error rate was not reported because each trial had to be completed successfully. They found that magnification does not affect performance until it approaches 20× magnification, at which point performance deteriorates because of finger tremor. They also conducted Fitts' law analyses for different limbs (fingers, wrist, and forearm), with results showing that the limbs with a small range of movement had greater aiming performance. Balakrishnan and MacKenzie (1997) found a similar trend in which the combined use of multiple fingers outperformed other limb segments but that a single finger in isolation did not necessarily perform better. Zhai et al. (1996) also found that user performance increased when coordinated fingers where used to control 6 DOF docking tasks.

These results are summarized in Figure 1, which shows that there is no clear result governing the effect of CD gain. Accot and Zhai (2001) and Jellinek and Card (1990) found that very low and very high CD gains reduced performance creating a U-shaped profile for movement time versus CD gain. Gibbs (1962) found that performance decreased with high CD gain, but Johnsgard (1994) found the inverse. Langolf et al. (1976) found that performance decreased sharply at a certain CD gain threshold, but Buck (1980) found no effect at all. In all of these studies, the range of CD gain evaluated was either small or had quantization problems. Also, the target distances and widths were conservative (see Figure 1).

2.4. Dynamic Gain: PA

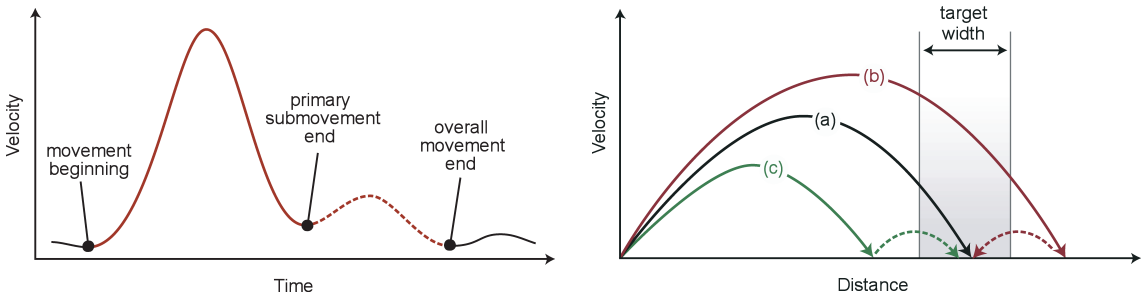
PA dynamically increases CD gain as the velocity of the control device increases. This behavior is motivated by the hybrid *optimized initial impulse* motor control model[1] for human pointing motions (Meyer, Smith, Kornblum, Abrams, & Wright, 1988, 1990). It works as follows: an initial high-velocity *ballistic movement* is made in the direction of the target. If the ballistic movement ends on the target, the task is complete, but if the movement under- or

1. Note that there are other models proposed for human pointing motion: Balakrishnan (2004) gave a brief summary; Rosenbaum (1991) provided more detail.

Figure 1. **Summary of Prior Work Evaluating the Effect of Control Display Gain.**

Study	Device	CD Gain	ID	Display W (mm)	Display D (mm)	Result
Gibbs (1962)	joystick	0.15–0.90	3	3	22.5	Effect Found: time increases with increased gain
Buck (1980)	joystick	0.5–2.0	4.2	0.85, 1.7	15, 30	No effect of gain
Arnaut and Greenstein (1990)	trackball & touch tablet	1–3	3.16	5–15	40–120	CD gain is not an adequate specification for performance. Optimization of performance requires an optimal combination of control input amplitude, display output amplitude, and display target width.
Johnsgard (1994)	mouse	1– 3	1–4	13–51	50–203	Effect Found: time decreases with increased gain
Jellinek and Card (1990)	mouse	1–32	1.6–5.0	7–28	56–223	Effect Found: U-shape with best gain of 2, but they dismiss results. Also, quantization present for gains greater than 2.7
Langolf, Chaffin, and Foulke (1976)	hand (tapping task)	1	1.7–7.4	0.076–50.8	2.5–305	Performance varies between the different limbs used (finger > wrist > arm)
Accot and Zhai (2001)	graphics tablet	1–16	4–33 (steering task)	7.6–17.8	63–254	Effect Found: U-Shape, best time between CD gain 2 and 4
Bohan, Thompson, and Samuelson (2003)	mouse	1–8	1.4–7.6	1.5–12	18.75–300	Effect Found: Longer movement times with movements involving proximal joints (forearm) compared to those involving distal joints (wrist, fingers)

Figure 2. **(left) Decomposition of a pointing movement into the ballistic and corrective phases (adapted from Meyer et al., 1988). (right) (a) Is the case where a single movement reaches the target. (b) and (c) are the more likely cases where the initial movement under or over shoots the target, requiring subsequent corrective movements.**



overshoots the target, a second lower velocity *corrective movement* is used in the direction of the target. Successively slower corrective movements are reapplied until the target is acquired (see Figure 2).

PA is one of many techniques that influence the motor-space through which the device travels during target acquisition: High gain reduces the motor distance during ballistic movement, and low gain increases the motor size of the target during corrective action. Other successful examples of motor-space adaptation include McGuffin and Balakrishnan's (2002) expanding targets; Grossman and Balakrishnan's (2005) bubble cursor; and Blanch, Guiard, and Beaudouin-Lafon's (2004) semantic pointing, all of which dynamically adjust motor-space to reduce the target distance, increase the width, or both. These techniques, fully reviewed in Balakrishnan (2004), are all target oriented: The CD gain or the target/cursor area is dynamically adjusted as a result of the cursor's proximity to the target. PA, in contrast, is more general because it is independent of the semantics of the target environment.

A PA function f produces a CD gain G from the device motor space velocity v (the function may map motor-space velocity directly to display space velocity, but this is equivalent).

$$G = f(v) \tag{5}$$

Most previous work has investigated variants of discrete two-level threshold functions (Jellinek & Card, 1990). These are easy to implement and were

2. Linux distributions still use the two-level threshold functions. The X server controls the PA using the threshold velocity and the second level of the functions which are set in the mouse configuration panel. The XChangePointerControl function is an alternative way of configuring these settings from within a software application.

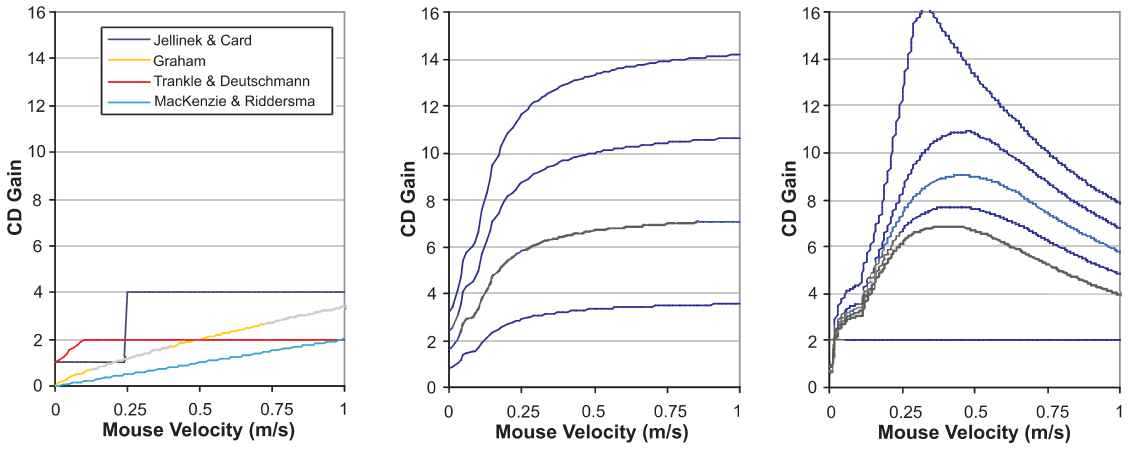
once common in commercial operating systems,[2] but they cause discontinuities where the pointer suddenly accelerates or decelerates, potentially reducing performance. The functions used in contemporary operating systems are continuous to smooth out changes in CD gain (see Figure 3).

Like constant CD gain, previous experiments investigating the effects of PA have produced divergent results. This is likely because of the inconsistent or poor quality of the PA function used, or the limited range of IDs, target distances, and widths. All evaluations used a mouse as the input device.

Jellinek and Card (1990) found that a two-level discrete threshold PA function did not improve user performance compared to constant CD gain of 2, and they claim that PA cannot improve performance or it would violate Fitts' law. However, their results need to be considered in light of their experimental conditions and apparatus, which used a discrete acceleration function with conservative upper CD gain levels (4 or 8), IDs below 5 bits, and a maximum display distance of 223 mm.

Graham (1996) found that a two-level discrete threshold function for 3D hand movement in virtual reality pointing tasks provided no advantage but that a continuous function impaired performance compared to a constant CD gain of 1. Like Jellinek and Card (1990), he used conservative functions in comparison to contemporary ones (Figure 2b and 2c). His two-level function

Figure 3. **Plotting the control device velocity against Control Display gain shows the characteristic curve of pointer acceleration functions. In previous research, conservative and discrete two-level pointer acceleration functions have been used (a) when compared with pointer acceleration functions used by modern operating systems (b, c) (calculated from the registry and source code; see Appendix).**



(a) Previous research (only one discrete function has been represented)

(b) Windows XP/Vista levels 0.25 to 1.0 corresponding to positions on preference slider

(c) Mac OSX levels 0.0 to 0.875 corresponding to positions on preference slider

had a CD gain of 1 until a control velocity of 200 mm/sec, at which the CD gain became 2, and his continuous function set $Gain = (velocity/200)^{p}$, where $p = .756$, which is likely to produce gain values below 1. The experiment evaluated IDs from 1.4 to 6.6 with a maximum target distance of 300 mm and a minimum width of 3 mm.

Trankle and Deutschmann (1991) found no difference between a continuous PA function and constant CD gain settings of 1 or 2. Their function linearly increased CD gain from 1 to a maximum of 2 when control device velocity reached 100 mm/sec. Their experiment evaluated IDs from 2.6 to 4.4 with a maximum target distance of 100 mm and a minimum width of 2.5 mm. Although not statistically significant, they noted mean movement times for PA were almost 10% higher.

Finally, MacKenzie and Riddersma (1994) tested three different scales of a continuous PA function found in the Apple Macintosh OS6 operating system, showing the medium scale setting to be significantly faster. The Macintosh function mapped control device velocity to CD gain as the product of squared control device velocity and a constant parameter. The experiment evaluated only a single ID of 3.2 with a target distance of 94 mm and a minimum width of 12 mm. They attribute the slower performance with the low setting to the observed predominant use of the forearm for device movement, whereas participants primarily used their wrist with the medium and high settings. They also found significantly lower error rates for the lower CD gain.

3. PA PERFORMANCE MODEL

PA causes dynamic modification to the target's distance and width in motor-space. This modification can be modeled to predict the extent to which the Fitts' law index of difficulty changes in motor space for each target. This section derives the motor space index of difficulty (ID_{mot}) formula for PA.

When constant CD gain is increased by a factor k, the distance and width of the target in motor-space are both reduced equally by a factor of k. Trivially, the ID in motor space equals the ID in display space so the difficulty of the pointing task has not changed. If we also assume unchanged constants for the intercept a and slope b (a reasonable assumption because the device has not changed), then Fitts' law predicts the exact same movement time regardless of the change in CD gain.

Recall that with PA, the mapping function is designed to produce high CD gain levels at high velocities and low CD gain levels at low velocities. Recall also that according to the optimized initial impulse motor control model, high velocities are used in the ballistic phase which attempts to get as close as possible to the target and low velocities are used during the subsequent corrective movements.

For the sake of illustration, assume we have constructed an ideal PA function, which instantly produces a high CD gain G_D for the duration of the ballistic phase (which mostly affects the distance to target, hence the subscript $_D$ for G_D), and a low CD gain G_W for the duration of the corrective phase (which mostly affects the target width, hence the subscript $_W$). Let G_D be k times greater than some baseline CD gain level and G_W be j times greater than the same baseline. By definition, $j < k$. Now, unlike the constant CD gain case, the distance and width of the target in motor space are not reduced equally; the distance is reduced by a factor of k and the width by a factor of j. The ID in motor space is now smaller than the ID in display space by a factor of j/k. According to Blanch et al.'s (2004) work with Semantic Pointing (which in some ways approximates the ideal pointer acceleration constructed for this argument) users are able to take advantage of a reduction of ID in motor-space and improve performance.

Of course, PA functions are not able to reliably produce a single high CD gain during exactly the duration of the entire ballistic phase. Because CD gain continuously changes with the velocity of the pointing device, we can compute the mean CD gain used to cover the distance (CD_D) and the mean CD gain used when near the target (CD_W). If a function is continuous, the mean of the function in an interval is the integral divided by the interval length. We approximate the ballistic phase as the movement occurring before the pointer crosses the target boundary at time T_1. CD_D is then the mean CD gain used in the interval T_1 and CD_W is then the mean CD gain used in the interval $T - T_1$, where T is the total time for the movement.

$$CD_D = \frac{1}{T_1}\int_0^{T_1} Gain(t)\,dt \qquad CD_W = \frac{1}{T - T_1}\int_{T_1}^{T} Gain(t)\,dt \tag{6}$$

From Fitts' law (equation 1), we find the ID in motor space ID_{mot}:

$$ID_{mot} = \log_2\left(\frac{\frac{D - W/2}{CD_D}}{\frac{W}{CD_W}} + 1\right) \tag{7}$$

Let the ratio of CD_W/CD_D be r and assume $D \gg W$, then:

$$ID_{mot} = \log_2\left(\frac{D}{W} + \frac{1}{r}\right) + \log_2(r) \tag{8}$$

If $1/r \ll D/W$, then:

$$ID_{mot} \approx ID + \log_2(r) \tag{9}$$

By definition $CD_W < CD_D$, so we can deduce that the index of difficulty in visual space (ID) is reduced in motor space by a quantity equal to the log of the ratio between mean CD gain used to cover the distance (CD_D) and the mean CD gain used when near the target (CD_W).

4. EXPERIMENT 1: DESKTOP SIZE DISPLAY

Our goal is to explore the effect of constant CD gain and PA on user performance. Because previous research comparing different levels of constant CD gain has been inconclusive, we want to confirm or refute any effect. If there is an effect we want to test previous hypotheses proposing that it is because of different limbs being used or the limits of fine motor control. Similarly, because previous research on PA has used noncontinuous mapping functions, and given that our theoretical analysis of PA using Fitts' law suggests that there should be a positive effect, we want to examine the effectiveness of a contemporary continuous PA function in comparison to constant CD gain. We used the default Microsoft Windows XP/Vista PA function (Microsoft, 2002), as it is arguably the most widely used.

4.1. Apparatus

We used a 20-in. 1600 × 1200 resolution 100 DPI LCD monitor and a 1600 DPI mouse (Logitech MX518 Gaming-Grade Optical Mouse). With our mouse and display configuration, this provided a maximum CD gain of 16 with no quantization problems (each pixel on the display is selectable). Our Windows C++/OpenGL application bypassed the standard mouse driver and read directly from the mouse hardware to get raw, real numbered coordinates at 60Hz, and updated the display at a regular 60FPS. To measure clutching time we mounted a feather weight switch under the mouse which recorded clutching events when the mouse was lifted off the surface. We also used a Vicon optical motion tracking system (http://www.vicon.com) to capture the absolute positions of the arm, hand, and mouse at 120Hz with submillimetre accuracy for limb movement analysis. Because the system is vision based and uses a custom predictive filter to smooth trajectories, it can be susceptible to tracking errors from marker occlusion or markers outside the calibrated tracking volume. However, with the small movement area in this experiment, tracking error was extremely low.

4.2. Task and Stimuli

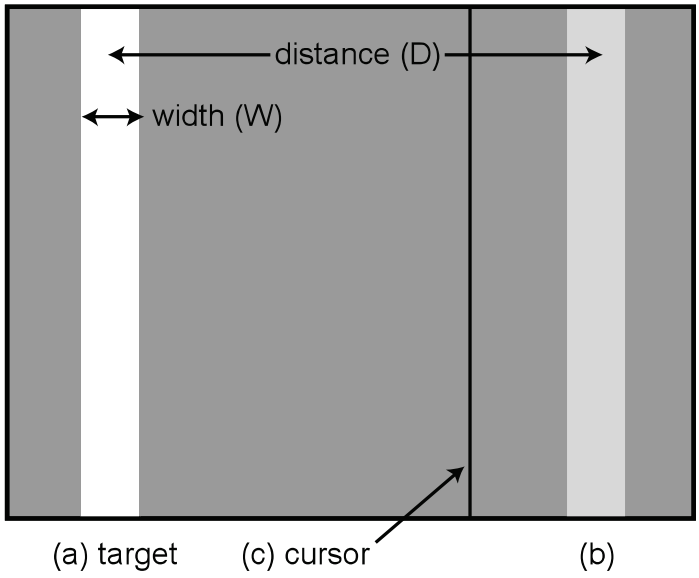
The task was a reciprocal one-dimensional pointing task, requiring participants to select two fixed-sized targets back and forth in succession (see

Figure 4). When participants correctly selected a target, the targets would swap colors, indicating the next target to select. If they missed a target, a sound was heard and the error logged. Participants had to successfully select the current target before moving to the next, even if it required multiple clicks. This design encourages participants to do the task to the best of their ability, rather than "racing through the experiment just to get done," as going too fast incurs errors that have to be corrected. After each block of trials, a cumulative error rate was displayed and a message encouraged participants to conform to an approximately 4% error rate by speeding up or slowing down. The pointer was not constrained to the bounds of the display to prevent using the edges to assist in target acquisition. Participants were encouraged to take breaks between blocks.

4.3. Participants

Eight volunteers (all male) with a mean age of 24.5 ($SD = 6.3$) participated. Compensation was in the form of credit for "experiment participation" in an undergraduate HCI course. We prescreened participants to form two groups of four: those that used Windows XP/Vista pointing acceleration on their own computer and those that did not.

Figure 4. **Experimental display. The targets were rendered as solid vertical bars, equidistant from the center of the display in opposite directions along the horizontal axis. The target to be selected was colored white (a), and the last target, which was the starting position, light gray (b). The cursor was represented by a one-pixel-thick vertical black line (c).**



4.4. Design

A within-subjects design was used. The independent variables were *Technique* (*CG* for Constant Gain and *PA* for Pointer Acceleration), *Level* (6 CD gain levels for *CG* and 6 scale factors for *PA*), distance between targets *D* ($D_L = 360$ mm, $D_M = 180$ mm, $D_S = 90$ mm), and target width *W* ($W_L = 8$ mm, $W_M = 4$ mm, $W_S = 2$ mm). *D-W* combinations were fully crossed with the exception of the combination $D_M W_M$, which was excluded to reduce experiment completion time because it had the same ID as D_L W_L and D_S W_S. The eight *D-W* combinations gave five task IDs: 3.6, 4.5, 5.5, 6.5, and 7.5. The CD gain *Levels* for the *CG* technique were 1, 2, 4, 6, 8, and 12 (left hand column of Figure 5); the scale *Levels* for the *PA* technique were 0.1, 0.25, 0.5, 0.75, 1.0, and 1.25, which span the scale factors available in the default "Mouse Properties" panel in Windows XP/Vista. Figure 5 summarizes and compares the six *CG* gain levels and *PA* scale levels—columns 3 and 4 show the minimum and maximum gain settings attainable with the *PA* levels, and columns 5 and 6 show the maximum and average gain levels used by the participants across all trials in the experiment. Our aim in supporting these settings for factor *Level* is to provide good coverage of settings for the *CG* and *PA* techniques. The analyses of results explicitly address the issues of conflating *Technique* and *Level*.

The presentation of the two techniques was fully counterbalanced across the participants in each PA usage group. Presentation of *Level* was counterbalanced between ascending or descending order. In that way we expect to lower the effect of learning between the different levels. For each technique and level, five blocks of trials were performed. Each block had each of the 8 *D-W* combinations presented in ascending order of ID, with 12 trials each. The presentation of the *D-W* combinations in ascending order of ID rather than randomly was an attempt to reduce drastic changes in the difficulty of trials from one set of 12 trials to the next. When each level was completed, the participant was asked to rate their performance in comparison to the previ-

Figure 5. **Comparison of Constant Gain and Pointer Acceleration Levels.**

CG Level (CD Gain)	PA Level (Scale)	Min CD Gain	Max CD Gain	Max CD Gain Used	Average CD Gain Used
1	0.1	0.31	1.46	1.4	0.7
2	0.25	0.79	3.66	3.1	1.4
4	0.5	1.59	7.32	5.7	2.2
6	0.75	2.38	10.98	7.8	3.0
8	1.0	3.18	14.65	9.6	3.6
12	1.25	3.98	18.31	11.1	4.2

ously completed level using a 5-point Likert scale. Because of the number of conditions and trials, the experiment was split across 2 days with each technique completed on 1 day (90 min per day).

In summary, the experimental design was:

8 participants ×
2 *Techniques* ×
6 *Levels* ×
5 *Blocks* ×
8 *D-W* combinations ×
12 trials
= 46,080 total trials

4.5. Results and Discussion

Error Rate

There is a significant effect of *W*, $F(2, 14) = 14.3$, $p < .0001$, on error rate. Error rate increases with small widths. A pairwise comparison[3] shows significant differences between each width: 6.8% for W = 2 mm, 4.5% for W = 4 mm, and 3.9% for W = 8 mm. The overall mean error rate was 5.0%. No other factors or interactions showed significant effects for error rate.

Movement Time

Movement time is the main dependent measure and is defined as the time taken to move from the previous target until the first click. Targets that were not selected on the first attempt were marked as errors but still included in the timing analysis. (We did the analysis with and without the errors removed and the same significant effects were found with comparable F and p values.)

Repeated measures analysis of variance (ANOVA) showed that the presentation order of techniques or levels had no significant effect on movement time (all $p > .3$), indicating that a within-subjects design is appropriate. We also found no significant effect or interaction for *Block* (all $p > .15$), indicating no learning effect was present, which is not surprising given the elemental nature of the task. We found no significant effect for PA usage group (whether the participant used Windows XP/Vista acceleration on their own computer or did not), $F(1, 6) = 1.2$, $p = .31$.

3. All post hoc pairwise analyses for all tests were performed using Bonferroni correction.

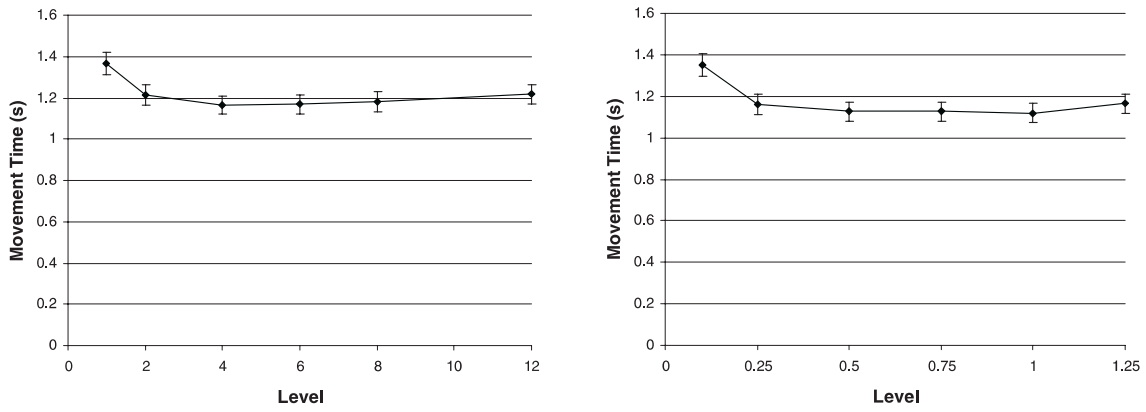
Although we found significant main effects for *Technique*, *D*, and *W,* we have to be cautious before drawing any conclusions because of the different meaning of the *Level* variable in each *Technique*. Thus, we proceed with an analysis of *Level*.

There was a significant main effect for *Level*, $F(5, 35) = 13.5$, $p < .001$. Post hoc pairwise comparisons found a significant difference between *CG Level* = 1 and *CG Levels* up to 8 ($p < .03$) with an 11% improvement between *CG Level* 1 and 2 (all $p < .05$). A significant difference was also found between *PA Level* = 0.1 and the other *PA Levels* (all $p < .05$) with a 14% improvement between 0.1 and 0.25 (Figure 6; recall that a *PA Level* of 0.1 corresponds to an effective CD gain range of 0.31 to 1.46, see Figure 5). No significant differences were found among the other *CG* or *PA Levels*. This shows that low CD gain does indeed have a negative effect on performance.

To see if the slower times observed for *CG Level* 1 and *PA Level* 0.1 are because of fatigue, we analyzed the variation of movement time across the blocks. We found that the time does not increase with *Block* for these levels (all $p > .1$) so we can reject this hypothesis. We also looked at clutching time and found it remained under 0.4% of the movement time, so it cannot be held responsible for the difference: Participants preferred to increase the operating range over which they operated the mouse rather than clutching.

To ensure that *CG* and *PA* are compared within their calibration "sweet spots," we removed *CG* levels 1, 2 and 12 (see Figure 4a) and *PA* levels 0.1, 0.25 and 1.25 (see Figure 6b) from further movement time analysis. We then find a marginal effect of *Technique* on movement time, $F(1, 7) = 4.8$, $p = .065$, with the *PA* mean of 1.16 sec ($SD = 0.3$) 3.3% faster than the *CG* mean of 1.2 sec ($SD = 0.3$).

Figure 6. **Effect of level on movement time for constant gain and pointer acceleration (error bars 95% confidence interval).**



(a) Constant Gain (b) Pointer Acceleration

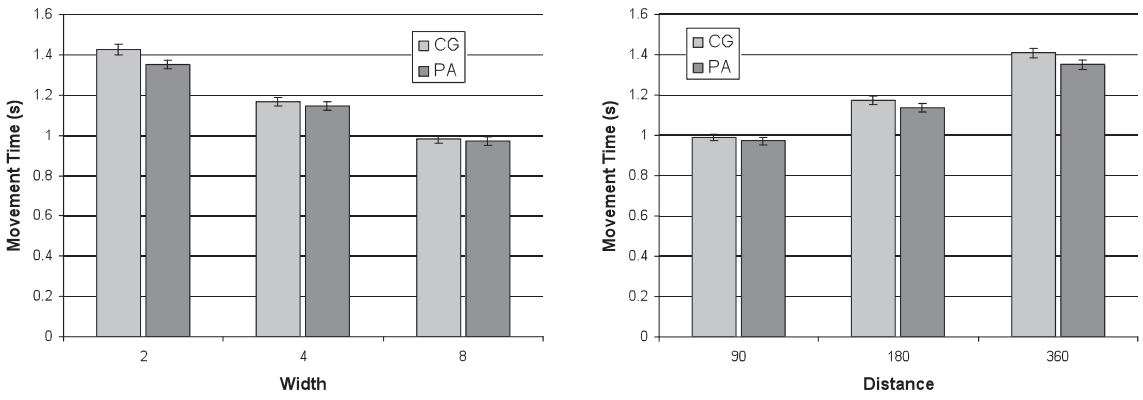
As predicted by Fitts' law, we found significant main effects for *D*, *W*, and *D-W*. We also found significant interactions with *Technique*. The *Technique* × *D-W* interaction, $F(4, 28) = 5.6$, $p < .01$, shows that performance with *PA* technique deteriorates less rapidly with increasingly difficult tasks than with the *CG* technique (further discussed in the Fitts' Law Analysis section later). The *Technique* × *W* interaction, $F(2, 14) = 6.8$, $p < .01$, depicted in Figure 7 (left), shows that *PA* has a 5.6% advantage over *CG* at small widths (pairwise significant, $W_S = 2$ mm, *PA* = 1.35 sec, *CG* = 1.43 sec, $p < .01$). This suggests that users encounter accuracy problems with *CG* because of the small motor space available for the target, particularly at high gain levels (also noted by Buck, 1980). With *PA*, however, users can maintain accuracy because of the low CD gains available at low device speeds. Finally, the *Technique* × *D* interaction, $F(2, 14) = 7.4$, $p < .01$, shown in Figure 7 (right), is caused by a similar effect, with *PA* allowing comparatively faster movement time as the target distance increases. *PA* has a 3.5% advantage over *CG* at large distances (pairwise significant, $D_L = 360$ mm, *PA* = 1.36 sec, *CG* = 1.41 sec, $p < .01$).

Mouse Operating Range and Limb Use

We define the mouse operating range as the maximum area on the desk traversed by the mouse. We calculated the operating range from motion tracking data, but because we are evaluating a one-dimensional task, we consider only the corresponding horizontal dimension.

There was a significant main effect for *Technique* on operating range, $F(1, 7) = 27.7$, $p < .005$, with a mean of 74 mm for the *CG* technique and 67 mm for the *PA* technique. Pairwise comparisons found significant differences between all *Levels* for each technique. As expected, the operating range decreases proportionally with increasing CD gain for *CG Levels*, with mean operating range

Figure 7. **Mean movement time for the two techniques, by width and distance (error bars 95% confidence interval).**



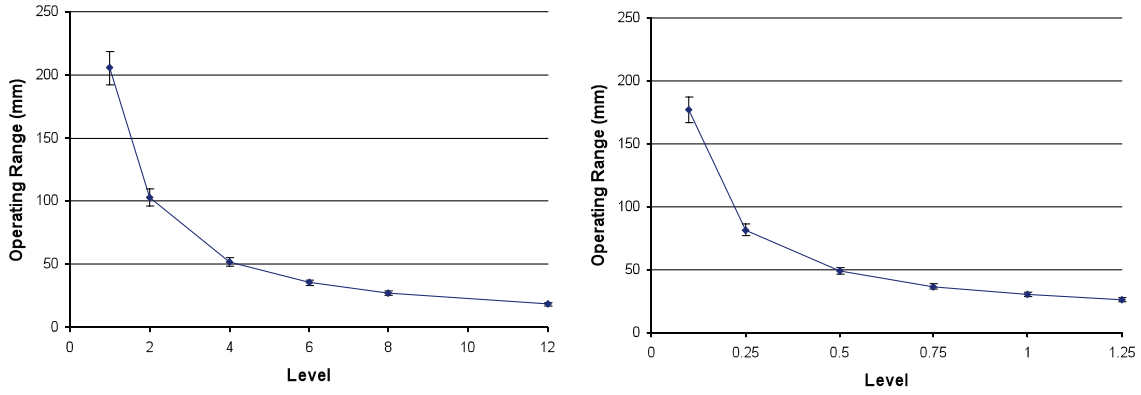
decreasing from 205 mm at *CG Level* = 1 to 18 mm at *CG Level* = 12 (Figure 8a). In contrast, the operating range decreased less dramatically across *PA Levels*, from 177 mm at *PA Level* = 0.1 to 26 mm at *PA Level* = 1.25 (Figure 8b).

Using the motion tracking data we also calculated the amount of forearm, hand, and finger movement in each frame within the parent limb's coordinate frame. We found a limb movement threshold based on the mean percentage of time in which the mouse velocity was zero during a trial. With this threshold, we calculated *limb usage profiles*: the percentage of time limbs or combinations of limbs moved in each frame during a trial (Figure 9).

For both techniques across *Levels*, limbs are rarely used in isolation. As the effective CD gain for each *Level* increases, there is progression from using all limbs together to using the hand and fingers in combination to using the hand or fingers individually. For example, to avoid clutching at a low CD gain of 1, users need to move the mouse a distance of 360 mm, equivalent to the on-screen target distance which is much too far for the hands or fingers alone. The arm is rarely used alone and fingers seem to play a role in maintaining accuracy at high CD gain levels.

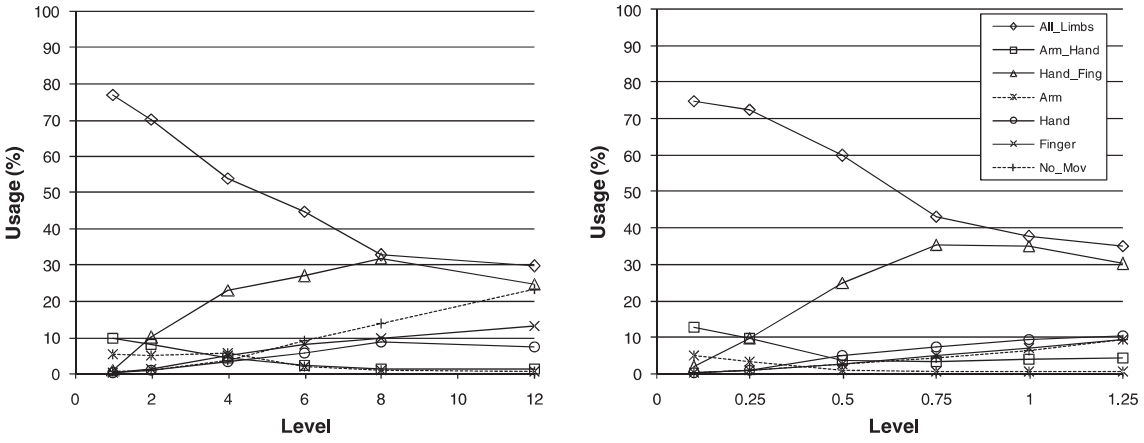
Previous work by Langolf et al. (1976) and Balakrishnan and MacKenzie (1997) found that finger throughput exceeds that of the arm in pointing tasks, so we anticipated that arm use would explain slow performance times at low CD gain levels. However the limb usage profiles do not have a strong correspondence to the movement time profile. For example, the differences in limb usage between *CG Level* = 1 and *CG Level* = 2 do not appear great enough to explain the 11% movement time difference. In fact, there is a larger

Figure 8. **Mean mouse operating range across levels with the two techniques (error bars 95% CI).**



(a) Constant Gain **(b) Pointer Acceleration**

Figure 9. **Limb usage profiles across all target distances.**



(a) Constant gain (b) Pointer acceleration

difference in limb usage between *CG Level* = 2 and *CG Level* = 4, yet the movement time difference in these two levels is smaller.

Overshooting

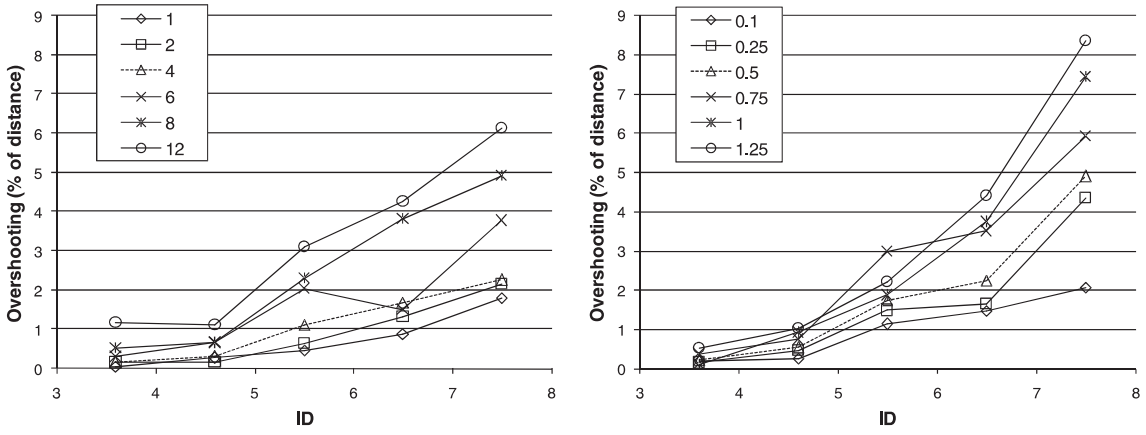
We define overshooting as the ratio of distance traveled past the extent of the target to the theoretical mean distance required for selection, which we define as $D + W/2$. A repeated measures ANOVA found a learning effect across *Blocks,* $F(4, 28) = 7.7$, $p < .01$, and pairwise analysis showed higher values for the first two blocks compared to the last three. As a result, we used only the last three blocks for overshooting analysis.

There was a significant effect of *Technique* on overshooting, $F(1, 7) = 6.3$, $p < .05$, with a *PA* mean of 2.2%, compared to *CG* with 1.6% . We also found significant main effects for *D-W,* $F(4, 28) = 10$, $p < .01$; *D,* $F(2, 14) = 9.9$, $p < .01$; and *W,* $F(2, 14) = 8.3$, $p < .01$, with more pronounced overshooting in high difficulty selections (more distant or smaller targets). Overshooting also increased with *Level,* $F(5, 35) = 8.8$, $p < .01$. Finally, there was a significant *Level* × *D-W* interaction, $F(20, 140) = 4.1$, $p < .01$, with high levels of CD gain causing more overshooting on difficult targets (Figure 10).

Peak Velocity in Motor and Display Space

Peak motor-space velocity (PMV) is the maximum velocity of the mouse during a trial and the peak display-space velocity (PDV) is the peak velocity of

Figure 10. **Overshooting across level and ID (D-W combination) for the two techniques.**



(a) Constant gain (b) Pointer acceleration

the on-screen pointer. These two measures are related by the function mapping motor movement to CD gain—constant for the *CG* technique and dynamic for the *PA* technique. Thus, for the *CG* technique, PDV is always a constant multiple of PMV depending on *CG Level*, but this not the case for the *PA* technique.

A repeated measures ANOVA showed significant main effects for *D* on PMV, $F(2, 14) = 220$, $p < .0001$, and *D* on PDV, $F(2, 14) = 383$, $p < .0001$, with PMV and PDV increasing with increased *D*. This confirms that the intensity of a ballistic movement is dependent on the distance to be covered (Plamondon & Alimi, 1997). A significant effect was also found for *Level* on PMV, $F(5, 30) = 160$, $p < .0001$, and on PDV, $F(5, 35) = 165$, $p < .0001$, with PMV decreasing with *Level*, and PDV increasing with *Level*.

We found no significant effect for *Technique* on PMV (*CG* mean 296 mm/sec, *PA* mean 279 mm/sec), but there was an effect for *Technique* on PDV, $F(1, 7) = 60.1$, $p < .0001$, with *CG* slower than *PA* (1031 mm/sec and 1314 mm/sec, respectively). This suggests that *PA*'s dynamic function provides a more pronounced ballistic movement in display space.

To estimate the maximum usable limb speed during pointing tasks we analyzed the peak velocity in motor space for the *CG* technique at *Level* 1.0. The maximum logged limb speed was 2441mm/s, with a 97th percentile value of 1536 mm/sec.

User Preference

After completing each set of blocks for a *Level*, the participants were asked if they found the current *Level* easier or more difficult to the previous *Level* us-

ing a 5-point Likert scale. On average, participants preferred *CG Level* 4 and *PA Level* 1 for the two techniques.

Fitts' Law Analysis and Relationship to the Model

Fitts' law models described here are based on regression analysis of the eight D-W combinations, rather than aggregate performance for each of the five IDs. This analysis allows independent effects of D and W to be exposed.

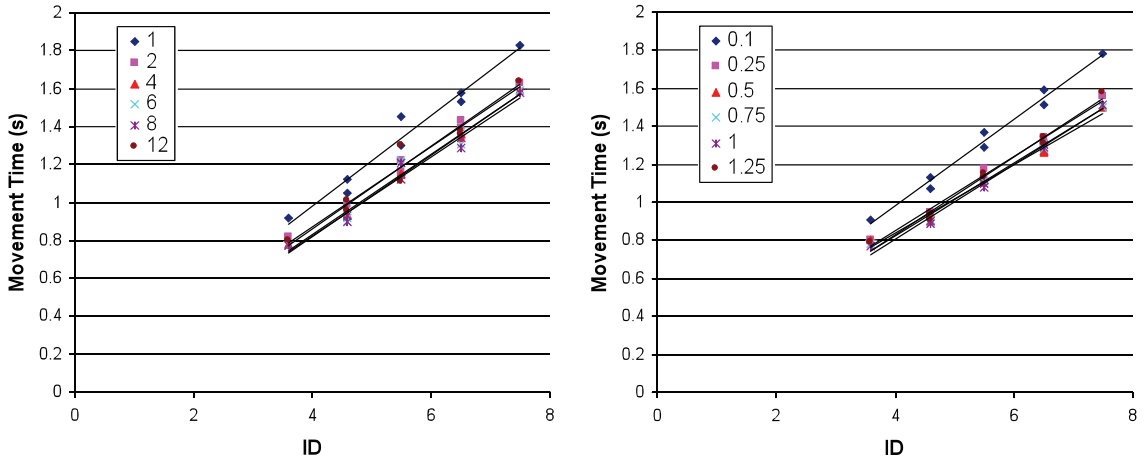
Regression analysis shows that both techniques closely adhere to Fitts' law (Figure 11 and Figure 12), refuting Jellinek and Card's (1990) claim that pointer acceleration would not.

From Equation 9 of our pointer acceleration performance model (see Section 3), we expect the index of difficulty in visual space (ID) to be reduced in motor space by a quantity equal to the log of the ratio between the mean CD

Figure 11. **Fitts' law regression across *Level*. The columns a and b are the standard constants from Fitts' law (equation 1).**

CG Level	a	b	r^2	PA Level	a	b	r^2
1	0.026	0.239	0.966	0.10	0.076	0.227	0.984
2	–0.011	0.218	0.982	0.25	0.059	0.197	0.987
4	–0.026	0.213	0.994	0.50	0.181	0.111	0.985
6	–0.042	0.215	0.975	0.75	0.055	0.191	0.990
8	–0.002	0.207	0.973	1.00	0.014	0.197	0.989
12	0.027	0.211	0.956	1.25	0.011	0.204	0.986
4–8	–0.023	0.212	0.984	0.5–1.0	0.060	0.190	0.991

Figure 12. **Fitts' law regression.**



(a) Constant gain **(b) Pointer acceleration**

gain used to cover the distance (CD_D) and the mean CD gain used when near the target (CD_W). As a first approximation, the time T_1 defined in our model (see Section 3) is taken as the time until the cursor crosses the border of the target. We computed the ratio CD_W/CD_D from the participants' performance data, finding it to be close to constant for all *PA Levels*, with a mean value of 0.5 ($SD = 0.14$). As a result, we can expect to decrease the ID in motor space (ID_{mot}) by approximately $\log_2(0.5) = 1$bit.

Computing the linear regression for ID_{mot} versus ID, we obtain good regression fitness for all PA Levels (with slopes and intercepts being close) revealing that the index of difficulty in motor space (ID_{mot}) is reduced at the same extent for all index of difficulty in visual space (ID). Using aggregated *PA Levels*, we obtain the following:

$$ID_{mot} = 0.93 * ID - 0.62 \quad r^2 = 0.931 \tag{10}$$

Although the slope follows our model's prediction because it is close to 1, we found that participants did not take full advantage of the 0.6 bit of ID reduction in motor space—there should have been an overall 10% time improvement compared to constant gain. The discrepancy between the theoretical prediction and the empirical data reveals the limitation of the PA technique. Although the theory is based on optimal performance without overshooting, the experimental data showed that the high CD gains used during the ballistic movement increased the amount of overshooting. Having overshot the target, the user must make corrective movements during which the device is likely to move slowly, resulting in low CD gains, and consequently time-consuming large motor distances. Although the theoretical model suggests that the index of difficulty in motor space can be reduced to zero with an ideal acceleration function which perfectly interprets the user's actions, in practice this is difficult to achieve. The increased overshooting and increased corrective movement time is symptomatic of the acceleration function incorrectly interpreting the user's actions preventing the user from reaching the full potential of pointer acceleration. Further work to tune the shape of pointer acceleration curves to match human performance might be a worthwhile endeavor.

5. EXPERIMENT 2: VERY LARGE, HIGH-RESOLUTION DISPLAY

Our goal with the second experiment is to explore the effect of high constant CD gain levels and high Micrsoft Windows XP/Vista PA scales on a very large, high-resolution display. With such a display, we can evaluate performance with very high index of difficulty pointing tasks, higher than those

conventionally studied in previous Fitts' law studies and in our first experiment. The high ID tasks should also induce enough clutching at low CD gain levels to allow us to inspect its impact on Fitts' law models.

5.1. Apparatus, Task, and Stimuli

The same apparatus, task, and stimuli were used as in Experiment 1, except the display was 4.7 m × 1.7 m and 25 DPI. With our 1600 DPI mouse, this provided a maximum CD gain of 64 with no quantization problems (each pixel on the display is selectable). Participants were seated at a large desk 3 m from the display at its center. They were free to use the entire desktop to operate the mouse. Although we used the same Vicon motion tracking setup as Experiment 1, the high amount of clutching resulted in large arm and forearm movements, which reduced the reliability of our tracking data.

5.2. Participants

Eight volunteers (6 male and 2 female) with mean age of 23.5 ($SD = 1.6$) participated; none had participated in Experiment 1. Compensation was in the form of credit for "experiment participation" in an undergraduate HCI course. Seven of the participants used Microsoft Windows XP/Vista PA on their own computer.

5.3. Design

The experiment design was similar to Experiment 1. A within-subjects design was used. The independent variables were *Technique* (*CG* for Constant Gain and *PA* for Pointer Acceleration), *Level* (6 CD gain levels for *CG* and 6 scale factors for *PA*), distance between targets D ($D_L = 4500$ mm, $D_M = 2250$ mm, $D_S = 1125$ mm), and target width W ($W_L = 36$ mm, $W_M = 18$ mm, $W_S = 9$ mm). The minimum width of 9 mm was chosen because people with 20/40 vision can read an 8.6 mm symbol from a distance of 3 m[4] (Millodot, 1997). As in Experiment 1, *D-W* combinations were fully crossed with the exception of the combination $D_M W_M$ which was excluded. The eight *D-W* combinations gave five task IDs: 5, 6, 7, 8, and 9. The CD gain *Levels* for the *CG* technique were 2, 5, 8, 12, 16, and 20. The scale *Levels* for the *PA* technique were 0.25, 0.5, 1.0, 1.25, 1.5, and 2.0.

4. The min decipherable symbol height h given distance d: $h = 2\, d \tan(\Theta/2)$, $\Theta = 5'$ of arc for 20/20 vision. With $d = 3$ m, $h = 4.36$ mm or 8.73 mm for 20/40 vision (Millodot, 1997).

As in Experiment 1, the presentation of the two techniques was fully counterbalanced across participants and presentation of *Level* was counterbalanced between ascending or descending order. To reduce total task completion time, and because there was no significant learning effect on movement time in Experiment 1, only three *Blocks* of trials were administered. Each block had each of the 8 *D-W* combinations presented in ascending order of ID, with six trials each. The experiment duration was between 75 and 90 min.

In summary, the experimental design was:

8 participants ×
2 *Techniques* ×
6 *Levels* ×
3 *Blocks* ×
8 *D-W* combinations ×
6 trials
= 13,824 total trials

5.4. Results and Discussion

Error Rate

The mean error rate was 4% with no significant difference across independent variables.

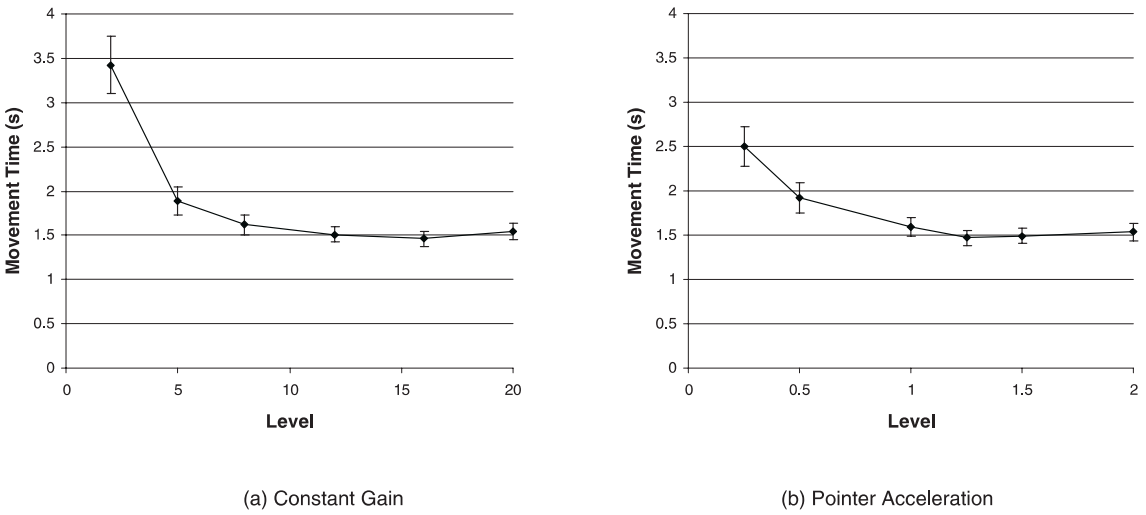
Movement Time

A repeated measures ANOVA showed a significant main effect for *Block,* $F(2, 14) = 23.4$, $p < .001$, caused by slower performance in Block 1 suggesting a learning effect; as we wished to study expert performance, we removed Block 1 data from further analysis. Subsequent analysis showed a significant main effect for *Technique,* $F(1, 7) = 18.6$, $p = .004$, with *PA* 10% faster than *CG* (*M*s = 1.697 sec and 1.882 sec, respectively). However, before drawing any conclusions, we must study the effect of *Level* for both *Techniques.*

There was a significant main effect of *Level,* $F(5, 35) = 159$, $p < .0001$, and pairwise comparison showed significant differences between the first two *CG* and *PA Levels* and all others. There was also a significant difference between the first *CG Level* and the first *PA Level* that can explain the difference observed between the two techniques. As a result, we removed the two first *CG Levels* and the two first *PA Levels* to fairly compare the two techniques within their "sweet spots" of calibration (Figure 13).

Subsequent analysis showed no significant difference between the two techniques, $F(1, 7) = 0.22$, $p = .653$. Unlike Experiment 1, where *PA* had a 6% advantage over *CG* for the smallest targets, there was no *Technique* × *W* inter-

Figure 13. **Effect of *Level* on movement time for the two techniques (error bars 95% confidence interval).**



(a) Constant Gain

(b) Pointer Acceleration

action—even with a *CG Level* of 20, accuracy was maintained with small targets. We suspect that W_S was too large to replicate the accuracy problem.

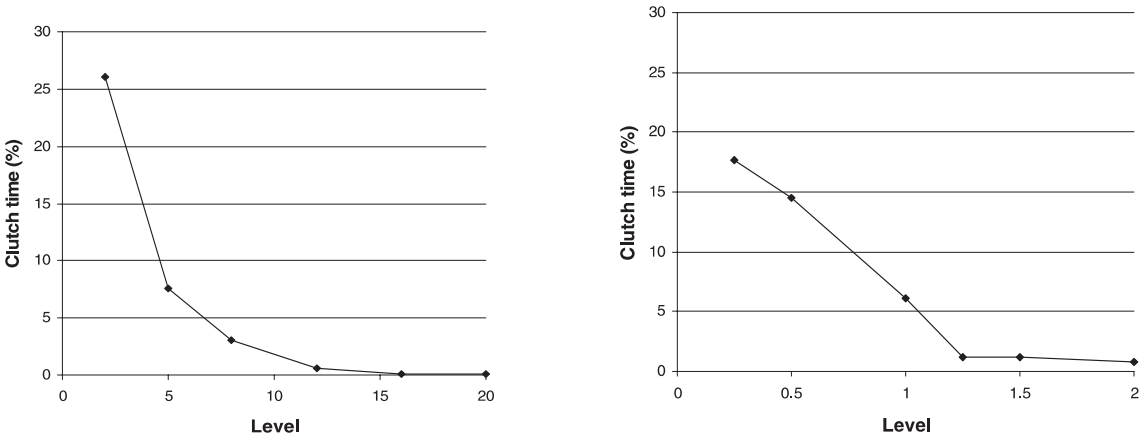
Clutching Time

There was a significant main effect for *Level* on clutching time, $F(5, 35) = 20.9$, $p < .001$. Pairwise comparison showed that clutching decreased significantly after the first two *CG* and *PA Levels* (Figure 14). *CG* Levels 2 and 5 had 26% and 7% clutching time, whereas both *PA* Levels 0.25 and 0.5 had 17% and 14%. As anticipated, increased clutching time also increased movement time, for example, by 124% with 24% clutching (Figure 13). This empirically confirms the assumptions of prior work (Jellinek & Card, 1990). We anticipated a significant effect of *D* on clutching, but the data did not confirm it—possibly because the participants formed a clutching strategy for high *D* tasks and maintained it for all levels of *D*.

Note that although the lowest level of *PA* had less clutching compared to the lowest level of CG, for other PA levels clutching was higher. For example, the amount of clutching for *CG Levels* above 8 is nearly zero, but with *PA Levels* above 1 clutching remained close to 1%. With the PA function, users need to accelerate through the low CD gain zones to get the benefit of high CD gain.

Because of the high amount of clutching that involved both arm and forearm in some conditions, we were not able to perform an analysis of limb usage like we did in Experiment 1 with the desktop-sized display. Unfortunately, we had not anticipated this behavior and our tracking equipment was

Figure 14. **Clutching time as percentage of the total movement time.**



(a) Constant Gain (b) Pointer Acceleration

not set up to track such large-scale movements. As a result, we are unable to infer reliable results on the limb usage.

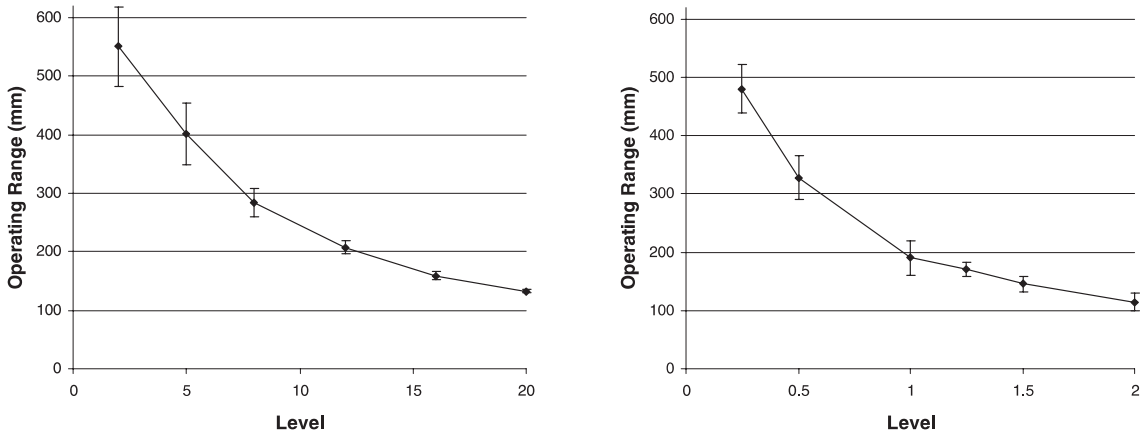
Mouse Operating Range

Analysis of the mouse operating range data showed significant main effects for *Technique,* $F(1, 7) = 65.5$, $p < .0001$; *Level,* $F(5, 35) = 140.8$, $p < .0001$; *D-W,* $F(7, 49) = 177.5$, $p < .0001$; and significant interactions between *Technique* × *Level,* $F(5, 35) = 4.3$, $p = .046$, and *Technique* × *W,* $F(2, 14) = 44.3$, $p < .0001$. The mean operating range is 335 mm for *CG* and 271 mm for *PA*. The significant effect of *Level* shows that the operating range decreases with increased *Level* (Figure 15). Pairwise comparison shows significant differences between all the *CG Levels* ($p < .035$) and all *PA Levels* ($p < .001$) expect for scales 1.0, 1.25 and 1.5.

Overshooting

Supporting the result of Experiment 1, overshooting is significantly more pronounced with *PA* ($M = 1.44\%$) than with *CG* (0.78%): $F(1, 7) = 16.8$, $p < .01$. There are also significant main effects for *Level,* $F(5, 35) = 6.5$, $p < .05$; *D,* $F(2, 14) = 11.12$, $p < .005$; and W, $F(2, 14) = 8.3$, $p < .01$. Overshooting increases with *Level* and *Distance.* Unlike Experiment 1, however, the significant effect of *Width* is not caused by increased overshooting on small targets ($W_S = 0.84\%$, $W_M = 0.73\%$, and $W_L = 1.06\%$). It seems that the participants remained accurate and confident with the small (9 mm) targets, using higher speeds to acquire them, resulting in higher overshooting.

Figure 15. **Mean mouse operating range at different levels (error bars 95% confidence interval).**



(a) Constant Gain

(b) Pointer Acceleration

User Preference

After completing each set of blocks for a *Level*, participants were asked if they found the current *Level* easier or more difficult to the previous *Level* using a 5-point Likert scale. On average, participants preferred *CG Level* 16 and a *PA Level* 1.5 for the two techniques.

Fitts' Law

As in Experiment 1, Fitts' models are based on regression analysis of the eight D-W combinations. Except for those Levels with significant clutching (*CG Level* 2 & 5 and *PA Level* 0.25), we found that performance followed Fitts' law (Figure 16 and Figure 13). This is a result that, to our knowledge, has not been shown before. The regression equations show a large negative intercept and relatively high slope, comparable to those previously observed with the touchpad (Epps, 1986), which also requires extensive clutching. We hypothesise that clutching accounts for the negative intercepts. Figure 17 shows the high slope (and hence low index of performance) for the two *Levels* with clutching.

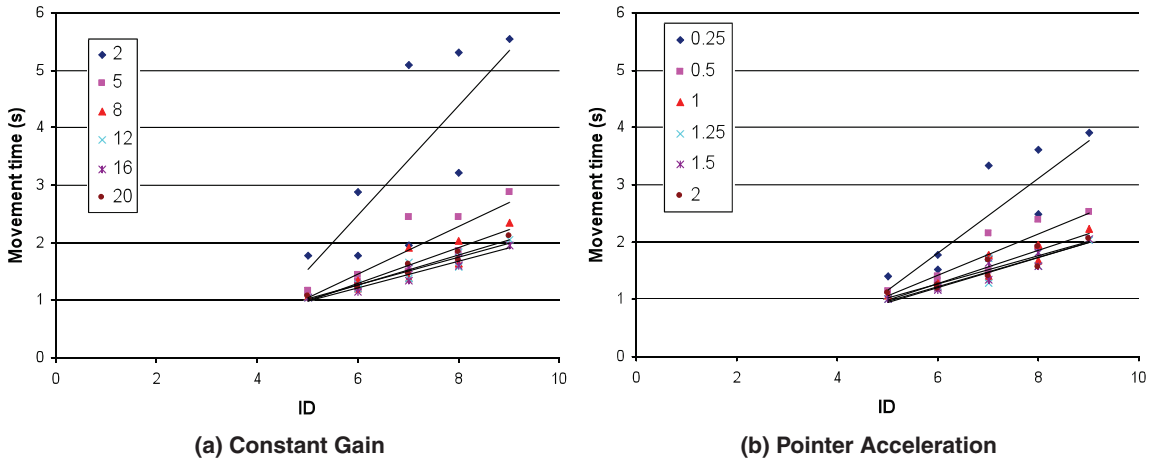
From Equation 9 of our pointer acceleration performance model (see Section 3), we expect the index of difficulty in visual space (ID) to be reduced in motor space by a quantity equal to the log of the ratio of the mean CD gain used to cover the distance (CD_D) and the mean CD gain used when near the target (CD_W). As in the first experiment, the time T_1 defined in our model (see

Figure 16. **Fitts' law regression across *level*. The columns a and b are the standard constants from Fitts' law (equation 1).**

CG Level	a	b	r^2	PA Level	a	b	r^2
2*	–3.207	0.950	0.577	0.25*	–2.079	0.650	0.711
5*	–1.007	0.412	0.734	0.5*	–0.720	0.359	0.810
8	–0.543	0.308	0.805	1.0	–0.438	0.287	0.882
12	–0.196	0.243	0.891	1.25	–0.325	0.258	0.868
16	–0.169	0.232	0.936	1.5	–0.353	0.261	0.909
20	–0.314	0.264	0.959	2.0	–0.212	0.247	0.861
8–20	–0.307	0.262	0.902	1.0–2.0	–0.333	0.263	0.884

*These levels have significant clutching.

Figure 17. **Fitts' law regression. The steep slopes correspond to levels with significant clutching.**



Section 3) is the time before the cursor crosses the border of the target. We computed the ratio CD_W/CD_D from the participants' performance data, finding it to be close to constant for all *PA Levels*, with a mean value of 0.37. As a result, we can expect to decrease the ID in motor space (ID_{mot}) by approximately $\log_2(0.37) = 1.4$ bit.

As in the first experiment, the linear regression for ID_{mot} versus ID has good regression fitness for all *PA Levels* (with slopes and intercepts being close). Using aggregated *PA Levels*, we obtain the following:

$$ID_{mot} = 0.91 * ID - 0.45 \quad r^2 = 0.993 \tag{11}$$

In spite of strong regression fitness, participants could not fully exploit the index of difficulty reduction in motor space. As in the first experiment, this was partly because of increased overshooting and increased correction time.

But here, clutching further eroded the theoretical performance advantage of PA.

6. CONCLUSIONS

We have evaluated mouse pointing performance with varying levels of CG and PA on a desktop display and on a very large, high-resolution display. We evaluated a continuous and mature pointer acceleration function used by the majority of GUI computer users. Our evaluation essentially eliminated any device quantization effects, recorded clutching actions, and used accurate 3D motion tracking equipment to analyze limb movements.

6.1. Gain Level

On both displays, and in both CG and PA techniques, we found that low levels of CD gain had a pronounced negative effect on performance. High levels of gain increased overshooting, indicating an issue with muscle control accuracy because of the reduced distances in motor-space.

Although previous research has suggested that limb bandwidth is responsible for decreased performance at low levels of CD gain (Accot & Zhai, 2001), our findings indicate that maximum limb speed and clutching time are better explanations. In our first experiment on the desktop- sized display we found that participants were limited by a maximum limb speed of about 1.5 m/sec (based on the 97th percentile of logged values with *CG* at Level 1) increasing the mean times at very low CD gain levels by 10% to 14%. In our second experiment on the very large, high-resolution display we found that clutch time was the dominant factor increasing movement time up to 124% with 24% clutching. This empirically confirms Jellinek and Card's (1990) clutching hypothesis. We also found that movement times that included clutching actions did not follow Fitts' law.

We were surprised that the seemingly high levels of CD gain in our experiments did not have a substantial impact on selection time, so we conducted a small three-participant pilot experiment on the large display. We evaluated very high CD gain levels of 8, 16, 20, 30, 40, and 50 with target distances of 4500, 2250, and 1125 mm and widths of 36, 18, and 9 mm. Surprisingly, the movement time appears to remain constant for the CD gain levels above 16. The resulting CD gain versus movement time profile is almost an L-shape, with a slight increase in time with very small targets at high CD gain levels. Essentially CD gain has little effect on pointing performance until human limits of speed and accuracy are approached.

From our results, we can define a usable range of CD gain settings between thresholds of speed and accuracy given the capabilities of a pointing device,

display, and the expected range of target widths and distances. These results have particular applications to device and pointer function developers, and future Fitts' law researchers to ensure they are selecting CD gain levels appropriate for the intended hardware, software, and application usage scenario.

To avoid clutching when acquiring distant targets, the user must increase the device operating range. Based on our experimental results, the maximum operating range used in the first experiment was 36 cm with a CD gain of 1, and in the second experiment it was 37 cm with a CD gain of 12 (where participants clutched less than 1%). We also found that device speed increased with larger operating range until a maximum limb speed affects performance. As a result, we make a conservative estimate that the maximum operating range (OR_{max}) should not exceed 30 cm. Using he largest expected target distance (D_{max}), the minimum usable CD gain (CD_{min}) can be calculated:

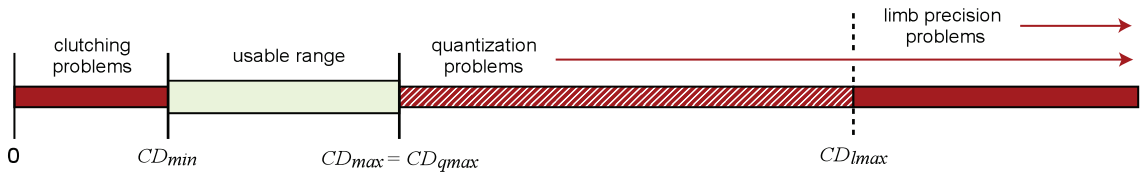
$$CD_{\min} = \frac{D_{\max}}{OR_{\max}} \tag{12}$$

The maximum usable CD gain (CD_{max}) is the lower bound of maximum usable CD gains given human limb precision and device quantization. The maximum CD gain given limb precision (CD_{lmax}) depends on the minimum expected target width (W_{min}) and the precision of the user's limbs. We observed accuracy problems with 2 mm targets and CD gain of 12. Because we used a very high resolution 1600 DPI mouse, these problems must be related to human accuracy rather than device quantization. Thus the minimum resolution of the hand and fingers ($Hand_{res}$) appears to be about 0.2 mm. Device quantization can also affect accuracy before this human threshold is reached, so we must also consider the maximum CD gain given device quantization (CD_{qmax}) which is the ratio of mouse and screen resolution ($Mouse_{res}$ and $Screen_{res}$).

$$CD_{\max} = \min\left(CD_{q\max} = \frac{Mouse_{res}(DPI)}{Screen_{res}(DPI)}, CD_{l\max} = \frac{W_{\min}}{Hand_{res}} \right) \tag{13}$$

A graphical interpretation of the usable range of CD gain is shown in Figure 18. For example, with a 400 DPI mouse, a 20″ display with 100 DPI resolution, a maximum 360 mm target distance, a minimum 2 mm target

Figure 18. **Usable CD gain range.**



width, and a maximum 250 mm operating range, we find $CD_{min} = 1.4$ and $CD_{max} = min\,(CD_{qmax}{=}4,\ CD_{lmax}{=}10) = 4$.

If a researcher calculates that CD_{max} is smaller than CD_{min} for an intended experiment, then the parameters of the experiment must to be changed—for example, the target width can be increased, the maximum distance reduced, or the resolution of the input device increased.

6.2. Pointer Acceleration Versus Constant Gain

On the standard desktop display, we found that pointer acceleration was 3.3% faster overall, and up to 5.6% faster with small targets. This confirmed the advantage predicted by our theoretical analysis; however, the benefit magnitude fell short of the theoretical potential, possibly because *PA* results in increased target overshooting. We also found that pointer acceleration follows Fitts' law with good regression fitness.

Finally we encourage researchers to use pointer acceleration rather than constant gain as a base technique for comparing new pointing technique performance. We found that the aggressive and continuous pointer acceleration functions used in modern operating systems perform better than constant gain, and many people use them already and are proficient with them.

NOTES

Acknowledgments. We thank all our study participants and Joe Laszlo for his help in finding the Mac OS X pointer acceleration curve.

Authors' Present Addresses. Géry Casiez, Bâtiment IRCICA, office 214, Parc scientifique de la haute borne, 50, Avenue Halley, 59650 Villeneuve d'Ascq, France. E-mail: gery.casiez@lifl.fr. Daniel Vogel, Department of Computer Science, University of Toronto, 10 King's College Road, Toronto, Ontario, Canada M5S 3G4. E-mail : dvogel@dgp.toronto.edu. Ravin Balakrishnan, Department of Computer Science, University of Toronto, 10 King's College Road, Room 3302, Toronto, Ontario, Canada M5S 3G4. E-mail : ravin@dgp.toronto.edu. Andy Cockburn, Computer Science and Software Engineering, University of Canterbury, Private Bag 4800, Christchurch 8140, New Zealand. E-mail: andy@cosc.canterbury.ac.nz.

HCI Editorial Record. First manuscript received September 21, 2006. Revision received June 10, 2007. Accepted by Scott MacKenzie. Final manuscript received July 9, 2007. — *Editor*

REFERENCES

Accot, J., & Zhai, S. (2001). Scale effects in steering law tasks. *Proceedings of the CHI'01 Conference on Human Factors in Computing Systems.* New York: ACM.

Arnaut, L., & Greenstein, J. (1990). Is display/control gain a useful metric for optimizing an interface. *Human Factors, 32*(6), 651–663.

Balakrishnan, R. (2004). "Beating" Fitts' law: Virtual enhancements for pointing facilitation. *International Journal of Human-Computer Studies, 61*(6), 857–874.

Balakrishnan, R., & MacKenzie, I. S. (1997). Performance differences in the fingers, wrist, and forearm in computer input control. *Proceedings of the CHI'97 Conference on Human Factors in Computing Systems.* New York: ACM.

Blanch, R., Guiard, Y., & Beaudouin-Lafon, M. (2004). Semantic pointing: improving target acquisition with control-display ratio adaptation. *Proceedings of the CHI'04 Conference on Human Factors in Computing Systems.* New York: ACM.

Bohan, M., Thompson, S. G., & Samuelson, P. J. (2003). Kinematic analysis of mouse cursor positioning as a function of movement scale and joint set. *Proceedings of the International Conference on Industrial Engineering–Theory, Applications and Practice* (pp. 442–447). Wichita, KS: Wichita State University.

Buck, L. (1980). Motor performance in relation to control-display gain and target width. *Ergonomics, 32,* 579–589.

Card, S. K., English, W. K., & Burr, B. J. (1978). Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection of a CRT. *Ergonomics, 21,* 601–613.

Epps, B. W. (1986). Comparison of six cursor control devices based on Fitts' law models. *Proceedings of the 30th Annual Meeting of the Human Factors Society* (pp. 327–331). Santa Monica, CA: Human Factors & Ergonomics Society.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology, 47,* 381–391.

Gibbs, C. B. (1962). Controller design: Interactions of controlling limbs, time-lags, and gains in positional and velocity systems. *Ergonomics, 5,* 385–402.

Graham, E. (1996). Virtual pointing on a computer display: Non-linear control-display mappings. *Proceedings of Graphics Interface 1996.* Burnaby, Canada: Human-Computer Communications Society.

Grossman, T., & Balakrishnan, R. (2005). The bubble cursor: Enhancing target acquisition by dynamic resizing of the cursor's activation area. *Proceedings of the CHI'05 Conference on Human Factors in Computing Systems.* New York: ACM.

Jellinek, H. D., & Card, S. K. (1990). Powermice and user performance. *Proceedings of the CHI'90 Conference on Human Factors in Computing Systems.* New York: ACM.

Johnsgard, T. (1994). Fitts' Law with a virtual reality glove and a mouse: Effects of gain. *Proceedings of Graphics Interface 1994.* Burnaby, Canada: Human-Computer Communications Society.

Langolf, G. D., Chaffin, D. B., & Foulke, J. A. (1976). An investigation of Fitts' law using a wide range of movement amplitudes. *Journal of Motor Behavior, 8,* 113–128.

MacKenzie, I. S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction, 7,* 91–139.

MacKenzie, I. S., & Riddersma, S. (1994). Effects of output display and control-display gain on human performance in interactive systems. *Behaviour and Information Technology, 1*(3), 328–337.

MacKenzie, I. S., Sellen, A., & Buxton, W. (1991). A comparison of input devices in elemental pointing and dragging tasks. *Proceedings of the CHI'91 Conference on Human Factors in Computing Systems.* New York: ACM.

McCormik, E. J. (1976). *Human factors in engineering and design.* New York: McGraw-Hill.

McGuffin, M., & Balakrishnan, R. (2002). Acquisition of expanding targets. *Proceedings of the CHI'02 Conference on Human Factors in Computing Systems.* New York: ACM.

Meyer, D. E., Smith, J. E., Kornblum, S., Abrams, R. A., & Wright, C. E. (1988). Optimality in human motor performance: Ideal control of rapid aimed movements. *Psychological Review, 95,* 340–370.

Meyer, D. E., Smith, J. E. K., Kornblum, S., Abrams, R. A., & Wright, C. E. (1990). Speed-Accuracy tradeoffs in aimed movements: Toward a theory of rapid voluntary action. In M. Jeannerod (Ed.), *Proceedings of Attention and Performance XIII* (pp. 173–226). Hillsdale, NJ: Erlbaum.

Microsoft. (2002). *Pointer ballistics for Windows XP.* Seattle, WA: Author. Retrieved July 9, 2007, from http://www.microsoft.com/whdc/device/input/pointer-bal.mspx

Millodot, M. (1997). *Dictionary of optometry and visual science.* Butterworth-Heinemann (http://en.wikipedia.org/wiki/Butterworth-Heinemann).

Phillips, J. G., & Triggs, T. J. (2001). Characteristics of cursor trajectories controlled by the computer mouse. *Ergonomics, 44,* 527–536.

Plamondon, R., & Alimi, A. M. (1997). Speed/accuracy tradeoffs in target-directed movements. *Behavioral and Brain Sciences, 20,* 279–349.

Rosenbaum, D. (1991). *Human motor control.* San Diego, CA: Academic.

Trankle, U., & Deutschmann, D. (1991). Factors influencing speed and precision of cursor positioning using a mouse. *Ergonomics, 34,* 161–174.

Zhai, S., Milgram, P., & Buxton, W. (1996). The influence of muscle groups on performance of multiple degree-of-freedom input. *Proceedings of the CHI'96 Conference on Human Factors in Computing Systems.* New York: ACM.

APPENDIX. SOURCE OF WINDOWS XP/VISTA AND MAC OS X POINTER ACCELERATION CURVES

The Windows XP and Mac OS X pointer acceleration curves are defined in a lookup table containing the speed of the cursor and the speed of the mouse. The speed of the cursor is then linearly interpolated. Windows XP pointer acceleration uses a mother curve stored in the registry (HKEY_CURRENT_USER\Control Panel\Mouse). When moving the cursor left and right in the user preference panel with "Enhance pointer precision" enabled, the mother curve is then scaled along the Y axis where the default setting corresponds to a scale of 0.5.

Mac OS X uses different pointer acceleration curves for each position of the slider on the mouse panel setting. The curves were found by analyzing the code for the mouse found on the Darwin Project (http://opensource.apple.com/

darwinsource./10.4/IOHIDFamily-164/IOHIDSystem/IOHIPointing) and using a developer utility on Mac OS X to dump the device tree info to get the HIDPointerAccelerationTable. It is interesting to see that it is possible to use the pointer acceleration technique or constant CD gain of various values on Windows XP but it is only possible to use the pointer acceleration technique on Mac OS X or a constant CD gain of 1.

The Effect of Spring Stiffness and Control Gain with an Elastic Rate Control Pointing Device

Géry Casiez[†] and Daniel Vogel[††]

[†]LIFL & INRIA Lille
University of Lille, FRANCE
gery.casiez@ lifl.fr

[††]Department of Computer Science
University of Toronto, CANADA
dvogel@ dgp.toronto.edu

ABSTRACT

Isometric and elastic devices are most compatible with a rate control mapping. However, the effect of elastic stiffness has not been thoroughly investigated nor its interaction with control gain. In a controlled experiment, these factors are investigated along with user feedback regarding ease-of-use and fatigue. The results reveal a U-shaped profile of control gain vs. movement time, with different profiles for different stiffness levels. Using the optimum control gain for each stiffness level, performance across stiffness levels was similar. However, users preferred lower stiffness and lower control gain levels due to increased controller displacement. Based on these results, design guidelines for elastic rate control devices are given.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Human factors

Keywords: elastic, control gain, stiffness, rate control

INTRODUCTION

The mouse is an efficient pointing device [4,13,14], but there are environments without a flat surface where the mouse is not practical. Laptop manufacturers have responded with alternative pointing devices such as the touch pad. Like the mouse, the touch pad is an *isotonic* input device (it is free-moving and uses X-Y position as input) with a *position control mapping* (the input is mapped to an X-Y cursor position) [21]. However, the touch pad has a very small input area and requires frequent clutching which degrades performance [5]. Clutching can be reduced by increasing the ratio of control movement to display movement (*Control-Display gain*, or *CD gain*) [1,3,9,10], but very high CD gain levels can hurt performance [1,9,10].

Alternatively, clutching can be removed altogether by using a *rate control mapping* where the device input is mapped to a cursor velocity and direction. A rate control mapping is more suitable for an *isometric* or *elastic device* since they have a *self-centering mechanism* to return the device to a neutral state when released [21]. Isometric devices, such as the TrackPoint [15,17], do not perceptibly move and instead measure the force applied. Unfortunately, isotonic devices seem to be faster than isometric devices [6,7,14,16]. However, this difference could be due to non-optimal device parameters. For example, isometric devices are also affected by *control gain* [9,11] and after some informal parameter tuning, Zhai found no difference between isotonic and isometric 6 DOF devices [21]. Another issue is that isometric devices lack proprioception, the human sense of position and movement of limbs, and may increase fatigue [21].

In contrast, elastic devices have an effector which can be displaced over a certain operating range, with a spring applying an opposite force to self-centre (Figure 1). Yet, with the exception of Zhai's small pilot experiment with a 6 DOF input device [21], little is known about the effect of elastic device spring stiffness and there is no clear conclusion for the added influence of control gain [4,7,9,11,18]. This, in spite of elastic devices appearing in the literature [5,8,12]. Without an understanding of the combined effect of elastic stiffness and control gain, tuning parameters for isometric or elastic devices will continue to be ad hoc.

In this paper we present an experiment that systematically evaluates the interaction between control gain and stiffness using a high performance force feedback device. We found that the control gain vs. movement time has a U-shaped profile and in addition, that proprioception influences the shape of the U: with a carefully chosen control gain, elastic and isometric devices can perform equally well. However, our participants preferred more elasticity. We also show that operating range is not only affected by stiffness, but also by control gain. Finally, using these results, we give guidelines for the design and use of elastic and isometric rate control devices given the stiffness and operating range.

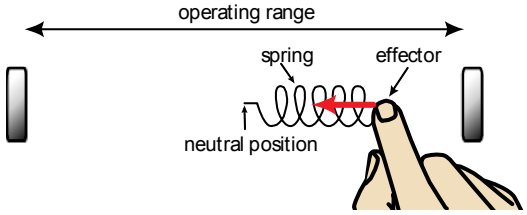


Figure 1. Elastic device composed of a spring attached to an effector. The resistive force is proportional to the effector displacement which is limited by the operating range.

CHI 2008, April 5–10, 2008, Florence, Italy.

BACKGROUND AND RELATED WORK

Elastic device stiffness

An elastic device is composed of an effector and spring placed in a constrained movement area, called the operating range (Figure 1). The user pushes on the effector and the device measures either the distance from the effector to the neutral position (the displacement) or the applied force, and uses this as input.

$$F = -k \cdot d \quad (1)$$

Using Hookes law [20], the reactive force F is a linear function of spring stiffness k and displacement d (Equation 1). Spring stiffness is measured in newtons per metre ($N.m^{-1}$). To give some idea of the range of stiffness, a typical office stationary rubber band is between 30 $N.m^{-1}$ and 300 $N.m^{-1}$. With high spring stiffness, the effective range of displacement decreases and force is a more appropriate measurement. In the extreme case, when stiffness is infinite with no displacement, the behavior is that of an isometric device. With lower spring stiffness, the effective range of applied force decreases and the effector distance is the more appropriate measurement. When stiffness is set to zero, the behavior is that of an isotonic device.

Effect of stiffness

To our knowledge, only Zhai has studied the influence of stiffness on user performance, but only in a pilot study with 2 users [21]. He used a custom 6 DOF built device that could vary the stiffness by changing the number of elastic bands. Although the range of stiffness evaluated is not reported he found optimal performance around 120 $N.m^{-1}$.

In a follow-up study, Zhai compared a 6 DOF isometric device with a 6 DOF elastic device (120 $N.m^{-1}$ – 20 mm operating range) in a 6 DOF docking task [21]. He did not find a significant difference in performance, but found differences in user fatigue. Unlike an isometric device, an elastic device can be displaced to provide *proprioception* – our sense of the joint positions in our body. Zhai summarizes the tradeoff with elastic stiffness settings: "For the sake of compatibility with rate control, a stiff (therefore strongly self-centered) elasticity is desirable. However, a stiff (near isometric) elastic device provides less rich proprioception than a loose one that allows more movement within a range of non-fatiguing forces." [21] In spite of Zhai's argument there are no design guidelines (aside from his initial pilot study results) for selecting an optimum stiffness value which balances strong self-centering and rich proprioception.

Control gain

With an *isotonic* position control device such as a mouse, *CD gain* is a unit free ratio since both input and output are corresponding movements. For example, with a CD gain of 2, the display pointer moves twice as far as the corresponding movement of the control device.

However, with isometric or isotonic rate control devices, input and outputs are expressed in different units. With an *isometric* device, the applied force F, expressed in N, is mapped to a pointer velocity V, expressed in $m.s^{-1}$. Thus, the isometric *control gain* ($CG_{isometric}$) is expressed in $m \cdot s^{-1} \cdot N^{-1}$ and the mapping function is:

$$V = CG_{isometric} \cdot F \quad (2)$$

As a simple illustration, given a control gain of 2 $m \cdot s^{-1} \cdot N^{-1}$, the same force on the isometric controller will produce twice the velocity of the cursor relative to a control gain of 1 $m \cdot s^{-1} \cdot N^{-1}$ [11].

With a pure isotonic rate control device, the displacement distance d, expressed in meters, is mapped to a pointer velocity V. Here, the isotonic control gain ($CG_{isotonic}$) is expressed in s^{-1} and the mapping function is:

$$V = CG_{isotonic} \cdot d \quad (3)$$

With an elastic device, as the pointer velocity can be computed from either displacement or applied force, either Equation 2 or 3 can be used. Moving from one mapping function to the other is simply done by dividing or multiplying the control gain by the spring stiffness, since the resistive force is proportional to displacement (Equation 1):

$$CG_{isotonic} = k \cdot CG_{isometric} \quad (4)$$

Effect of control gain

Constant control gain transfer functions

Gibbs [9] tested 7 values of control gain with an elastic rate control device and found that performance follows a U-shape, described by the following equation:

$$T = 0.949 + 0.255 \cdot CG_{isotonic} + \frac{0.444}{CG_{isotonic}} \quad (5)$$

This gives an optimal performance at approximately 1.3 s^{-1}, and then degrades with higher or lower levels. He used a "slightly loaded" joystick with a 100 mm operating range (±25° of angular movement and 120 mm joystick length). For comparison sake, we estimated "slightly loaded" to be similar to the elasticity of a loose rubber band and assigned a stiffness of 30 $N.m^{-1}$. With this, we can estimate the control gain range, using Equation 4, to be 0.01 – 0.1 $m.s^{-1}.N^{-1}$ The stimulus was a 3 mm target at a distance of 22.5 mm (with an index of difficulty, or ID of 3). Error rate was not a reported factor since each pointing task had to be completed successfully.

Using an isometric rate control device, Kantowitz and Elvers [11] tested two levels of control gain (0.04 and 0.08 $m.s^{-1}.N^{-1}$) and found no significant difference. Error rates were between 15 and 30%. Interestingly, their study also evaluated an isometric position control device with control gain levels of 1 and 2. They found that Fitts' law held for both devices, with higher slopes for the rate control condition and negative intercepts close to 0.8s. Using an isometric rate control device, Epps [7] also found good regression fitness with a negative intercept equal to 0.587 s.

Study	Device	Control gain	ID	Width (mm)	Dist (mm)	Result
Gibbs [9]	Elastic joystick	Constant Function 7 values (0.01 – 0.1 $m.s^{-1}.N^{-1}$)*	3	3	22.5	**U-Shape**: best control gain at 1.3 s^{-1} (0.03 $m.s^{-1}.N^{-1}$*)
Kantowitz & Elvers [11]	Isometric joystick	Constant Function (0.04, and 0.08 $m.s^{-1}.N^{-1}$)	3.5 - 5.5	7 - 11	63 - 168	**No effect**
Card et al. [4]	Isometric joystick	Non-Linear Function (0 to 0.008 $m.s^{-1}.N^{-1}$)	0.5 – 6	2.46 – 24.6**	10 – 160	**High Error Rates**
Rutledge et al. [18]	Isometric pointing stick	3 Non-Linear and 3 Constant Functions (0.075, 0.15, 0.3 $m.s^{-1}.N^{-1}$)	1.75 - 6	2.8 – 14	Random	**Negative Effect for High control gain:** performance decreased with control gain 0.3
Zhai [21] (2 subj. pilot)	Elastic device	Non-Linear Function (control gain values unclear)	6 DOF Docking Task			**U-Shape**

Table 1: Summary of prior work evaluating the effect of control gain with rate control devices. (*control gain levels estimated using 30 $N.m^{-1}$ stiffness; **width between 1 and 10 characters with 1 character = 2.46 mm [13])

Non-Linear Control Gain Transfer Functions

Researchers have also experimented with non-linear transfer functions for control gain.

Card et. al [4] tested an isometric joystick with a simple non-linear parabolic transfer function. The transfer function mapped forces below 4 N to a control gain of 0 $m.s^{-1}.N^{-1}$ (the cursor did not move) and 0.008 $m.s^{-1}.N^{-1}$ for forces above. In spite of good Fitts' law regression fitness, the joystick had an error rate between 10 and 15% and lower performance than the mouse. The authors attributed this to the non-linear transfer function.

Rutledge et al. [17,18] compared a custom non-linear transfer function with three constant control gain settings (0.075, 0.15, 0.3 $m.s^{-1}.N^{-1}$) and 2 parabolic non-linear functions for an isometric rate control device. For all functions, performance decreased with control gain 0.3. Also, the constant functions were found to be faster when the index of difficulty is below 3.5 bits and slower above. Results for all functions were found to follow Fitts' law. Their function has been improved by adding a negative inertia filter to the pointer motion [2].

Zhai ran a small pilot experiment with two subjects (an expert and a novice) to tune the control gain of an elastic device and an isometric device [21]. He used the non-linear function: $V = F^{CG}$. Like Gibbs, he found that performance was optimal with a mid-level control gain, but since each level was tested with only 12 trials, it is unclear if his findings are statistically significant. Zhai characterized this as a "U-Shape," based on a plot of mean selection times over a range of control gains. However, he does not report the maximum force and the units of control gain are unclear.

Summary

There is no clear conclusion for the effect of control gain on user performance for rate control devices (Table 1). Gibbs [9] and Zhai [21] found a U-shaped performance curve; Rutledge et al. [18] found performance degraded with high control gains; and Kantowitz and Elvers [11] did not find any effect. However, in these experiments, the range of control gain levels was too small or the target distances and widths were conservative. Only Rutledge et al. [18] compared non-linear and constant control gain transfer functions, and they found that performance decreased with high control gain. In addition, only Zhai's 2-person pilot study has examined the influence of stiffness on user performance [21]. Yet, in his follow up evaluation of elastic and isometric devices, no significant effect on performance was found. This, in spite of participants noting increased fatigue with the isometric device and Zhai's argument that the elastic device provides richer proprioception. Finally, the interaction between control gain and stiffness has not been evaluated.

EXPERIMENT

We wished to investigate the effect of control gain and stiffness for elastic devices ranging from isotonic to nearly isometric. Rather than confound the experiment with a custom non-linear control gain transfer function, we focus only on constant transfer functions. If there is an effect for control gain, then researchers can design non-linear transfer functions guided by these base line results.

Apparatus

We used a single Phantom Desktop haptic device to simulate the different elastic devices [19]. This eliminated extraneous intra-device differences, such as ergonomics, size and sensitivity, while also providing an efficient way to administer the experiment without having to frequently swap custom-built elastic devices during a session.

The Phantom uses a stylus connected to a force-feedback armature to produce haptic feedback. It has an 1100 DPI nominal resolution and an operating range of 160 mm. With a maximum resistive force of 7.9 N, the Phantom can simulate a maximum stiffness of 1860 $N.m^{-1}$. This allowed us to simulate a large stiffness range, from isotonic to near isometric (the maximum stiffness felt like an inflated bicycle tire). Using the force feedback, the movement of the

effector was constrained to a single degree of freedom parallel to the display and 2 cm above the table. To avoid instability when selecting the target, participants used their non-dominant hand to trigger a button on a mouse.

The pointer velocity was computed from the resistive force and control gain using Equation 2. After running a pilot experiment, we observed that a dead-band was needed to prevent the pointer from moving without any apparent force applied to the device. The problem is that the Phantom will not fully self-centre due to the presence of *back-drive friction*. We found that using a deadband force, F_{DB}, of 0.055 N stabilized self-centering across all stiffness levels. This value is close to the Phantom's rated backdrive friction of 0.06 N and similar to values used previously [4,18]. Including the deadband, the pointer velocity is computed as:

$$V = \begin{cases} 0 & if \left|F\right| < F_{DB} \\ Sgn(F) \cdot CD \cdot (\left|F\right| - F_{DB}) & o.w. \end{cases} \quad (6)$$

Where Sgn(x) = -1 if x < 1, 1 if x > 1, 0 if x = 0.

The simulation is coded in C++ and uses OpenGL for the display and OpenHaptics toolkit to control the Phantom. The frequencies of the visual and haptic renderings were 60 Hz and 1000 Hz respectively.

Task

The task was a reciprocal one dimensional pointing task (Figure 2). Each experimental trial began after the previous target was successfully selected and ended with the selection of the current target. After the current target was successfully selected, it turned grey, and the next target to be selected, on the other side of the screen, turned white. If a participant missed a target, a sound was heard and an error was logged. Participants had to successfully select the current target before moving to the next, even if it required multiple clicks. The pointer was not constrained to the bounds of the screen to avoid using the edges to assist in target acquisition.

Participants

Sixteen right-handed people (14 male, 2 female) with a mean age of 25.2 (SD = 2.9) participated. All were regular computer users (at least 2 hours per day) and had little or no experience with isometric devices such as the TrackPoint or SpaceMouse, nor did they have previous experience with the Phantom.

Design

A repeated measures within-subjects design was used. The independent variables were STIFFNESS (0, 30, 120, 800, 1600 $N.m^{-1}$), Control Gain (CG) (0.05, 0.1, 0.3, 0.5, 0.7 $m.s^{-1}.N^{-1}$), target DISTANCE (D_L = 300 mm, D_M = 150 mm, D_S = 75 mm), and target WIDTH (W_L= 8 mm, W_M = 4 mm, W_S = 2 mm). The operating range is treated as a dependant variable since it is a function of CG and STIFFNESS.

The nine DISTANCE and WIDTH (D-W) combinations give five Fitts' Indices of difficulty (ID) ranging from 3.4 to 7.2. This design gives us redundant points at each ID, allowing us to examine different effects of DISTANCE and WIDTH. The tradeoff is that our ID range of 3.8 is narrower than Card et al. [4] or Epps [7] (with 6.18 and 6.04 respectively). Mackenzie [13] discusses this issue, noting extremely narrow ID ranges, such as 2, as problematic. Our mid-point choice is an attempt to balance these range breadth with redundant points for each ID.

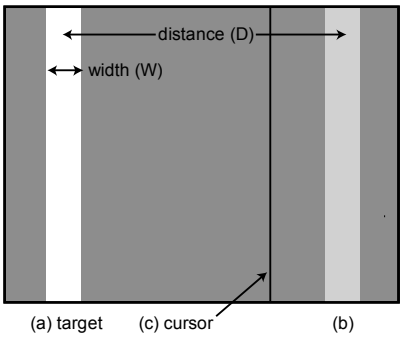


Figure 2. Experimental display. The targets are solid vertical bars, equidistant from the centre of the display. The target to be selected was coloured white (a), and the previous target, which was the starting position, light grey (b). The cursor was represented by a one pixel thick vertical black line (c).

The STIFFNESS levels range from pure isotonic (0 $N.m^{-1}$) to nearly pure isometric (1600 $N.m^{-1}$ with an operating range less than 1.0 mm). The intermediate STIFFNESS levels of 120 $N.m^{-1}$ and 30 $N.m^{-1}$ enable comparison with the results of Zhai and Gibbs respectively. We also added a stiff spring of 800 $N.m^{-1}$. The CG levels were chosen after running a pilot study with two participants. Since there is no resistive feedback with a stiffness of 0 $N.m^{-1}$, we used a 30 $N.m^{-1}$ stiffness to compute the velocity making the only difference between these two levels the resistive force. Note that the *SpaceMouse Cadman* has a stiffness of approximately 2500 $N.m^{-1}$ and an operating range of 4mm.

Participants completed two consecutive BLOCKs of trials for each combination of CG and STIFFNESS. Each BLOCK consisted of 27 trials: 3 repetitions of the 9 D-W combinations. The D-W combinations were presented in ascending order of ID within a single BLOCK to avoid drastic changes in difficulty. After 3 trials, a message displayed the cumulative error rate and encouraged participants to conform to a 4% error rate by speeding up or slowing down. The presentation order of CG and STIFFNESS was counterbalanced from low-to-high and high-to-low across participants using a Latin Square design.

Before starting the experiment, participants had a 5-10 minute training period to get used to controlling the cursor with 120 $N.m^{-1}$ STIFFNESS and CG of 0.1 $m.s^{-1}.N^{-1}$. After the two blocks were completed for each STIFFNESS and CG combination, the participant rated the settings ease-of-use

and fatigue using a 5 point Likert scale. The experiment lasted approximately 120 minutes.

In summary, the experimental design was:

16 participants ×

5 STIFFNESS × 5 CG ×

2 BLOCKS × 9 D-W combinations × 3 trials

= 21,600 total trials

RESULTS

The dependent variables were error rate, movement time, operating range, and maximum force.

Performance

Error rate

Repeated measures analyses of variance showed that the order of presentation of STIFFNESS or CG had no significant effect or interaction on error rate, indicating that a within-participants design was appropriate. We also found no significant effect or interaction for BLOCK indicating there was no presence of a learning effect. Repeated measures analysis of variance found an expected significant main effect for WIDTH ($F_{2,30} = 48.5$, $p < 0.0001$) on error rate, but also significant main effects for STIFFNESS ($F_{4,60} = 10.5$, $p < 0.0001$) and CG ($F_{4,60} = 25.1$, $p < 0.0001$).

However, we have to be cautious before drawing any conclusions regarding these main effects because of a significant interaction between STIFFNESS and CG ($F_{16,240} = 4.3$, $p < 0.0001$) which shows that error rate increases at different rates according to CG and STIFFNESS (Figure 3) – no other significant interaction was found. For example, while participants maintained an error rate between 2.8 and 4.7% for 30 $N.m^{-1}$ STIFFNESS, they failed to do so with a STIFFNESS of 1600 $N.m^{-1}$ where the error rate ranged from 2.6% to 16.5%. This is in spite of encouraging participants to try and maintain a 4% error rate for all settings (note that Kantowitz et al. [11] and Card et al. [4] also found error rates between 10 and 15%).

Given this STIFFNESS and CG interaction, a more meaningful comparison of error rate across STIFFNESS levels should use optimum levels of CG. We first define the *optimum CG range* for each STIFFNESS level as the range of CG levels with a statistically significant lower error rate. In other words, the error rates for each CG within the optimum range are statistically better than those for all CG levels outside the range, and the error rates for CG levels within the range have no statistical difference. Using pairwise comparisons, we found the following optimum ranges:

- STIFFNESS 0: CG [0.05 – 0.5] (all $p < 0.02$)
- STIFFNESS 30: CG [0.05 – 0.3] (all $p < 0.044$)
- STIFFNESS 120: CG [0.05 – 0.1] (all $p < 0.014$)
- STIFFNESS 800: CG [0.05 – 0.1] (all $p < 0.022$)
- STIFFNESS 1600: CG [0.05 – 0.1] (all $p < 0.004$)

Repeated measures analysis of variance using the optimum CG range found a significant main effect ($F_{4,60} = 4.1$, $p = 0.006$) for STIFFNESS on error rate. Pairwise comparisons found that STIFFNESS 0 had a significantly lower error rate of 5% compared to STIFFNESS 30, 120, and 1600 ($p < 0.007$) (Figure 4).

Our results indicate that CG has an impact on the error rate, and by comparing the optimal CG ranges, we found a slight error rate advantage with a STIFFNESS of 30 $N.m^{-1}$. However, we note that all error rates are within an acceptable range at close to 4%.

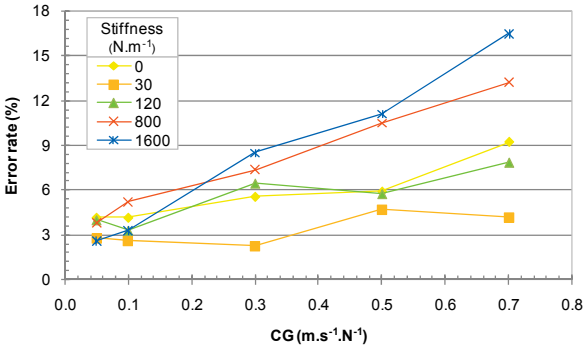


Figure 3: Mean error rate for CG and STIFFNESS.

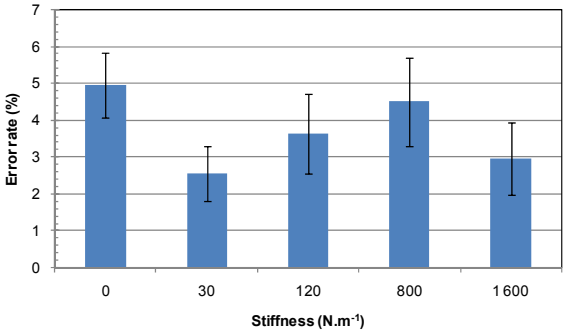


Figure 4: Mean error rate for STIFFNESS using optimal CG range. Error bars represent 95% confidence interval. All confidence intervals are computed from the standard deviation quantifying dispersion among the sample of participants.

Movement time

Movement time is defined as the time it took to move from the previous target to the current target and correctly select it. Targets that were not selected on the first attempt were marked as errors, but were still included in the timing analysis (the same significant effects were found with and without errors).

Repeated measures analyses of variance showed that the order of presentation of STIFFNESS or CG had no significant effect or interaction on movement time, indicating that a within-participants design was appropriate. We also found no significant effect or interaction for BLOCK indicating there was no presence of a learning effect. As predicted by

Fitts' law, repeated measures analysis of variance found significant effects for DISTANCE ($F_{2,16}$ = 280, $p < 0.0001$) and WIDTH ($F_{2,16}$ = 190, $p < 0.0001$) on movement time, but also main effects for STIFFNESS ($F_{4,32}$ = 3.6, p = 0.016) and CG ($F_{4,32}$ = 6.9, $p < 0.0001$).

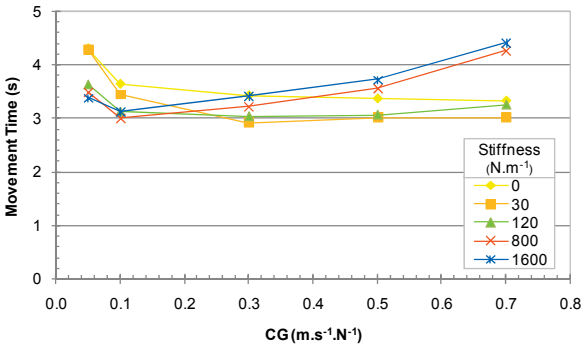


Figure 5: Mean movement time for CG and STIFFNESS.

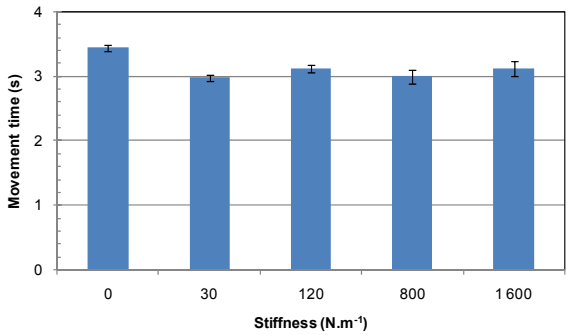


Figure 6: Mean movement time for STIFFNESS using the optimal CG range. Error bars represent 95% confidence interval.

Just as with error rate, the significant interaction between STIFFNESS and CG ($F_{16,28}$ = 9.2, $p < 0.0001$) provides more meaningful interpretation (Figure 5). Large differences were found between the best and worst performing CG levels for a given STIFFNESS. For example, we found a 42% increase between the movement times for the lowest and highest CG for the 30 $N.m^{-1}$ STIFFNESS, and 41% increase between the 0.1 and 0.7 CG levels for the 1600 $N.m^{-1}$ STIFFNESS. Thus, as with error rate, more meaningful comparisons across STIFFNESS levels should be done using the optimum CG range. For the most part, we found that the range decreased as STIFFNESS increases:

- STIFFNESS 0: CG [0.1 – 0.7] (all $p < 0.01$)
- STIFFNESS 30: CG [0.3 – 0.7] (all $p < 0.002$)
- STIFFNESS 120: CG [0.1 – 0.7] (all $p < 0.02$)
- STIFFNESS 800: CG 0.1 (all $p < 0.05$)
- STIFFNESS 1600: CG 0.1 (all $p < 0.05$)

Repeated measures analysis of variance using the optimum CG range found a significant main effect of STIFFNESS ($F_{4,52}$ = 5.7, p = 0.001) on movement time (Figure 6). Pairwise comparisons found a significant difference between STIFFNESS 0 and all other STIFFNESS values ($p < 0.02$). At 3.8s, the isotonic equivalent STIFFNESS value of 0 was 15% slower than the elastic or isometric STIFFNESS values. This confirms the results found by Zhai [21] while extending it to a wider range of stiffness values. However, Zhai found a 47% difference between the isotonic and isometric rate control devices, which may be explained by the 6 DOF docking task or the different device form factors he used. Perhaps most interesting is that our results suggest that even with a very low stiffness of 30 $N.m^{-1}$, participants can take advantage of the resistive feedback to more accurately control the device.

Fitts' Law analysis

We also found a significant STIFFNESS × CG × ID interaction on movement time ($F_{64,768}$=3.5, $p<0.01$) which led us to a Fitts' law analysis. To perform the Fitts law analysis, we computed the effective width for each STIFFNESS × CG × D-W combination, according to MacKenzie's formulation since the error rate is not constant [13]. By using the nine D-W combinations rather than an aggregate time for the five IDs, independent effects of DISTANCE and WIDTH may be exposed. We found negative intercepts between -1.75 s and -0.11 s and slopes between 0.64 and 1.0 $s.bit^{-1}$ for those settings which followed Fitts' law. These are close to those found in previous work [9,7].

Fitts' law does not hold for all settings (Table 2 & Figure 8). The settings which do not hold are with the largest distance and low CG levels, or the smallest WIDTH and high CG levels. These results explain the STIFFNESS × CG × ID interaction.

Operating range

The operating range is defined as the maximum distance between the two extreme positions of the effector. We expected the main effect of STIFFNESS on operating range ($F_{4,32}$ = 173.5, $p < 0.0001$) since there is a dependency, but we were more interested in how CG and DISTANCE affected operating range. Repeated measures analysis of variance found a significant main effect for DISTANCE ($F_{2,16}$ = 529.6, $p < 0.0001$) with operating range increasing with DISTANCE. Pairwise comparisons found significant differences between all DISTANCE values ($p < 0.0001$).

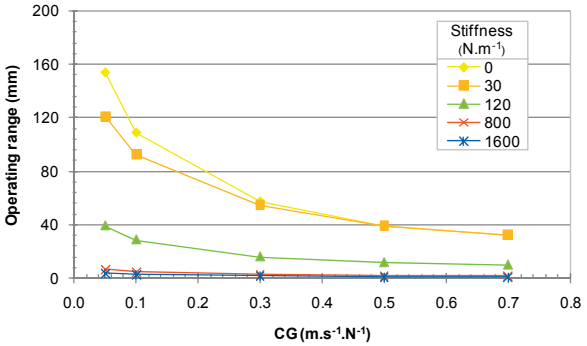


Figure 7: Mean operating range for CG and STIFFNESS.

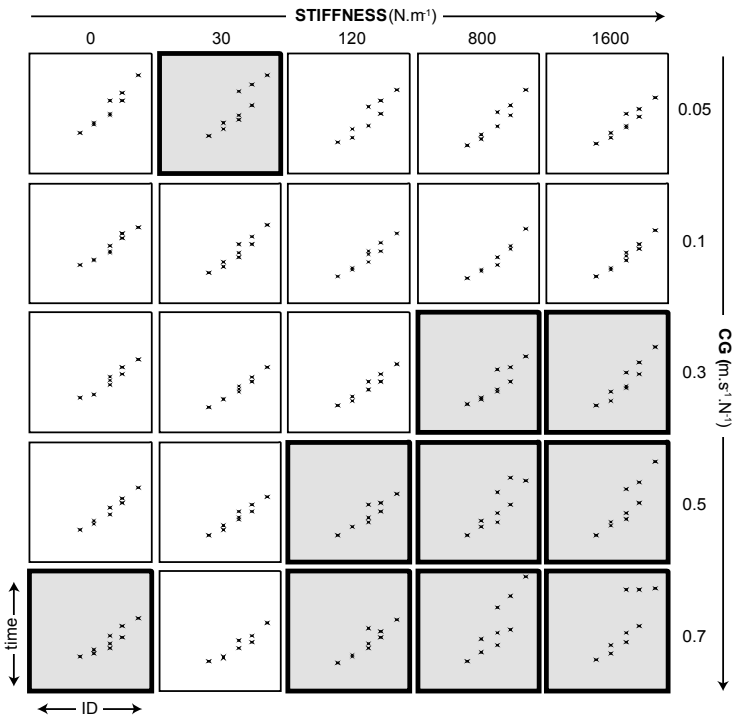


Figure 8. Fitts' law regression plots by STIFFNESS and CG. Plots highlighted in bold have low fitness ($r^2 < 0.8$). In these cases, different D-W combinations have different movement times in spite of having the same ID. For example, the combinations D_LW_L, D_MW_M, and D_SW_S all have ID = 5.3, but for STIFFNESS and CG settings of 1600 N.m^{-1} and 0.7 m.s^{-1}.N^{-1} the movement times appear different.

STIFFNESS N.m^{-1}	CG m.s^{-1}.N^{-1}	a	b	r^2
0	0.05	-0.92	1.00	0.90
	0.1	-0.11	0.71	0.94
	0.3	-0.84	0.81	0.83
	0.5	-0.75	0.80	0.94
	0.7	0.10	0.65	0.57
30	0.05	-1.25	1.04	0.75
	0.1	-0.57	0.75	0.88
	0.3	-0.73	0.69	0.93
	0.5	-0.83	0.73	0.81
	0.7	-0.39	0.64	0.82
120	0.05	-1.32	0.94	0.83
	0.1	-0.62	0.70	0.86
	0.3	-0.79	0.75	0.83
	0.5	-0.19	0.62	0.69
	0.7	-0.29	0.70	0.64
800	0.05	-1.50	0.93	0.89
	0.1	-1.75	0.89	0.88
	0.3	-0.99	0.85	0.53
	0.5	-1.51	1.05	0.45
	0.7	-3.07	1.65	0.42
1600	0.05	-1.05	0.82	0.87
	0.1	-1.02	0.76	0.92
	0.3	-1.84	1.03	0.63
	0.5	-2.50	1.27	0.52
	0.7	-2.94	1.67	0.36

Table 2: Fitts' Law regression values for STIFFNESS and CG: *a* is the intercept, *b* is the slope, r^2 is the fitness (computed as means over participants). Lines highlighted in grey have low fitness ($r^2 < 0.8$).

We also found a main effect for CG ($F_{4,32} = 137.5$, $p < 0.0001$), but again the significant interactions are more relevant. We found a STIFFNESS × CG interaction ($F_{16,128} = 50.8$, $p < 0.0001$) which shows how the operating range decreases with increased CG but at different rates across STIFFNESS (Figure 7). For example, the operating range decreases from 154 mm to 32 mm for STIFFNESS 0 but decreases from 3.5 mm to 0.7 mm for STIFFNESS 1600 N.m^{-1}. A significant CG × STIFFNESS × DISTANCE interaction ($F_{32,256}=10.8$, $p < 0.0001$) revealed that with low STIFFNESS and low CG, the operating ranges for different DISTANCE can vary by as much as 40 mm. But, as the STIFFNESS and CG increase, the difference across DISTANCE becomes very small (less than 1mm in some cases).

These operating range curves can assist designers of elastic rate control devices. For example, very little physical movement is required for devices with stiffness above 800 N.m^{-1} with most distances. In addition, the relationship between these three factors can be further formalized using a multiple regression analysis.

Using a regression analysis with a variety of possible model predictors, we found the ratio of distance to stiffness *D/k* and the inverse of the product of stiffness and control gain *1/(k · CG)* to be significant predictors ($p<0.0001$). A forward stepwise analysis found the contribution of each predictor to be:

$$OR = 2 + \frac{5 \cdot D}{k} + \frac{146}{k \cdot CG} \quad (7)$$

With this model, *1/(k · CG)* explains 83.7% of the variance (std err 5.2) and *D/k* 13.8% (std err 0.28). With a significant r^2 value of 0.975 ($F_{2,59}=1114$, $p < 0.0001$) and no within-predictor collinearity (VIF=1.37, tolerance 0.73), this model predicts the operating ranges used by our experimental settings within -80% and 41% of the real data, with a mean prediction within -15%.

Maximum force

The maximum force is the maximum absolute amplitude of force applied to the end effector during a trial. Repeated measures analysis of variance show a significant main effect of STIFFNESS ($F_{4,32} = 57.0$, $p < 0.0001$), CG ($F_{4,32} = 84.2$, $p < 0.0001$), and DISTANCE ($F_{2,16} = 58.4$, $p < 0.0001$) on maximum force. But again, the significant STIFFNESS × CG interaction is most interesting ($F_{16,128} = 17.2$, $p < 0.0001$) (Figure 9). It shows that the maximum force applied on the effector decreases at different rates across stiffness as the CG increases. For example, it decreases

from 2.8 N to 0.6 N for the 1600 N.m^{-1} STIFFNESS and from 1.6 N to 0.5 N for the 30 N.m^{-1} STIFFNESS.

We were pleased to see that participants did not exceed the maximum force of the Phantom (7.9 N), and in fact applied forces much less than that. These values can help elastic device designers when choosing appropriate force sensing hardware.

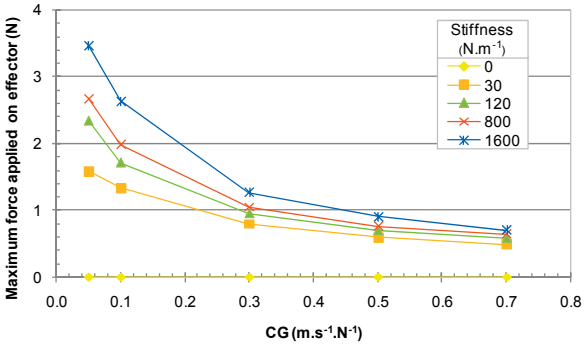


Figure 9: Mean maximum force applied for CG and STIFFNESS.

User feedback of ease-of-use and fatigue

After completing all BLOCKS for a STIFFNESS and CG combination, we asked the participants to answer two questions about *ease-of-use* and *fatigue* using a Likert scale. The ease-of-use question was worded "How easy is it to move the pointer the way you want?" and the fatigue question "How tiring is it to move the pointer?" The Likert scale ranged from 1 ("very easy to use" or "not tiring at all") to 5 ("very difficult to use" or "very tiring"). Since this produces ordinal data which is not normally distributed, the analysis used rank-transformed data – replacing each original value by its rank from 1 for the smallest value to N for the largest. Note that there is no equivalent non-parametric method for two-way or three-way designs.

Ease-of-use

Repeated measures analysis of variance found a significant main effect for STIFFNESS ($F_{4,60} = 13.3$, $p < 0.0001$) and CG ($F_{4,60} = 12.9$, $p < 0.0001$) on ease-of-use scale. Again, a significant STIFFNESS × CG interaction ($F_{16,240} = 4.9$, $p = 0.0001$) is most relevant (Figure 10). It shows that the easiest to use control gain depends on stiffness. For low values of STIFFNESS, pairwise comparisons found no significant difference across CG. However, as the STIFFNESS increases, significant differences were found for a stiffness 120 N.m^{-1} with a significant difference between CG 0.7 m.s^{-1}.N^{-1} and all other CG levels ($p < 0.015$) and for STIFFNESS 800 N.m^{-1} and 1600 N.m^{-1}, CG ranges [0.05-0.1] and [0.3-0.7] were significantly different ($p < 0.03$).

If we compare the optimum CG ranges for each STIFFNESS, we see a significant difference between 0 N.m^{-1} and all other values except 1600 N.m^{-1} ($p < 0.01$) (Figure 11).

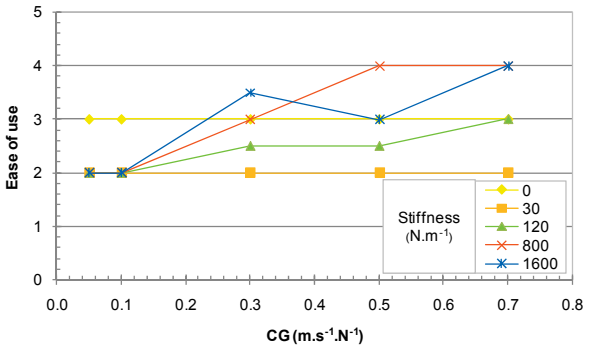


Figure 10: Median ease-of-use for CG and STIFFNESS.

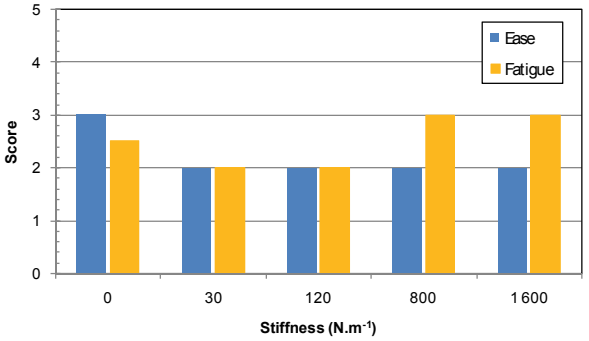


Figure 11: Median ease-of-use and fatigue score for STIFFNESS using the optimal CG range.

Fatigue

Repeated measures analysis of variance found a significant main effect across STIFFNESS ($F_{4,60} = 7.8$, $p < 0.001$) and CG ($F_{4,60} = 10.4$, $p = 0.001$) on fatigue scale. Once again, a significant STIFFNESS × CG interaction ($F_{16,240} = 2.6$, $p = 0.001$) is most relevant. Pairwise comparisons found no significant differences for CG with 0 N.m^{-1} STIFFNESS. However, for higher STIFFNESS levels, participants found either the lowest or highest extremes of CG to be most tiring (Figure 12). For STIFFNESS 30 N.m^{-1}, the lower range of CG [0.05 - 0.1] were most tiring ($p < 0.0001$); for STIFFNESS 120 N.m^{-1}, the lower range [0.05-0.1] were most tiring ($p < 0.03$); but for STIFFNESS 800 N.m^{-1}, the high range [0.1 – 0.5] were least tiring ($p < 0.02$); for STIFFNESS 1600 N.m^{-1} there was no significant difference.

By using the optimum CG ranges reported as least tiring, we find significant differences between STIFFNESS ranges [30 – 120 N.m^{-1}] and [800 – 1600 N.m^{-1}] ($p < 0.02$) (Figure 11). Participants found higher STIFFNESS more tiring confirming Zhai's argument that although performance may be similar, people find isometric feedback more tiring.

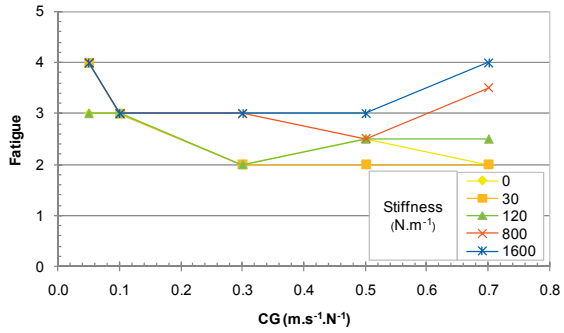


Figure 12: Median fatigue for CG and STIFFNESS.

DISCUSSION

Influence of control gain on user performance

We found a strong effect of control gain on user performance (both error rate and movement time), which shows that it is a critical factor to tune in rate control input devices. Plotting control gain against movement time exhibits a U-shaped curve with the optimum control gain levels dependent on the elastic stiffness. This confirms the general shape of the performance curve reported by Gibbs [9] and Zhai [21]. In addition, we show that the optimum values (the bottom of the U-shape) shift towards low control gain values with increased stiffness and optimum value range (the flatness of the bottom of the U-shape) increase with decreasing stiffness.

Our Fitts' law analysis explains why this stiffness-dependent U-shape curve occurs. Low control gain values make corrective movements easier and faster, but take more time to cover the distance to the target. In contrast, high control gain values quickly cover the target distance, but the corrective movements are difficult which increases overall movement time. With these extreme settings, moving to the target and acquiring the target are almost like two different devices. To provide a good tradeoff moderate values of control gain are recommended.

Influence of proprioception and control gain

If we had found the same optimal value of control gain across all stiffness levels, it would have meant that participants rely only on force to control the pointer velocity. Our findings demonstrate that participants also rely on displacement, using a combination of force *and* displacement sensing for velocity control. In fact, we note that the maximum force for 120, 800 and 1600 $N.m^{-1}$ stiffness with 0.7 $m.s^{-1}.N^{-1}$ control gain are very close (Figure 9), but there is a large difference in movement time and error rate (Figure 3 and Figure 5). The only difference between these settings is the effector displacement, which is greater with the lower 120 $N.m^{-1}$ stiffness resulting in better performance. This suggests that participants use displacement more than force to control velocity. The lower fatigue with decreased stiffness and control gain further supports the relationship between effector displacement and performance. Lower control gain values increase displacement, and therefore can increase performance with high stiffness devices (but the range of usable control gain levels is also reduced with higher stiffness).

Influence of stiffness

With an appropriate selection of control gain, devices with different stiffness perform equally well, except for stiffness 0, the pure isotonic condition, which had slightly higher error rates and a 15% increase in movement time. The only difference between 0 and 30 $N.m^{-1}$ stiffness levels is the self centering mechanism, so our study confirms that an elastic or isometric device is more suited to rate control. But, we also demonstrate that even a moderate level of stiffness is sufficient. The difference in performance between the pure isotonic and the elastic/isometric condition is lower than Zhai [10] likely because we use a 1D controller and a single device form factor across conditions.

Design guidelines

Considering the importance of proprioception, fatigue and the difficulty of tuning control gain with high stiffness, we recommend that elastic device designers use lower stiffness values. However, with lower stiffness values comes a larger device operating range. This can be an issue because rate control devices are often used for situations where the physical device space is restricted. If this is an issue, then we recommend using the upper limit of the optimum range of control gain values, and continue to use a stiffness value as low as possible to obtain the desirable operating range.

Before setting the control gain, first select a deadband force to counteract the natural backdrive friction of the device. Also, to accurately control the cursor velocity, the display resolution and control loop frequency must be used.

Now if we combine the optimum control gain range given an acceptable error rate (around 4%), movement time, ease of use and fatigue, we have the following optimum ranges:

- Stiffness 0: control gains [0.1 – 0.5]
- Stiffness 30: control gains [0.3 – 0.7]
- Stiffness 120: control gain 0.1
- Stiffness 800: control gain 0.1
- Stiffness 1600: control gain 0.1

In other words, if the stiffness is greater than 120 $N.m^{-1}$, a control gain equal to 0.1 $m.s^{-1}.N^{-1}$ will give the optimal performance for movement time and error rate. For stiffness below 120 $N.m^{-1}$, this value can be increased up to 0.7 $m.s^{-1}.N^{-1}$ as the stiffness gets closer to 0. Higher stiffness accommodates a device with a small operating range.

Using Figure 9 or Equation 7, the selected control gain value should be checked for a compatible operating range given the device space constraints, the desired spring stiffness and device display size (with the display diagonal considered as the maximum target distance). If Equation 7

results in an operating range which cannot be accommodated by the device, it suggests that the operating range is too constrained. However, the control gain can be increased to accommodate, but it may degrade performance.

As an example, consider selecting the best control gain for the Nintendo Wii Nunchuck rate control device. The Nunchuck has a 2D elastic joystick with a 20mm operating range and a 60 $N.m^{-1}$ stiffness. With a maximum target distance of 300 mm, Equation 7 finds an operating range of 30 mm and a control gain of 0.7 $m.s^{-1}.N^{-1}$. Considering the wide optimum control gain range for this stiffness, we can increase control gain without hurting performance to fall within the 20 mm operating range of the Nunchuck.

CONCLUSIONS AND FUTURE WORK

Thus far, tuning elastic device parameter settings has been guided by inspiration and small pilot experiments. With this study we provided a systematic analysis of not only stiffness, but also its interaction with control gain. We have shown that control gain is a critical factor when tuning rate control devices and demonstrated how the stiffness impacts the optimum range of control gains. With optimum control gain values, we found that elastic and isometric devices can perform equally well. However, operating range and its effect on user fatigue is another important factor to consider. Considering the stiffness, operating range and control gain, we provide the first guidelines for choosing stiffness, and control gain for rate control devices. The results of our experiment give a base line for comparisons involving rate control devices and for experimentation with non-linear functions. As future work, we plan to begin this work by investigating the optimal design of non-linear functions for rate control devices.

ACKNOWLEDGEMENTS

We thank all the study participants and Adel Abderrahmani for his help in running the experiment. We also thank the IRCICA for its support.

REFERENCES

1. Accot, J., and Zhai, S. Scale effects in steering law tasks. In *Proc. of CHI'01*, 2001, pp. 1–8.
2. Barrett, R. C., Selker, E. J., Rutledge, J. D., and Olyha, R. S. Negative inertia: a dynamic pointing function. In *Proc. of CHI '95*, 1995, pp. 316–317.
3. Buck, L. Motor performance in relation to control-display gain and target width. *Ergonomics, 32 (1980)*, 579 – 589.
4. Card, C., English, W., and Burr, B. Evaluation of mouse, rate controlled isometric joystick, step keys, and text keys for text selection on a crt. *Ergonomics, 21 (1978)*, 601–613.
5. Casiez, G., Vogel, D., Pan, Q., and Chaillou, C. RubberEdge: reducing clutching by combining position and rate control with elastic feedback. In *Proc. of* UIST*'07*, 2007, pp. 129–138.
6. Douglas, S. A., and Mithal, A. K. The effect of reducing homing time on the speed of a finger-controlled isometric pointing device. In *Proc. of CHI'94*, 1994, pp. 411–416.
7. Epps, B.W. Comparison of six cursor control devices based on Fitts' law models. In *Proc. of* the 30th Annual Meeting of the Human Factors Society, 1986, 327–331.
8. Froehlich, B., Hochstrate, J., Skuk, V., and Huckauf, A. The GlobeFish and the GlobeMouse: two new six degree of freedom input devices for graphics applications. In *Proc. of CHI'06*, 2006, pp. 191–199.
9. Gibbs, C. Controller design: interactions of controlling limbs, time-lags and gain in positional and velocity systems. *Ergonomics, 5 (1962)*, 385–402.
10. Jellinek, H.D., and Card, S.K. Powermice and user performance. In *Proc. of CHI '90*, 1990, pp. 213–220.
11. Kantowitz, B. H., and Elvers, G. G. Fitts' law with an isometric controller. Effects of order of control and control-display gain. *Journal of Motor Behavior, 20 (1988)*, 53–66.
12. Kawachiya, K., and Ishikawa, H. NaviPoint: an input device for mobile information browsing. In *Proc. of CHI'98*, 1998, pp. 1–8.
13. MacKenzie, I. S. Fitts' law as a research and design tool in human computer interaction. *Human Computer Interaction*, 7 (1992), 91–139.
14. MacKenzie, I. S., Sellen, A., and Buxton, W. A. S. A comparison of input devices in element pointing and dragging tasks. In *Proc. of CHI'91*, 1991, pp. 161–166.
15. Mithal, A. K., and Douglas, S. A. Differences in movement microstructure of the mouse and the finger-controlled isometric joystick. In *Proc. of CHI'96*, 1996, pp. 300–307.
16. Phillips, J. D., Triggs, T. J., and Bellgrove, M. Computer Screen Cursor Trajectories as Controlled by an Accupoint: A Kinematic Analysis. In *Proc. of Computer Human Interaction Conference*, 1998, pp. 314–319.
17. Rutledge, J. D., and Selker, E. J. Controller for improved pointing devices. US Patent 5,764,219. 1998.
18. Rutledge, J. D., and Selker, T. Force-to-Motion Functions for Pointing. In *Proc. of Interact'90*, 1990, pp. 701–706.
19. Sensable, Phantom haptic device, www.sensable.com.
20. Ugural, A. C., and Fenster, S. K. Advanced strength and applied elasticity. Prentice Hall PTR. 2003.
21. Zhai, S., Human Performance in Six Degree of Freedom Input Control. Ph.D. Thesis, University of Toronto, 1995.

RubberEdge: Reducing Clutching by Combining Position and Rate Control with Elastic Feedback

Géry Casiez[†], *Daniel Vogel*[††], *Qing Pan*[†] *and Christophe Chaillou*[†]

[†]LIFL & INRIA Futurs
University of Lille, FRANCE
{*gery.casiez|qing.pan|christophe.chaillou*}*@lifl.fr*

[††]Department of Computer Science
University of Toronto, CANADA
dvogel@dgp.toronto.edu

ABSTRACT

Position control devices enable precise selection, but significant clutching degrades performance. Clutching can be reduced with high control-display gain or pointer acceleration, but there are human and device limits. Elastic rate control eliminates clutching completely, but can make precise selection difficult. We show that hybrid position-rate control can outperform position control by 20% when there is significant clutching, even when using pointer acceleration. Unlike previous work, our RubberEdge technique eliminates trajectory and velocity discontinuities. We derive predictive models for position control with clutching and hybrid control, and present a prototype RubberEdge position-rate control device including initial user feedback.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces

General terms: Design, Human Factors

Keywords: hybrid, pointing, clutching, mobile, elastic

INTRODUCTION

For the most part, a relative position control device, such as the mouse, will perform better than a rate control device, such as a joystick [6,9]. However, a potential issue with position control devices is when *clutching* – the momentary recalibration to avoid running out of input area – becomes more frequent, taking additional time [12,16]. Recently the resolution of digital displays has increased significantly, while the input area remains fixed, making clutching more of an issue. For example, laptops are available with 38cm (15") displays with resolutions in excess of 1400 × 1050 pixels, yet the touch pad input space remains at about 4cm. With wall-sized displays, the difference is even greater.

Clutching can be reduced by increasing the ratio of display movement to control movement (*Control-Display gain*, or *CD gain*), but high CD gain can hurt performance [1,12,13,26]. An alternative is to dynamically adjust CD gain based on the input velocity. Called *pointer acceleration*, [12,21] this technique uses low CD gain at low velocity to improve precision and high CD gain at high velocity to cover large distances with minimal clutching.

UIST'07, October 7–10, 2007, Newport, Rhode Island, USA.

Clutching can be avoided altogether by using a rate control device such as the TrackPoint [26]. This may increase performance for long distance movements, but for shorter movements, where a position control device could be used without clutching, performance will suffer [9].

To preserve the benefits of medium-distance position control and still accommodate long movements without clutching, simple hybrid position-and-rate control techniques have been proposed [2,22]. But without any haptic feedback, the transition between position and rate mode is difficult to distinguish and the rate is difficult to control. Zhai found that elastic feedback is well suited for rate control [26] and Dominjon et al. used elastic feedback for 3D hybrid position-and-rate control [8]. However, their mapping function has trajectory and velocity discontinuities when transitioning from position to rate control, further highlighting the challenges in designing a usable hybrid device.

In this paper we present RubberEdge, a 2D hybrid position-and-rate control technique using elastic feedback. Unlike past work, we designed a mapping function which enables a smooth transition from position to rate control. We conducted an experiment to evaluate its performance and explore the interaction of CD gain and pointer acceleration. We found that our hybrid control technique outperforms position-only control by 20% with a small input area similar to a laptop touch pad. We derive two predictive models for selection time with clutching and hybrid control. Finally, we discuss a class of RubberEdge devices (Figure 1) and present our first physical RubberEdge prototype device for laptop touch pads, with initial user feedback.

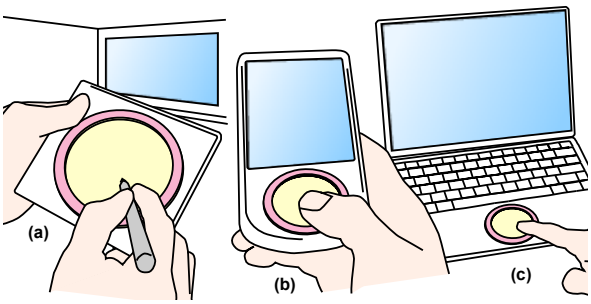


Figure 1: Design Concepts for RubberEdge Devices: (a) handheld pen tablet for a large display; (b) PDA with touch pad; (c) laptop touch pad

BACKGROUND AND RELATED WORK

Isotonic Position and Elastic Rate Control

Zhai defines three classes of devices, *isotonic*, *isometric* and *elastic* [26]. Isotonic devices are free-moving and use position for input. Isometric devices do not perceptibly move and use force for input. In between isotonic and isometric devices are elastic devices, where resistance increases with displacement. Elastic devices can use either position or force for input.

With any class of input device, there are different ways to map to output, the two most popular being *position control* (zero-order) and *rate control* (first-order) [26]. Zhai found that isotonic devices are better suited to position control whereas isometric and elastic devices should use rate control [26]. The latter require a self-centering mechanism – a way for the device to return to a neutral rate control state – difficult to achieve with isotonic devices. Rate control maps input to a velocity vector and moves the display pointer in that direction and speed. Position control maps device input to an output position. *Absolute* position control assigns a unique output position to every input position [5,7]. This is typically done when the input and output spaces are coincident, like on a pen-input display. *Relative* position control uses a displacement vector from an *initial input position* to the current input position [5,7]. The current output position is calculated with the vector and a corresponding initial output position. The definition of this initial input position is achieved by *clutching,* where the input stream is suspended as the device is repositioned at a new initial input position.

A transfer function uses *Control Display Gain* (*CD gain*) to scale the relative displacement vector. Jellinek et al. [12], Accot et al. [1] and Zhai [26] found that performance degraded at low and high CD gain levels. This may be partially due to limits of human motor control [3] and limited device resolution preventing selection of every pixel at high levels, and increased clutching at low levels [12, 16]. Unfortunately the effect of CD gain is not conclusive [1,12,13,15].

The amount of clutching is dependent on the maximum area of unconstrained physical movement (the *operating range*[1]), target distance, and transfer function. A small operating range causes more clutching with large target distances, and the maximum operating range is dependent on the transfer function. High CD gain increases the size of the operating range but with a potential performance cost. In theory, *Pointer Acceleration* [21]*,* a dynamic transfer function which uses high CD gain for long, fast movements and low CD gain for slow precise movements, should help minimize clutching without the performance cost of a uniform high CD gain. However, there is no published research showing a benefit for pointer acceleration [12, 19], and its effect on clutching has yet to be been shown.

[1] Jellinek & Card [12] use the term *footprint*, but this can be confused with the static area occupied by an object.

Hybrid Position-Rate Control

Hybrid position-rate control techniques combine both input control modes into one device. This can be done *simultaneously* with two different physical position and rate controls mapping each to different outputs [28]; however, controlling just the pointer position with two controls simultaneously is not feasible. A more general solution is to make the device bimodal, using either position or rate control, and always controlling the pointer position directly. A common example is used in many common applications utilizing scrolling windows. When dragging and selecting items, the input switches from position to rate control as the pointer crosses the boundary of the visible window. In practice, it is difficult to move in arbitrary 2D directions and rate control is difficult, because without feedback the position-to-rate transition point is difficult to perceive and self-centre [26].

In virtual environments, Bowman and Hodges' Stretch Go-Go technique [2] uses visual feedback to help control the rate and self-centre. A virtual hand is controlled with position control, but the arm length is expanded or contracted with (constant) rate control when the hand enters circular near or far regions. The use of circular zones allows rate-control movement in any direction. Tactile 3D [24] is a commercial 3D file browser using hybrid position-to-rate control with visual and audio feedback. Rate control is used to rotate the camera with rotation speed proportional to the distance from the circular zone.

Synaptics touch pads include a hybrid technique called EdgeMotion™ [22]. At the edge of the touch pad, an isometric rate control mode is activated by switching to a downward pressure. In practice, transitioning from horizontal movement to vertical pressure for rate control may not be intuitive. Also, because of the rectangular shape of the position control zone, continuing pointer movement in the same direction in the rate control mode is difficult. No user evaluations have been reported.

Dominjon et al.'s 3D hybrid position-rate control technique uses elastic feedback with a large Virtuose 6 DOF force feedback device [8]. A spherical volume is simulated in physical space and visualized as a transparent sphere on the display. When the input point is inside the volume, movement is by position control with constant CD gain. When the input is moved beyond the spherical volume, the device uses rate control with elastic feedback. However, when we adapted their straightforward mapping functions to 2D, it exposed trajectory and velocity discontinuities at the transition point affecting its usability. Moreover, to our knowledge the authors have not conducted any sound user evaluation, and there is no satisfactory theoretical basis.

No previous examples of hybrid position-to-rate control devices have demonstrated a benefit, and little work has been done on the effect of clutching. We present our experiment which investigates these related issues later.

RUBBEREDGE HYBRID CONTROL

Through an analysis of Dominjon et al.'s [8] straightforward mapping functions, we were able to determine the reason for erratic behaviour when transitioning to rate control. This motivated our design for improved RubberEdge mapping functions with a smooth transition from isotonic position control to elastic rate control.

Straightforward Mapping Functions

Dominjon et al.'s [8] mappings (Equations 1, 2) introduce trajectory and speed discontinuities when transitioning from isotonic to elastic zones. In Equation 1, the feedback force F is proportional to the distance between end effector P and the isotonic-to-elastic boundary N given spring stiffness k. The force direction is always radial with $\vec{r}$, the radial direction from the centre of the isotonic circle to P. In Equation 2, the input control rate V is a third degree polynomial with a scaling constant K. Dominjon et al.'s implementation set $k = 200\ N.m^{-1}$ and $K = 0.03\ N^{-3}.s^{-1}$.

$$\vec{F} = -k \cdot (P - N) \cdot \vec{r} \quad (1)$$

$$\vec{V} = K \cdot F^3 \cdot \vec{r} \quad (2)$$

This formulation introduces a trajectory discontinuity as long as the isotonic trajectory is not radial to the isotonic circle. The pointer will jump to the radial trajectory defined by Equation 2 the moment it enters the rate control zone, regardless of its initial path (Figure 2). A speed discontinuity also occurs because according to equation 1, the initial force in the elastic zone will be zero, and thus the velocity will be set to zero with equation 2. Continuity of speed is important, since a noticeable drop could affect the pre-planned trajectory, impairing user performance [20].

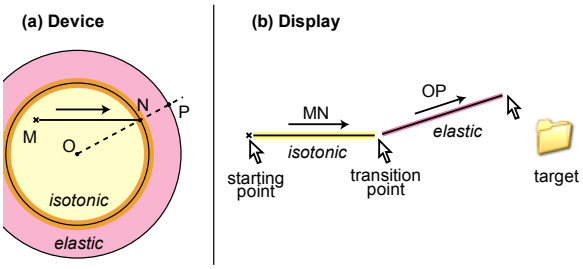


Figure 2: Trajectory Discontinuity with Straightforward Mapping Function: (a) Using the device to select a distant target, the user moves from position *M* to *N* in the isotonic zone, then transitions to the elastic zone; (b) On the display, the pointer will deviate from its trajectory of *MN* at the transition point, instantly changing to *OP* because the elastic zone always uses a direction vector radial from *O* through transition point *N* to an end effector *P*.

RubberEdge Mapping Functions

Our mapping functions enable a consistent trajectory by rotating and translating the isotonic-to-elastic boundary after transition and we smooth the transition velocity by mixing pre- and post-transition velocities. At first it appears that simply using the same isotonic direction vector (*MN* in Figure 2) as the direction vector $\vec{r}$ for the rate in Equation 2 is the solution. However, the pre-planned trajectory direction in the isotonic zone is not always correct and the user may want to adjust it in the elastic zone. This could be done by saving the exit point *N* and translating the pointer according to *NP*. However any change in *P* produces an important variation in the pointer direction.

To create a consistent trajectory we translate and rotate the isotonic-to-elastic boundary zone as the user penetrates the elastic zone. We do this smoothly, by giving mass and inertia to the boundary zone using a simple physical simulation to align it with the isotonic direction vector. The intuition behind this technique is to consider how a real circular object, like a dinner plate, would rotate and translate when pulled by a string attached to its edge (Figure 3). When the user exits the isotonic zone, the exit point *N* is saved. In the elastic zone, the vector from *N* to the end-effector *P* gives the force direction applied by the user on the plate. By applying angular momentum, *N* rotates smoothly to *N'* and the force direction vector becomes radial to *O*, the centre of the isotonic-to-elastic boundary. Past user interface researchers have utilized similar physics-based rotation and translation functions, but for rotating graphical objects with direct manipulation [14] and smoothly rotating or peeling back GUI windows [4].

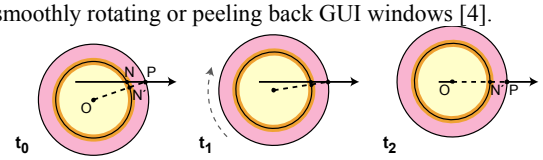


Figure 3: Rotation and translation with momentum over time: like pulling a dinner plate with a string.

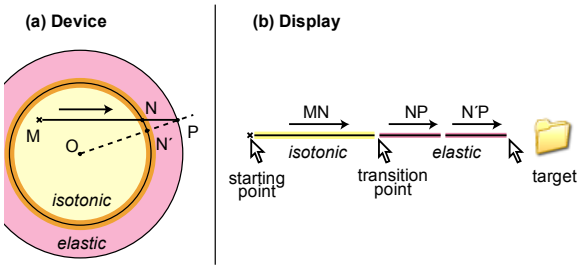


Figure 4: Continuous Trajectory with Enhanced Technique: (a) Using the device, the user moves from *M* to *N* in the isotonic zone, then transitions; (b) On the display, the initial trajectory *NP* smoothly changes to *N'P* by applying angular momentum.

The angular speed of the isotonic zone is computed by the theorem of angular momentum (Equation 3). $\vec{\omega}$ is the rotation vector of the isotonic zone and J is its moment of inertia with a friction term (μ) added to avoid instability. The translation is proportional to the vector *NP*. We found that using a mass of *1Kg* and a friction coefficient of $3*10^{-3}\ N.s.rad^{-1}$ smoothes out the trajectory nicely without the sharp direction changes.

$$J\frac{d\vec{\omega}}{dt} - \mu\vec{\omega} = \overrightarrow{ON} \wedge \overrightarrow{NP} \quad (3)$$

To smooth the transition velocity, we mix the pre-transition velocity in the isotonic zone V_0 with the input control rate computed in the elastic zone. Equation 4 find the mixed velocity V_t where t is the time after the isotonic exit and A is a constant to adjust the mixing time ($A=0.3s$).

$$V_t = \left(V_0 \cdot e^{-At} + K \cdot NP \cdot (1 - e^{-At})\right) \frac{\overrightarrow{NP}}{NP} \qquad (4)$$

While iterating this design we found the technique worked well except when the displacement is tangent to the boundary circle. Then, the elastic force is near zero, making control difficult. However, the friction coefficient makes this occurrence rare.

EXPERIMENT

Since no previous work has demonstrated a benefit for hybrid position-to-rate control, we conducted an experiment comparing RubberEdge to pure position control. Note that after our theoretical analysis established that the 2D adaptation of Dominjon et al.'s unproven technique has control discontinuities, it cannot be considered state-of-the art and an empirical comparison would be of limited value.

Goals and Hypotheses

Since the motivation for hybrid position-to-rate control is to eliminate (or at least reduce) clutching, clutching is a key factor in the experiment. We experimentally manipulated clutching by holding the device operating range constant and adjusting the target distance. Since the transfer function affects the device operating range, we included conditions for both constant CD Gain and pointer acceleration. Pointer acceleration should reduce clutching, and could negate the benefit of hybrid control. Unlike past work [12], we use the more aggressive Windows XP/Vista pointer acceleration function [21].

H1: The hybrid technique will outperform pure position control when there is clutching. Clutching with a position control device takes time because the pointer movement stops as the user recalibrates their position, whereas with the hybrid technique the pointer continues to move in the direction of the target. For long distances, we expect the inclusion of an elastic zone to be an advantage for the hybrid device in spite of the lower performance of pure elastic devices. This is because hybrid control still enables isometric control for fine adjustment near the targets.

H2: Pointer acceleration improves position control performance by reducing clutching. A dynamic transfer function uses high CD gains at high speeds which should increase the effective operating range and reduce the amount of clutching, but without hurting low speed precision.

Apparatus

To avoid device related confounding factors in the experiment, we simulated both 2D position control and 2D hybrid control on a Phantom Omni haptic device. The Phantom uses a stylus connected to a force-feedback armature to produce haptic feedback. By simulating both techniques with a single device, we were able to compare them without introducing extraneous intra-device differences such as ergonomics, size and sensitivity. The Phantom also enabled rapid prototyping – we could iterate the RubberEdge technique and parameters with synthesized haptic feedback.

To ensure that position control performance is not adversely affected when using the Phantom, we conducted a 4 participant pilot experiment comparing it to the mouse. We used 3 target distances (70, 140, 280 mm) and a constant CD gain of 2. With these settings, no clutching was needed by either device (we found constraining the maximum mouse operating range difficult, so did not compare it with clutching). A keyboard key was used for target selection, since the Phantom and mouse have different buttons. We found mean movement times of 1.24s for the Phantom and 1.23s for the mouse. Fitts' Law analysis gave similar regression coefficients: T = -0.03 + 0.24 ID for the Phantom (R^2=0.97) and T = 0.09 + 0.22 ID for the mouse (R^2=0.99). This is consistent with previous mouse results [17]. Although not definitive, the results of this pilot bolstered our confidence that using the Phantom would be comparing position control comparable to a mouse, perhaps the best performing position control device.

Simulating the Techniques on the Phantom

For both techniques, the Phantom stylus moves on a simulated haptic surface 1cm above the desk. The size of the isotonic area in each technique was constrained to a circle 40 mm in diameter by simulating a vertical wall around the perimeter (Figure 5). This size was selected to be similar to typical laptop touch pads. For the hybrid control technique, the elastic zone was accessed beyond the perimeter wall by pushing against a simulated radial spring with a stiffness of 60 $N.m^{-1}$. This setting was chosen after running a pilot experiment testing different stiffness values and it approximates the elasticity of a typical thick rubber band. For the position control technique, the simulated perimeter wall was rigid, and clutching was performed by lifting the pen above the simulated surface. To avoid instability when selecting a target with the button on the Phantom stylus, participants instead pressed a keyboard key with their non-dominant hand.

Our experiment was conducted on a 3 GHz PC with dual 19 inch, 85 DPI LCD monitors. Our C++ software displays the stimulus at 60 Hz. The Phantom Omni has a 450 DPI nominal resolution with 1000Hz haptic rendering.

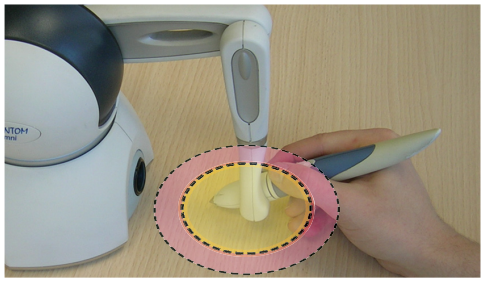


Figure 5: Simulating the RubberEdge hybrid technique with the Phantom haptic device. A 2D simulated haptic surface constrains the pen movement and the elastic zone is created using force feedback.

Task and Stimuli

The task was a reciprocal two dimensional pointing task, requiring participants to select round targets back and forth in succession. The positions of the targets were randomly pre-computed using the position of the previous target and the current distance. When participants correctly selected a target, the target disappeared and the next one appeared on the other side of the screen. If a participant missed a target, a sound was heard and an error was logged. Participants had to successfully select the current target before moving to the next one, even if it required multiple attempts. This prevented participants from "racing through the experiment" by clicking anywhere. To avoid using the edges to assist in target acquisition, the pointer was not constrained to the bounds of the screen. Participants were encouraged to take breaks between sets of trials.

Participants

Eight people (5 male, 3 female) participated with a mean age of 26.3 (SD = 1.5). Three participants used Windows XP/Vista pointer acceleration exclusively, two did not, and the remaining used both.

Design

A repeated measures within-subjects design was used. The independent variables were *Technique* (*Position* control and *Hybrid* control), *Transfer Function* (*CG* - constant gain and *PA* - pointer acceleration), target *Distance* (D_L – 688mm, D_M – 344mm, D_S – 172mm), and target *Width* (W_L – 8mm, W_M – 4mm, W_S – 2mm). The nine *Distance-Width* combinations give five Fitts' indices of difficulty (ID) [18] ranging from 4.5 to 8.4. We selected long distances to promote clutching, so our ID range is high. With short distances (and corresponding low IDs) the *Hybrid* and *Position* control techniques are equivalent since the elastic zone is not needed in the *Hybrid* technique.

For the *CG Transfer Function* we used a constant CD gain of 2 to encourage clutching. For the *PA Transfer Function*, we used the default Windows XP/Vista setting [21]. Using this setting, the CD gain increases continuously with the speed, from about 1.6 for low speeds to 7.3 for high speeds.

The presentation order of the 2 *Techniques* and 2 *Transfer Functions* was fully counterbalanced across participants. For each *Technique* and *Transfer Function* combination, participants completed a training period of approximately 5 minutes. Each *Distance-Width* combination was repeated 36 times with 4 Blocks of 9 trials each. *Distance-Width* combinations were presented in ascending order of ID within a single block allowing participants to leverage repetitive, ballistic movements while steadily increasing task difficulty.

After all blocks were completed for a *Technique* and *Transfer Function* combination, a short questionnaire asked participants to compare it to the previous combination. At the end of the experiment, a final questionnaire asked for an overall ranking of the four *Technique* and *Transfer Function* combinations. The experiment lasted approximately 120 minutes.

In brief, the experimental design was:

8 *Participants* × 2 *Techniques* × 2 *Transfer Functions* × 4 *Blocks* × 3 *Distances* × 3 *Widths* × 9 repetitions = 10,368 total trials.

RESULTS

The dependent variables were movement time, error rate, and measurement of clutching and elastic zone usage.

Error rate

Participants had an overall mean error rate of 1.5%, and a repeated measures analysis showed no significant effect of the different independent variables on error rate. In this type of experiment, a 4% error rate represents a good trade combination of speed and accuracy, so our lower error rate suggests greater emphasis on accuracy. As a result, movement times were somewhat higher and we computed the effective width for our Fitts' Law analysis [18].

Selection Time

Selection time is the time from the beginning of the trial until the first target selected attempt. Targets that were not selected on the first attempt were marked as errors, but were still included in the timing analysis (the analysis was run with and without error trials and the same significant effects were found with similar F and p values).

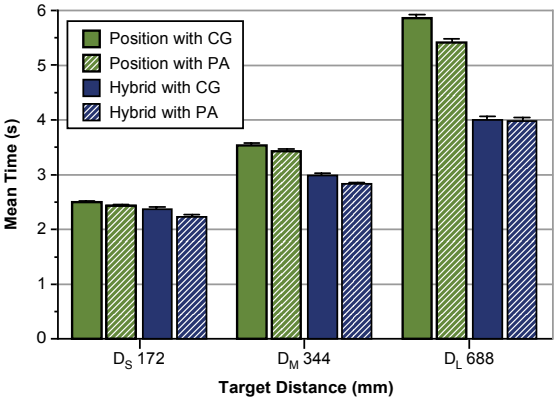


Figure 6: *Technique* × *Transfer Function* × *Distance* interaction on selection time (error bars 95% CI).

Repeated measures analysis of variance showed that the presentation order of *Technique* and *Transfer Function* had no significant effect on movement time, indicating that a within-participants design was appropriate. No significant effect for *Block* was found indicating that there was no learning effect present. There was a significant main effect for *Technique* on selection time ($F_{1,7} = 16.0$, $p < 0.005$) with the *Hybrid* control technique outperforming the *Position* control technique by 20.6%. As expected in a target selection experiment, there were also significant main effects for *Distance* ($F_{2,14} = 471.0$, $p < 0.0001$) and *Width* ($F_{2,14} = 231.0$, $p < 0.0001$) on selection time. The significant interactions for *Technique* × *Distance* ($F_{2,14} = 50.2$, $p < 0.0001$) and *Technique* × *Transfer-Function* × *Distance*

($F_{2,14}$ = 9.7, p < 0.017) are perhaps most relevant. These show that the selection time increases with *Distance* at different rates given the *Transfer-Function* (Figure 6). Pair-wise comparisons found no significant difference between the two *Techniques* for the smallest *Distance* D_S but significant differences for D_M (p < 0.017) and D_L (p < 0.001) with 16% and 29% improvements for *Hybrid* control over *Position* control respectively. The high selection times for distant targets with the *Position* control technique are due to heavy clutching, which we discuss in detail below.

Pair-wise comparisons revealed a significant difference between the two *Transfer Functions* for the *Position* technique and D_L (p < 0.032). *PA* reduces the selection time by 7.5% compared to *CG*. It appears that participants were able to harness the higher speeds for distant targets and thus use higher CD gains to avoid clutching.

Fitts' Law Analysis

The significant interaction between *Technique*, *Transfer-Function* and *Distance* leads us to a Fitts' Law analysis. We aggregated the *Distance-Width* combinations for each *Technique* and *Transfer-Function* and computed the effective width since the error rate is not equal to 4% [18].

Unlike many past studies, we found poor regression fitness suggesting that Fitts' Law may not hold in the presence of significant clutching or for a technique combining two different control mappings. The index of difficulty (ID) is expressed as a ratio between target distance and width, giving the same importance to each. Therefore, Fitts' Law predicts that any *Distance-Width* combination with the same ID will yield the same selection time. However, in looking at a plot of ID and selection time (Figure 7), it appears that distance alone affects position control clutching or encourages hybrid control mode switching.

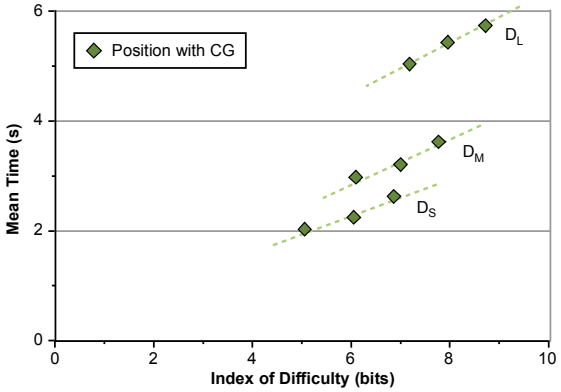


Figure 7: Selection times given *Distance-Width* combinations for the *Position* technique and *Constant* transfer function. Trend lines for *Distance-Width* groups are highlighted, suggesting *Distance* has a greater effect on selection time. Plots of other *Technique* and *Transfer Function* combinations are similar.

Usage of Clutching and Elastic Zone

To help characterize technique usage and to investigate possible explanations for the poor conformance to Fitts' Law, we analyzed the amount of time spent clutching in *Position* or using the elastic zone control in *Hybrid*, as well as the number of invocations for each. Since cross-technique statistical comparisons of these two measures would not be meaningful, separate ANOVAs were used for each technique.

Technique	*a*	*b*	r^2
Position with *CG*	-3.9	1.1	.73
Position with *PA*	-2.4	0.9	.70
Hybrid with *CG*	-1.0	0.6	.86
Hybrid with *PA*	-1.0	0.6	.74

Table 1: Fitts' Law regression values for *Technique* and *Transfer Function*: *a* is the intercept of the regression line, *b* is the slope, r^2 is the fitness.

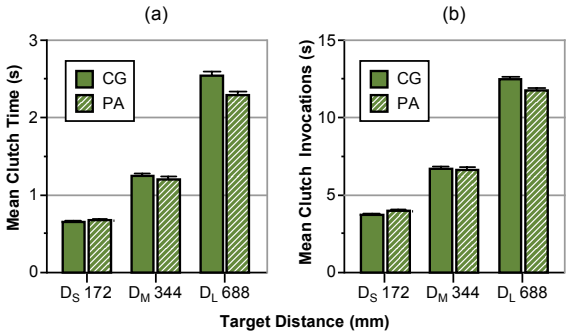


Figure 8: Position Control Transfer Func. × Distance interaction on: (a) clutch time; (b) clutch invocations

Clutch Time and Elastic Zone Time

Clutch time is the total time in which the stylus was lifted during a trial, and elastic zone time is the time spent outside the isotonic zone. For the *Position* technique, there is a significant main effect of *Distance* ($F_{2,14}$ = 165.5, p < 0.0001) showing that clutch time increases as *Distance* increases. Pair-wise comparisons revealed that the total clutch time increased from 0.7s for D_S to 1.2s for D_M and 2.4s for D_L for *Position* control and *CG* (all p < 0.0001). Results for *PA* are similar (Figure 8a). A *Transfer Function* × *Distance* interaction ($F_{2,14}$ = 21.9, p < 0.0001) and pair-wise comparison show that clutch time is reduced by 10% with the *PA* Transfer function for the largest distance D_L (p = 0.018). For the *Hybrid* technique, we found a significant main effect for *Distance* ($F_{2,14}$ = 956.7, p < 0.0001) showing that elastic zone time increases with *Distance*.

Clutch and Elastic Zone Invocations

Comparing the mean number of *Position* invocations to *Hybrid* elastic zone transitions can help characterize technique usage. With 1.3 and 7.5 invocations respectively, we see that frequent clutching actions can be replaced often with a single transition to the elastic zone.

For the *Position* technique, a significant main effect was found for *Distance* ($F_{2,14}$ = 430.6, p < 0.0001), but the *Transfer Function* × *Distance* interaction ($F_{2,14}$ = 19.1, p < 0.0001) is more relevant. Pair-wise comparison revealed that the number of *Position* clutch invocations for D_L is

dependent on *Transfer Function*: 12.4 for CG and 11.6 for PA (p = 0.013) (Figure 8b). *Hybrid* invocations were near 1 regardless of *Distance* or *Transfer Functions*.

Table 2 compares the actual clutching invocations with an ideal minimum number calculated by dividing the floor of target distance by maximum device operating range (accounting for CD gain). We can see that the ratio remains constant around 0.75, indicating that participants did not use the maximum available operating range for each clutch.

	Position with *CG*		
Distance	D_S	D_M	D_L
Ideal	3	5	9
Actual	3.75	6.7	12.5
Ratio	0.8	0.75	0.72

Table 2: Comparison of Actual and Ideal numbers of invocations used in the Position technique.

User Feedback

Overall preference for *Technique* was split. Those preferring *Hybrid* found it faster for long-distance targets and disliked the repetitive clutching motion with *Position*. Those preferring *Position* found it acceptable to use conventional clutching for short and medium distances, and felt that it was difficult to accurately exit the elastic zone and fine-tune their selection. Our observations during the experiment reinforced this last comment: when participants exited the elastic zone having undershot the target, any movement back in the direction of the target transitioned them back to the elastic zone. This left no room for isotonic movement to fine-tune the selection. One way to address this is to allow conventional clutching in the isotonic zone of the *Hybrid* technique. Later in the paper, we present a prototype device which does exactly that.

DISCUSSION

Our experiment confirmed our two hypotheses and illustrated a negative performance impact with clutching.

Our results confirmed hypothesis H1: a hybrid position-rate control technique has a performance advantage over pure position control when faced with significant clutching. Our experimental design intentionally provoked clutching, and overall we found *Hybrid* control improved performance by 20.6%. Specifically, *Hybrid* control improved performance by 16% and 29% over *Position* control for D_M and D_L respectively. Increased clutching with position control appears to be the reason. Clutch times went from 0.7s at D_S to 1.2s at D_M and 2.4s at D_L with the *CG* transfer function. One reason why clutching may be slower than *Hybrid*, is the number of invocations required. Users had to clutch more than 12 times to reach D_L.

We confirmed hypothesis H2: pointer acceleration reduces clutching with *Position* control. The effect is somewhat slight; we saw it only at the longest distance where PA clutch time was 10% lower, requiring an average of 11.6 invocations compared to 12.4 for CG. This suggests that participants are able to utilize the high CD gain levels with quick ballistic movements. Our results differ from past researchers who did not see an effect [12]. We attribute this to using a more aggressive pointer acceleration function and a task requiring significant clutching.

The experiment demonstrates the negative impact of clutching on user performance and shows that selection times with significant clutching do not conform to Fitts' Law. With clutching, task difficulty appears to be primarily dependent on distance, rather than the ratio of distance to width as in Fitts' Law. Past researchers have not reported this [12, 16], perhaps because their experiments did not promote significant clutching.

FORMAL MODELS

We developed two formal models to predict position control performance with clutching, and performance with a hybrid position-rate control technique.

Clutching Model

We base our model for position control movement with clutching on a two-part, idealized movement. In the first part, the user clutches several times to bring the pointer within a "clutch-free" distance to the target. In the second part, the user completes the movement without clutching, and selects the target. The total movement time T is the sum of the time for the first part T_1 and the second part T_2:

$$T = T_1 + T_2 \quad (5)$$

T_1 is dependent on the number of clutches N and the time for each clutch T_C, which we assume to be constant. We also assume the time the cursor is engaged *between* two clutches to be equal to T_C, hence the factor 2:

$$T_1 = 2 \cdot N \cdot T_C \quad (6)$$

N is dependent on the target distance D (in mm) and the effective device operating range d_e:

$$N = \left\lfloor \frac{D}{d_e} \right\rfloor \quad (7)$$

The effective operating range of the device, d_e is calculated from the physical device operating range d (in mm), the CD gain CD, and a corrective parameter c. Recall that our experimental results showed that in practice, only a portion of the operating range is actually used:

$$d_e = c \cdot d \cdot CD \quad (8)$$

The movement time in the second part, T_2, can be calculated using Fitts' Law [18] with the remaining target distance D_2 and Fitts' device parameters a_i and b_i.

$$T_2 = a_i + b_i \log_2\left(\frac{D_2}{W} + 1\right) \quad (9)$$

Where D_2 is equal to:

$$D_2 = D - N \cdot d_e \quad (10)$$

By substituting Equations 6 to 10 into Equation 5 and simplifying, we have a model which accounts for clutching when predicting target selection time:

$$T = 2\ N\ T_c + a_i + b_i \log_2\left(\frac{D - N \cdot c \cdot d \cdot CD}{W} + 1\right) \quad (11)$$

Hybrid Control Model

A similar idealized model exists for a hybrid position-rate control technique. Similar to clutching, the movement has two parts (Equation 5). T_1 is the time to move to the isotonic-elastic boundary and T_2 is the remaining time in the elastic zone to the target. Note that if the target is within reach of isotonic movement, then $T_2=0$ and T_1 can be predicted by Fitts' Law with the parameters a_i and b_i of an isotonic device. Otherwise, we suppose the movement distance in the first part is equal to the effective device operating range and $T_1=T_C$. T_2 can then determined using Fitts' Law with parameters a_e and b_e for an elastic device, and the remaining distance D_2 to the target:

$$T_2 = a_e + b_e \log_2\left(\frac{D_2}{W}+1\right) \quad (12)$$

Where D2 is simply:

$$D_2 = D - CD \cdot d \quad (13)$$

By substituting Equations 12 and 13 into Equation 5 and simplifying, we have a model for hybrid movement predicting target selection time:

$$T_a = T_c + a_e + b_e \log_2\left(\frac{D - CD \cdot d}{W}+1\right) \quad (14)$$

Comparison to Experimental Results

To test the validity of our models, we compared their predicted selection times with the results of our experiment. The following model parameters were used $d=40mm$, $CD=2$, $c=0.75$, $T_C=0.2s$ (T_C from our experiment). The Fitts' law parameters were from the literature $a_i=0$, $b_i=4.5$ [17,9], $a_e=0$, $b_e=2.0$ [9]. Considering the simplicity of the model, we found good fitness (Figure 9). The root mean square (RMS) is 0.4s for clutching and 0.2s for hybrid (The RMS for Fitts' law are respectively 1.1s and 0.6s). At D_S, the clutching model was 25% lower, likely due to the floor in Equation 7, while the experimental data presents a mean value. For example, at D_S, the predicted number of clutches is 3, but the experimental data is 3.75.

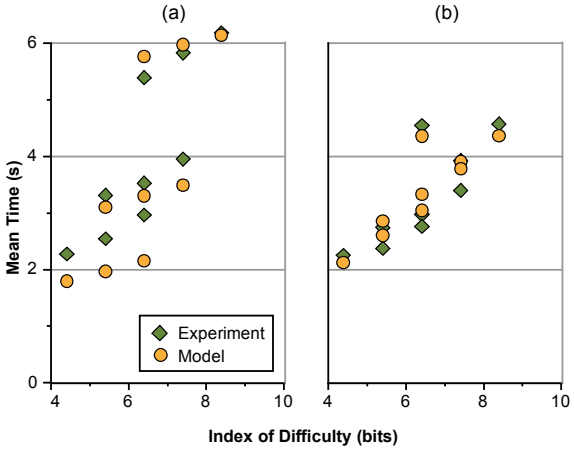


Figure 9: Predicted Model Time vs. Actual Time for: (a) Clutching Model; (b) Hybrid Model.

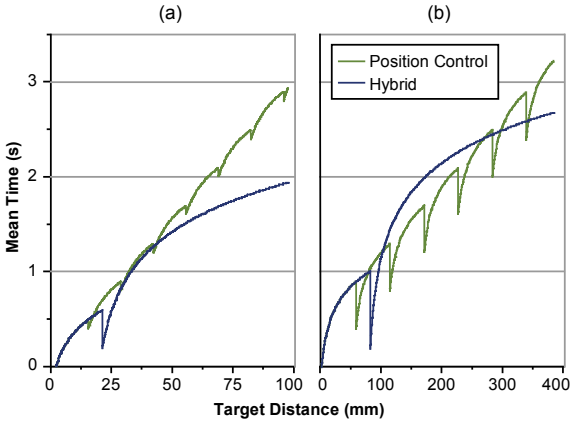


Figure 10: Theoretical Comparison for *W=4mm*: (a) Touch Pad PDA: *d=10mm*, *CD=2* (b) High Resolution Laptop: *d=40mm, CD=2*.

When is Hybrid Control Advantageous?

We can use the models to predict when hybrid control will have a performance advantage over position control.

For example, consider a laptop with a 38cm (15"), 1400 × 1050 pixel display, and 4cm touchpad. Using our theoretical model, RubberEdge hybrid control will outperform position control when targets are more than 30cm apart (nearly three-quarters of the maximum possible target distance) (Figure 10b). A second example is the HP iPAQ hx4700 PDA which has a 1cm touchpad and a 10cm (4") display. Here our model predicts an advantage for hybrid control above 5 cm (half the display distance) (Figure 10a).

The examples in Figure 10 also illustrate a potential drawback with hybrid control. Depending on the operating range size, one or two manual clutches can be faster than using the rate control zone. In our model this is attributed to the lower performance of elastic devices, but there may be other factors not accounted for, such as a constant mental transition time. Regardless of the reason, it appears that a hybrid device should allow standard isotonic clutching as well as an elastic zone. This way, the user can develop their own optimized strategy for reaching near or far targets.

PROTOTYPE DEVICE

With this more flexible hybrid model in mind, we built a device prototype that enabled a mix of isotonic clutching and elastic rate control. Our initial requirements were:

- cheap and compatible with current notebooks
- support for high resolution absolute input to measure the elastic zone penetration and compute pointer velocity
- support for relative position input and clutching in the same way as an existing device

We found that modifying a standard laptop touch pad fulfilled these requirements. Creating an elastic zone with the right feel and stiffness similar to the Phantom required some trial and error. We experimented with elastic materials like latex gloves, balloons, and elastic fabrics mounted on different types of frames. Eventually, we converged on

a simple design using a 1mm thick plastic frame cut from a old phone card. The frame has a 40mm hole with a plastic ring suspended by four rubber bands for elastic feedback (Figure 11). The ring is 36mm in diameter leaving 2mm for elastic movement. Plastic lets the ring slide easily on the touch pad surface and has just enough tactile feedback to define the boundary of the 30mm isotonic zone. We would have preferred creating a larger isotonic zone, but the borders of the frame had to support the elastic force. Adding physical constraints in this way is reminiscent of Wobbrock et al's EdgeWrite [25].

Our driver uses the Synaptics SDK [23] to measure the absolute finger position at 2000 DPI. In the isotonic zone, the pointer behaves like a standard Windows touch pad. When the finger enters the elastic zone, we transition using the RubberEdge mapping functions and compute the pointer velocity (Equations 3, 4). Our driver works like a standard Windows' pointing device with any application.

Early designs revealed that isotonic-elastic boundary accuracy is critical since there is only 2mm of movement. Imperfections in our fabrication and the non-uniform way in which a finger contacts positions around the ring led us to develop a two-step calibration (Figure 12). First, the boundary is defined by tracing around the perimeter of the ring. Then, to calibrate the maximum force (penetration distance) in each direction, the user pushes into the elastic zone at eight radial positions. We interpolate between these measurements when computing elastic rate control.

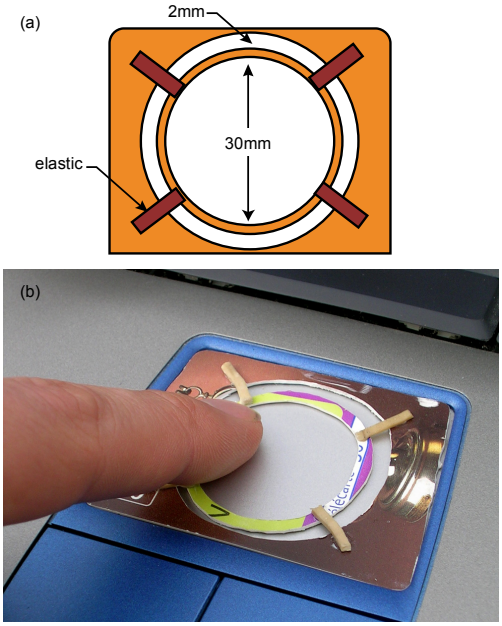


Figure 11: RubberEdge prototype device: (a) Schematic; (b) implementation.

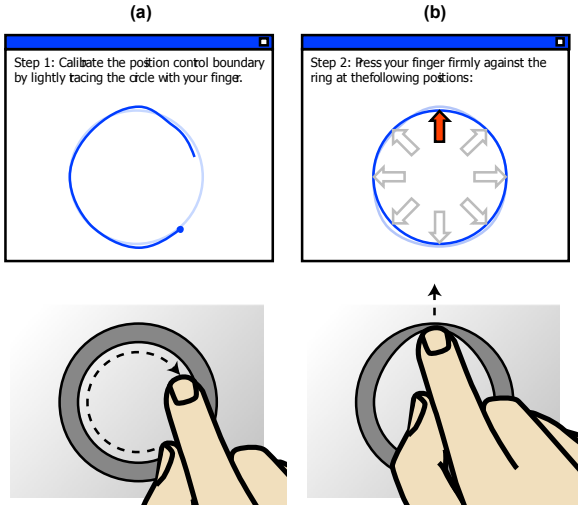


Figure 12: Two Step Prototype Calibration addresses caused by finger angle and fabrication: (a) Calibrating the boundary of the isotonic zone by tracing the finger clockwise around the perimeter; (b) Calibrating the maximum force by pushing the finger into the elastic zone at eight radial positions.

Initial Evaluation

We ran a pilot study with four participants to gather initial feedback about our prototype. We targeted people with touch pad experience since we were interested in the usability of our device, not touch pads in general; 3 participants used a touch pad daily and the fourth occasionally. All were right-handed. For approximately 20 minutes, participants used the device with common Windows tasks: file browsing, viewing PDF documents, painting, and web browsing. The tasks included pointing, window scrolling and steering through menus.

To grow accustomed to the device's reduced operating range, participants used only the isotonic zone with the rate control disabled for the first 2 minutes. We then enabled the elastic zone using a generic calibration, and gave no explanation or instructions. Participants immediately grasped that the pointer moved in two different ways depending whether you were pushing into the ring. The most difficulty was with elastic rate control: participants would at first overshoot the target, then sometimes overcompensate with hesitant and slow rate control. Past researchers have found that elastic rate control has a steep learning curve [26]. Two of the participants used overshooting as a kind of strategy: in the elastic zone, they shot the pointer as fast as possible past the target, then moved back to the target under isotonic control. However, after more practice, participants generally moved the pointer more accurately and with less hesitation using the elastic zone. Overall, participants said they liked using rate control for continuous movement of the pointer over far distances and appreciated the ability to use the isotonic zone for tasks like drawing.

CONCLUSIONS AND FUTURE WORK

RubberEdge hybrid position and rate control enables users to reach distant targets without clutching, yet still maintains benefits of position control for precise movements. Our mapping functions eliminate trajectory and velocity discontinuities when transitioning from isotonic position control to elastic rate control. The results of our controlled experiment found that hybrid control outperforms pure position control by 20% when there is significant clutching. This advantage is in spite of our related finding that a pointer acceleration transfer function will decrease clutching. We present theoretical performance models for position control clutching and hybrid position rate control, enabling designers to determine when hybrid control is beneficial. Based on our experimental and theoretical investigations, we developed a RubberEdge hybrid device for laptops which revealed design considerations such as construction, material, and calibration. With promising initial user feedback, we plan to further iterate our current prototype and investigate other types of RubberEdge hybrid devices.

REFERENCES

1. Accot, J., and Zhai, S. Scale effects in steering law tasks. In *Proc. of CHI'01*, 2001, pp. 1–8.
2. Bowman, D. and Hodges, L. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proc. of SI3D*, 1997, pp. 35–38.
3. Balakrishnan, R., and MacKenzie, I. S. Performance differences in the fingers, wrist, and forearm in computer input control. In *Proc. of CHI'97*, 1997, pp. 303–310.
4. Beaudouin-Lafon, M. Novel interaction techniques for overlapping windows. In *Proc. of UIST '01*, 2001, pp. 153–154.
5. Buxton, W. Lexical and Pragmatic Considerations of Input Structure*s. Comp. Graphics,* 17,1 (1983), 31–37.
6. Card, C., English, W., and Burr, B. Evaluation of mouse, rate controlled isometric joystick, step keys, and text keys for text selection on a crt. *Ergonomics*, 21 (1978), 601–613.
7. Card, S. K., Mackinlay, J. D., and Robertson, G. G. A morphological analysis of the design space of input devices. *ACM Trans. Inf. Syst.*, 9, 2 (1991), 99–122.
8. Dominjon, L., Lécuyer, A., Burkhardt, J.M., Andrade-Barroso, G., and Richir, S. The bubble technique: Interacting with large virtual environments using haptic devices with limited workspace. In *Proc. of IEEE World Haptics*, 2005, pp. 639–640.
9. Douglas, A. and Mithal, A. The effect of reducing homing time on the speed of a finger-controlled isometric pointing device. In *Proc. of CHI'94*, 1994, pp. 411-416.
10. Fitts, P. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47, 6 (1954), 381–391.
11. Gibbs, C.B. Controller design: interactions of controlling limbs, timelags, and gain in positional and velocity systems. *Ergonomics*, 5 (1962), 385– 402.
12. Jellinek, H. and Card , S. Powermice and user performance In. *Proc. of CHI'90*, 1990, pp. 213–220.
13. Johnsgard, T. Fitts' Law with a virtual reality glove and a mouse: Effects of gain. In *Proc. of Graphics Interface Conference,* 1994, pp. 8–15.
14. Kruger, R., Carpendale, S., Scott, S. D. and Tang, A. Fluid integration of rotation and translation. In *Proc. of CHI '05,* 2005, pp. 601–610.
15. Langolf, G. D., Chaffin, D. B. and Foulke, J.A.. An investigation of Fitts' law using a wide range of movement amplitudes. *Journal of Motor Behavior*, 8 (1976), 113–128.
16. MacKenzie, S. and Oniszczak, A. A comparison of three selection techniques for touch pads. In *Proc. of CHI'98,* 1998, pp. 336–343.
17. MacKenzie, I. S, Sellen , A., and Buxton, W. A comparison of input devices in element pointing and dragging tasks. In *Proc. of CHI'91,* 1991, pp. 161–166.
18. MacKenzie, I.S. Fitts' law as a research and design tool in human computer interaction. *Human Computer Interaction,* 7 (1992), 91–139.
19. MacKenzie, S. and Riddersma, S. Effects of output display and control-display gain on human performance in interactive systems. *Behavr. and Info. Tech.*, 1,3 (1994), 328–337.
20. Plamondon, R., and Alimi, A. M. Speed/accuracy tradeoffs in target-directed movements. *Behavioral and Brain Sciences*, 20, 1997, 279–349.
21. Pointer ballistics for Windows XP/Vista. http://www.microsoft.com/whdc/device/input/pointer-bal.mspx.
22. Synaptics, EdgeMotion. http://www.synaptics.com.
23. Synaptics, SDK. http://www.synaptics.com/support/dev_support.cfm
24. Tactile 3D. http://www.tactile3d.com
25. Wobbrock, J. O., Myers, B. A., and Kembel, J. A. EdgeWrite: a stylus-based text entry method designed for high accuracy and stability of motion. In *Proc. of UIST '03*, 2003, pp. 61–70.
26. Zhai, S. Human Performance in Six Degree of Freedom Input Control. PhD thesis, University of Toronto, 1995.
27. Zhai, S. Characterizing computer input with Fitts' law parameters - the information and non-information aspects of pointing. *Intl. Jrnl. of Human-Computer Studies,* 61,6 (2004), 791–809.
28. Zhai, S., Smith, B. A., and Selker, T. Dual stream input for pointing and scrolling. In *Proc. of CHI '97 Ext. Abstracts*, 1997, pp. 305–306.

Giving a Hand to the Eyes: Leveraging Input Accuracy for Subpixel Interaction

Nicolas Roussel[1], *Géry Casiez*[1,2,3], *Jonathan Aceituno*[1] *and Daniel Vogel*[4]
[1]Inria Lille & [2]LIFL, [3]University of Lille, France
[4]Cheriton School of Computer Science, University of Waterloo, Canada
nicolas.roussel@inria.fr, gery.casiez@lifl.fr, jonathan.aceituno@inria.fr, dvogel@uwaterloo.ca

ABSTRACT
We argue that the current practice of using integer positions for pointing events artificially constrains human precision capabilities. The high sensitivity of current input devices can be used to enable precise direct manipulation "in between" pixels, called subpixel interaction. We provide detailed analysis of subpixel theory and implementation, including the critical component of revised control-display transfer functions. A prototype implementation is described with several illustrative examples. Guidelines for subpixel domain applicability are provided and an overview of required changes to operating systems and graphical user interface frameworks are discussed.

ACM Classification: H.5.2 [Information interfaces and presentation]: User interfaces - Graphical user interfaces.

General terms: Design, Human Factors

Keywords: Display density; indirect pointing; input device sensitivity; device's human resolution; subpixel interaction; direct manipulation

INTRODUCTION

Over the last thirty years, there have been tremendous increases in computer processing power, storage capacity, and network bandwidth. Graphical user interfaces (GUIs) have played a crucial part in making these resources available, enabling the direct manipulation of data, which has also increased substantially in diversity and size. However, while processing, storage, and communication capabilities have experienced a hundred, thousand, or million-fold increase to handle these increased data manipulation requirements, the situation is quite different for computer displays. Most modern displays typically have a pixel density lower than 150 PPI (pixels per inch) and cannot display more than 2.5 megapixels[1]. Display capabilities have increased by only a factor of 21 compared to the original Macintosh.

[1]see `http://libpointing.org/resolution/` for information on the sensitivity and pixel density of commercial mice and displays

UIST'12, October 7–10, 2012, Cambridge, MA, USA.

When data density exceeds display density, direct manipulation becomes a problem. For example, selecting an item from an overview of a large discrete set, or fine adjustment of a continuous variable. Solutions usually involve a scale adjustment in visual space or layering transfer functions to reach the desired control precision. But these approaches take for granted the curious way in which operating systems map input device movements to data.

When you move a pointing device, motion deltas (in device-specific units) are sent to the I/O subsystem and transformed by a transfer function [5] into on-screen pointer motions (in pixels). The system then moves the pointer accordingly and generates *movement events* which are routed via the window subsystem and GUI framework to a widget to manipulate data (Figure 1a). What is curious is that although these are commonly called "mouse events," they are actually "pointer events" because the information they carry describe on-screen *pointer movements*, not *mouse movements*. So, a pointing device is not really how we interact with data: it is a device through which we interact with an on-screen pointer, through which we interact with data. Thirty years ago, mouse sensitivity and display density were comparable, so this kind of input mapping seemed reasonable. The original Macintosh mouse was 90 CPI (counts per inch), quite close to its ≈72 PPI display density. The problem is that this mapping still serves as the basis for all graphical interactions,

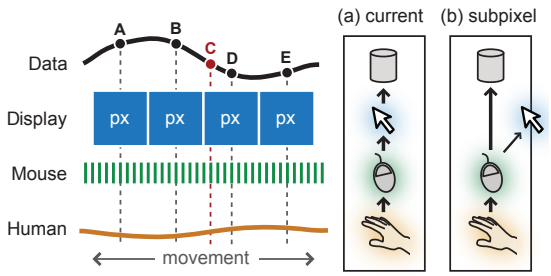


Figure 1: Input mappings: (a) currently, human movements are discretized by mouse sensitivity, then again by display density: data points "in between" pixels like 'C' are unreachable; (b) a subpixel mapping discretizes human movements by mouse sensitivity only, for precise data manipulation.

and since movements are measured in pixels, they are only as accurate as the display density.

Fifteen years ago, in his classic paper "*The Eyes Have It (...)*" [12], Shneiderman pointed out how our remarkable perceptual abilities were often underutilized. From what we just described, it is fair to say that our remarkable motor abilities are currently underutilized. To put it simply, the eyes could surely use a hand. The sensitivity of modern mice is now typically over 800 CPI and can reach more than 10000 CPI, and some touchpads are 1000 CPI — levels of precision more suitable for capturing high human sensitivity [4]. It is time the current input mapping is revised.

We are proposing *subpixel interaction*, a subtle, but critical alteration to current system architectures where device movements are mapped directly to data, enabling interaction "between pixels" (Figure 1b). Our solution requires a small software-level modification, yet increases the accuracy of standard interaction techniques and remains compatible with complimentary approaches like Focus+Context lenses. After describing motivating situations where pixel-precision is inadequate and reviewing related work, we describe the details of our solution. Specifically, guidelines for useful subpixel resolution and, since our focus is on indirect pointing devices like mice and touchpads, we describe revised transfer function characteristics which leverage subpixel capability. Our work is a rallying cry with implementation details to leverage human input capability for precise direct manipulation.

WHY PIXELS ARE NOT ENOUGH

To illustrate the current problem more concretely, consider navigating a video player where frame selection accuracy is ultimately limited by the pixel width of the display. For example, the Apple OS X QuickTime player has a fixed timeline slider width of 315 pixels (Figure 2a). Moving the slider 1 pixel skips a 5 minute video by 1 second, but a 1.5 hour movie is skipped by 17 seconds (Table 1). This may be fine for coarse positioning, but tasks like skipping *only* commercials becomes tedious or impossible as the video length increases. Even more extreme is accurately selecting specific frames with Apple Keynote's video inspector which uses a 198 pixel, fixed width slider.

There are a myriad of other examples where there are not enough pixels to provide the required precision: picking a particular item in a large list using a slider; image navigation and editing (e.g. accurately cropping a megapixel image, picking a precise location on a map); resizing a calendar event to minute precision (e.g. when booking flight departures or lawyers tracking billing time); high precision vector drawing tasks (e.g. drawing a structural wall to centimeter precision or aligning objects in vertex-dense areas); or manipulating objects in 3D applications (e.g. setting orientation of CAD objects to within 0.1°).

	Duration	Keynote	QT Player
CHI video	05:00	00:01.515	00:00.952
TV Show	52:00	00:15.758	00:09.905
Movie	1:30:00	00:27.273	00:17.143
Gone with the Wind	3:58:00	01:12.121	00:45.333
Bergensbanen	7:14:13	02:11.581	01:22.708

Table 1: Time between pixels in the Keynote inspector and QuickTime Player sliders for different video durations.

A solution to these pixel precision problems could be to simply add more pixels. Returning to the video player example, consider enabling a more reasonable 1 second selection accuracy with a 1.5 hour video by increasing the timeline slider width. Unfortunately, the slider would need to be 5400 pixels wide, the width of three 1080p HDTVs. This is clearly unworkable, so a common way to overcome the limitation is to introduce alternate navigation controls such as arrow keys, but these can be slow and error prone, partly because they no longer follow principles of direct manipulation. Next, we discuss how researchers have approached the pixel precision problem while trying to maintain the benefits of direct manipulation.

RELATED WORK

To make direct manipulation more accurate, previous work introduces explicit *precise pointing modes* such as discrete or continuous zooming and Focus + Context Lenses. In addition to problems inherent with modes, these techniques are impeded by the assumption that whole-pixel input resolution is a hard constraint – they ignore the high resolution capabilities of modern pointing devices and human limb accuracy.

Increasing Pointing Accuracy with Modes

A straightforward way to increase input accuracy is to *zoom* and magnify the desired target area to make pointing easier. For example, the OS X QuickTime Player 10.1 has what is essentially a dwell-while-dragging zoom mode to se-

Figure 2: Manipulating visual space to increase pointing accuracy: (a) dwelling while dragging the OS X QuickTime Player 10.1 timeline slider *zooms* the timeline in to provide 1 second precision in a 10 second window; (b) some YouTube videos have a Focus+Context timeline with a secondary slider providing 1 second precision.

lect video frames more precisely (Figure 2a). Other examples include Popup Vernier [3] which supports explicit incremental cursor space magnification during drag operations, and Ramos and Balakrishnan's Zliding technique [10] which continuously maps pen tip pressure to view scale. These examples are part of the more general class of zoomable user interfaces (ZUIs) [8] where the zoom level is adjusted using an explicit control like dwell, pressure, keyboard keys, mouse scroll wheel, or GUI buttons, sliders, or textfields. However, zooming requires an explicit cognitive decision, locating the optimal zoom level to facilitate precise pointing can be time consuming and difficult, and the overall context of the information space is lost while zoomed in.

To maintain context, Focus+Context Lenses embed a magnified portion of the information space in the overview view [9]. Although this addresses some shortcomings of ZUIs, Appert et al. [2] argue that when the input device is used to simultaneously move the lens *and* point at targets in the lens, there is a *quantization problem* limiting the usable range of lens magnification levels. Their solution is also to introduce an explicit mode switch, this time to switch between positioning the lens and pointing at targets inside the lens. Techniques like Alphaslider [1] and YouTube's secondary magnified timeline slider (Figure 2b) are one-dimensional examples of the Focus+Context strategy. Since they are one-dimensional, the switch to the accurate mode is achieved spatially, by acquiring the magnified slider.

Another solution is to switch to a mode with a custom precise pointing transfer function. This can be achieved with a technique called *pointer lock* where "*input methods of applications [are] based on the movement of the mouse, not just the absolute position of a cursor*" [11]. This is often used for first-person navigation in games, but can also be used for modal precise control of parameters like object rotation in 3D applications. Since direct manipulation no longer follows the system pointer, the pointer is usually hidden or "locked" in place to reduce user confusion.

Unfortunately, using pointer lock to enable a modeless subpixel solution is not practical. For the second transfer function to increase accuracy, it must scale down the relatively large integral movement deltas received from the system event (Figure 3). This will increase motor space which lowers comfort and performance due to device clutching. In addition, since precise pointing is necessarily mapped to low speeds which have a lower-bound constrained by integral deltas, then larger 'normal' pointing movements will be harder to control since the speed-to-position mapping is condensed. Finally, since the precise transfer function is layered, it is has to compensate for different hardware, software, and user settings which determine the underlying system transfer function [5]. Like the other techniques above, using pointer lock does not harness the potential precision of modern input devices.

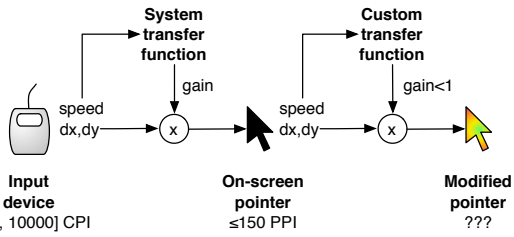


Figure 3: Using pointer lock for subpixel interaction amounts to layering a second precise transfer function in which low speeds are constrained by integral movement deltas.

Lost Capabilities of Input Devices and Human limbs

Traditional GUI toolkits like Java Swing report pointer movement events as integers. Even toolkits like Qt or WPF which are designed for display resolution independence[2] report motion events as integers, regardless whether they are specified as floating point (with WPF, this is due to integral `WM_MOUSEMOVE` messages). On OS X, position events are floating point, but these only hold non-integral values with absolute pointing devices such as tablets. For relative devices, Casiez and Roussel [5] show that pointing transfer functions used by Microsoft Windows, OS X, and X.Org apply a speed-dependent float factor on motion deltas but move the pointer according to the sole integral part of the result. Although the fractional part is preserved internally for later accumulation, it is inaccessible from high-level software.

Raw motion deltas can usually be obtained (e.g. through `WM_INPUT` messages) and are commonly used by games for custom high-sensitivity pointer control. But until recently [5], the lack of appropriate knowledge and tools made it quite difficult for this custom pointer to behave like the system one for "normal" pointing tasks. The idea of using floating coordinates to improve the precision of the system pointer with relative devices was once suggested within the X.Org community[3], but received very little attention.

The sensitivity of modern pointing devices ranges from 400 up to 10000 CPI or more for high-end mice, enabling minimal measurable displacements of 0.0635 mm and 0.00254 mm respectively. Whether people can leverage this level of precision depends on their ability to control fine movements of input devices in motor space. In a multi-scale pointing experiment, Guiard et al. [7] found that users can comfortably acquire 0.06 mm targets in motor space (423 CPI), but this is not an upper bound since it was the smallest width they evaluated. More recently, Bérard et al. [4] defined a *Device's Human Resolution* (DHR) as "*the smallest target size that users can acquire with an ordinary amount of effort using a particular device*". They found that the DHR depends on the input device and the user's sensorimotor capabilities. For computer mice they found DHR values from 700 CPI to 1400 CPI. This means that humans can easily exploit input resolutions up to 7 to 14 times higher than a typical 100 PPI display. The challenge is how to translate these human

[2] i.e. where graphical objects are positioned using floating point coordinates in a space independent of the screen

[3] `http://johan.kiviniemi.name/blag/making-x-report-the-mouse-position-with-subpixel-precision/`

capabilities into effective control of direct manipulation interfaces.

LEVERAGING INPUT ACCURACY

Direct manipulation interfaces translate objects of interest into physically manipulatable, graphical representations. Each object is composed of a collection of underlying data, often referred to as a model. Models can represent discrete elements (e.g. integers, video frames) and continuous elements (e.g. floating values). The range of elements may be bounded (e.g. choosing a video frame from a clip) or unbounded (e.g. specifying a CAD object's length). In practice, unbounded models can be considered bounded given reasonable minimum and maximum values for the task, and continuous models may be considered discrete given a reasonable level of precision for the task. By treating all models as discrete, we can define the model's cardinality (N) as the number of elements which users expect to select from.

When display density prevents the object from being represented at the desired granularity (i.e. with the right level of detail), auxiliary representations are needed such as text labels, zoomed views, or even sounds or haptic effects. This solves the output granularity problem, but the problem remains for input. This is what motivates us: we want to overcome the problem of input granularity being limited by display density; we want to support physical actions on objects with a higher precision than the display. As discussed earlier, there is a hidden potential in input devices that is compatible with the degree of control that people can exert. We would like to take advantage of this potential in a way that smoothly integrates with current practices.

In this section we develop a way to provide subpixel input while maintaining current graphical object representations and without introducing any explicit "high precision" mode. We preserve the on-screen pointer since it provides essential visual feedback for direct manipulation. We alter the mapping between movements in motor space and visual space, but we do this without changing the current "feeling" of pointing: we avoid situations where the cursor feels stuck and then suddenly moves quickly, and we do not extend the device operating space. People continue to interact as they do now, but with higher, subpixel precision when needed.

Taking the input device into account

To leverage input accuracy, the system first needs to know the device sensitivity, expressed in device-independent units (such as CPI). This is important so that the system understands what it measures: a higher sensitivity should result in a more precise movement, not a change of magnitude. Unfortunately devices and hardware drivers typically do not provide this information (and probably for that reason, the Windows and X.Org pointing transfer functions completely ignore it [5]).

As already explained, pointing transfer functions apply a dynamic gain factor on device displacements, move the pointer according to the integral part of the result, and store the fractional part internally. We want that information all the way up: the fractional part should be accessible in an easy way to the code actually responsible for the interaction. We propose to generalize what some systems do for absolute devices: expose the fractional part by using floating values for the pointer coordinates. To summarize, we propose first that transfer functions should take into account device sensitivity; and second, that float values resulting from these computations should be forwarded *as is* to higher-level code.

This solution takes the sensitivity of the input device into account so that movements in motor space are no longer magnified by input sensitivity or constrained by display density. In theory, granularity of control can be increased 100 times for a high end 10000 CPI mouse (assuming a display density of 100 PPI). However, standard transfer functions must be adapted to unlock this potential, and this adaptation must be done carefully to maintain normal cursor behavior.

Taking the user into account

Having a device with a high sensitivity does not necessarily mean one can fully take advantage of it: users' capabilities and limitations should also be taken into account. As noted above, Bérard et al. found the *Device's Human Resolution* ($\text{RES}_{\text{human}}$) for mice to be in the range of 700 - 1400 CPI [4]. Devices with a sensitivity ($\text{RES}_{\text{input}}$) below $\text{RES}_{\text{human}}$ are not accurate enough to capture fine movements in motor space. But at the same time, users are not likely to benefit from higher sensitivities. We can thus define the *useful resolution* ($\text{RES}_{\text{useful}}$, in CPI) of a device and *epsilon* (ε, in inches), the smallest measurable displacement one can produce with it:

$$\text{RES}_{\text{useful}} = \min\left(\text{RES}_{\text{input}}, \text{RES}_{\text{human}}\right) \quad (1)$$

$$\varepsilon = \frac{1}{\text{RES}_{\text{useful}}} \quad (2)$$

Considering the current unitless gain applied by the pointing transfer function (G), we can calculate the number of practicable subpixels (S) as:

$$S = \frac{1}{\text{RES}_{\text{screen}}} \times \frac{\text{RES}_{\text{useful}}}{G} \quad (3)$$

where $\text{RES}_{\text{screen}}$ denotes the pixel density of the display (in PPI). S is actually the number of *epsilons* required to move from one pixel to another. Note that S necessarily varies over time since G is dynamic (speed-dependent, based on user movements). Note also that the highest S value $S_{\max}$ corresponds to the lowest gain level $G_{\min}$.

Adapting the transfer function

Modern pointing transfer functions are discrete, speed-dependent, and were generally designed for a 400 CPI mouse sampled at 125 Hz and a 96 PPI display refreshed at 60 Hz [5]. They typically produce gain values between 0.8 and 4.1 for motor speeds under 2.5 $cm.s^{-1}$ and then smoothly transition from these values to high ones at high speeds. Figure 4 shows a sample sigmoid function producing gain values in the same range for a particular hardware configuration. Assuming $\text{RES}_{\text{human}}$ = 1000 CPI and applying Equation 3 to $G_{\min} = 1.0$, this function would provide access to $S_{\max} \approx 11$ subpixels. But what if this was not enough?

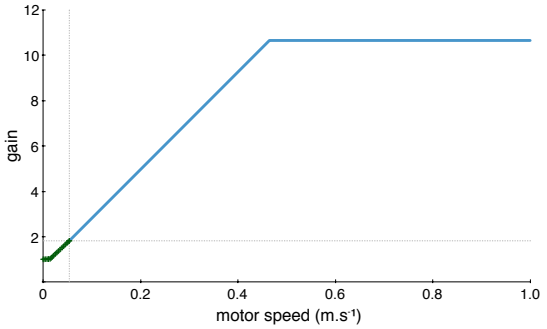


Figure 4: Sample sigmoid transfer function. The plot was made considering a 4000 CPI mouse sampled at 500 Hz and a 90 PPI display refreshed at 60 Hz. Green crosses indicate the 22 gain levels that generate subpixel motion.

We can determine the optimal gain G_{opt} which provides enough subpixels to select among all model elements with cardinality N given P pixels of screen space:

$$G_{opt} = \frac{P}{\text{RES}_{screen}} \times \frac{\text{RES}_{useful}}{N} \quad (4)$$

From this equation, it can be seen that when $P < N$ (i.e. $G_{opt} < \text{RES}_{useful} / \text{RES}_{screen}$) subpixels are beneficial. For example, using the video durations in Table 1 with RES_{useful} = 1000 CPI and RES_{screen} = 90 PPI, a one second accuracy in the QuickTime Player corresponds to a G_{opt} of 1.121 for the TV show, 0.648 for the generic movie, 0.245 for *Gone with the Wind*, and 0.134 for *Bergensbanen*. If we compare these numbers to the gain values in Figure 4, we see that the function should work for the TV show as-is (the corresponding G_{opt} is higher than G_{min}), but not the other videos. Achieving frame accuracy is even more challenging. Assuming a frame rate of 30 fps, G_{opt} values are much lower (0.037, 0.022, 0.008, and 0.004). In order to approach these optimum gain levels, we need a strategy for function adaptation.

To maintain normal pointer behavior, we alter an existing transfer function so that it produces G_{opt} at low speeds. Specifically, we calculate V_{min}, the minimum speed in meters-per-second on which the transfer function operates, and V_{use}, the speed associated with RES_{useful}:

$$V_{min} = \frac{0.0254}{\text{RES}_{input}} \times \text{FREQ}_{input} \quad (5)$$

$$V_{use} = \frac{0.0254}{\text{RES}_{useful}} \times \text{FREQ}_{input} \quad (6)$$

where FREQ_{input} is the frequency of the pointing device. Since there is no guarantee that people can actually move the device at such a slow speed like V_{min}, we maintain the G_{opt} value for all speeds under at least V_{use}. To maintain normal pointer behavior, we smoothly transition between this point (V_{use}, G_{opt}) and the point where the function starts producing

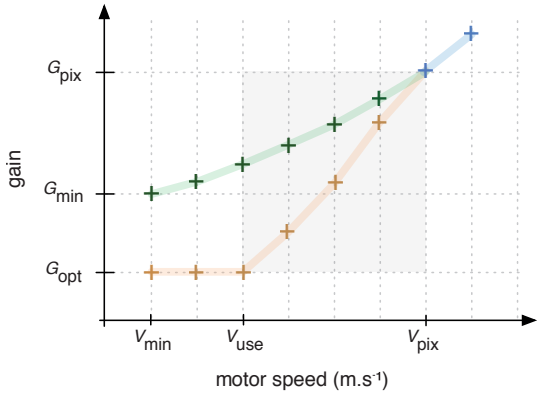


Figure 5: Closeup of the low-speed domain of a transfer function adapted to a particular model. Green and blue crosses show the original gains (subpixel & pixel). Yellow crosses show a possible adaptation.

motions of 1 pixel or more (V_{pix}, G_{pix}). Note that like G_{min} and unlike V_{min} and V_{use}, the values of V_{pix} and G_{pix} depend on the function definition.

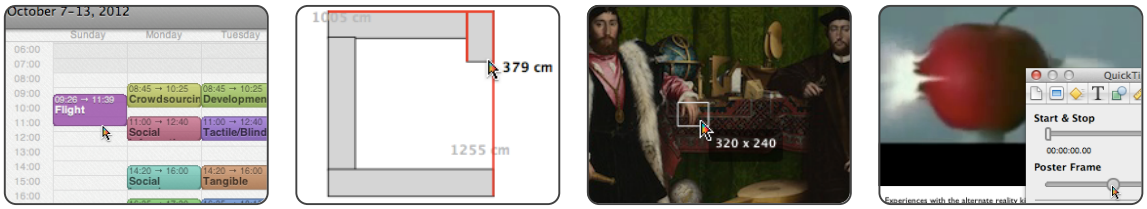
Figure 5 shows a closeup of the low-speed domain for a hypothetical transfer function that one might want to adapt. The blue crosses show the points of the original function that produce pixel motions, the green ones subpixel motions. The yellow crosses illustrate a possible interpolation between (V_{use}, G_{opt}) and (V_{pix}, G_{pix}). The distance between V_{use} and V_{pix}, the one between G_{opt} and G_{pix}, and FREQ_{input} constrain the interpolation. One needs enough time steps to keep a reasonable distance between interpolated gain levels to limit the risks of overshooting. At the same time, one does not want to alter the gain values corresponding to pixel motions since this would result in a possibly perceivable modification of the pointer behavior. Note that in a worse-case scenario, there might be no subpixel speed/gain combination (i.e. green cross) in the original function.

In the next section, we provide concrete examples of how this method can be applied to adapt a transfer function to a particular model, taking into account device and human capabilities.

ILLUSTRATIVE EXAMPLES

To create illustrative examples of subpixel interaction (Figure 6), we developed a cross platform software application written in C++ that runs subpixel-enabled applications through the WebKit browser engine. We used libpointing [5] to get raw information from devices, apply transfer functions taking into account hardware characteristics (input and output resolutions, frequencies) as well as DHR, and access remainders to produce floating pointer coordinates to control a custom pointer.

We used a sigmoid transfer function, similar to Figure 4 which is representative of transfer functions used by modern operating systems. Equation 7 describes this function



(a) (b) (c) (d)

Figure 6: Illustrative subpixel interaction applications: (a) manipulating calendar events with minute precision; (b) precise CAD object dimensioning; (c) image pixel accurate cropping; (d) selecting one frame of a video.

which we refer to as F_G. It produces unitless gain values between 1.0 and 10.0, where dX represents a relative distance measured in motor space (in meters), dt is the elapsed time since the last input event (in seconds), $v = dX/dt$, and v_1 and v_2 are equal to 0.15 $m.s^{-1}$ and 0.5 $m.s^{-1}$ respectively. Thus, $F(dX,dt) = dX \times F_G(dX,dt)$ is the transfer function used by default to control the pointer.

$$F_G(dX,dt) = \begin{cases} 1 & : v \leq v_1 \\ 1 + 9 \times \frac{v-v_1}{v_2-v_1} & : v \in]v_1; v_2[\\ 10 & : v \geq v_2 \end{cases} \quad (7)$$

When interacting with an object, we first compute G_{opt} as defined by Equation 4. If G_{opt} is below the minimal gain value defined by Equation 7 ($F_G(dX,dt) = 1$), then the transfer function is blended with G_{opt} to yield the function $H(dX,dt) = dX \times H_G(dX,dt)$, where H_G is described by Equation 8:

$$H_G(dX,dt) = \begin{cases} G_{opt} & : v \leq V_{use} \\ F_G(dX,dt) & : v > V_{pix} \\ (1-q)G_{opt} + q\, F_G(dX,dt) & : \text{elsewhere} \end{cases} \quad (8)$$

$$q = \frac{v - V_{use}}{V_{pix} - V_{use}}$$

We used a 22" Dell 2208WFP display at its native resolution of 1680×1050 at 60 Hz, providing a pixel density of about 90 PPI. Our input device was a Microsoft Sidewinder X8 mouse with a 500 Hz polling rate. The mouse sensitivity was set to 4000 CPI, corresponding to its maximum configurable value. We fixed $RES_{useful} = 1000$ CPI. Combined with our particular transfer function, this configuration leads to the following values:

$$S_{max} \approx 11 \text{ for } G_{min} = 1.0$$
$$V_{use} = 0.0127 m.s^{-1}$$
$$G_{pix} = 1.0$$
$$V_{pix} = \frac{0.0254}{RES_{screen}} \times \frac{1.0}{G_{pix}} \times FREQ_{input} \approx 0.141 m.s^{-1}$$

$V_{pix}/V_{min} \approx 44$ discrete speed values below V_{pix} are available for subpixel interaction and transfer functions blending.

Calendar

It is often convenient to enter and modify calendar events using direct manipulation. Depending on the number of hours in the current view, it may not be possible to set event times like flight or train departures requiring one minute precision: a height of 720 pixels is needed for a 12 hour view and 1080 pixels for 18 hours. These heights may exceed typical calendar window space. For example, an 18 hour view represented in a 400 pixel window requires a modest 2.7 subpixels for one minute precision. With subpixel input, it is even possible to represent a month view while still enabling individual event manipulation to the minute. The calendar application we created represents 17 hours in 272 pixels, yielding 16 pixels per hour (Figure 6a). With our particular setup, this view requires 3.75 subpixels, which our transfer function F_G provides. For subpixel feedback, start and end times are shown on each calendar event.

Computer-aided design

Computer-aided design (CAD) applications require precise specification of object dimensions and placement. Direct manipulation is preferable given the graphical nature of the task, but to achieve the required precision, current applications resort to text entry or zooming. Precise feedback is often shown already as a numerical measurement near the manipulated object, but with subpixel input, the manipulation can be made equally precise. We developed a small subset of an architectural CAD application where it is possible to dimension and position walls with one centimeter precision while maintaining a view of the entire structure. For example, consider a room plan where three walls are already drawn, and a fourth vertical wall of length 2205 cm must be created and aligned exactly. The whole drawing is viewed at a scale which maps 10 cm to 1 pixel (i.e. 1 : 354). Here 10 subpixels are required to perform the task with required precision (1 cm), which our transfer function F_G again provides.

Cropping a high resolution image

Many digital cameras capture images exceeding 10 megapixels, yet most computer displays cannot show more than 2.5 megapixels. Cropping such an image to specific dimensions, e.g. 320 by 240 pixels, usually requires zooming, and the crop area can be so large that multiple zoom and pan operations are required making the task difficult. For example,

consider a 17.8 megapixel image measuring 4215 by 4215 pixels. To show this entire image in a 300 by 300 pixel window, it must be scaled to 7% where one display pixel corresponds to 14 image pixels. To achieve image pixel precision, 14 subpixels are required. Since this is more than $S_{\max}$, we need to switch from F_G to H_G (Equation 8) with $G_{opt} \approx 0.794$ (Equation 4). To facilitate subpixel output, a tooltip shows image-space coordinates for the subpixel pointer and image-space size of cropping rectangle (Figure 6c). When cropping according to visual details, lens-like feedback would be more appropriate.

Selecting a frame in a video

We replicated the Apple Keynote video interface where a 198 pixel wide slider is used to select a single "*Poster Frame*" (Figure 6d). As an example, consider the TV show of Table 1. At 30 frames per second, one pixel of the slider corresponds to $52 * 60 * 30/198 = 472.72$ frames (i.e. 15.75 s) in the video. To achieve frame-level precision, 473 subpixels are required: we again need to switch from F_G to H_G (Equation 8), this time with $G_{opt} \approx 0.023$ (Equation 4). We informally tested this scenario with several participants. In spite of the low value for G_{opt}, all managed to comfortably reach the intended frame and no one spontaneously noted any change in pointer behavior. Note that selecting one frame in this scenario corresponds to a 16.51 bit task.

The strategies described in the above examples are specific to the transfer function F_G and the particular hardware configuration that we used. Had we used the same monitor with a low-end mouse (400 CPI, 125 Hz) and the default transfer function of Windows, OS X or Xorg, $S_{\max}$ would have been less than 4. In all cases except the simplest one of the Calendar example, it would have been necessary to switch from F_G to H_G.

DISCUSSION

In this section we provide guidelines for applying subpixel in generalized data manipulation situations and enumerate modifications which must be made to current operating systems and GUI toolkits.

Domain of applicability for subpixel interaction

There are three parameters which determine the applicability of a subpixel-enabled floating pointer and custom transfer functions: the number of available pixels (P), the number of practicable subpixels (S, from Equation 3) and the cardinality of the underlying model (N). Using these we can define two critical values for N: $N1 = S \times P$, the value above which custom transfer functions must be used to reach all values of the model; and N2, the value above which custom transfer functions can no longer operate due to usability issues. These determine four zones (illustrated in Figure 7):

- $N \leq P$: all values of the model can be addressed with a standard integer pointer, subpixel interaction is unnecessary but compatible;
- $P < N \leq N1$: all values of the model can only be addressed with subpixel interaction, but a standard transfer function is compatible without any change in pointer behavior;
- $N1 < N \leq N2$: to address all model values with subpixel interaction, a custom transfer function like those described earlier is required;
- $N > N2$: all model values cannot be addressed with subpixel interaction.

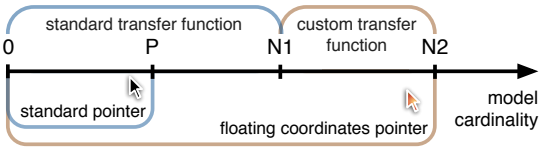


Figure 7: Four zones of applicability for subpixel and custom transfer functions (see text for description).

Impact on operating systems and GUI frameworks

As stated above, our subpixel prototypes were implemented using libpointing [5] to create a custom transfer function taking into account actual input device CPI and screen PPI. We also took into account human precision capabilities (DHR) and the cardinality of the model being manipulated to adjust the transfer function when required[4].

To support subpixel floating point coordinates universally, operating systems and GUI frameworks need to make relatively small, but fundamental changes. First, system transfer functions must take into account input device sensitivity so that higher device sensitivities result in higher precision. Currently, operating systems generally translate higher device sensitivity to faster pointer movements either because they do not take the sensitivity information into account or because it is not provided by the device [5]. Device configuration interfaces definitely need to be re-designed.

Operating systems then need to use the remainders stored between two input events to create floating point coordinates. These subpixel coordinates would be dispatched by the windowing system, and ultimately received by the GUI framework. Framework event loops, methods, callback methods, etc. also need to be updated to handle floating point coordinates.

GUI frameworks also need to provide developers with a way to take into account human limb resolution directly or, even better, indirectly. Ideally, a developer would specify model cardinality and the framework would alter the transfer function used when a subpixel widget is manipulated. This also requires that operating system transfer functions can be changed dynamically, which is already achievable to some extent on some systems. Frameworks with dynamic layout capabilities can also adjust the size of widgets, making a subpixel enabled slider smaller without sacrificing precise control of the underlying data. For example, the preferred size of a slider would correspond to N1, and the minimal size of a widget would correspond to N2.

Since limb resolution varies between individuals [4], the operating system should provide a method to specify the current user's level of limb precision. This could be part of the input

[4]Basic source code to create a subpixel libpointing application is available from `http://libpointing.org/`

device settings with default values set conservatively based on the literature. Ideally, a simple calibration step would tune this value to specific individual, perhaps using a game-like procedure [6] rather than a dry experiment task like Bérard et al. [4]. Since human resolution can improve with practice, the calibration process should be updated intermittently, or automatically adapted over time based on patterns of use – like the strategy of adjusting the offset distance in the Shift pointing technique [13].

CONCLUSION AND FUTURE WORK

Subpixel interaction is a fundamental way to increase direct manipulation accuracy. For too long, positional input has been artificially constrained by design decisions made when input devices were about as accurate as the pixel density of a display. Now that input device sensitivity far surpasses display capabilities, the subpixel methods, transfer functions, and guidelines described above can enable interaction at a level of precision bounded only by human capability. Best of all, subpixel interaction does not change the way people interact, remains compatible with other precision techniques like ZUIs and Focus+Context, and would only require moderate changes to current operating systems and toolkits.

Our focus so far has been to enable subpixel interaction, but there is future work to extend its application context and test the potential benefit in actual settings. For example, formal testing of subpixel-enabled interfaces with realistic direct manipulation scenarios, experimental analysis examining the trade-off between pure subpixel interaction and subpixel interaction augmented by traditional precision techniques like ZUIs and Focus+Context, and investigating subpixel applications to very sensitive absolute touchscreens. There is also opportunity to extend the work of Bérard et al. [4] by measuring DHR with a wider range of devices and a broader range of participants. This will add even more substance to the overarching subpixel philosophy: pixels may be the limit of what we can see, but they should not limit what we can do.

REFERENCES

1. C. Ahlberg and B. Shneiderman. The alphaslider: a compact and rapid selector. *Proceedings of CHI'94*, 365–371. ACM, 1994.
2. C. Appert, O. Chapuis, and E. Pietriga. High-precision magnification lenses. *Proceedings of CHI'10*, 273–282. ACM, 2010.
3. Y. Ayatsuka, J. Rekimoto, and S. Matsuoka. Popup vernier: a tool for sub-pixel-pitch dragging with smooth mode transition. *Proceedings of UIST'98*, 39–48. ACM, 1998.
4. F. Bérard, G. Wang, and J. R. Cooperstock. On the limits of the human motor control precision: the search for a device's human resolution. *Proceedings of INTERACT'11*, 107–122. Springer-Verlag, 2011.
5. G. Casiez and N. Roussel. No more bricolage! Methods and tools to characterize, replicate and compare pointing transfer functions. *Proceedings of UIST'11*, 603–614. ACM, Oct. 2011.
6. D. R. Flatla, C. Gutwin, L. E. Nacke, S. Bateman, and R. L. Mandryk. Calibration games: making calibration tasks enjoyable by adding motivating game elements. *Proceedings of UIST'11*, 403–412. ACM, 2011.
7. Y. Guiard, M. Beaudouin-Lafon, and D. Mottet. Navigation as multiscale pointing: extending fitts' model to very high precision tasks. *Proceedings of CHI'99*, 450–457. ACM, 1999.
8. K. Perlin and D. Fox. Pad: an alternative approach to the computer interface. *Proceedings of SIGGRAPH'93*, 57–64. ACM, 1993.
9. E. Pietriga and C. Appert. Sigma lenses: focus-context transitions combining space, time and translucence. *Proceedings of CHI'08*, 1343–1352. ACM, 2008.
10. G. Ramos and R. Balakrishnan. Zliding: fluid zooming and sliding for high precision parameter manipulation. *Proceedings of UIST'05*, 143–152. ACM, 2005.
11. V. Scheib. Pointer lock. W3C Editor's draft, Web Applications Working Group, Apr. 2012.
12. B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. *Proceedings of VL'96*, 336–343. IEEE Computer Society, 1996.
13. D. Vogel and P. Baudisch. Shift: a technique for operating pen-based interfaces using touch. *Proceedings of CHI'07*, 657–666. ACM, 2007.

Hand Occlusion with Tablet-sized Direct Pen Input

Daniel Vogel[1,2], Matthew Cudmore[2], Géry Casiez[3], Ravin Balakrishnan[1], and Liam Keliher[2]

[1] Dept. of Computer Science
University of Toronto, CANADA
{dvogel|ravin}@dgp.toronto.edu

[2] Dept. of Math & Computer Science
Mount Allison University, CANADA
{dvogel|mcudmore|lkeliher}@mta.ca

[3] LIFL & INRIA Lille
University of Lille, FRANCE
gery.casiez@lifl.fr

ABSTRACT

We present results from an experiment examining the area occluded by the hand when using a tablet-sized direct pen input device. Our results show that the pen, hand, and forearm can occlude up to 47% of a 12 inch display. The shape of the occluded area varies between participants due to differences in pen grip rather than simply anatomical differences. For the most part, individuals adopt a consistent posture for long and short selection tasks. Overall, many occluded pixels are located higher relative to the pen than previously thought. From the experimental data, a five-parameter scalable circle and pivoting rectangle geometric model is presented which captures the general shape of the occluded area relative to the pen position. This model fits the experimental data much better than the simple bounding box model often used implicitly by designers. The space of fitted parameters also serves to quantify the shape of occlusion. Finally, an initial design for a predictive version of the model is discussed.

Author Keywords: Hand occlusion, pen input, Tablet PC.

ACM Classification: H5.2. Information interfaces and presentation: User Interfaces - Input devices and strategies.

INTRODUCTION

Given our familiarity with using pens and pencils, one would expect that operating a tablet computer by drawing directly on the display would be more natural and efficient. However, issues specific to direct pen input, such as the user's hand covering portions of the display during interaction – a phenomena we term *occlusion* (Figure 1a) – create new problems not experienced with conventional mouse input [12].

Compared to using pen on paper, occlusion with pen computing is more problematic. Unlike paper, the results of pen input, or system generated messages, may be revealed in occluded areas of the display. Researchers have suggested that occlusion impedes performance [7,10] and have used it as motivation for interaction techniques [1,14,24], but as of yet there has been no systematic study or model to quantify the amount or shape of occlusion.

CHI 2009, April 4–9, 2009, Boston, Massachusetts, USA.

Certainly, any designer can simply look down at their own hand while they operate a Tablet PC and take the perceived occlusion into account, but this type of ad hoc observation is unlikely to yield sound scientific findings or universal design guidelines. To study occlusion properly, we need to employ controlled experimental methods.

In this paper we describe an experimental study using a novel combination of video capture, augmented reality marker tracking, and image processing techniques to capture images of hand and arm occlusion from the point-of-view of a user. We call these images *occlusion silhouettes* (Figure 1b). Analyses of these silhouettes found that the hand and arm can occlude up to 47% of a 12 inch display and that the shape of the occluded area varies across participants according to their style of pen grip, rather than basic anatomical differences.

Based on our findings, we create a five parameter geometric model, comprised of a scalable circle and pivoting rectangle, to describe the general shape of the occluded area (Figure 1c). Using non-linear optimization algorithms, we fit this geometric model to the silhouette images captured in the experiment. We found that this geometric model matches the silhouettes with an F1 score [18] of 0.81 compared to 0.40 for the simple bounding box which designers often use implicitly to account for occlusion. The space of fitted parameters also serves as to quantify the shape of occlusion, capture different grip styles, and provide approximate empirical guidelines. Finally, we introduce an initial scheme for a predictive version of the geometric model which could enable new types of *occlusion-aware* interaction techniques.

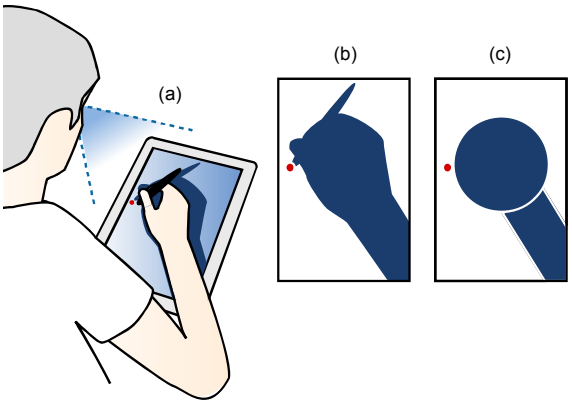


Figure 1: (a) Occlusion caused by the hand with direct pen input; (b) an *occlusion silhouette image* taken from the point-of-view of a user and rectified; (c) a simplified circle and rectangle geometric model capturing the general shape of the occluded area.

RELATED WORK

Few researchers have investigated occlusion directly, but many have speculated on its effect or considered it in the design of interaction techniques. Those who investigate it have done so without a strict control for occlusion. In practice it is very difficult to actually control without resorting to a different input paradigm such as indirect pointing.

Hancock and Booth [7] found that right-handed users selected targets more slowly when located East of the current pen position and attribute this effect to hand occlusion. However, the second slowest time occurred when targets were located in a North-East direction, so it is difficult to conclude. They recommend that pop-up menus should be placed South-West of the current pen location to minimize occlusion for right-handed users.

Based on experimental results, Forlines and Balakrishnan [4] argue that tactile pen feedback can make up for loss of visual feedback due to pen and hand occlusion. They also argue that occlusion is less problematic for serial compared to continuous input because the user can lift their hand to survey the display as part of the task.

Inkpen et al. [10] found a performance advantage and user preference for left-handed scrollbars with left-handed users. All participants cited occlusion problems when using the right-handed scrollbar and the authors note that some participants raised their grip on the pen or arched their hand over the screen to reduce occlusion.

There are several examples of pen interaction techniques that use occlusion as motivation. Ramos and Balakrishnan [14] designed a sinusoidal shaped slider that reduced occlusion from the user's hand. In Apitz and Guimbretières' [1] crossing based interface, they utilized a predominant right-to-left movement direction to counteract occlusion from right-handed users. Zeleznik and Miller [24] describe a tear-off menu technique to reduce occlusion problems.

In the related field of touch screen and tablet top interaction, occlusion is also cited as motivation. Shen et al. [16] discuss table top techniques to combat occlusion, including remote manipulation of objects and visual feedback that expands beyond the area typically occluded by a finger. Other strategies include: placing the hand behind [22] or under the display [23]; and shifting a copy of the intended selection area up and out of the area occluded by the finger [20].

Other researchers have cited problems with occlusion in unrelated experiments and usability studies. Grossman et al. [6] found that users sometimes moved away from the experimental target so they could invoke a hover widget without hand occlusion. Hinkley et al. [9] discovered that conventional GUI tooltips could be easily blocked by the hand. Hinckley et al. [8] found that users needed a chance to lift their hand to view the screen and verify progress when making a lasso selection. Dixon, Guimbretière, and Chen [2] located a start button below their main experimental stimulus to counteract hand occlusion. Ramos et al. [15] argue that accuracy is impaired when using a direct pen because of pen tip occlusion, but provide no evidence.

INITIAL OBSERVATIONAL STUDY

To investigate how occlusion affects usability in more detail, we conducted an initial observational study of Tablet PC interaction with realistic tasks and common software applications. This allowed us to discover a wider range of issues without knowing what they were beforehand. We considered using a performance-oriented experiment design, but controlling for occlusion a-priori is difficult without deeper knowledge about the shape and location of occlusion.

Twelve right-handed participants completed the study. A moderator guided then through an imagined scenario where an office worker must complete a presentation using a Tablet PC while away from their desk. They used typical office applications like a web browser, spreadsheet, and presentation tool. Text input was not required to complete the scenario. During the study, we asked participants to *think-aloud* as we recorded their actions with video and logged pen movements. The scenario took about 50 minutes to complete.

We found that occlusion likely contributed to user errors, led to fatigue, and forced inefficient movements:

- *Hidden Status Messages*. Several participants missed system status messages shown near the bottom of the display. In one case, a participant assumed a "file being saved" confirmation message had been shown beneath their arm, but in fact they missed selecting the save button.
- *Missed Previews*. The presentation application featured real time document previews when browsing text formatting menus. Unfortunately, many participants did not notice this feature, and some assumed their formatting choices had been successful, when they had mistakenly unselected the text behind their arm.
- *Inefficient Movements.* When dragging to highlight text or drawing a selection marquee, we observed large movement deviations past or away from the intended target when moving in a predominately left-to-right direction.
- *Occlusion Contortion*. Like Inkpen et al. [10], we observed participants occasionally arching their wrist while selecting formatting options to simultaneously preview document changes that would otherwise be occluded.
- *Rest Position*. We found that participants had a neutral rest position for their hand located at the right side of the display (all participants were right-handed). Participants commented that this enabled them to survey the display before a task, without their hand getting in the way.

Our results reinforce and expand those of previous research. Yet, we still do not have a thorough understanding of the fundamental characteristics of hand and arm occlusion. For this reason, we continued by examining the shape and area occluded by the hand in a methodical manner. These results could be used by designers to more effectively counteract the effect of occlusion with refined layouts or enable new types of occlusion-aware interaction techniques that compensate for occlusion in real time.

FORMAL EXPERIMENT

Our goal is to measure the size and shape of the occluded area of a tablet-sized display. To accomplish this, we record the participant's view of their hand with a head-mounted video camera as they select targets at different locations on the display. We then extract key frames from the video and isolate *occlusion silhouettes* of the participant's hand as they appear from their vantage point.

Participants

22 people (8 female, 14 male) with a mean age of 26.1 (SD 8.3) participated. All participants were right-handed and pre-screened for color blindness. Participants had little or no experience with direct pen input, but this is acceptable since we are observing a lower level physical behaviour.

At the beginning of each session, we measured the participant's hand and forearm since anatomical dimensions likely influence the amount of occlusion (Figure 2). We considered controlling for these dimensions, but recruiting participants to conform to anatomical sizes proved to be difficult, and the ranges for each control dimension were difficult to define.

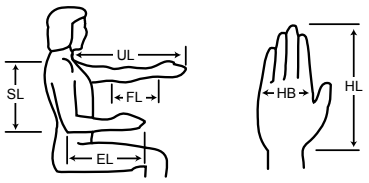


Figure 2. Anthropomorphic measurements (diagram adapted from Pheasant and Hastlegrave [13]).

- EL - elbow to fingertip length
- SL - shoulder to elbow length
- UL - upper limb length including hand
- FL - upper limb length, elbow to crease of wrist, EL - HL
- HL - hand length, crease of the wrist to the tip of finger
- HB - hand breadth, maximum width of palm

Apparatus

The experiment was conducted using a Wacom Cintiq 12UX direct input pen tablet. It has a 307 mm (12.1 inch) diagonal display, a resolution of 1280 by 800 pixels (261 by 163 mm), and a pixel density of 4.9 px/mm (125 DPI). We chose the Cintiq because it provides pen tilt information which is unavailable on current Tablet PCs.

We positioned the tablet in portrait-orientation and supported it such that it was at an angle of 12 degrees off the desk, oriented towards the participant. Participants were seated in an adjustable office chair with the height adjusted so that the elbow formed a 90 degree angle when the forearm was on the desk. This body posture is the most ergonomically sound according to Pheasant and Hastlegrave [13].

To capture the participant's point-of-view, we use a small head-mounted video camera to record the entire experiment at 640 × 480 px resolution and 15 frames-per-second (Figure 3a). The camera is attached to a head harness using hook-and-loop strips making it easy to move up or down so that it can be positioned as close as possible to the center of the eyes, without interfering with the participants' line of sight. In pilot experiments, we found that we could position the camera approximately 40 mm above and forward of the line of sight, and the resulting image was very similar to what the participant saw.

Printed fiducial markers were attached around the bezel of the tablet to enable us to transform the point-of-view frames to a standard, registered image perspective for analysis. Details of the image analysis steps are in the next section.

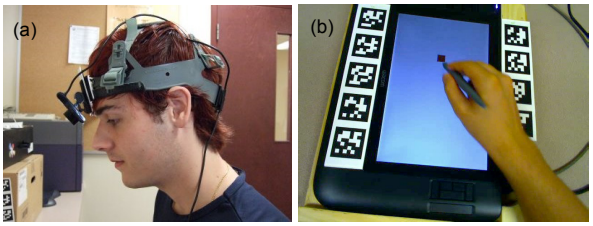


Figure 3. Experiment apparatus: (a) head mounted camera to capture point-of-view; (b) fiducial markers attached to tablet bezel (image is taken from head mounted camera video frame).

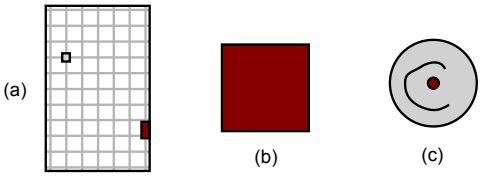


Figure 4. (a) 7 x 11 grid for placement; (b) square; (c) circle target (targets are printed actual size).

Task and Stimuli

Participants were presented with individual trials consisting of an initial selection of a *home target*, followed by selection of a *measurement target*.

The 128 px tall and 64 px wide home target was consistently located at the extreme right edge of the tablet display, 52 mm from the display bottom. This controlled the initial position of the hand and forearm at the beginning of each trial. We observed participants instinctively returning to a similar rest position in our initial observational study.

The location of the measurement target was varied across trials at positions inscribed by a 7 × 11 unit invisible grid (Figure 4a). This created 77 different locations with target centers spaced 122 px horizontally and 123 px vertically.

We observed two primary styles of pen manipulation in our initial observational study: long, localized interactions where the participant rested their palm on the display (such as adjusting a slider), and short, singular interactions performed without resting the hand (such as pushing a button). Based on this, our task had two types of target selection: *tap* – selection of a 64 px square target with a single tap (Figure 4b); and *circle* – selection of a circular target by circling within a 28 px tolerance between a 4 px inner and 32 px outer radius (Figure 4c). The circle selection is designed to encourage participants to rest their palm, while the tap selection can be quickly performed with the palm in the air. The different shapes for the two selection tasks were intended to serve as a mnemonic to the user as to what action was required.

The circle selection used an ink trail visualization to indicate progress. Errors occurred when the pen tip moved beyond the inner or outer diameter. We wanted this to be difficult enough to require a palm plant, but not tedious. In practice, participants took at least half-a-second to circle the target, which seemed to be enough to plant the palm.

At the beginning of each trial, a red home target and a gray measurement target were displayed. After successfully selecting the home target, the measurement target turned red and the participant selected it to complete the trial. We logged all aspects of pen input, including pressure and tilt.

Design

We presented 3 blocks of trials for each of the two tasks. A block consisted of 77 trials covering each target position in the grid, making 3 repetitions for each grid position and task type. Trials were presented in randomized order within a block and the presentation order of tasks was balanced across participants. Before beginning the first block of a task, the participant completed 40 practice trials. In summary:

2 *Tasks* (*Tap, Circle)* × 3 *Blocks* × 77 Target Positions
= 462 data points per participant

IMAGE PROCESSING

To transform the point-of-view video into a series of occlusion silhouettes, we performed the following steps with custom built software (Figure 5):

Frame Extraction. We extracted video frames taken between successful down and up pen events for the tap target, or just before the circular target was completely circled. To do this, we had to synchronize the video with the data log. We used a visual time marker which functions similar to a movie clapperboard. The time marker is a large red square containing a unique number. When this square is tapped, it disappears and a timestamp is saved to our data log. After the experiment, we scrubbed through the video and found the video time where the time marker disappeared. Then, using linear interpolation between bounding time marks, we located the corresponding video frame for a given log time. In most cases, the frame captured the pen at the intended target location, but occasional lags during video capture produced a frame with the pen separated from the target location.

Rectification. We used the ARToolkitPlus augmented reality library [21] to track the fiducial markers in each frame and determine the location of the four corners of the display. In practice, this sometimes required hand tuning when the markers were occluded by the hand or were out of frame due to head position. Using the four corner positions, we unwarped the perspective using the Java Advanced Image [17] functions `PerspectiveTransform` and `WarpPerspective` with bilinear interpolation, and cropped it to a final 267 × 427 px image. Note that due to our single camera setup, the unwarping will shift the image of the hand down slightly relative to the actual eye view. As an example, if the eye position is at the end of a vector 500 mm and 50° from the centre of the tablet, and the camera is located 40 mm above and forward of the eye, the unwarped image of a point on the hand 40 mm above the tablet will be shifted down by 6.2 mm (about 4 px in our unwarped image). The exact error will vary according to participant size and grip style, but the values above are typical. Rather than try to compensate for this slight shift and possibly introduce additional errors, we accepted this as a reasonable limitation of our technique.

Isolation. We used simple image processing techniques to isolate the silhouette of the hand. First, we applied a light blur filter to reduce noise. Then we extracted the blue color channel and applied a threshold to create an inverted binary image. We were able to use the blue channel to isolate the hand because the camera's color balance caused the display background to appear blue (it was actually white). Since the color space of skin is closer to red, this made isolating the hand relatively easy. To remove any edge pixels from the display bezel, we applied standard dilation and erosion morphological operations [3]. Finally, we filled holes based on the connectivity of pixels to produce the final silhouette.

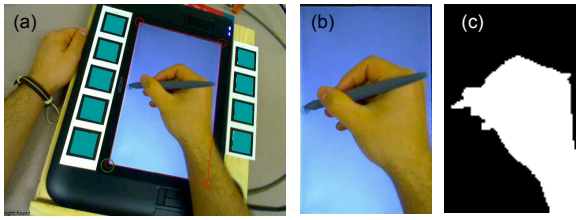


Figure 5. Image processing steps: (a) frame extraction; (b) rectification; (c) silhouette isolation.

RESULTS

Unfortunately, lighting and video problems corrupted large portions of data for participants 7, 14, 21, and 22 making isolation of their occlusion silhouettes unreliable. Capture problems with participant 8 corrupted the first block, but we kept this participant and their remaining blocks. In the end, our analysis included 18 out of the original 22 participants (6 female, 12 male) with a mean age of 26.3 (SD 8.4). In addition, we removed data trials when capture lag produced silhouettes more than 20 mm from the target location (7.8% of trials). These types of problems are typical when using video capture to generate empirical data: it is difficult to produce the same kind of "clean" data generated by experiments recording straightforward variables such as performance time and errors. Researchers attempting similar work should recruit extra participants and run multiple trials as we did, to ensure a reasonable amount of clean trials can be obtained.

Participants occasionally produced errors (mean 4.4%), but we included the silhouette regardless. Since each target must be successfully tapped or circled before continuing, the final video frame for an error trial would not differ. Also, the logged pen tilt values were very noisy, in spite of silhouette images suggesting tilt should be more uniform. Our attempts to filter them were unsuccessful, and we were forced to leave them out of our analysis.

Occlusion Ratio

We define the *occlusion ratio* as the percentage of occluded pixels within all possible display pixels. We used a ratio, rather than actual area, for unit independence. The actual area can be computed using the display area of 42,543 mm^2.

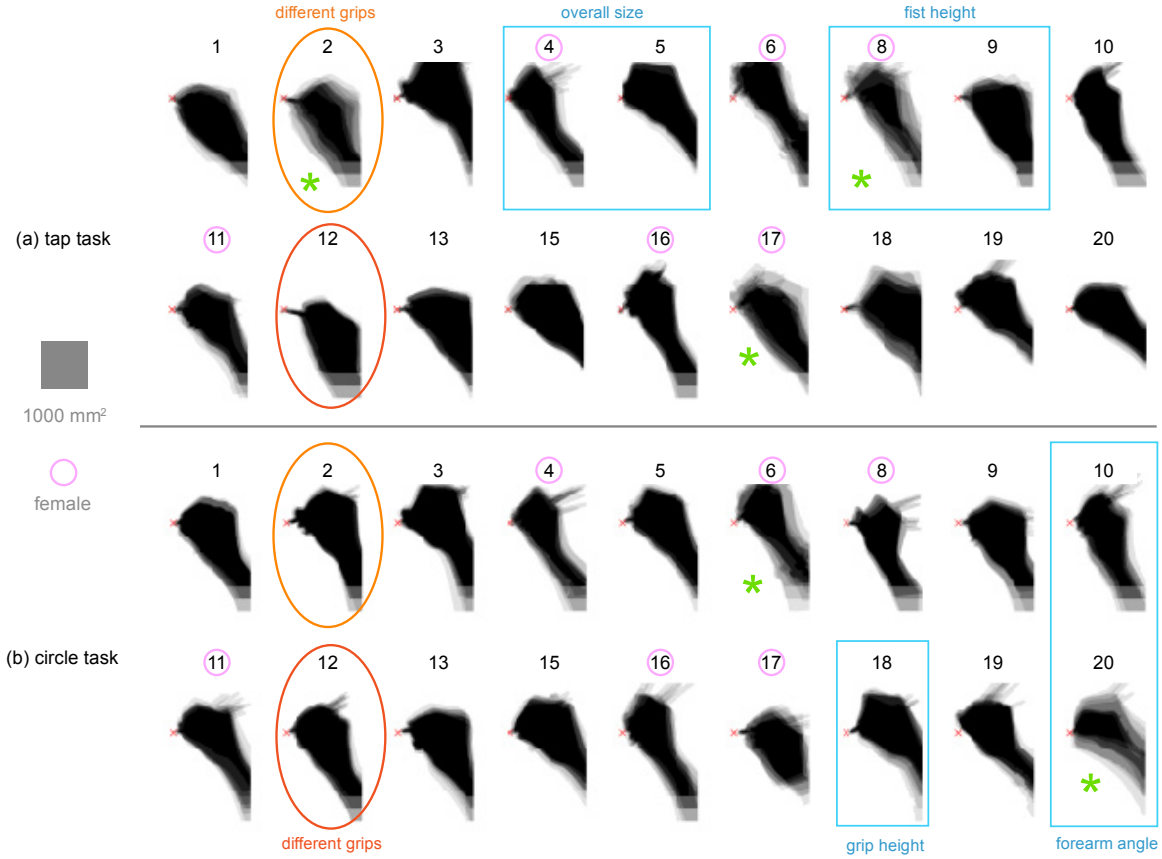


Figure 6. Occlusion shape silhouettes for each participant for: (a) tapping task and (b) circling task. 9 samples from 3 pen positions at middle-left portion of display; see text for discussion of participant and task comparison highlights.

Since occlusion ratio varies according to pen location, we calculate the occlusion area for each X-Y target location in the 7 x 11 grid. Not surprisingly, we found the highest occlusion ratios when the pen was near the top left of the display. However, the highest value did not occur at the extreme top, but rather a short distance below (Figure 7). The highest values did not differ greatly by task with 38.6% for circle (SD 6.2) and 38.8% for tap (SD 14.2). Participant 1 had the highest occlusion ratio with 47.4% for tap and 46.3% for circle.

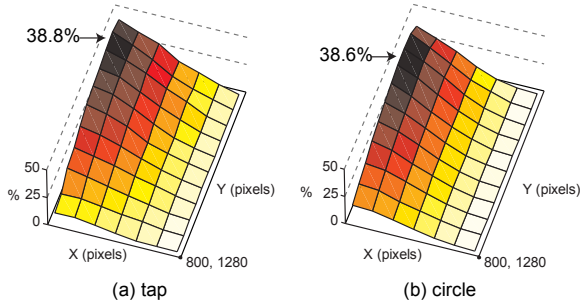


Figure 7. Occlusion ratio, plotted by X-Y display location for: (a) tap task; (b) circle task.

These mean ratios may reflect a sampling bias among our participants since controlling for aspects such as anatomical size and pen grip style is difficult to do a-priori. To help address this, we compare occlusion ratios given participant size.

Influence of Participant Size

We established a simple size metric *S* to capture the relative size of each participant's arm and hand compared to the general population. *S* is the *mean* of three ratios between a participant measurement and 50th percentile values from a table of anthropomorphic statistics[1]. We use measurements for shoulder length (SL), hand length (HL), and hand breadth (HB). Since tables of anthropomorphic statistics are divided by gender, we compute *S* for men and women using different 50th percentile values. We found mean *S* values of 0.99 (SD 0.04) and 1.01 (SD 0.06) for men and women respectively, indicating that the size of our participants was representative.

We expected to see a relationship between S and the maximum occlusion ratio since, larger hands and forearms should cover more of the display. However, a plot of S vs. maximum occlusion ratio does not suggest a relationship (Figure 8).

[1] Anthropomorphic statistics for U.S Adults 19 to 65 years old [13].

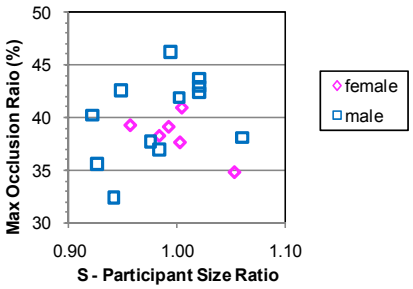


Figure 8. Participant size (S) vs. max occlusion ratio.

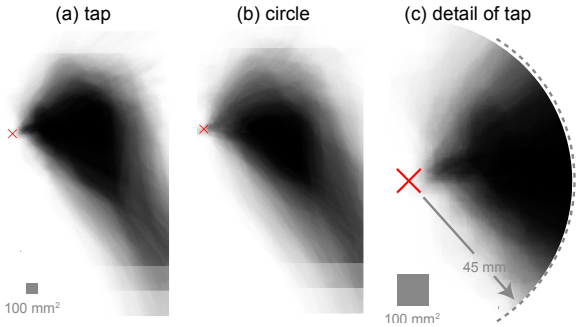


Figure 9. Mean occlusion shapes: (a) tap task; (b) circle task; (c) detail of tapping mean shape.

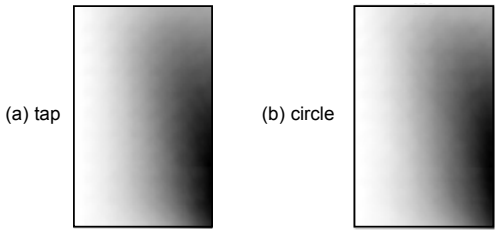


Figure 10. Pixels most likely to be occluded given a uniform distribution of pen positions: (a) tap task; (b) circle task (darker pixels are occluded more often).

Occlusion Shape

Although occlusion ratio gives some sense of the scope of occlusion, it is the shape of the occluded pixels relative to the pen position that is most useful to designers. Figure 6 illustrates the mean shapes for participants for circling and tapping tasks. Since the captured image of the forearm and hand are increasingly cropped as the pen moves right and downward, we illustrate shapes for positions sampled near the middle-left portion of display.

It is immediately apparent that occlusion shape varies between participants. There are differences which are likely due to anatomical size, possibly related to gender: compare how slender female participant 4 appears compared to male participant 5. Some participants adopt a lower hand position occluding fewer pixels above the target: contrast the height of participant 8 with participant 9. The forearm angle also often varies: for example, participant 20 has a much higher angle than participant 10. A few participants grip the pen far away from the tip, occluding fewer pixels around the target: participant 18 in the tapping task is one example.

When comparing individual participant shapes between the tap and circle tasks, the visual differences are more subtle and inconsistent. For example, we expected the higher speed of the tapping task to create a more varied posture resulting in blurry mean shapes. This seems to be the case for participants 2, 8, and 17, but there are contrary examples when circling shapes are more blurred: see participants 6 and 20. Only participants 2 and 12 seemed to adopt very different postures for tapping (low) and circling (high).

The combined participant mean shape gives an overall picture of occluded pixels near the pen position across all participants (Figure 9). As with individual participants, differences between tasks are subtle. The tapping task mean shape appears slightly larger, higher, and sharper compared to the circling task. In both cases, the typically occluded pixels form a circular blob centered far to the right of the pen tip with fewer pixels occluded in the immediate vicinity of the pen's position (Figure 9c).

Pixels Most Likely to be Occluded

Another way to view occlusion shape is to look at which display pixels are most likely to be occluded given a distribution of pen positions. To create a simple baseline for analysis, we assume that the probability of accessing any position on the display is uniform. Under this distribution, commonly occluded display pixels across participants and target positions form a cluster of frequently occluded pixels emanating from the lower two-thirds along the right edge (Figure 10). There appears to be no difference between circle and tap tasks.

A uniform distribution of pen positions is not representative of common application layouts: consider the frequency of accessing menus and toolbars located along the top of the display. With this in mind, the often occluded pixels near the bottom right may be even more likely to be occluded.

DISCUSSION

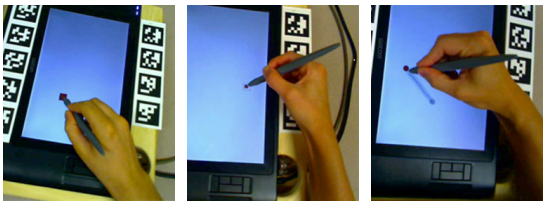
The results of our experiment reveal four main findings:

1. A large portion of the display can be occluded depending on pen position; with our participants it was typically as high as 38%, but could range up to 47%.
2. The pixels immediately below the pen position are not occluded by the hand as much as we expected, but more pixels are occluded above the pen tip horizon than previously thought. Given that our experimental set-up tended to shift the capture silhouette down slightly, this could be even more pronounced than what we observed.
3. Individuals seem to have a signature occlusion silhouette, but comparing silhouettes between different individuals can reveal large differences.
4. There appears to be no simple relationship between the size of the occluded area and anatomical size.

The Impact of Grip Style

The largest differences in occlusion silhouettes are due to the different styles of pen grips used by our participants (Figure 11). We searched the motor behaviour and graphonomics literature for a definitive classification of pen grip. Greer and Lockman [5] observed three different styles of pen grips used by adults, but do not describe these in detail. With our par-

ticipants, we found that grip style varied predominately across three dimensions: size of fist, angle of pen, and height of grip location on pen. We believe it is these characteristics of grip style that interact with anatomical measurements and ultimately govern occlusion area.



(a) (b) (c)

Figure 11. Grip styles: (a) loose fist, low angle, medium grip height; (b) tight fist, high angle, high grip height; (c) loose fist, straight angle, low grip height.

Left-handed Users

We conducted a small follow-up study with two left-handed users. Similar to Hancock and Booth's finding with performance [7], we found that the left-handed data mirrored the right-handed individuals.

Influence of Clothing

We gathered our data for sleeveless participants to maintain a consistent baseline, but we recognize that size of the occlusion silhouette could be much larger when clothed (consider using a tablet while wearing a loose fitting sweater or jacket). As a general rule, Pheasant and Hastlegrave [13] suggest adding 25mm to all anatomical dimensions for men and 45mm for women to account for thickness of clothing.

GEOMETRIC MODEL OF OCCLUSION SHAPE

The experiment revealed that the occlusion shape was somewhat uniform within a participant and high level similarities appeared across participants. We wondered if a simple geometric model could describe the general shape and position of the occlusion silhouettes. If so, by fitting this model to the actual silhouettes, the resulting model parameters could serve as empirical guidelines for designers. Moreover, this geometric representation could form the basis for a predicative version of model: in real time, a system would be aware of occluded portions of the interface without the aid of elaborate sensors. For example, imagine an interface that knows when a status message is occluded, and re-displays it as a bubble in a nearby non-occluded area instead.

There are many ways to approach modeling the shape of the occlusion silhouettes. Perhaps the most straightforward approach is to assume pixels below and to the right of the pen's position are occluded, an approach which some designers and researchers seem to use implicitly. We refer to this as a *bounding rectangle model* (Figure 12c). This model is constant relative to the pen's position and requires no other input, but the accuracy is poor. At the other end of the spectrum, we could create a model with a flexible shape such as one composed of Bézier spline segments (Figure 12a). While this would certainly yield a very accurate representation of the occluded area, the huge number of parameters would make fitting and interpreting the model difficult and hence impractical for creating empirical guidelines. Our aim then is to create a simple model with a small number of parameters**,** yet still produce a reasonable degree of accuracy.

Scalable Circle and Pivoting Rectangle Model

We noticed that the occlusion silhouettes produced by the experimental data often resembled a lopsided circle for the fist, a thick narrowing rectangle sticking out the bottom for the arm, and, with some participants, there was also a thinner rectangle puncturing the top of the ball for the pen. This meant that a single *oriented* bounding box would be unlikely to capture all grip styles accurately. Our first approach then, was to create a geometric model using an ellipse for the fist, an isosceles trapezoid for the arm, and a rectangle for the pen. However, even this model had 11 parameters and automatically fitting the geometry to our experimental data was problematic. Instead, we simplified our representation further to an offset circle and a rectangle with only the following 5 parameters (also illustrated in Figure 12b):

- q is the offset from the pen position **p** to the circle edge,
- r is the radius of the circle over the fist area,
- Φ is the rotation angle of the circle around **p** (expressed in degrees where $\Phi = 0°$ when the centre is due East, $\Phi = -45°$ for North-East, and $\Phi = 45°$ for South-East),
- Θ is the angle of rotation of the rectangle around the centre of the circle (using the same angle configuration as Φ),
- w is the width of the rectangle representing the forearm.

Note that the length of the rectangle is infinite for our purposes. If we were building a model for larger displays, this may become another parameter, but at present we are concerned with tablet-sized displays like the portable Tablet PC.

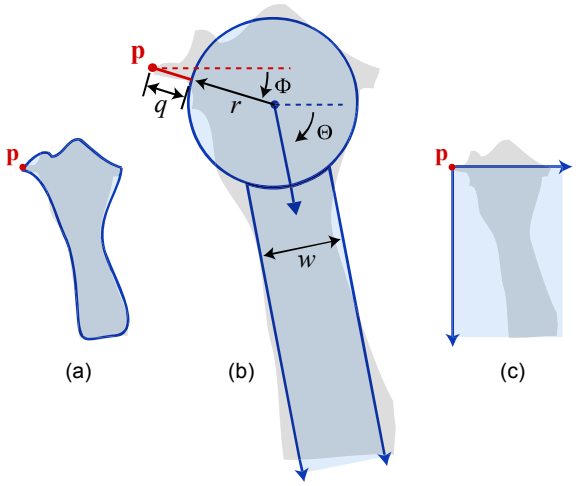


Figure 12. Three occlusion shape models: (a) Bézier spline; (b) circle and rectangle; (b) bounding rectangle. **p** is the position of the pen.

Fitting the Geometric Model to Captured Silhouettes

For each silhouette image from our experiment, we use nonlinear optimization techniques to set the five parameters of the geometric model so that it "fits" over the silhouette as accurately as possible. Note that other optimization algo-

rithms, or other fitting techniques, can be used – we describe our process as an example of one possible procedure.

To guide the optimizers to an *optimal fit*, we create an objective function. Our objective function returns 0 for a perfect fit, when the geometry matches the silhouette exactly, and increases as the alignment diverges. This is computed using two area ratios: the percentage of the silhouette image not covered by the geometry, and the percentage of geometry area not covering the silhouette. We give slightly more weight to the first ratio to favour covering more occluded pixels at the potential cost of covering non-occluded pixels as well. To compute these area ratios, we converted the silhouette binary images to polygons and computed the ratios analytically. The inverse would have worked as well, converting the geometric model to a binary image and "counting pixels" to calculate the ratios. To reduce the chance of the optimizer finding anatomically improbable configurations, we constrained the possible angles for Θ and Φ to be in (0, 90) and (-90,90) respectively. We also added smaller objective terms to encourage a smaller rectangle width w and shorter distance from circle to pen position q.

One problem during our initial optimization attempts was caused by cropped occlusion silhouette images. As the pen moves towards the bottom right, more and more of the forearm and fist are outside the display area and were cropped during image processing, making it difficult for the optimizer to find an optimal placement of the geometry. We solved this by fitting the geometry in two stages for each participant and target type (circle and tap). In the first stage, we optimized all parameters using 3 pen positions near the upper left portion of the display, since the hand and forearm would not be cropped. Using these values, we found mean values for r and w. In stage two, we locked r and w to these mean values and optimized over the remaining parameters. We rationalize this two-stage strategy by arguing that the size of silhouettes produced by the fist and forearm is unlikely to vary greatly according to X- and Y-coordinate, but their position and angle may change. If we had silhouette images capturing the entire image of the fist and forearm including parts outside the display, we would not have needed this step.

We ran the optimization using two algorithms in sequence over all target locations except the rightmost where the hand was completely off the display. First, a pattern search algorithm found a candidate neighbourhood for the global minima, and then a standard gradient search found a local minima (see [19] for algorithm descriptions). We could not use gradient search alone since our objective function produced a rough error surface. The total time for optimization was approximately 12 hours on a 2.66 GHz quad processor.

Testing the Accuracy of the Geometric Model

We use precision-recall plots and F1 scores, standard measures used in information retrieval [18], to test our geometric model's fidelity. This can be justified by considering the geometric model as a binary classifier which labels each pixel as occluded or not occluded. High precision means that pixels labelled by the model as occluded are actually occluded, but other occluded pixels may have been missed. High recall means that the model is correctly labelling occluded pixels, but could also be labelling non-occluded pixels as occluded.

By plotting the results of each fitted silhouette in precision-recall space, we get a sense for how well the model performs (Figure 13). A near-perfect model will have a concentration of points in the upper right corner and an F1 score close to 1. We calculate mean F1 scores across all cases.

Our geometric model has a mean F1 score of 0.81 (SD 0.20) and the precision-recall plots suggests very high recall, but some falloff for precision (Figure 13b). This precision falloff is expected since we designed our optimization objective function to fit the model in a more conservative manner, favouring covering more occluded pixels at the potential cost of covering non-occluded pixels. A designer would probably be more comfortable over compensating for occlusion, but this is a limitation. We included the bounding box model as a baseline comparison. It has a F1 score of 0.40 (SD 0.20) and a precision-recall plot (Figure 13a) suggesting a poor fit in terms of both precision and recall.

Note that our geometric model is only one of many potential models. For example, although we ruled out the *oriented* bounding box initially, it may be satisfactory in some situations. Evaluating our model, or any others, in real applications remains future work.

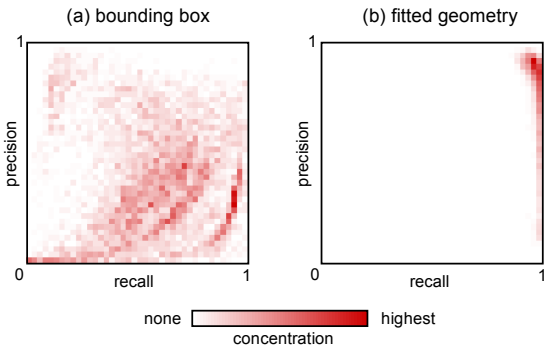


Figure 13. Precision-recall concentration plots illustrating performance of: (a) simple bounding box model; (b) fitted circle and rectangle geometry. More points in the upper right indicate better performance.

Space of Fitted Parameters

We can use the space of optimized parameters to further investigate the shape of the occlusion silhouettes. To enable comparison with Figure 6, in Table 1 we summarize the participant mean parameters for the circle task across the same 9 pen positions at the middle-left portion of the display. This focuses our comparison on positions in which the entire hand and forearm silhouette is captured without cropping and reduces variance from parameters, such as Θ (the forearm angle), as it changes across pen coordinate positions.

For the most part, the fitted parameters match visual intuition from the mean silhouette images in Figure 6. For example, a low value of Φ indicates a high grip and a high value of Φ indicates a low grip: the two lowest Φ values of -25.5 and -21.4 for participants 6 and 16 match the high grips seen in Figure 6, the high Φ values of 12.1 and 11.9 for participants

9, and 17 match their low grips. Likewise, q captures how close participants hold the pen relative to the pen tip: high q values of 28.3 and 26.1 for participant 2 and 17 indicate they hold the pen far from the tip, and low q values of 5.3 and 4.1 for participants 16 and 19 indicate the opposite. A comparison of other mean parameters with the silhouettes in Figure 6 reveals similar patterns.

We expected more variance in parameter values between participants than within a participant. For the most part, this was the case, but there are exceptions. Participants 6 and 20 have high variance, but we expected this from their blurry mean silhouettes in Figure 6. The high variance for participant 17 is somewhat surprising; we speculate that this may be due to image cropping caused by the grip style.

	q	*r*	Φ	Θ	*w*
1	12.3 (2.2)	61.5 (1.4)	10.1 (3.7)	58.0 (2.3)	58.9 (1.8)
2	28.3 (3.8)	64.0 (6.6)	-4.9 (3.8)	63.5 (3.6)	62.7 (2.3)
3	14.9 (2.5)	64.5 (1.1)	-13.9 (2.6)	57.7 (3.7)	72.8 (3.3)
4	7.1 (4.7)	50.3 (0.8)	-7.4 (5.1)	60.1 (3.3)	49.0 (2.2)
5	17.6 (4.3)	59.9 (0.8)	-7.9 (3.9)	53.8 (2.1)	61.8 (2.2)
6	15.3 (4.8)	58.4 (13.3)	-25.5 (8.1)	60.5 (5.0)	58.9 (5.5)
8	14.1 (5.5)	53.8 (1.4)	8.6 (6.3)	68.6 (4.0)	50.1 (1.7)
9	21.5 (1.9)	63.2 (1.1)	12.1 (4.5)	62.0 (3.6)	59.5 (1.1)
10	9.5 (3.4)	55.3 (1.7)	-1.6 (3.9)	69.2 (3.2)	54.9 (4.0)
11	15.5 (2.7)	56.8 (1.1)	-7.0 (5.9)	53.8 (5.4)	56.1 (3.3)
12	14.9 (3.5)	59.5 (0.8)	1.9 (3.5)	61.5 (2.6)	57.0 (3.5)
13	23.9 (3.8)	65.4 (1.2)	7.6 (4.1)	56.6 (3.4)	61.0 (5.7)
15	13.0 (3.8)	64.6 (1.6)	-9.1 (2.7)	45.8 (5.0)	63.9 (3.9)
16	5.3 (3.0)	52.6 (1.8)	-21.4 (6.9)	61.0 (3.2)	50.7 (2.4)
17	26.1 (7.9)	63.4 (6.8)	11.9 (7.7)	39.9 (23.3)	60.5 (19.1)
18	23.5 (2.9)	60.4 (0.9)	-16.4 (3.9)	46.9 (6.5)	73.1 (1.8)
19	4.1 (3.5)	58.6 (1.0)	-11.0 (2.8)	48.1 (3.8)	48.7 (2.9)
20	11.2 (5.4)	56.5 (9.5)	-11.9 (12.4)	39.0 (8.3)	58.6 (1.4)
all	**15.5** (7.9)	**59.5** (6.2)	**-5.1** (12.3)	**55.6** (10.8)	**59.0** (8.5)

Table 1. Summary statistics of fitted geometric model parameters for each participant for circle task (9 samples from 3 pen positions at middle-left portion of display, standard deviations shown in parenthesis).

DESIGN IMPLICATIONS

Our findings suggest three main design implications.

1. Avoid showing visual feedback or related widgets in the area described by the circular area to the right of the pen (see Figure 9c).
2. Avoid displaying status or alert messages in the bottom right area of the display since it may be often occluded by the hand (Figure 10).
3. When designing for occlusion, be aware that real users have a wide range of pen grips and postures.

We can use our fitted geometry model parameters to make implications 1 and 2 more specific. Assuming our experimental sample is representative, the mean parameter values across all participants (bottom row of Table 1) could form a universal mean configuration for the geometric model (Figure 14). In practice, this may not be the most accurate solution given implication 3 and because these mean values include only a subset of pen positions to avoid introducing higher variance, but it may suffice as a rough guide for designers.

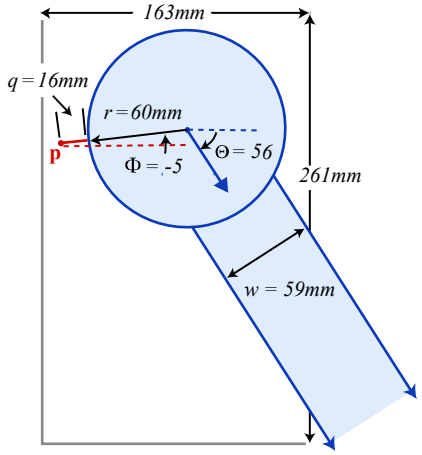


Figure 14. Mean configuration for the geometric model (with our display size context as context).

Towards A Predictive Model

To address the limitation with the geometric model described above, we developed an initial scheme for a predictive geometric model which would adjust according to pen coordinate position and user grip style. This would provide designers with more accurate models to fine tune layouts according to a custom set of users and display positions, in addition to enabling new types of occlusion-aware interfaces introduced above. This model is a work in progress, and we briefly describe it here with initial test results.

We simplified the problem significantly by first assuming constant values could be found for Φ, *q*, *r*, and *w* for each user. We imagine a short calibration process where a user fine tunes the size and position of the rendered geometric model shapes to match their grip style, indirectly setting these values. Note that the usability of the model may be limited if the calibration process is too long or cumbersome. Also, if reliable pen tilt information is available, we believe that Φ could be determined directly.

In early model prototypes, we found that an incorrect Θ could drastically alter the position of the forearm and throw the model off. To correctly model Θ, we use a kinematic model [11] to estimate the posture of the forearm. This requires several simplifying assumptions to make the problem tractable, and required us to estimate the position of the torso and arm segment lengths using gender-specific anthropometric data. In spite of these approximations and simplifying assumptions, testing our model against the experimental data yielded an F1 score of 0.77 (SD 0.16). More work remains to be done to test this model in real world applications and compare its performance against static occlusion models such as the mean version of our geometric model or bounding box.

CONCLUSIONS AND FUTURE WORK

Previous researchers and designers have, for the most part, made implicit assumptions like the bounding box model to determine what areas of the display are likely occluded by the silhouette of the hand and forearm. To move beyond such assumptions, we have provided a systematic study of occlusion using experimental data gathered with a head mounted video camera, augmented reality marker tracking, and image processing techniques. Our results suggest that the shape of occlusion varies among participants due to anatomical size and grip style, and for the most part, individuals adopt a consistent posture for long and short interactions. Moreover, the general shape of occluded pixels is higher relative to the pen than previously thought. Despite varying occlusion shapes across users, we were able to develop a simple five parameter geometric model that captures the general shape of the occluded area and use the space of parameters to characterize and further quantify the shape of occlusion.

It is important to note that we have focused on occlusion resulting from a typical, neutral posture. Inkpen et al. [10] report that users will contort their hand to overcome occlusion problems. We are currently investigating this phenomenon.

Extending our results to very large vertical displays or very small hand-held devices has yet to be explored. In terms of our geometric model, we expect that most parameters relating to grip style are unlikely to change (q, Φ , r, and w), but the values of these parameters would differ as the size and orientation of the display diverges from the tablet-sized display used in our study. In the case of large displays, the variance of parameter values would likely increase substantially.

As future work, we plan to refine the predictive version of the model introduced here, and utilize it to create occlusion-aware interfaces. We also plan to contrast our findings with occlusion silhouettes resulting from touch input.

REFERENCES

1. Apitz, G. and Guimbretiére, F. CrossY: a crossing-based drawing application. *Proceedings of UIST*, ACM (2004), 3-12.
2. Dixon, M., Guimbretière, F., and Chen, N. Maximizing Efficiency in Crossing-Based Dialog Boxes. *Proceedings of SIGCHI*, ACM (2008), 1623 - 1632.
3. Dougherty, E.R. *An Introduction to Morphological Image Processing*. Bellingham, Wash., USA, 1992.
4. Forlines, C. and Balakrishnan, R. Evaluating tactile feedback and direct vs. indirect stylus input in pointing and crossing selection tasks. *Proceedings of SIGCHI*, ACM (2008), 1563-1572.
5. Greer, T. and Lockman, J.J. Using writing instruments: invariances in young children and adults. *Child Development 69*, 4 (1998), 888(15).
6. Grossman, T., Hinckley, K., Baudisch, P., Agrawala, M., and Balakrishnan, R. Hover widgets: using the tracking state to extend the capabilities of pen-operated devices. *Proceedings of SIGCHI*, ACM (2006), 861-870.
7. Hancock, M.S. and Booth, K.S. Improving menu placement strategies for pen input. *Proceedings of Graphics Interface*, (2004), 221-230.
8. Hinckley, K., Guimbretiére, F., Agrawala, M., Apitz, G., and Chen, N. Phrasing techniques for multi-stroke selection gestures. *Proceedings of Graphics Interface*, (2006), 147-154.
9. Hinckley, K., Zhao, S., Sarin, R., et al. InkSeine:In Situ search for active note taking. *Proceedings of SIGCHI*, ACM (2007), 251-260.
10. Inkpen, K., Dearman, D., Argue, R., et al. Left-Handed Scrolling for Pen-Based Devices. *International Journal of Human-Computer Interaction 21*, 1 (2006), 91-108.
11. Isaacs, P.M. and Cohen, M.F. Controlling dynamic simulation with kinematic constraints. *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ACM (1987), 215-224.
12. Meyer, A. Pen computing: a technology overview and a vision. *SIGCHI Bull 27*, 3 (1995), 46-90.
13. Pheasant, S. and Hastlegrave, C. *Bodyspace: Anthropometry, Ergonomics and the Design of the Work*. CRC, 2006.
14. Ramos, G. and Balakrishnan, R. Fluid interaction techniques for the control and annotation of digital video. *Proceedings of UIST*, ACM (2003), 105-114.
15. Ramos, G., Cockburn, A., Balakrishnan, R., and Beaudouin-Lafon, M. Pointing lenses: facilitating stylus input through visual-and motor-space magnification. *Proceedings of SIGCHI*, ACM (2007), 757-766.
16. Shen, C., Ryall, K., Forlines, C., et al. Informing the Design of Direct-Touch Tabletops. *IEEE Comput. Graph. Appl. 26*, 5 (2006), 36-46.
17. Sun Microsystems. Java Advanced Imaging API. http://java.sun.com.
18. Van Rijsbergen, C.J. *Information Retrieval*. London ; Toronto, 1979.
19. Venkataraman, P. *Applied Optimization with MATLAB Programming*. New York, 2002.
20. Vogel, D. and Baudisch, P. Shift: a technique for operating pen-based interfaces using touch. *Proceedings of SIGCHI*, ACM (2007), 657-666.
21. Wagner, D. and Schmalstieg, D. ARToolKitPlus for Pose Tracking on Mobile Devices. *Proceedings of 12th Computer Vision Winter Workshop (CVWW'07)* , (2007).
22. Wigdor, D., Forlines, C., Baudisch, P., Barnwell, J., and Shen, C. Lucid touch: a see-through mobile device. *Proceedings of UIST*, ACM (2007), 269-278.
23. Wigdor, D., Leigh, D., Forlines, C., et al. Under the table interaction. *Proceedings of UIST*, ACM (2006), 259-268.
24. Zeleznik, R. and Miller, T. Fluid inking: augmenting the medium of free-form inking with gestures. *Proceedings of Graphics Interface*, (2006), 155-162.

Hand Occlusion on a Multi-Touch Tabletop

Daniel Vogel
Cheriton School of Computer Science
University of Waterloo, Canada
dvogel@uwaterloo.ca

Géry Casiez
LIFL & INRIA Lille
University of Lille, France
gery.casiez@lifl.fr

ABSTRACT
We examine the shape of hand and forearm occlusion on a multi-touch table for different touch contact types and tasks. Individuals have characteristic occlusion shapes, but with commonalities across tasks, postures, and handedness. Based on this, we create templates for designers to justify occlusion-related decisions and we propose geometric models capturing the shape of occlusion. A model using diffused illumination captures performed well when augmented with a forearm rectangle, as did a modified circle and rectangle model with ellipse "fingers" suitable when only X-Y contact positions are available. Finally, we describe the corpus of detailed multi-touch input data we generated which is available to the community.

Author Keywords
occlusion; multi-touch; hand; tabletop; tablet; finger

ACM Classification
H.5.2 [Information Interfaces and Presentation]: User Interfaces - Input devices and strategies;

INTRODUCTION
Operating a computer by directly touching the display surface has many benefits, and in tabletop computing, multi-touch is arguably the most natural form of input. However, with any form of *direct* input, where the input device and the output display are coincident, the hand and arm cover — or *occlude* — part of the display. This can be a problem, because compared to manipulating objects on a real tabletop, a tabletop computer is dynamic and can display relevant information, sequential widgets, and system messages in occluded areas. Researchers are aware of occlusion: they suggest it impedes performance [9,21,22] and use it to motivate the design of interaction techniques [7,13,15,20,24]. Yet, there has not been a systematic study of hand occlusion with multi-touch tabletops.

Vogel et al. [25] developed a methodology to study direct pen occlusion by capturing the actual shape of occlusion from a person's point-of-view. We adapt their methods for video capture, augmented reality marker tracking, and image processing to a multi-touch tabletop (Figure 1). Compared to pen input, examining multi-touch occlusion is more challenging due to the wide vocabulary of touch contact types using different combinations of fingers and postures across different hands. We studied 18 conditions covering typical combinations of 9 different contact types with 3 tasks: tapping, dragging, and transforming. By examining the mean occlusion shapes, we find that individuals use consistent hand postures, and although there are differences between individuals, there is enough commonality to use overall mean shapes to inform interface design. For this purpose, we create calibrated occlusion template shapes to guide designers with interface layouts which reduce occlusion.

We conducted our experiment on a diffused illumination (DI) table top. The raw infrared (IR) image blob near the contact points should be a good estimate of the occluded area, and we propose a geometric model of occlusion combining the IR blob with a "forearm" rectangle. However, input technologies like capacitance only sense X-Y contact positions, so we also created an alternative model. We add ellipses for extended fingers to Vogel et al.'s [25] pen occlusion model without increasing the number of parameters. In a test fit of these models to occlusion silhouettes, the DI model achieves a F1 score of 0.80, while the multi-touch geometric model slightly outperforms it at 0.82. This suggests that real-time prediction of the occluded area, even with only X-Y contact positions, is possible — enabling occlusion-aware interaction techniques [23] on any multi-touch table regardless of hardware technology.

In the course of this project, we generated a large corpus of images synchronized with logged multi-touch data for common tasks. We feel that this is also a contribution, and make it available for related research.

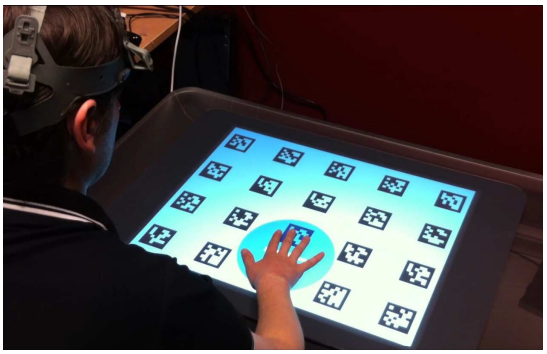


Figure 1. Experiment set-up: the occluded area is captured by a head-mounted camera and rectified using fiducial tracking.

CHI'12, May 5–10, 2012, Austin, Texas, USA.

RELATED WORK

Vogel and Balakrishnan [22] list hand occlusion as one of five direct pen input problems. They observed hidden status messages, missed previews, inefficient movements, and occlusion contortion. Pen input work by Hancock and Booth reached similar conclusions [9], but in practice it is difficult to strictly control for occlusion without resorting to different input paradigms like indirect pointing. With touch screens, finger occlusion has long been known to be problematic [21]. Recent work has argued that finger occlusion is not directly responsible for errors [12], but it is undeniable that one cannot see what is beneath their finger. With larger multi-touch tables and tablets, this is compounded as multiple fingers, hands, and forearms cover more of the display. So it makes sense that reducing occlusion is an important aspect of direct input interface design. Vogel et al. [25] provide a summary of occlusion and direct pen input; we focus on direct touch input here.

Effect of Occlusion

Several touch techniques address occlusion directly. For example, expanding feedback beyond occluding fingers [20], shifting a copy of the hidden area out from under the finger [24], and creating methods to manipulate objects remotely to avoid occluding them [27]. More radical solutions like moving touch to the back [28] or underside [29] also work, but they reduce the directness of touch.

Other work uses occlusion as a motivation for the spatial layout of interface designs. For example, FingerGlass [13] and ShadowGuides [7] use spatial offsets to reduce the chance of occlusion, but it is unclear how the exact offset was determined, or if it is optimal. Eden [15] describes multiple design decisions to reduce occlusion and argues for occlusion-awareness in multi-touch applications.

Understanding Occlusion

When making such occlusion-motivated design decisions, there is an implied reference to the shape of a "typically occluded" area. However, this is typically determined in an ad hoc manner. With direct pen input, empirically based occlusion-aware layout decisions have been used, such as Hancock and Booth's [9] context menu placement by radial selection time and Brandl et al.'s [4] paper-based experiment for pie menu orientation.

Although encouraging, in neither of these cases is the shape of occlusion analyzed directly as in Vogel et al. [25]. In a controlled experiment, they capture images of the occluded area using a head-mounted camera. These *occlusion silhouette* images are used to visualize mean shapes and develop a simple geometric model. This work led directly to Vogel & Balakrishnan's [23] design for an individually configurable, real-time occlusion model to realize occlusion-aware interfaces and interaction techniques.

Our methodology is based closely on this work, but we introduce new experiment tasks and refined geometric models tailored for multi-touch input. We also contribute other methodology refinements such as placing fiducial tracking markers in the display and introducing more meaningful descriptive statistics for occlusion shapes.

Understanding Multi-touch Postures

It is impossible to study multi-touch occlusion without considering the shape of the hand. Past work has looked at what postures people use for various types of multi-touch interactions in controlled experiments [14], elicitation studies [17,30], and in the field [11]. This has provided insights such as: people use different fingers for the same contact type [14]; people use any number of fingers for operations like dragging [17,30]; and people use different open- and closed-hand postures for single finger contacts [11]. In our experimental tasks and posture conditions, we balance these "almost anything goes" results with the reality of research and commercial systems which already map the type of contact [27] or number of finger contacts to specific tasks (e.g. selecting vs. scrolling vs. paging [2]).

Rather than looking at what postures are used, we examine the literal posture shape from a person's point-of-view. While our focus is different, our methodology and the corpus of data we generated can be applied to investigations of other posture characteristics like the studies above.

EXPERIMENT

The goal of our experiment is to study the shape of the occluded area for canonical multi-touch table postures and interaction tasks. We focus on a multi-touch table because smaller tablet form factors use fewer gestures due to their size and capacitive sensing makes some postures impractical. Limitations are discussed in our conclusion.

We adapt the methodology introduced by Vogel et al. [25]. As participants perform common multi-touch gestures, we record a video of their hands using a head-mounted camera. Then we extract key frames and isolate occlusion silhouettes to create a rectified view of the hands from their vantage point. We are not interested in performance time.

Participants

24 people (16 male, 8 female) with a mean age of 30 (SD 6.6) participated. 3 participants were left-handed (2 male, 1 female). 7 participants said they had experience with a multi-touch table and 21 said they had used a multi-touch phone or tablet. We recorded the height of all participants and found a mean of 176.3 cm (SD 9.4). This measurement is to suggest a reasonable sampling of person size, not to search for correlations between anatomical size and occlusion shape since Vogel et al. found this unlikely.

Apparatus

We conducted the experiment on a Microsoft Surface 1.0 multi-touch table. It has a 610 × 458 mm display with a display resolution of 1024 × 768 px (1.679 px per mm). The table-like case is 560 mm high with a 690 × 1080 mm top, approximating a small coffee table. Participants sat in a fixed chair centred along one of the long sides and we asked them to refrain from leaning from side-to-side. We did not observe anyone having difficulty reaching distant target locations. A fixed body position is a necessary

experimental control, but our relative results could be transformed to a tracked body location [1].

The small head-mounted video camera for the participant's point-of-view recorded the experiment at 960 × 720 px resolution and 15 FPS. It was positioned as closely as possible to the centre of the eyes, without interfering with the participants' line of sight. Since the camera is mounted above the centre of the eyes, it does not capture the *exact* point-of-view. Vogel et al. estimated that rectified occlusion silhouettes would be shifted down by 6 mm on average. Since we have a greater distance from the eye to the hand, our error would be reduced further.

To enable us to track and rectify the Surface display in the camera image, we displayed a 4 × 5 grid of 59 mm fiducial markers. We could not put the markers on the bezel due to the size of the display and the field-of-view of the camera. We were initially hesitant to show this pattern under our experimental stimulus, but found that participants effectively ignored it within the first few practice trials.

The experiment code is in C# using the Surface SDK. During task activity, the position, ellipse size, and ellipse orientation of all touch contacts were logged at more than 60 Hz along with 15 FPS of 768 × 576 px (1.26 px/mm) raw IR captures. In addition to the head-mounted camera, we recorded everything with a stationary camera above the Surface, but did not use this in the present analysis.

Gesture Set

Compared to Vogel et al.'s study where the pen is the only type of contact, we needed to select a representative set of multi-touch gestures: contact postures paired with interaction movements. We consulted multi-touch device SDKs and user guides (nicely summarized in [31]) and related research [3,5,7,11,16,17,30]. To keep the study reasonable and more ecologically valuable, we selected only gestures which use a single posture (e.g. no opening palms [3]) and simple movements along single paths (e.g. no L-shapes or X-shapes [17]). We focus on single-handed gestures since many two-handed gestures may be factored into two separate gestures for the purpose of shape analysis.

We identified three main types of interaction movements: tapping, dragging, and object transformation. A fourth choice would have been "flicks," but these resemble a short, high speed drag. To avoid redundancy between dragging and transforming, we restrict transformations to simultaneous rotation and scale only.

We identified eight common types of postures: using 1, 2, 3, 4, or 5 digits ('digit' includes 'fingers' and 'thumb'); a flat palm, the side of the fist, and the side of the hand. All 8 of these postures are paired with tapping, but only the first 6 with dragging since dragging with the fist or the side of the hand are less common. We paired transforming with 2 digit and 5 digit poses only since a 2 digit posture captures the common pinch gesture and the 5 digit posture is also common for transformations [11]. Like gestures, many two-handed postures can be factored into two one-handed postures (e.g. a non-dominant flat palm setting the mode for a 1 digit dominant hand drag [27]). We only include a two-handed transform gesture since both hands work together.

	1 Digit	2 Digits	3 Digits	4 Digits	5 Digits	Palm	Fist	Side	Two Handed
Tap	25	80	100	120	160	200	100	180	
Drag	25	80	100	120	160	200			
Transform		120			160				140

Table 1. *Postures* and *Tasks* with target diameters (in mm).

Task and Stimuli

The gesture movements define three tasks: *Tap*, *Drag*, and *Transform*. Each task has a main circular target with diameter set according to posture (see Table 1). The smallest diameter is three times the minimum recommended touch target size [12], the largest based on anthropomorphic palm size [18], and intermediate sizes selected to easily accommodate postures. This balances ease-of-selection with location constraints.

In the same spirit of Kin et al.'s experiment [14], the inherent ambiguity of a single circular target, together with the generic term 'digit,' allows participants to use different posture strategies. For example, a 2 digit posture could be a thumb and index finger, or an index and middle finger. Overly suggestive targets or terms like those for teaching specific gestures [3,7] would prevent natural posture strategies, leading to different shapes.

For most postures, our code prevents interaction unless the correct number of contacts are on the target. This worked well for 1 to 5 digits, but had to be relaxed for palm, fist, and side since the Surface detects an irregular number of contacts in these cases. In the spirit of allowing participants to adopt posture strategies, we do not control for the number of contacts with the two-handed condition. We wanted to see if people used two index fingers or some other combination of fingers across hands.

Tap Task. To complete the *Tap* task, the participant touches a circular target using the required posture for 333 ms. This short delay reduces motion blur and increases the tolerance for event log synchronization for the point-of-view frame captures, addressing problems reported by Vogel et al. [25] The centre of the current target is positioned at one of 9 locations spaced evenly on a 3 × 3 grid. Vogel et al. use a more granular 7 × 11 grid, but their findings do not suggest this is necessary. We cover a range of extreme positions by centering the grid in the display and spacing columns and rows at 192 mm and 112 mm respectively.

Drag Task. To complete the *Drag* task, the participant uses the required posture to drag a circular target from the centre of the display to one of 8 circular dock locations on the same 3 × 3 grid. The outer and inner diameters of the ring-shaped dock are set so that the error tolerance for the target

is 30 mm. Like the *Tap* task, the target must be held within the dock using the correct posture for 333 ms.

Transform Task. To complete the *Transform* task, the participant rotates and scales a circular target until a pin aligns with a 30 mm rotation tolerance "key" and the target border fits within a 15 mm outer ring tolerance. This position must also be held for 333 ms to complete the task. All *Transform* tasks are at the center grid position, but with 4 rotation and scale conditions: clock-wise (CW) and counter-clockwise (CCW) 60° rotation and scaling up or down by 45 mm. The initial target angle is -60° for CW and 0° for CCW to minimize key occlusion.

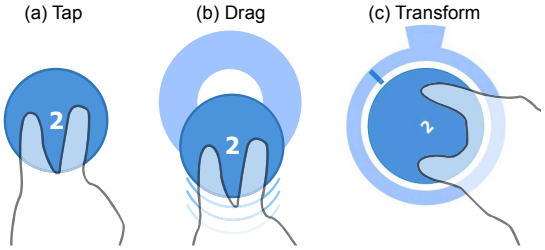


Figure 2. Experiment tasks (using 2 Digits as an example): (a) Tap on circular target; (b) Drag target into ring-shaped dock; (c) Transform target rotate pin to key and scale to fit in ring.

Design

The main experiment had 3 *Blocks*, with each block consisting of 3 *Task Sections*. Each *Task Section* contained all permutations of *Posture* and *Task Condition* for a *Task*, grouped by posture: the *Tap Task* had 8 posture groups, each at 9 grid position *Conditions*; the *Drag* task had 6 *Posture Groups*, each at 8 grid position *Conditions*; and the Transform task had 3 *Posture Groups*, each with 4 rotation and scale *Conditions*. The *Postures* were presented in approximate order of increasing difficulty (the column ordering in Table 1). Within each *Posture Group*, the order of *Conditions* were randomized. All blocks had the same *Task* ordering, but this order was counter-balanced across participants. In summary:

3 Blocks ×
8 *Postures* × 9 *Conditions* (*Tap* task section)
+ 6 *Postures* × 8 *Conditions* (*Drag* task section)
+ 3 *Postures* × 4 *Conditions* (*Transform* task section)
= 540 data points per participant

Before beginning the main experiment blocks, participants completed 26 practice trials: 1 centre *Tap* trial for each *Posture*; 1 *Drag* trial for each *Posture* to a random outer grid position; and all permutations of *Transform* trials. After the main experiment, participants also completed 72 trials with their non-dominant hand covering a subset of *Conditions* for all *Tasks* and *Postures*: 5 *Tap* trials for each *Posture* at all grid positions except corners; 4 *Drag* trials for each *Posture* to all grid positions except corners; and 8 *Transform* trials covering all *Conditions* except the two-handed posture. We elected to use this subset of conditions to reduce experiment time and fatigue, but provide enough data to show any pronounced differences between hands. The total experiment took less than 1 hour to complete.

IMAGE PROCESSING

To transform the point-of-view video into occlusion silhouettes, we use the same steps as Vogel et al.:

Frame Extraction. After synchronizing the video and the data log using visual time markers, we capture one video frame at the end of all tasks and one frame when the participant first touched the target in *Transform* and *Drag*.

Rectification. We wrote custom software using the ARToolkitPlus augmented reality library [26] to track the fiducial markers. After tracking the image-space marker positions, we use OpenCV to calculate the homography matrix and rectify the image of the hand against the display. The rectified display area is 610 × 458 px, so 1 px = 1 mm. Our software application enables us to manually track markers when automatic tracking fails (i.e., when less than 4 non-collinear markers are detected due to motion blur, cropping, or hand occlusion). About 5% of the frames had to be partially tracked manually.

Isolation. To isolate the binary occlusion silhouette images for analysis, we use similar image processing steps as Vogel et al. Since the fiducial markers are in the display space, we add a median background subtraction step to remove them. This works reasonably well, but we realized that colouring the fiducial markers saturated blue instead of black would have greatly simplified this task.

RESULTS

Compared to Vogel et al., we had no corrupted participants and very few corrupted images (less than 0.001%). This is largely due to consistent camera settings and room lighting.

Shape of Occluded Area

Since our goal is to study the shape of the occluded area, we begin with an examination of the overall mean occlusion silhouette shapes shown in Figure 8 (second last page). These are created by registering all silhouettes by target grid position (the actual centroid is not robust for palm, side, and fist postures), then finding average pixel values across all participants and conditions for each *Task* and *Posture*. Using the usual experimental assumption that our 24 participants provide a reasonable population sample, the darker areas are more likely to be occluded.

For *Tap* and *Drag*, the darker areas show most *Postures* clearly, suggesting homogeneity across participants and grid positions, but heterogeneity between *Postures*. Differences between *Fist* and *Side* are subtle, but consistent. Medium grey areas suggest different *Posture* strategies. For example, the ghost-like shape of other fingers for *1-Digit* and *2-Digits* suggest a mixture of open and closed hand postures. There is also surprising similarity between *Tap* and *Drag* tasks. For most postures, the differences are unperceivable; the largest differences are with the *Palm*.

For *Transform*, the darker areas are less defined. This is partly due to high positional variance in the different conditions, but also indicates a greater variety of *Posture* strategies. A general thumb and index finger pinch shape can be seen for *2-Digits*, perhaps because most participants reported experience with multi-touch devices where this strategy is standard. For *Two Hands*, the hands can be discerned, but it is unclear what digits are used. The posture for *5 Digits* is particularly heterogeneous. While *2-Digit Transform* has similarities to *2-Digit Tap* and *Drag*, *5-Digit* and *Two Hand* similarities to *Tap* and *Drag* are less so.

Individual Participants

We also examined all mean silhouettes for *Participants* by *Task* and *Posture.* Due to space, our discussion here focuses on *Tap* and *Transform* for 8 participants chosen for diversity, which are reproduced in Figure 9 (last page).

Individual differences in posture strategy, hand shape, and hand size are apparent, but there are common shape characteristics overall. This is most clear in *Tap* where the palm resembles a circle with one or more fingers extended, and the forearm typically projects down to the lower-right. This is more difficult to see with *Transform* where start and end positions are merged together.

Open- and closed-hand strategies are clearly seen for *1-Digit*, *2-Digits* and *3-Digits*. For example, participants 8, 3, 21, and 24 extend their touch fingers from a closed fist, whereas others extend all fingers regardless. Although we observed some variation in digit used [14], each participant generally used the same digit(s) for a given *Posture*. For Palm posture, most participants spread their fingers (participant 24 is an exception) and differences between the fist and side are subtle. With *Two Handed* postures, the non-dominant hand is often positioned lower (e.g. participants 22, 3) and individual hands are similar to *1-Digit Tap* silhouettes.

Across postures and tasks, forearm angle appears consistent. Wrist angle appears consistent in *Tap* (e.g. participant 23 has the most acute angle for *3-Digits)*. With *2-* and *5-Digit Transform*, there is more wrist variability.

Left-Hand and Non-Dominant Hand

We found no pronounced differences for left-handed participants (e.g. participants 22 and 24). We also found no pronounced differences between dominant and non-dominant hand, but the smaller set of data limits this result.

Descriptive Statistics

To help quantify these observations, we devised three statistics (Figure 3): the size of the occluded *area* near the target; the *breadth* of the fingers in the hand posture; and the *angle* of the forearm. Our motivation is to reinforce mean silhouette observations not quantitative tests.

Occluded Area

The mean areas for *Tap* and *Drag* across common *Postures* are very similar at 107 cm^2 (SD 17) and 105 cm^2 (SD 19), supporting our mean shape observations. The mean area for *Transform* is somewhat lower at 98 cm^2 (SD 23). Mean areas by *Posture* for *Tap* and *Drag* also support visual observations. The areas of *1-Digit Tap* and *Drag* are both 82 cm^2 (SD 15). The trend continues with *Palm's* large task discrepancy and area, 152 cm^2 (SD 21) for *Tap* and 141 cm^2 (SD 28) for *Drag*. With *Transform*, we observed visual differences for common *Postures* with *Tap* and *Drag*, but quantitatively the difference in area is within 10 cm^2 with measurements of 89 cm^2 (SD 17) for *2-Digits* and 106 cm^2 (SD 18) for *5-Digits*. The mean area for *Two Hands* is 98 cm^2 (SD 34) which is closer to *3-Digit Tap* and *Drag*.

Breadth of Posture

Mean breadth for *Tap* and *Drag* across common *Postures* is similar at 170° (SD 48) and 181° (SD 39), but with higher variance. Differences between the same *Posture* for *Tap* and *Drag* are less than 4°, except *Palm* at 59° which supports our observations. Overall, the spread monotonically increases from 119° to 225° for *1-Digit* to *5-Digits*. For *Transform*, *2 Digit* spread is greater than *2-Digit Tap* and *Drag* reflecting the variability in the pinch gesture.

Forearm Angle

Arm angle is remarkably consistent for common *Postures* between *Tap* and *Drag*. The overall means are 58° (SD 16) and 57° (SD 16) respectively, with individual variations less than 3° (centre grid positions removed since all *Drag* tasks begin at this position). We can compare *Transform* and *Tap* at the centre location only. Here the values diverge, with means of 45° (SD 21) and 58° (SD 9) respectively, likely due to the variability of the *Transform Task*. As the hand reaches target locations, the forearm angle changes according to kinematics. While arm angle varies according to contact location, these angles are remarkably consistent between *Drag* and *Tap* and across participants.

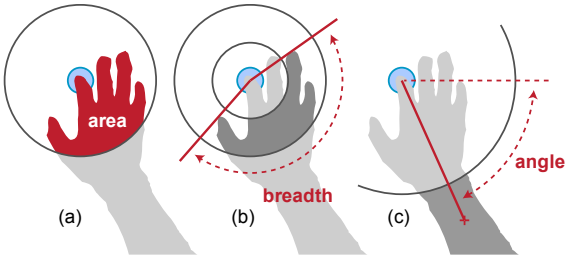


Figure 3. Descriptive statistics for hand occlusion: (a) occluded *area* in cm^2 within a 100 mm radius of the target center; (b) angular hand posture *breadth* within a 50 to 100 mm ring; (c) arm *angle* from horizontal, calculated from the target center to the centre of mass beyond a 150 mm circle. Segmentation radii were selected by trial and error: for *breadth* and *angle*, they crudely segment silhouettes into hand and forearm slices; for *area*, this gives a measure of localized occlusion (an admittedly arbitrary size, but more generalisable than Vogel et al.'s occlusion area measure as the ratio of a 12" display).

Discussion

Our results yield these main findings for occlusion shape:

- Postures have distinct shapes, but with common elements like a hand blob and protruding forearm.

- Tapping and dragging have very similar shapes, but transforming shapes are different.
- Different people may use different posture strategies, but each individual adopts a consistent strategy making posture shapes within an individual similar.
- Left- and right-handed people, and dominant and non-dominant hands, have no pronounced differences.
- Forearm angle is consistent between tapping and dragging, transform is more divergent.

These results are of theoretical interest to researchers and provide causal evidence for the potential impact of occlusion during interaction. But, how can it be used? Next, we show how this occlusion shape information is made accessible to tabletop interface designers, and describe a simple model to capture the essence of the occluded area setting the stage for multi-touch occlusion-aware interfaces.

DESIGN IMPLICATIONS AND RESOURCES

Without empirical guidelines, designers rely on rules-of-thumb or ad hoc observation to reduce occlusion in interface layouts. This is difficult to justify and may be sub-optimal as non-occluded or occluded locations are missed.

Occlusion-awareness Templates for Designers

Using the overall mean silhouettes, we created design-time "occlusion-awareness" templates for designers (Figure 4). Each template is a dimension-calibrated image showing areas which may be occluded relative to the expected contact centroid. Two bands of occlusion severity are illustrated, calculated from pixel density thresholds in the associated mean image: the possibly occluded area (>10%) and the often occluded area (>50%). These templates are available for download[1] as a layer-separated PDF and can be imported into common design applications like Adobe Photoshop or Illustrator. After scaling the template to match the real world units of an interface design, the designer can use it as an overlay to make occlusion-aware layout decisions. We created templates for all postures across *t* and *Transform* task (since drag is very similar to tap).

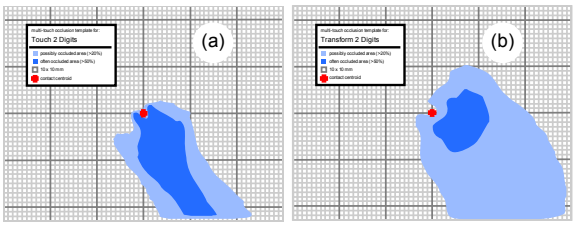


Figure 4. Occlusion-awareness template examples for: (a) 2 Digit Tap; (b) 2 Digit Transform. Light blue is the possibly occluded area (>10%), dark blue is the often occluded area (>50%). The red cross is the expected contact centroid.

All templates are generated as single right-handed shapes, but can be easily flipped for left-handed designs. Although people prefer using their dominant hand for single-handed multi-touch interactions [14], this is not guaranteed. We did not see pronounced differences between dominant and non-dominant silhouettes in the subset of tested conditions, so designers should be able to mirror and union the right-handed template when designing for either hand.

Example Applications

Figure 5 illustrates two examples of how designers can use these templates to guide occlusion related decisions:

- When specifying the position of an information bubble opened with a two finger tap, a designer can use the 2 digit tap template to position the bubble at an offset and angle least likely to be occluded, but also minimizing distance for ideal Gestalt association (Figure 5a).
- A design for a small multi-touch rotary dial can use the 2 digit transform template to position labels on the circumference of the dial to minimize occlusion (Figure 5b). While labels should avoid the "often occluded area," the designer can utilize the "possibly occluded" band to place lower priority labels. The template shape enables further refinements, such placing highest priority labels at '1', since they are least likely to be occluded.

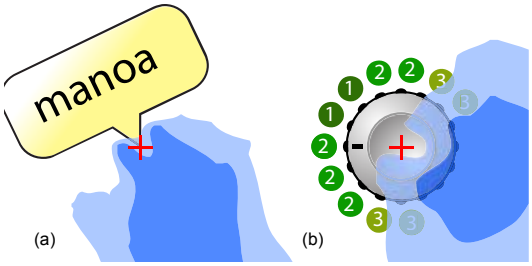


Figure 5. Example applications of design-time templates: (a) the best place to put an information bubble activated with 2 digits; (b) the best option indent locations for a rotary dial operated with 2 digits: 1 is best, 2 are good, 3 are acceptable.

Other scenarios include contextual menu placement [9], pie menus [4], and visualizations [27]. The designers of FingerGlass [13] and ShadowGuides [7] could use these templates to justify their choice for spatial offsets and there are multiple opportunities to refine layouts to minimize occlusion in a complex application like Eden [15].

Although helpful, these are design-time decisions using mean shapes. Knowledge of the currently occluded area at any given moment would be even better.

GEOMETRIC MODEL OF OCCLUSION SHAPE

In this section, we develop and test different ways to model the occluded area suitable for high fidelity technologies like DI and more limited hardware like capacitance. Our aim is to show that applying a single model to a wide range of postures is possible, and establish an upper bound on potential performance. This sets the stage for a configurable real time model of occlusion as future work.

Diffused Illumination (DI) Capture Model

Unlike a pen tablet, a DI tabletop captures an image of the hand and forearm near the surface. Without a switchable diffuser [10] or calibrated overhead camera, this is the best

available method for capturing actual hand shape. With minor rotation and offset transformations relative to the contact centroid, this should match a portion of the occluded area and forms a key part of our first geometric model. The problem is that the whole forearm is not usually captured due to its height above the diffuser, so we add a rectangle with a constant offset of 100 mm from the same centroid (Figure 6a). This DI model has five parameters: a distance and angle to describe the offset of the DI image, an angle for rotation of the DI image, and a rotation angle and width for the rectangle.

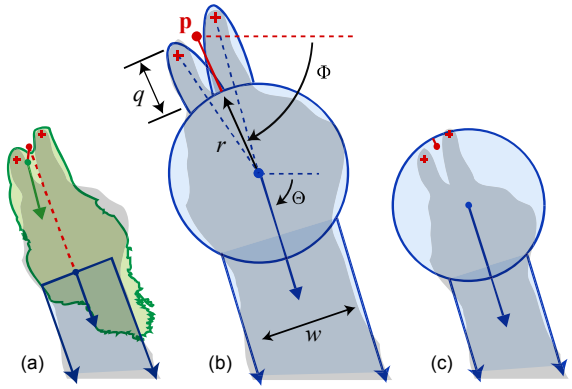


Figure 6. Three occlusion shape models: (a) DI and rectangle; (b) multi-touch circle and rectangle; (c) Vogel et al.

Multi-Touch Circle and Rectangle Model

Typically, FTIR only provides the shape and size of the contacts [8], and capacitive only X-Y contact positions. To cover a wider range of devices, we extended Vogel at al.'s circle and rectangle model (Figure 6c) to multi-touch postures. Our model has exactly the same five parameters (Figure 6b): distance q and angle Φ describe the offset of circle from point **p**, the centroid of the *actual* finger contact points; r is the circle radius; Θ is angle of the rectangle; and w is the rectangle width. To represent extended fingers, we add an ellipse for each contact and position it relative to the circle. Specifically, the major axis is aligned with the vector from the contact point to the circle centre and its length is such that the minor axis forms a chord on the circle. We set the minor axis to 15 mm and scale the major axis such that the tip extends 10 mm beyond the contact. Since these are constants, no additional model parameters are introduced for the ellipses. When faced with modeling extended finger postures, the ellipses should increase accuracy compared to only Vogel et al.'s circle and rectangle (Figure 6c). A limitation is that we only have actual X-Y contacts, so a single finger contact with an open hand (like participant 8 in Figure 9) would only have a single contact. In this case, the model can increase r and decrease q to remove the ellipse and cover the entire hand with the circle.

Testing Models by Fitting to Captured Silhouettes

To test and compare the models, we use the same approach as Vogel et al. [25]: we "fit" the model to each silhouette as accurately as possible and use precision-recall plots and F1 scores to compare fidelity. Note that we are not *learning* model parameters, but rather estimating an upper bound for model capability. The reader can consult the prior art for methodology details noting these changes: we use a single fitting stage; we only used pattern search; we use the posture contact centroid; and our objective function simply maximizes the F2 score. F2 favours recall over precision so more of the occluded area is covered creating more false positives, but fewer false negatives — a quality Vogel et al. argue is desirable. To remain consistent with past work, we compare fidelity with the equally weighted F1 score. Since the models are one-handed, we remove two-handed transform cases. Fitting each model to the 9209 test cases took more than 12 hours with a 2.66 GHz quad processor.

Mean F1 scores for the DI model and our multi-touch circle and rectangle model are comparable: 0.801 (SD 0.078) and 0.808 (SD 0.064) respectively. We also tested the "fingerless" Vogel et al. model, which has a very respectable F1 of 0.785 (SD 0.066). Since our model is based closely on it, a similar score is expected, but it is encouraging to see the finger ellipses improve fidelity without additional parameters.

Our geometric model is primarily useful for non-DI devices where only individual finger contacts are sensed, not postures like palm, fist, and side. Thus, comparing mean F1 scores using only 1 to 5 digit contacts is more relevant. In this test, the DI model achieves a similar F1 score of 0.802 (SD 0.074) and a precision-recall plot illustrates a precision bias (Figure 7a). Our multi-touch circle and rectangle model improves with 0.819 (SD 0.055) and the plot suggests very high recall and good precision (Figure 7b). The Vogel et al. model also improves to 0.797 (SD 0.057).

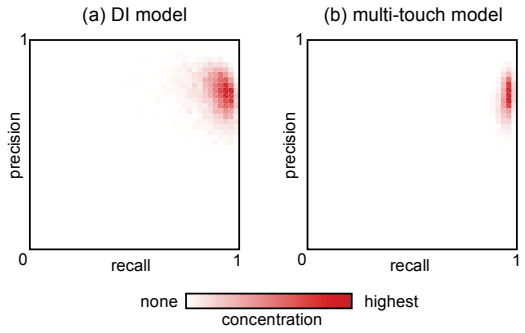


Figure 7. Precision-recall concentration plots: (a) DI shadow; (b) multi-touch circle and rectangle. Points in upper right indicate better performance.

IMAGE CORPUS

Generating the occlusion design-time templates and testing different occlusion models leverages the large corpus of images and metadata we created in our experiment. This includes 16,320 sets of images synchronized with contact positions, sizes, and orientations. Each image set has an occlusion silhouette, raw and rectified versions of a point-of-view frame capture, a DI capture, and an overhead frame

capture. We believe this corpus will be a valuable resource for related research such as palm rejection, finger to hand mapping [6], and identifying users by hand contour [19]. The entire corpus is publically available to download[1].

CONCLUSIONS AND FUTURE WORK

We adapted an established image-based methodology [25] to study the shape of occlusion on a multi-touch tabletop. By examining the shapes of mean occlusion silhouette visualizations and calculating descriptive statistics, we found common characteristics across people, postures, and tasks. Based on this, we created occlusion awareness templates to guide interface layout decisions, and tested different geometric models suitable for high- and low-fidelity multi-touch input technologies. This latter contribution is a necessary step towards a real-time, configurable model to enable multi-touch occlusion-aware techniques such as those created for direct pen input [23].

Using a tabletop allowed us to test a wider assortment of postures and test DI captures as a potential model, but we had to accept potential limitations. Although the Microsoft Surface is popular and has spawned other similarly-sized tables, larger and taller tables could influence body posture and resulting occlusion. More broadly, an obvious question is how well our results generalize to other multi-touch phones, tablets, tables, inclined desks, and vertical walls. We argue that at least for near horizontal cases, the relative relationship and viewing angle of operator to device is similar. The biggest shape change was due to forearm angle when reaching, which does not apply to small devices. However, body postures may contort more drastically when reaching targets at extreme edges of a large surface. A second limiting factor is that our results are relative to body location. For larger surfaces, we assume that the location of people around the table can be determined using sensors [1], or perhaps using the angle or shape of the hand in the spirit of Dang et al. [6].

Finally, we are eager to see how the community might use the large corpus of data we created.

ACKNOWLEDGEMENTS

This work was partially funded by Interreg IV-A 2 seas SHIVA project, and ERDF funds.

REFERENCES

1. Annett, M., Grossman, T., Wigdor, D., and Fitzmaurice, G. Medusa: a proximity-aware multi-touch tabletop. *Proc. UIST'11*, ACM (2011), 337–346.
2. Apple Computer. iPad 2 User Guide. 2011.
3. Bragdon, A., Uguray, A., Wigdor, D., Anagnostopoulos, S., Zeleznik, R., and Feman, R. Gesture play: motivating online gesture learning with fun, positive reinforcement and physical metaphors. *Proc. ITS'10*, ACM (2010), 39–48.
4. Brandl, P., Leitner, J., Seifried, T., Haller, M., Doray, B., and To, P. Occlusion-aware menu design for digital tabletops. *Proc. CHI'09,* ACM (2009), 3223-3228.
5. Cao, X., Wilson, A., Balakrishnan, R., Hinckley, K., and Hudson, S.E. ShapeTouch: Leveraging Contact Shape on Interactive Surfaces. Proc. *Horizontal Interactive Human Computer Systems*, IEEE (2008).
6. Dang, C.T., Straub, M., and André, E. Hand distinction for multi-touch tabletop interaction. *Proc. ITS '09*, ACM (2009), 101-108.
7. Freeman, D., Benko, H., Morris, M.R., and Wigdor, D. ShadowGuides: visualizations for in-situ learning of multi-touch and whole-hand gestures. *Proc. ITS'09*, ACM (2009), 165–172.
8. Han, J.Y. Low-cost multi-touch sensing through frustrated total internal reflection. *Proc.UIST'05,* ACM (2005), 115-118.
9. Hancock, M.S. and Booth, K.S. Improving menu placement strategies for pen input. *Proc. GI'04*, CIPS (2004), 221-230.
10. Hilliges, O., Izadi, S., Wilson, A.D., Hodges, S., Garcia-Mendoza, A., and Butz, A. Interactions in the air: adding further depth to interactive tabletops. *Proc. UIST'09*, ACM (2009), 139-148.
11. Hinrichs, U. and Carpendale, S. Gestures in the wild: studying multi-touch gesture sequences on interactive tabletop exhibits. *Proc. CHI'11*, ACM (2011), 3023–3032.
12. Holz, C. and Baudisch, P. Understanding touch. *Proc. CHI'11*, ACM (2011), 2501-2510.
13. Käser, D.P., Agrawala, M., and Pauly, M. FingerGlass: efficient multiscale interaction on multitouch screens. *Proc. CHI'11*, ACM (2011), 1601-1610.
14. Kin, K., Agrawala, M., and DeRose, T. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. *Proc. GI'09*, CIPS (2009), 119-124.
15. Kin, K., Miller, T., Bollensdorff, B., DeRose, T., Hartmann, B., and Agrawala, M. Eden: a professional multitouch tool for constructing virtual organic environments. *Proc. CHI'11*, ACM (2011), 1343-1352.
16. Micire, M., Desai, M., Courtemanche, A., Tsui, K.M., and Yanco, H.A. Analysis of natural gestures for controlling robot teams on multi-touch tabletop surfaces. *Proc. ITS'09*, ACM (2009), 41–48.
17. Morris, M.R., Wobbrock, J.O., and Wilson, A.D. Understanding users' preferences for surface gestures. *Proc. GI'10*, CIPS (2010), 261-268.
18. Pheasant, S. and Hastlegrave, C. *Bodyspace: Anthropometry, Ergonomics and the Design of the Work*. CRC, 2006.
19. Schmidt, D., Chong, M.K., and Gellersen, H. HandsDown. *Proc. NordCHI'10*, ACM (2010), 432-441.
20. Shen, C., Ryall, K., Forlines, C., et al. Informing the Design of Direct-Touch Tabletops. *IEEE Comput. Graph. Appl. 26*, 5 (2006), 36-46.
21. Shneiderman, B. Touch Screens Now Offer Compelling Uses. *IEEE Software 8*, 2 (1991), 93-94, 107.
22. Vogel, D. and Balakrishnan, R. Direct Pen Interaction with a Conventional Graphical User Interface. *Human-Computer Interaction 25*, 4 (2010), 324-388.
23. Vogel, D. and Balakrishnan, R. Occlusion-aware interfaces. *Proc. CHI'10*, ACM (2010), 263-272.
24. Vogel, D. and Baudisch, P. Shift: a technique for operating pen-based interfaces using touch. *Proc. CHI'07*, ACM (2007), 657-666.

[1] http://www.nonsequitoria.com/mt_occlusion

25. Vogel, D., Cudmore, M., Casiez, G., Balakrishnan, R., and Keliher, L. Hand Occlusion with Tablet-sized Direct Pen Input. *Proc. CHI'09*, ACM (2009), 557-566.
26. Wagner, D. and Schmalstieg, D. ARToolKitPlus for Pose Tracking on Mobile Devices. *Proc. CVWW'07*, (2007).
27. Wigdor, D., Benko, H., Pella, J., Lombardo, J., and Williams, S. Rock & rails: extending multi-touch interactions with shape gestures to enable precise spatial manipulations. *Proc. CHI'11*, ACM (2011), 1581–1590.
28. Wigdor, D., Forlines, C., Baudisch, P., Barnwell, J., and Shen, C. Lucid touch: a see-through mobile device. *Proc. UIST'07*, ACM (2007), 269-278.
29. Wigdor, D., Leigh, D., Forlines, C., et al. Under the table interaction. *Proc. UIST'06*, ACM (2006), 259-268.
30. Wobbrock, J.O., Morris, M.R., and Wilson, A.D. User-defined gestures for surface computing. *Proc. CHI'09*, ACM (2009), 1083-1092.
31. Wroblewski, L. *Touch Gesture Reference Guide*. 2010.

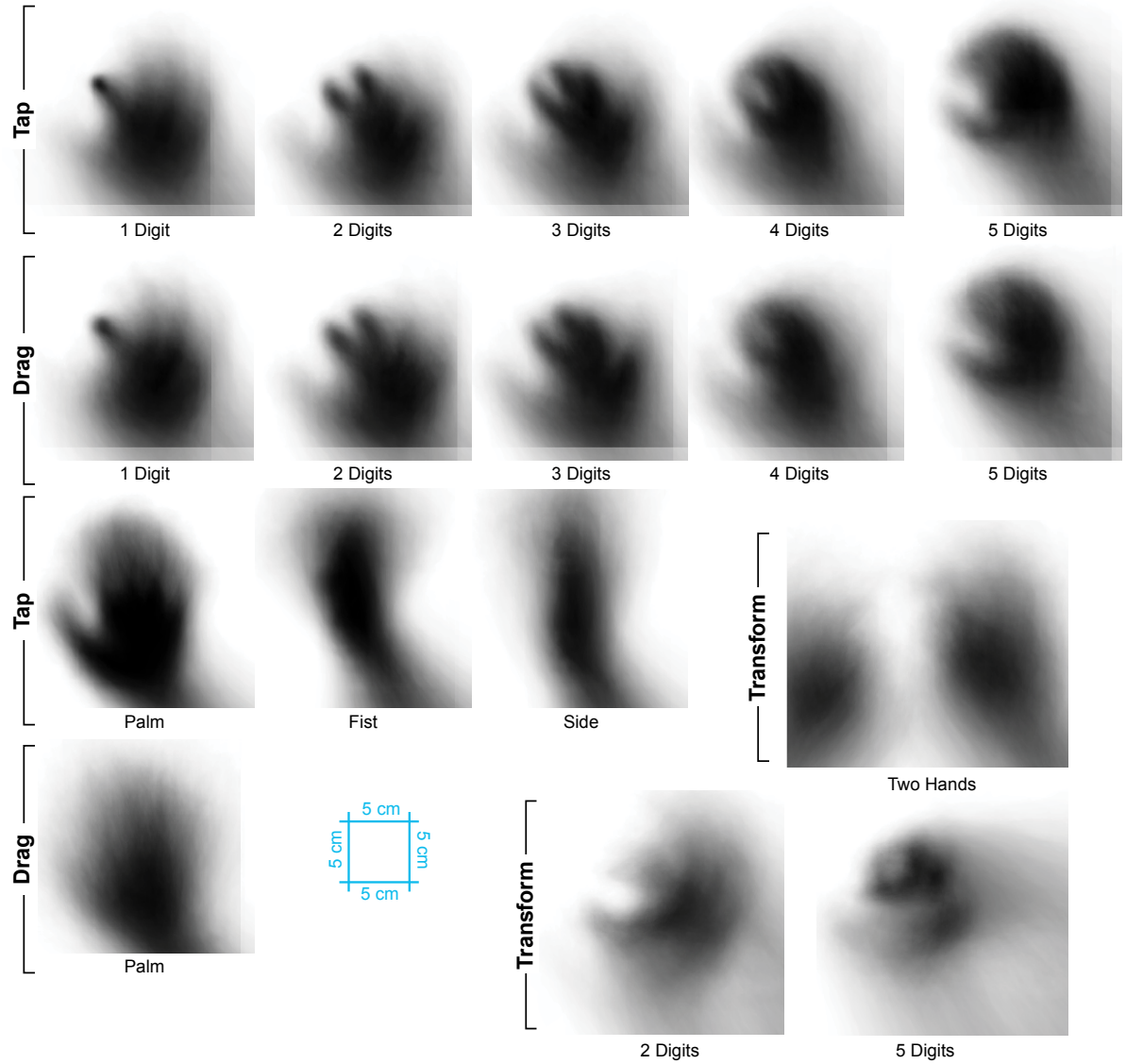


Figure 8. Mean occlusion silhouettes for *Tap*, *Drag*, and *Transform* by *Posture*. *Tap* and *Drag* silhouettes use end of task capture, Transform silhouettes use beginning and end. Horizontal and vertical hatching is due to rectification and camera cropping.

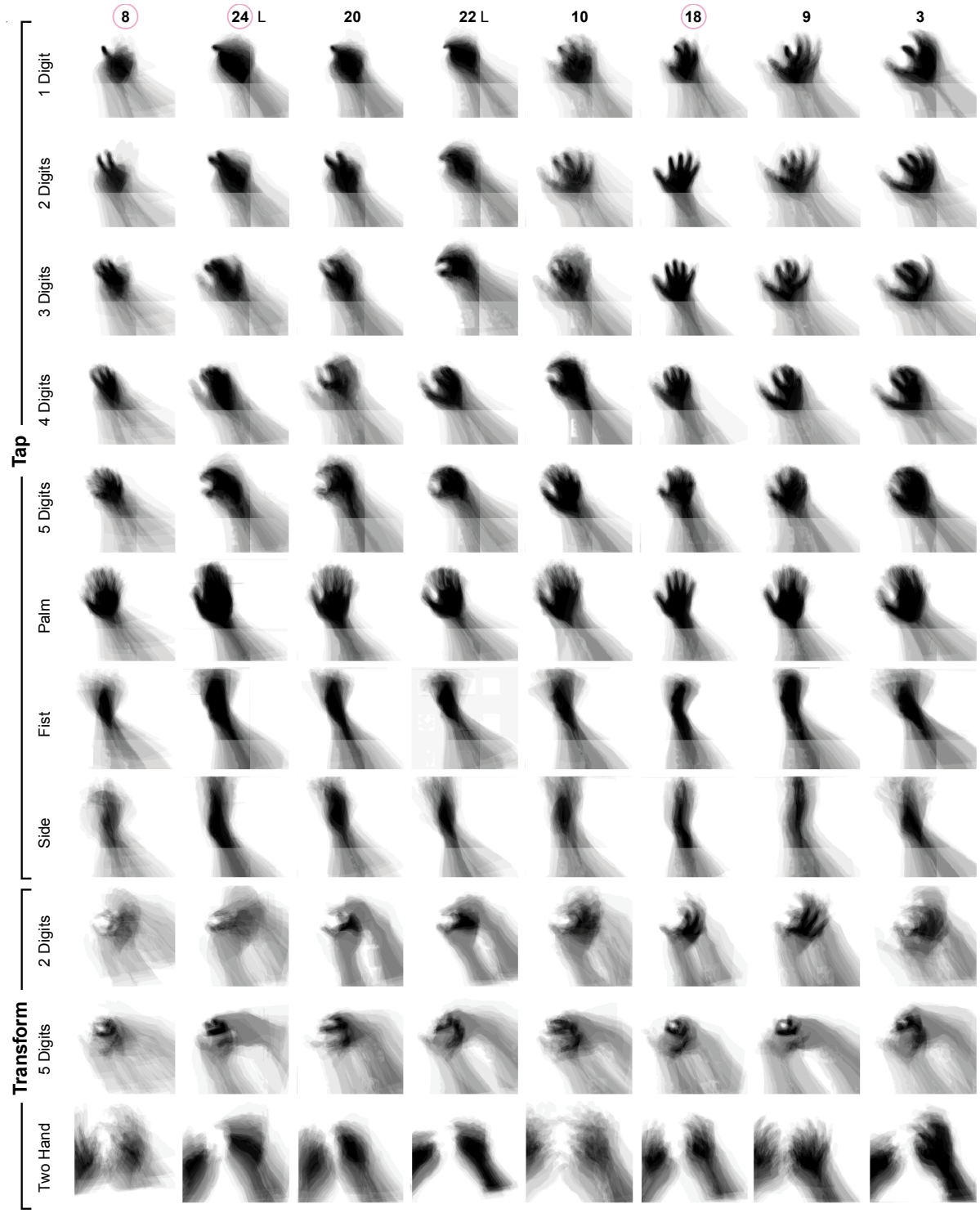


Figure 9. Selected participant mean occlusion shapes by *Posture* for *Tap* and *Transform* tasks. Individuals are in columns with participant numbers adored with a pink circle for females and a 'L' for left-handed participants (shown mirrored)

Surfpad: Riding Towards Targets on a Squeeze Film Effect

Géry Casiez[1,3,4], **Nicolas Roussel**[3], **Romuald Vanbelleghem**[3] **& Frédéric Giraud**[2,3,4]
[1]LIFL, [2]L2EP, [3]INRIA Lille & [4]University of Lille, France
gery.casiez@lifl.fr, nicolas.roussel@inria.fr, frederic.giraud@polytech-lille.fr

ABSTRACT
We present *Surfpad*, a pointing facilitation technique that does not decrease target distance or increase target width in either control or display space. This new technique operates instead in the tactile domain by taking advantage of the ability to alter a touchpad's coefficient of friction by means of a squeeze film effect. We report on three experiments comparing *Surfpad* to the *Semantic Pointing* technique and constant control-display gain with and without distractor targets. Our results clearly show the limits of traditional target-aware control-display gain adaptation in the latter case, and the benefits of our tactile approach in both cases. *Surfpad* leads to a performance improvement close to 9% compared to unassisted pointing at small targets with no distractor. It is also robust to high distractor densities, keeping an average performance improvement of nearly 10% while *Semantic Pointing* can degrade up to 100%. Our results also suggest the performance improvement is caused by tactile information feedback rather than mechanical causes, and that the feedback is more effective when friction is increased on targets using a simple step function.

ACM Classification Keywords
H.5.2 [Information interfaces and presentation]: User interfaces - Graphical user interfaces.

General Terms
Design, Performance, Experimentation, Human Factors

Author Keywords
Pointing facilitation, target-aware, control-display gain adaptation, squeeze film effect

INTRODUCTION

Pointing is a fundamental task of modern human computer interfaces and has been extensively studied by the HCI research community. Fitts' law has proven to be one of the most robust and widely adopted models in this area [29]. It expresses the movement time to acquire a target of width W at a distance D as a linear function of the index of difficulty $ID = \log_2(\frac{D}{W} + 1)$.

CHI 2011, May 7 - 12, 2011, Vancouver, BC, Canada.

Numerous techniques have been proposed that attempt to beat Fitts' law, i.e. to make virtual pointing easier than it is in the physical world [5]. Most of these techniques attempt to decrease D, to increase W, or both. Most of them are also inherently target-aware [34]: they take advantage of some knowledge about the size and position of the targets and sometimes modify them. In cases where pointing involves the indirect control of a visual cursor, some techniques operate by dynamically adapting the control-display gain $CDgain = V_{\text{cursor}}/V_{\text{device}}$ [12]. Other techniques supplement the visual display with auditory or haptic feedback. Yet despite their demonstrated efficiency in simple configurations, most target-aware pointing techniques are difficult to use in practice. One of the key problems that affects them in real-life situations is the potential interferences caused by intervening targets on the way to the primary one (distractors), a problem that is still largely understudied.

In this paper, we present *Surfpad*, a pointing facilitation technique that does not decrease D or increase W in either control or display space. This new technique operates instead in the tactile domain by taking advantage of the ability to alter the coefficient of friction of a particular touchpad, the STIMTAC [9], by means of a squeeze film effect (Figure 1). We report on three experiments comparing *Surfpad* to the *Semantic Pointing* technique [10] and constant control-display gain with and without distractor targets. Our results clearly show the limits of traditional target-aware CD gain adaptation in the latter case, and the benefits of our tactile approach in both cases. Our results also suggest the performance improvement is caused by tactile information feedback rather than mechanical causes, and that the feedback is more effective when friction is increased on targets using a simple step function.

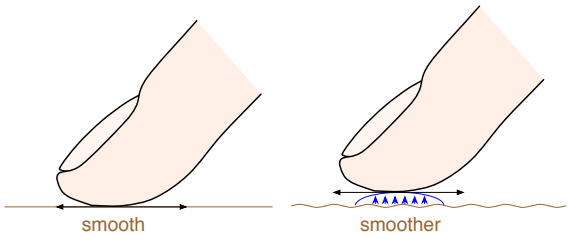


Figure 1. The squeeze film effect: controlled vibration of a surface creates an air film which reduces its coefficient of friction.

The paper is organized as follows. After reviewing related work on pointing facilitation techniques and haptic feedback, we describe the STIMTAC device and our *Surfpad* technique. We then describe our three experiments. We conclude with some directions for future work.

RELATED WORK

A detailed review of pointing facilitation techniques can be found in [5]. As explained, most of these techniques are target-aware and involve the reduction of target distance, the increase of target width, or both. In what follows, we briefly discuss the most relevant examples, focusing on the use of a haptic modality and the impact of distractors.

Reducing D and Increasing W

Different methods have been proposed to reduce target distance. *Drag-and-pop* [7], for example, temporarily brings potential targets closer to the pointer. Other techniques use endpoint prediction to make the cursor automatically jump over empty spaces and potential distractors [19, 4]. The *ninja cursors* [25] reduce D in yet another way by attaching multiple cursors to the same device and using knowledge about the targets to resolve pointing ambiguities.

Different methods have also been proposed to increase target width. *Expanding targets* [27], for example, dynamically grow to provide a larger area to interact with at the focus of attention. Expansion usually occurs in visual space, but sometimes also in control space (more on this below). Research has also shown that the W term of Fitts' law can apply to the width of the cursor, rather than that of the target, which led to the design of different *area cursors* [23, 35, 18].

A problem with the above techniques is that they are often visually distracting because of the displacement, growing or shrinking of objects. Other techniques have been proposed that preserve the display by operating only in control space. Described as *semantic pointing* [10] or using a stickiness or force field metaphor, these control space techniques operate by adapting the CD gain [35, 14, 10] or warping the cursor [31, 14, 13, 1, 21]. The CD gain is typically reduced when the cursor is over targets or approaching them, thereby expanding them in control space. Warping the cursor additionally supports trajectory adjustments in any direction.

A particular case of CD gain adaptation is described in [24], where it is not used to reduce D or increase W but to create a "cursor-catching effect". By requiring more movement effort to leave than to enter the target centre without increasing the total amount of effort to enter and leave the target area, the proposed *dynamic cursor gain* preserves the pointing task's index of difficulty. CD gain adaptation can be seen as a feedback mechanism in this context, rather than a Fitts' law optimization enabler, an approach that was also used to successfully simulate haptic percepts [26].

Haptic Feedback for Pointing Facilitation

Following the ISO standard [30], we use the term *haptic* to refer to two different types of feedback: *tactile* feedback (information received through nerve receptors in the skin) and *kinesthetic* feedback (information sensed through movement and/or force to muscles and joints). Haptic feedback has long been used as an assistive technology for disabled users, or to supplement the visual modality in teleoperation systems and virtual environments. But researchers have also investigated its potential for facilitating routine target acquisition in graphical interfaces.

Discrete $2D$ pointing tasks have received the most attention [3, 2, 24, 28, 16, 22]. But reciprocal $1D$ pointing [13], reciprocal $2D$ pointing and crossing [17], steering [11, 15], as well as ecological tasks [28, 15, 13], have also been studied. The device used for these studies is typically a haptic-enabled mouse. Other devices include a customized trackpoint [11], stylus [17] or trackball [24], a force-feedback joystick [22] and a 6DOF haptic device [28]. Feedback is provided by either exerting a force on the device to constrain its movement [2, 24, 28, 15, 16, 22] or by moving or vibrating a small part of it [3, 2, 11, 13, 17]. In [17], the haptic mechanism was used to confirm the selection of the target. In all the other studies, it was used to provide feedback about the cursor's relative position to the target, or tunnel, during the selection movement.

Haptic feedback has mostly been evaluated against "normal" pointing, i.e. pointing with no additional feedback indicating the cursor is over a target or has been selected. All the above studies showed it can improve users' targeting performance in this context by reducing the overall movement time [2, 11, 15, 16, 22, 13, 17], the time to stop after entering the target [3], or the error rate [28]. Some studies suggest that tactile feedback might be particularly effective at reducing selection times for small targets at the cost of higher error rates, although the reasons for the additional errors remain unclear [2, 13]. It has also been suggested that tactile feedback does not aid in direct input configurations [17].

Moving and vibrating parts of haptic devices usually generate audible sounds that one might want to filter out during experiments [13]. Comparisons of tactile and auditory feedback indeed showed similar positive effects on target acquisition [3, 13]. Tactile feedback has also been compared to visual feedback [3, 11] and CD gain adaptation [13], and researchers have investigated whether these different modalities can combine in a positive way [3, 2, 11, 13]. As Cockburn and Brewster put it, "some do while others do not" and the actual result depends a lot on the nature of the task: a promising technique poorly applied to a simple ecological task can damage interaction by distracting users from it [13].

Most of the devices we mentioned were based on simple and well-tested electromagnetic technologies (e.g. solenoids, voice coils, or vibratory motors with an offset mass). The problem with these technologies is that the haptic sensations they support are rather coarse. Yet recent advances in haptic technologies offer significant promise for extending their use in HCI in general, and pointing facilitation in particular, by supporting more subtle sensations. Recent works on friction reduction are particularly interesting in this context.

While most tactile feedbacks rely on active stimulation using pin-based arrays, Watanabe & Fukui proposed a method to create a smoother feeling on a surface by applying ultrasonic

vibration with only a few micrometers amplitude [32]. The perceived feeling, caused by a *squeeze film effect* (Figure 1), has been used recently to simulate bumps and holes [33]. Electrovibration has also been shown to support similar sensations [6]. In contrast to traditional vibrotactile approaches, devices based on these technologies provide information passively, acting as *texture displays* [20]: they do not transfer energy to the user but modify how energy is dissipated within the contact area by a user-initiated friction process.

Impact of Distractors

Target-aware pointing techniques tend to work best on sparse layouts. For intrinsic reasons, many of them do not scale well to situations where multiple potential targets are closely packed together [5]. In real-world applications however, locally dense clusters of potential targets emerge for various reasons [8]. Surprisingly, although the problem is clearly identified in the literature, little research has been done to systematically evaluate the impact of distractors on existing techniques or design new ones that take them into account.

Among the studies of control space techniques and haptic feedback we discussed, a few took distractors explicitly into account – although in limited ways. One variable of the experiments described in [24] and [35] was the presence or absence of a single distractor along the target path (multiple distractors were actually used to make sure one would always be on the path when needed). In [22], one condition involved a distractor located at 180, 90, 0 or -90° relative to the task axis. The second experiment described in [16] displayed 13 targets arranged in a cross shape and required the user to randomly move from one to another, all the others acting as potentially avoidable distractors. The second experiment described in [13] is one of the very few that evaluated the impact of multiple distractors on a control space technique (*sticky targets*) and tactile feedback in a simple ecological task (menu selection). Results from all these studies suggest a negative impact of distractors on movement time, error rate, or user satisfaction. All the authors recommend further investigation.

STIMTAC AND THE SURFPAD TECHNIQUE

Previous research has clearly demonstrated the potential of control space techniques and haptic feedback for pointing facilitation. Yet, the impact of distractors on these techniques remains largely unknown. At the same time, recent advances in haptic technologies offer significant promise for supporting a wider range of sensations and thus more subtle barehand interactions. All these elements contributed to our initial motivation for the *Surfpad* technique, which relies on a particular device, the STIMTAC [9].

STIMTAC

The STIMTAC is a touchpad-like device based on the squeeze film effect described above. The tactile plate is made of 36 piezoelectric cells bonded on a $79\ mm \times 49\ mm$ copper–beryllium plate. This monomorph structure constitutes a mechanical resonator excited by a $40\ V$ sinusoidal voltage provided by a $0.5\ W$ power supply (to reduce power consumption, the device can be configured so that it vibrates the plate only if finger contact is detected). The overall design results in a compact and lightweight form factor that

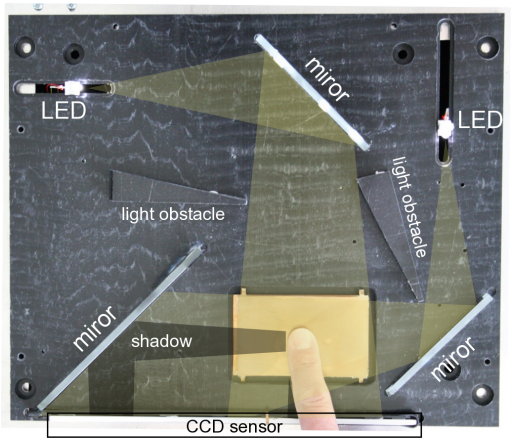


Figure 2. Picture of the STIMTAC device with its shell removed. The shell has only one opening for the tactile surface.

allows free exploration of the tactile surface (Figure 2). The plate is coated with a thin plastic layer to make finger contact more comfortable. It vibrates at the ultrasonic frequency of $28.55\ kHz$ and thus emits no perceptible noise during operation. Since the frequency is outside skin mechanoreceptors' bandwidth, users do not feel the vibration. Instead, they feel its effect on tribological contact mechanisms: the touchpad feels more slippery as one raises the vibration amplitude. The device is typically configured for a maximum amplitude of $1\ \mu m$ which can reduce friction up to 50% depending on surface preparation (e.g. cleaning the surface also affects its coefficient of friction).

Traditional touchpad sensors are incompatible with the squeeze film effect due to the relatively high voltage and frequencies (a resistive sensor would damp the vibrations and a capacitive one would be perturbed by the electric field). A custom-made optical sensor is thus used to locate the user's finger. The sensor was built from two white LEDs, three mirrors and a linear $200\ dpi$ CCD array. An on-board DSP computes the centroids of two shadow images created by the user's finger and sends them on a serial line as absolute (x, y) coordinates at a rate of $120\ Hz$. The final resolution of the sensor is $170\ dpi$ due to optical constraints and post-treatments. The serial line allows to specify the desired coefficient of friction by controling the amplitude at a rate up to $120\ Hz$ using a 7-bit encoded integer between 0 (no squeeze film effect, maximum friction) and 127 (maximum amplitude, maximum effect, minimum friction).

The Surfpad Technique

Tactile feedback through the STIMTAC builds on the relative displacement that exists between a fingertip and a surface when a user is probing for friction. A user moving a finger on the switched-off plate will find it hard, smooth, and not sticky. But, because of its high level of friction, the skin will be stretched laterally, which will become obvious at any direction change. Once the squeeze film effect is activated, the surface retains its original properties but with the reduced friction, the skin becomes less stretched. The sensation can approach the feeling of touching a silk scarf. If the

effect is disabled while the finger is moving, the increased amount of friction will be quite noticeable.

Considering these new possibilities, we were excited to investigate ways in which they could be used for pointing facilitation. Influenced by the related work, we started with the idea of designing a target-aware technique. As we saw, many of these techniques operate by modifying the mechanics of motion around targets in the virtual world (for control space techniques) or in the physical one (for haptic techniques). The purpose of the modifications is to facilitate pointing and the desired result is to slow down the cursor or guide its movements. In fact, existing techniques can be thought of as increasing the friction of specific objects.

The STIMTAC can only reduce friction. In order to increase it on specific objects, one needs to decrease it everywhere else. The technique we propose originates from this figure-ground reversal in pointing facilitation: instead of slowing down the cursor around targets, why not facilitate its movement on the background? While most target-aware techniques tend to ignore the background, ours is background-aware. A surfing metaphor seemed appropriate, the low friction background corresponding to the ocean, the objects to the shore, and the finger-controlled cursor to the board.

The *Surfpad* technique uses the programmable squeeze film effect of the STIMTAC to reduce the touchpad's coefficient of friction at all times except when the cursor is over a target. We have implemented it in two ways. Similar to traditional sticky targets, *Surfpad*Π uses the following step function:

$$\textit{Surfpad}\Pi(x) = \begin{cases} 0 & \textit{maximum friction if over a target} \\ 127 & \textit{minimum friction otherwise} \end{cases}$$

Instead of a step function, *Surfpad*Ω uses the Ω bell-shaped mixing function defined in [10] to avoid discontinuities in the amount of friction.

EXPERIMENT 1: SURFPAD

The goal of this first experiment is to investigate the effect of *Surfpad* on performance in a pointing task and compare it to target-aware CD gain adaptation and constant CD gain in the absence of distractors. We used the *Semantic Pointing* technique [10] for target-aware CD gain adaptation as it is well documented and considered as a reference in this domain.

Apparatus

The STIMTAC device described in the previous section was used as the input device for all the techniques to eliminate extraneous intra-devices differences such as ergonomics, size and sensitivity. We used a 15" LCD display at a 1280×800 pixel resolution. The experiment was coded in C++ and OpenGL. The frequency of the visual and haptic renderings were $60\ Hz$ and $120\ Hz$ respectively.

Task

We used a reciprocal one dimensional pointing task (Figure 3). Each trial began after the previous target was successfully selected and ended with the selection of the current target. After a target was successfully selected, it turned grey and the next one (on the other side of the screen) turned green. If a participant missed a target, a sound was heard and an error was logged. Participants had to successfully select the current target before moving to the next one, even if it required multiple clicks. The pointer was not constrained to screen bounds to avoid using the edges to facilitate target acquisition. Participants used the left `Ctrl` key on a keyboard with their non-dominant hand to select targets. After each block of trials, a cumulative error rate was displayed and a message encouraged participants to conform to an approximately 4% error rate by speeding up or slowing down.

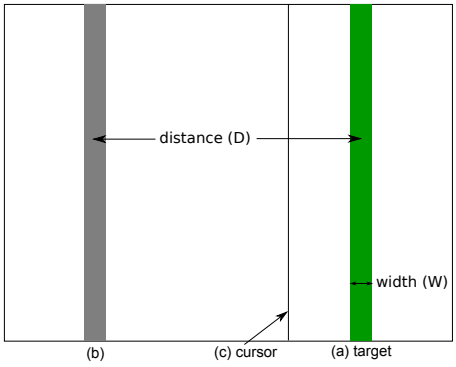


Figure 3. Experimental display. Targets were rendered as solid vertical bars equidistant from the center of the display in opposite directions along the horizontal axis. The target to be selected was colored green (a), and the previous one gray (b). The cursor was represented by a one-pixel-thick vertical black line (c).

Participants

Twelve unpaid volunteers with a mean age of 28.9 ($SD = 7.0$) served in the experiment (9 male and 3 female, 10 right-handed and 2 left-handed).

Design

A repeated measures within-subjects design was used. The independent variables were the technique used (TECHNIQUE) and the target distance (DISTANCE) and width (WIDTH). DISTANCE was evaluated with three levels (D_L = 100 mm, D_M = 50 mm, D_S = 25 mm) and WIDTH as well (W_L = 4.136 mm = 16 pixels, W_M = 2.068 mm = 8 pixels, W_S = 1.034 mm = 4 pixels)[1]. The index of difficulty thus ranged from 2.8 to 6.6.

The techniques were constant CD gain with no actuation of the STIMTAC (*Control*), constant CD gain with full actuation of the STIMTAC (*Control-*), *Semantic Pointing* using the Π step function (*SemPoint*Π), *Semantic Pointing* using the Ω mixing function (*SemPoint*Ω), *Surfpad*Π and *Surfpad*Ω. For *Semantic Pointing*, we chose to quadruple the size of targets in motor space as this was reported by Blanch et al. as yielding the best performance [10]. But while they had set their baseline CD gain to 1, we instead used one of 2 for all techniques to reduce clutching[2] with the largest distance considering the dimensions of our input surface.

[1] All distances and sizes are given in display space.

[2] Clutching consists in temporarily breaking the link between the physical device and the virtual pointer.

Participants had a few minutes to get used to the device in the *Control* condition before starting the experiment. They then completed 4 successive BLOCKS of trials for each TECHNIQUE. Each BLOCK consisted of 27 trials: 3 repetitions of the 9 DISTANCE×WIDTH combinations. The DISTANCE and WIDTH were presented in descending order. The presentation order of TECHNIQUE was counterbalanced across participants using a Latin Square design. Participants were encouraged to take a break every 9 trials. The experiment lasted approximately 50 minutes.

In summary, the experimental design was: 12 participants × 6 TECHNIQUE × 4 BLOCKS × 3 DISTANCE × 3 WIDTH × 3 trials = 7,776 total trials.

RESULTS

The dependent variables were the error rate, the movement time, the approaching time, the stopping time, the click time and the clutch time.

Error Rate

Targets that were not selected on the first attempt were marked as errors. Participants emphasized speed over accuracy with an overall error rate of 6.5%. A repeated measures ANOVA showed a significant effect of WIDTH on error rate, the latter increasing as target width decreases ($F_{2,22}$=24.3, p<0.001; W_S : 10.9%, W_M : 4.6%, W_L : 3.0%). There was also a significant effect of TECHNIQUE on error rate ($F_{5,55}$=7.06, p<0.001). Post-hoc analysis showed significant differences between *SemPoint*Π and all the other techniques (p<0.012; *Control* : 6.4%, *Control-* : 8.6%, *SemPoint*Π : 3.4%, *SemPoint*Ω : 5.0%, *Surfpad*Ω : 7.6%, *Surfpad*Π : 6.0%).

Movement Time

Movement time is the main dependent measure and is defined as the time taken to move from a target to the next one and click on it. Targets marked as errors were removed from the timing analysis. We also considered trials at least three standard deviations away from the mean for each TECHNIQUE×DISTANCE×WIDTH condition as outliers and removed them from the data analysis (1.6% of the trials).

A repeated measures ANOVA showed that the presentation order of TECHNIQUE had no significant effect or interaction on movement time, indicating that a within-participants design was appropriate. We also found no significant effect or interaction for BLOCK indicating there was no presence of a learning effect.

A repeated measures ANOVA showed a significant effect of TECHNIQUE on movement time ($F_{5,55}$=14.2, p<0.001). Pairwise comparisons showed no significant difference between *Surfpad*Ω and *Surfpad*Π (p=0.09), but while *Surfpad*Π was significantly different from all the other techniques (p<0.03), *Surfpad*Ω was only significantly different from *SemPoint*Π and *SemPoint*Ω (p<0.009). No significant difference was found between *SemPoint*Π and *SemPoint*Ω, but significant differences were found between these variants and the others techniques (p<0.008). No significant difference was found between *Control* and *Control-*, but significant differences were found between these two techniques and the others (p<0.012)

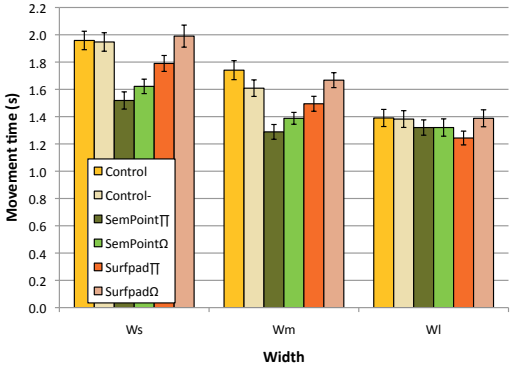


Figure 4. Mean movement time for TECHNIQUE and WIDTH. Error bars represent 95% confidence interval.

except *Surfpad*Ω (*Control* : 1.58s, *Control-* : 1.58s, *SemPoint*Π : 1.29s, *SemPoint*Ω : 1.31s, *Surfpad*Ω : 1.57s, *Surfpad*Π : 1.44s).

As predicted by Fitts' law, there was significant main effects of DISTANCE ($F_{2,22}$=106.4, p<0.001) and WIDTH ($F_{2,22}$=57.6, p<0.001) and a significant DISTANCE×WIDTH interaction ($F_{4,44}$=7.4, p<0.001). More interestingly, we also observed a TECHNIQUE×WIDTH interaction ($F_{10,110}$=11.2, p<0.001, Figure 4). Subsequent pairwise comparisons showed significant differences between the techniques as WIDTH gets smaller. For W_L, there was no significant difference between techniques except between *Surfpad*Π and *Surfpad*Ω (1.34s vs. 1.17s, p=0.032). For W_M and W_S, we observed similar patterns. There was no significant difference between the two control conditions and no significant difference between *SemPoint*Π and *SemPoint*Ω. We found a significant difference between *Surfpad*Π (1.39s) and *Surfpad*Ω (1.55s) for W_M (p=0.03), but not for W_S. For W_M and W_S, *Surfpad*Π, *SemPoint*Π and *SemPoint*Ω significantly improved movement time compared to the two control conditions (p<0.05). On these target sizes, *SemPoint*Π and *SemPoint*Ω were significantly better than *Surfpad*Π (p<0.02).

To better understand the effects observed on movement time, we split it in three parts: *approaching time*, *stopping time* and *click time*. As we noticed participants clutching during the experiment, we also analyzed the corresponding time.

Approaching Time

Approaching time is the time between the beginning of the movement and the instant the target border is crossed. There was a significant main effect of TECHNIQUE ($F_{5,55}$=7.9, p<0.001), DISTANCE ($F_{2,22}$=229.2, p<0.001) and WIDTH ($F_{2,22}$=17.5, p<0.001) on it as well as significant TECHNIQUE×WIDTH ($F_{10,110}$=18.6, p<0.001) and DISTANCE×WIDTH ($F_{4,44}$=11.4, p<0.001) interactions. Pairwise comparisons showed significant differences between the techniques in a trend similar to the one observed for movement time. In particular, *SemPoint*Π, *SemPoint*Ω and *Surfpad*Π showed a significantly lower approaching time compared to the two control conditions (p < 0.05). Pairwise comparisons showed that approaching time increased with larger target distances and smaller target widths.

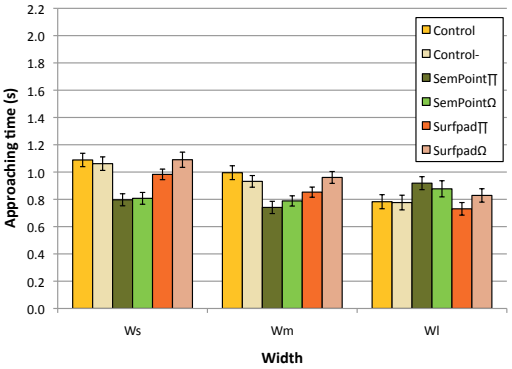


Figure 5. Mean approaching time for TECHNIQUE and WIDTH. Error bars represent 95% confidence interval.

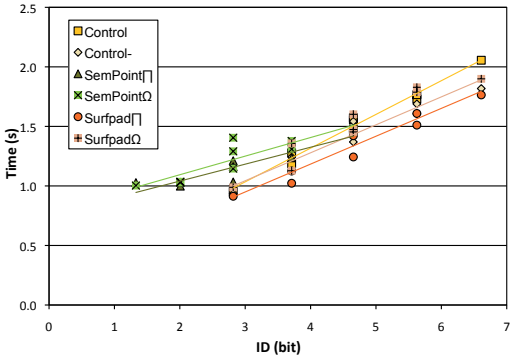


Figure 6. Fitts'law regression for each TECHNIQUE. IDs were computed in visual space for *Control*, *Control-*, *Surfpad*Ω and *Surfpad*Π and motor space for *SemPoint*Π and *SemPoint*Ω.

More surprisingly the TECHNIQUE×WIDTH interaction exhibited a different behavior for *SemPoint*Π and *SemPoint*Ω for W_L with a significantly higher approaching time compared to *Control*, *Control-* and *Surfpad*Π ($p<0.008$, Figure 5). This result might explain why no significant difference was found for the movement time on W_L between the *Semantic Pointing* variants and the two control conditions.

Stopping Time

Stopping time is the time between the first crossing of the target border and the stopping of the cursor. We observed a significant main effect of TECHNIQUE on it ($F_{5,55}$=3.7, p=0.038; *Control* : 0.44s, *Control-* : 0.44s, *SemPoint*Π : 0.34s, *SemPoint*Ω : 0.34s, *Surfpad*Ω : 0.41s, *Surfpad*Π : 0.36s). Pairwise comparisons showed significant differences ($p<0.05$) between the *Semantic Pointing* variants and the control conditions as well as *Surfpad*Ω. We also observed significant differences between *Surfpad*Π and the two control conditions ($p<0.02$). These results might partially explain the significant differences observed for the movement time.

We also found a significant main effect of WIDTH ($F_{5,55}$=20.1, p=0.001) on stopping time, the latter increasing with smaller widths. Pairwise comparisons showed significant differences between W_S and the two other widths (W_S = 0.48s, W_M = 0.36s, W_L = 0.33s; $p<0.004$).

Click Time

Click time is the time during which the pointer remains still before the button is pressed. We observed significant effects of TECHNIQUE ($F_{5,55}$=9.9, $p<0.001$) and WIDTH ($F_{2,22}$=210.4, p=0.001) on it. Pairwise comparisons showed significant differences between the *Semantic Pointing* variants and the other techniques (*SemPoint*Π = 0.15s, *SemPoint*Ω = 0.16s, *Control* = 0.20s, *Control-* = 0.19s, *Surfpad*Π = 0.21s, *Surfpad*Ω = 0.19s; $p<0.02$). This result might explain the significant difference observed between the *Semantic Pointing* variants and *Surfpad*Π for the movement time. There were significant differences between the widths, the click time increasing as the target width decreases (W_S = 0.24s, W_M = 0.18s, W_L = 0.13s; $p<0.001$).

Clutch Time and Number of Clutches

Clutch time is the total time the finger is lifted during a trial. There was a significant effect of DISTANCE ($F_{2,22}$=10.2, p=0.003) on it. Pairwise comparisons showed that it increases with larger target distances, with significant differences between all distances (D_S=0.018, D_M=0.045, D_L=0.147; $p<0.034$). There was also a significant effect of DISTANCE ($F_{2,22}$=6.2, p=0.014) on the number of clutches. Pairwise comparisons showed significant differences between all distances (D_S=0.36, D_M=0.58, D_L=1.73; $p<0.05$). Although not significant, we observed that the clutch time increased with target width (W_S=0.057s, W_M=0.062s, W_L=0.079s).

Fitts' Law Analysis

We ran a Fitts' law analysis on movement time removing trials where an error or clutching occured. For *SemPoint*Π and *SemPoint*Ω, we computed the index of difficulty in motor space, as suggested in [10]. We aggregated the data for each target width and distance, producing a total of 9 points for each TECHNIQUE. As shown on Figure 6, we obtained good regression fitness for *Control* ($MT = 0.185 + 0.283\,ID$, $r^2 = 0.98$), *Control-* ($MT = 0.339 + 0.235\,ID$, $r^2 = 0.90$), *Surfpad*Ω ($MT = 0.295 + 0.258\,ID$, $r^2 = 0.92$) and *Surfpad*Π ($MT = 0.247 + 0.234\,ID$, $r^2 = 0.93$), but reduced regression fitness for *SemPoint*Π ($MT = 0.756 + 0.142\,ID$, $r^2 = 0.84$) and *SemPoint*Ω ($MT = 0.781 + 0.156\,ID$, $r^2 = 0.79$). These last two results are mainly explained by the outlier point (D_L,W_L) for which we have the highest amount of clutching.

DISCUSSION

The experiment compared six techniques with the same baseline CD gain. No significant difference was found between the two control conditions. Our results show that *Surfpad*Π and the two *Semantic Pointing* variants significantly improve performance by 8.8% and 17.7% respectively, compared to the control conditions. *Surfpad*Ω did not result in any significant performance improvement compared to the control conditions. Results also show that these differences can be explained by a significant decrease in approaching time and stopping time for *Surfpad*Π, *SemPoint*Π and *SemPoint*Ω compared to the control conditions, and a significant decrease in click time for *SemPoint*Π and *SemPoint*Ω compared to the other techniques.

Mechanical Effect or Information Feedback?

Our results do not show any significant difference between *Control* and *Control-* for movement time, approaching time,

stopping time or click time. This inclines us to conclude that the friction reduction provided by the full actuation of the STIMTAC does not help to achieve faster movements.

The approaching time is significantly lower for *SemPoint*Π and *SemPoint*Ω than for the control conditions. As the step function and the bell-shaped mixing function operate only in the close vicinity of the targets, we assume this is the consequence of an anticipation phenomenon already observed in the use of *expanding targets* [27].

The approaching time is also significantly lower for *Surfpad*Π than for the control conditions. As friction reduction does not help to achieve faster movements, we assume this is also caused by anticipation of some later perceivable effect. Since the approaching time for *Surfpad*Ω is not significantly different from the control conditions, the perceived effect must be inherent to the Π function and incompatible with the Ω one.

Our results show a significantly lower stopping time for *Surfpad*Π compared to the control conditions. We hypothesize two reasons for this: (H1) a mechanical braking effect related to the friction increase, or (H2) tactile information feedback, i.e. a cognitive response to the perception of this increase. Although Π and Ω differ, their integral is the same[3]. From a mechanical perspective, the braking effect of H1 should also be observed with *Surfpad*Ω, which was not the case. H1 is also contradicted by the fact that *Surfpad*Π is more efficient on small target sizes where it should be more difficult to take advantage of a mechanical effect. We thus favor the second hypothesis, H2, which is also supported by previous evidence that the addition of tactile information can reduce response times by providing a confirmation without the need for visual attention [3, 17].

Target Size Matters

We observed a significant interaction between target sizes and techniques on movement time. Compared to the two control conditions, the mean movement time for target sizes W_M and W_S is reduced by 8.8% for *Surfpad*Π and 17.7% for the *Semantic Pointing* variants. Yet the three techniques fall short for W_L although according to Blanch et al., *Semantic Pointing* reduces the index of difficulty in motor space independently of target width [10].

Although clutching remained limited, we hypothesize that the slightly higher amount of it observed for W_L may have disrupted finger movements. Further experiments are required to validate this hypothesis. Still, the more pronounced effect of *Surfpad*Π as target width decreases agrees with results from previous work on tactile feedback [2].

EXPERIMENT 2: ANTI-SURFPAD

Results from Experiment 1 suggest that the performance improvement observed with *Surfpad*Π is the result of information feedback provided by the sudden increase of friction when the cursor crosses the target border. This second experiment was designed to better understand the nature of this feedback. We wanted to investigate if it can also be provided by a sudden decrease of friction. We call this condition *Anti-Surfpad*Π: the friction is minimal if the cursor is over a target, and maximal otherwise.

The apparatus and task for this experiment were the same as in the first one. The techniques were *Control*, *Surfpad*Π and *Anti-Surfpad*Π. Nine participants with a mean age of 27.3 (SD=4.7) took part in the experiment (8 male and 1 female, 8 right-handed and 1 left-handed).

A repeated measures ANOVA showed a significant main effect of TECHNIQUE ($F_{2,16}$=17.8, $p<0.001$) and a significant TECHNIQUE$\times$WIDTH interaction ($F_{4,32}$=4.4, $p=0.02$) on the movement time. Significant differences were found between the three techniques (*Control*=1.60s, *Anti-Surfpad*Π=1.82s, *Surfpad*Π=1.49s; $p<0.007$). The significant interaction showed that *Anti-Surfpad*Π increased the movement time for all target widths ($p<0.05$). *Surfpad*Π significantly improved performance compared to *Control* and *Anti-Surfpad*Π for W_S and W_M ($p<0.009$). However, the difference with *Control* was no longer significant for W_L.

A repeated measures ANOVA also showed a significant main effect of TECHNIQUE ($F_{2,16}$=17.8, $p<0.001$) on stopping time with significant differences between all techniques (*Control*=0.46s, *Surfpad*Π=0.39s, *Anti-Surfpad*Π=0.60s, $p<0.01$). We again hypothesize two reasons for this: (H3) a negative mechanical effect stronger than the information feedback, or (H4) counter-effective information feedback. Further experimentation is needed to validate these compatible hypotheses.

EXPERIMENT 3: DISTRACTORS

In Experiment 1, we showed that *Surfpad*Π and the *Semantic Pointing* variants significantly improve the movement time compared to the two control conditions, especially for small target sizes. The goal of this third experiment was to investigate the impact of distractors on the *Surfpad* and *Semantic Pointing* techniques.

As we found no significant difference between *SemPoint*Ω and *SemPoint*Π in Experiment 1, we decided to focus on *SemPoint*Ω which is the implementation described in [10]. We also decided to focus on *Surfpad*Π since it showed significant differences with the control conditions in Experiment 1 while *Surfpad*Ω did not show any. *Surfpad* and *Semantic Pointing* will thus refer to *Surfpad*Π and *SemPoint*Ω in this section. Lastly, as we found no significant difference between *Control* and *Control-*, we decided to focus on *Control* which corresponds to the default state of the STIMTAC.

Apparatus and Task

We used the exact same apparatus as in Experiment 1 and 2. The task was also the same except for the presence of distractors evenly spaced between the two opposite clickable targets (Figure 7).

Participants

Twelve unpaid volunteers with a mean age of 28 ($SD = 9$) served in the experiment (10 male and 2 female, 9 right-handed and 3 left-handed).

[3]Friction depends on control input nonlinearly for variable friction devices. In the case of the STIMTAC however, the non-linearity is negligible. The difference between the integrals of *Surfpad*Π and *Surfpad*Ω is below 1%.

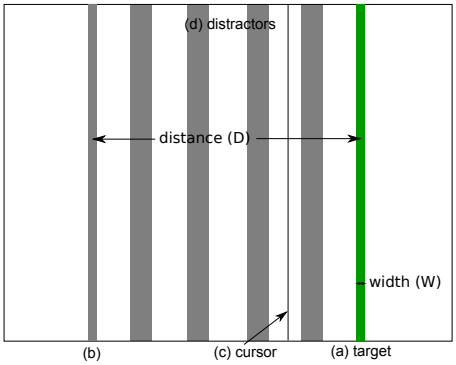


Figure 7. Experimental display. Targets were rendered as solid vertical bars equidistant from the center of the display in opposite directions along the horizontal axis. The target to be selected was colored green (a) and the last target gray (b). The cursor was represented by a one-pixel-thick vertical black line (c). Distractors (d) were evenly spaced between the targets (a) and (b) and were also colored gray.

Design

A repeated measures within-subjects design was used. The independent variables were TECHNIQUE, target width (WIDTH) and distractor density (DENSITY). TECHNIQUE was evaluated with three levels (*Control*, *Semantic Pointing*, *Surfpad*), WIDTH with two levels (W_L = 4.136 mm = 16 pixels, W_S = 1.034 mm = 4 pixels) and DENSITY with 6 levels (0, 1, 2, 4, 8, 12). The distractors were evenly spaced between the extremum targets with a size equal to W_L across all conditions. We used this width and these densities for distractors as they are representative of buttons size and densities in toolbars or menus. The target distance was kept constant to 100 mm to allow evaluating the different distractor densities while keeping a reasonably small amount of clutching.

Participants had a few minutes to get used to the device in the *Control* condition before starting the experiment. They then completed four successive BLOCKS of trials for each TECHNIQUE. Each BLOCK consisted of 36 trials: 3 repetitions of the 6 DENSITY × 2 WIDTH combinations. The WIDTH was presented in descending order and the DENSITY in ascending order. The presentation order of TECHNIQUE was counterbalanced across participants using a Latin Square design. Participants were encouraged to take a break after every 6 trials. The experiment lasted approximately 35 minutes.

In summary, the experimental design was: 12 participants × 3 TECHNIQUE × 4 BLOCKS × 2 WIDTH × 6 DENSITY × 3 trials = $5,184$ total trials

RESULTS

The dependent variables were the error rate, the movement time, the clutch time and the overshooting distance. They were computed the same way as in Experiment 1.

Error Rate

A repeated measures ANOVA showed a significant effect of WIDTH ($F_{1,11}$=96.3, $p<0.001$) on error rate with significant difference between W_S (10.2%) and W_L (1.9%) and an overall error rate of 6.1%.

Movement Time

Targets marked as errors were removed from the timing analysis. Trials at least three standard deviations away from the mean for each condition were considered as outliers and also removed from the data analysis (1.5% of the trials).

A repeated measures ANOVA showed that the order of presentation of TECHNIQUE had no significant effect or interaction on movement time, indicating that a within-participants design was appropriate. We also found no significant effect or interaction for BLOCK indicating there was no presence of a learning effect. As predicted by Fitts' law, a repeated measures ANOVA found a significant effect of WIDTH ($F_{1,11}$=199.4, $p<0.001$) on movement time with the smaller width increasing the movement time.

There was a significant main effect for TECHNIQUE ($F_{2,22}$ =119.1, $p<0.001$) and DENSITY ($F_{5,55}$=67.8, $p<0.001$) and a significant TECHNIQUE×DENSITY interaction ($F_{10,110}$=92.6, $p<0.001$) on movement time (Figure 8). Pairwise comparison showed significant differences ($p<0.001$) between all techniques: 2.1s for *Control*, 2.9s for *Semantic Pointing*, and 1.9s for *Surfpad*. It shows that *Surfpad* improves performance by 9.5% compared to *Control* and 52.6% compared to *Semantic Pointing*. *Semantic Pointing* deteriorates performance by 38.1% compared to *Control*.

Subsequent pairwise comparison for the significant TECHNIQUE × DENSITY interaction showed that the degradation of performance for *Semantic Pointing* increased with DENSITY. No significant difference between techniques was found for density 0, but we found significant differences ($p<0.04$) between *Control* and *Surfpad* for densities greater than 1. Significant differences ($p<0.03$) were found between DENSITY levels for *Semantic Pointing* except between 0 and 1 ($p=0.16$), and 2 and 4 ($p=0.07$). No significant difference was found between DENSITY levels for *Control* and *Surfpad*.

Clutch Time

A repeated measures ANOVA showed a significant main effect of TECHNIQUE ($F_{2,22}$=64.4, $p<0.001$) and DENSITY ($F_{5,55}$=164.4, $p<0.001$) and a significant TECHNIQUE×DENSITY interaction ($F_{10,110}$=121.9, $p<0.001$) on clutch time. Pairwise comparison showed significant differences ($p<0.001$) between

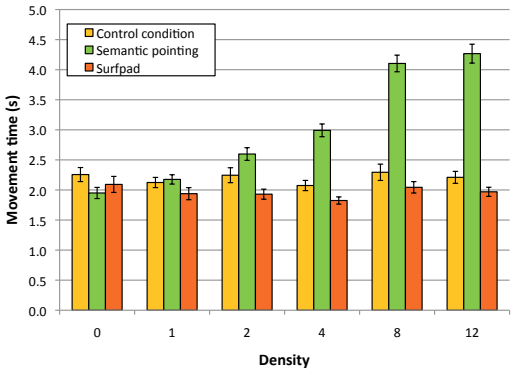


Figure 8. Mean movement time for TECHNIQUE and DENSITY. Error bars represent 95% confidence interval.

Semantic Pointing (0.59s), *Surfpad* (0.12s) and *Control* (0.15s). No significant difference was found across DENSITY for *Surfpad* or *Control*, but we found significant differences ($p<0.003$) between the following density groups for *Semantic Pointing*: (0,1): 0.28s, (2): 0.42s, (4): 0.54s, (8,12): 1.02s (Figure 9).

Overshooting Distance

We define overshooting as the distance traveled past the extend of the target. A repeated measures ANOVA showed no significant main effect or interaction on the overshooting distance. The mean overshooting distance was equal to 1.7 mm (SD=4.8 mm) and the 90^{th} percentile was equal to 4.8 mm. The 90^{th} percentile for overshooting was equal to 3.7 mm in Experiment 1 and 3.4 mm in Experiment 2. Considering this relatively small overshooting distance, users' strategy to acquire the target was probably not to overshoot the target and then correct to select it.

User Feedback

Most of participants comments on *Semantic Pointing* concerned the clutching required to move the pointer, especially when the number of distractors becomes important. Participants did not spontaneously comment on distractors for *Surfpad*. After debriefing, they explained they did not feel disrupted in their movement by the tactile feedback on distractors. Participants were also asked which technique they would use. Eleven chose *Surfpad* and one chose *Control*.

DISCUSSION

We compared *Surfpad* to *Control* and *Semantic Pointing* in the same conditions as in the first experiment with additional control on the density of distractors. Our results show that *Surfpad* significantly improves the movement time by 9.5% compared to *Control*, independently of the density of distractors. In contrast, we showed that *Semantic Pointing* significantly degrades the movement time compared to *Control* with a rate related to the density of distractors (from 22.4% for density 2 to 100% for density 12). Results show that the significant increase in clutching for *Semantic Pointing* compared to *Surfpad* and *Control* can be held responsible to its significant increase of movement time.

In the worse case scenario for *Semantic Pointing*, the distance to the target gets fully covered with distractors: the indices of difficulty in motor and display spaces become equal (considering there is no overlapping between distractors), but the distance in motor space gets multiplied by the scale factor. Such a situation typically occurs in hierarchical menus [13]. If the device operating range is set to cover the entire display surface without clutching, multiplying the distance by the scale factor in motor space will inevitably lead to clutching and a deterioration of performance.

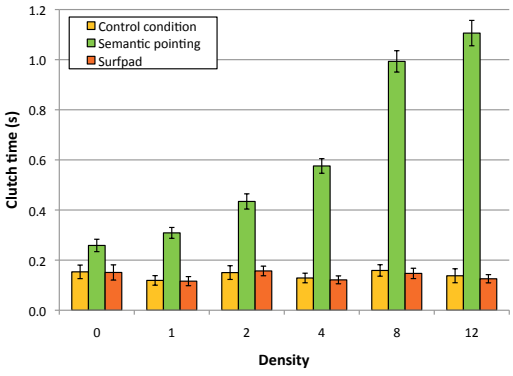


Figure 9. Mean clutch time for TECHNIQUE and DENSITY. Error bars represent 95% confidence interval.

There was no negative effect of distractors on *Surfpad* which still showed a significant improvement of 9.5% on movement time compared to *Control* in their presence. This reinforces our belief that the *Surfpad*Π implementation mainly provides information feedback and little or no mechanical effect. Participants did not make any negative comment on the tactile feedback associated to distractors. This makes *Surfpad* a good alternative to *Semantic Pointing* and probably target-aware CD gain adaptation in general, especially for limited workspaces where clutching is likely to occur in presence of distractors.

CONCLUSION AND DIRECTIONS FOR FUTURE WORK

We presented *Surfpad*, a new pointing facilitation technique based on STIMTAC, a tactile touchpad that supports friction reduction. *Surfpad* preserves the nominal coefficient of friction of the touchpad when the cursor is on targets but reduces it in all other places. We reported on three experiments comparing it to *Semantic Pointing* and constant CD gain. Our results show that *Surfpad* leads to a performance improvement close to 9% compared to unassisted pointing on small targets without distractors. It is also robust to high distractor densities, keeping an average performance improvement of nearly 10% whereas the performance of *Semantic Pointing* can degrade up to 100% due to increased clutching caused by distractor expansion in motor space. Our results also show that *Surfpad* needs to be implemented using a step function (*Surfpad*Π) to improve performance.This implementation provides a sudden reduction in the amount of friction when the pointer crosses the target border which, we hypothesize, results in an information feedback that helps users reduce the approaching and stopping times.

Surfpad's robustness to distractors is particularly novel. This characteristic has exciting implications since it no longer requires the careful determination of targets to enable a pointing facilitation technique. Our prototype STIMTAC device can be easily carried for demonstrations, but it is still too large to incorporate in a mobile computing device such as a laptop. New prototypes are being developed which use more compact sensing techniques. Once the size is reduced, its low power consumption ($0.5\ W$) makes it feasible to use in place of a conventional laptop touchpad. However, the tactile feedback it provides is intrinsically mono-touch. The extension to multi-touch will be addressed as future work.

Acknowledgments

This work was supported by the ANR project n°ANR-09-CORD-013 "InSTInCT".

REFERENCES

1. D. Ahlström, M. Hitz, and G. Leitner. An evaluation of sticky and force enhanced targets in multi target situations. In *Proc. of NordiCHI'06*, 58–67. ACM, 2006.

2. M. Akamatsu and I. S. MacKenzie. Movement characteristics using a mouse with tactile and force feedback. *IJHCS*, 45(4):483–493, 1996.
3. M. Akamatsu, I. S. MacKenzie, and T. Hasbrouc. A comparison of tactile, auditory, and visual feedback in a pointing task using a mouse-type device. *Ergonomics*, 38(4):816–827, 1995.
4. T. Asano, E. Sharlin, Y. Kitamura, K. Takashima, and F. Kishino. Predictive interaction using the delphian desktop. In *Proc. of UIST'05*, 133–141. ACM, 2005.
5. R. Balakrishnan. "Beating" Fitts' law: virtual enhancements for pointing facilitation. *IJHCS*, 61(6):857–874, 2004.
6. O. Bau, I. Poupyrev, A. Israr, and C. Harrison. Teslatouch: electrovibration for touch surfaces. In *Proc. of UIST'10*, 283–292. ACM, 2010.
7. P. Baudisch, E. Cutrell, M. Czerwinski, D. C. Robbins, P. Tandler, B. B. Bederson, and A. Zierlinger. Drag-and-pop and drag-and-pick: techniques for accessing remote screen content on touch- and pen-operated systems. In *Proc. of Interact'03*, 57–64. IOS Press, 2003.
8. P. Baudisch, A. Zotov, E. Cutrell, and K. Hinckley. Starburst: a target expansion algorithm for non-uniform target distributions. In *Proc. of AVI'08*, 129–137. ACM.
9. M. Biet, F. Giraud, and B. Semail. Squeeze film effect for the design of an ultrasonic tactile plate. *IEEE Transactions on Ultrasonic, Ferroelectric and Frequency Control*, 54(12):2678–2688, 2007.
10. R. Blanch, Y. Guiard, and M. Beaudouin-Lafon. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proc. of CHI'04*, 519–526. ACM, 2004.
11. C. S. Campbell, S. Zhai, K. W. May, and P. P. Maglio. What you feel must be what you see: Adding tactile feedback to the trackpoint. In *Proc. of Interact'99*, 383–390. IOS Press, 1999.
12. G. Casiez, D. Vogel, R. Balakrishnan, and A. Cockburn. The impact of control-display gain on user performance in pointing tasks. *Human-Computer Interaction, Taylor and Francis*, 23(3):215–250, 2008.
13. A. Cockburn and S. Brewster. Multimodal feedback for the acquisition of small targets. *Ergonomics*, 48(9):1129–1150, 2005.
14. A. Cockburn and A. Firth. Improving the acquisition of small targets. In *Proc. of HCI'03*, 77–80. BCS, 2003.
15. J. T. Dennerlein, D. B. Martin, and C. Hasser. Force-feedback improves performance for steering and combined steering-targeting tasks. In *Proc. of CHI'00*, 423–429. ACM, 2000.
16. J. T. Dennerlein and M. C. Yang. Haptic force-feedback devices for the office computer: Performance and musculoskeletal loading issues. *Human Factors*, 43(2):278–286, 2001.
17. C. Forlines and R. Balakrishnan. Evaluating tactile feedback and direct vs. indirect stylus input in pointing and crossing selection tasks. In *Proc. of CHI'08*, 1563–1572. ACM, 2008.
18. T. Grossman and R. Balakrishnan. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proc. of CHI'05*, 281–290. ACM, 2005.
19. Y. Guiard, R. Blanch, and M. Beaudouin-Lafon. Object pointing: a complement to bitmap pointing in GUIs. In *Proc. of GI'04*, 9–16. CHCCS, 2004.
20. C. Harrison and S. E. Hudson. Texture displays: a passive approach to tactile presentation. In *Proc. of CHI'09*, 2261–2264. ACM, 2009.
21. A. Hurst, J. Mankoff, A. K. Dey, and S. E. Hudson. Dirty desktops: using a patina of magnetic mouse dust to make common interactor targets easier to select. In *Proc. of UIST '07*, 183–186. ACM, 2007.
22. F. Hwang, S. Keates, P. M. Langdon, and P. J. Clarkson. Multiple haptic targets for motion-impaired computer users. In *Proc. of CHI'03*, 41–48. ACM, 2003.
23. P. Kabbash and W. Buxton. The "prince" technique: Fitts' law and selection using area cursors. In *Proc. of CHI'95*, 273–279. ACM/Addison-Wesley, 1995.
24. D. V. Keyson. Dynamic cursor gain and tactual feedback in the capture of cursor movements. *Ergonomics*, 40(12):1287 – 1298, 1997.
25. M. Kobayashi and T. Igarashi. Ninja cursors: using multiple cursors to assist target acquisition on large screens. In *Proc. of CHI'08*, 949–958. ACM, 2008.
26. A. Lécuyer, J.-M. Burkhardt, and L. Etienne. Feeling bumps and holes without a haptic interface: the perception of pseudo-haptic textures. In *Proc. of CHI'04*, 239–246. ACM, 2004.
27. M. J. McGuffin and R. Balakrishnan. Fitts' law and expanding targets: experimental studies and designs for user interfaces. *ACM ToCHI*, 12(4):388–422, 2005.
28. I. Oakley, M. R. McGee, S. Brewster, and P. Gray. Putting the feel in 'look and feel'. In *Proc. of CHI'00*, 415–422. ACM, 2000.
29. R. W. Soukoreff and I. S. MacKenzie. Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI. *IJHCS*, 61(6):751–789, 2004.
30. J. B. F. van Erp, K.-U. Kyung, S. Kassner, J. Carter, S. A. Brewster, G. Weber, and I. Andrew. Setting the standards for haptic and tactile interactions: ISO's work. In *Proc. of EuroHaptics'10*, 353–358. Springer, 2010.
31. K. van Mensvoort. What you see is what you feel: exploiting the dominance of the visual over the haptic domain to simulate force-feedback with cursor displacements. In *Proc. of DIS'02*, 345–348. ACM, 2002.
32. T. Watanabe and S. Fukui. A method for controlling tactile sensation of surface roughness using ultrasonic vibration. In *Proc. of ICRA'95*, 1134–1139. IEEE, 1995.
33. L. Winfield, J. Glassmire, J. E. Colgate, and M. Peshkin. T-pad: Tactile pattern display through variable friction reduction. In *Proc. of World Haptics Conf.*, 421–426. IEEE, 2007.
34. J. O. Wobbrock, J. Fogarty, S.-Y. Liu, S. Kimuro, and S. Harada. The angle mouse: target-agnostic dynamic gain adjustment based on angular deviation. In *Proc. of CHI'09*, 1401–1410. ACM, 2009.
35. A. Worden, N. Walker, K. Bharat, and S. Hudson. Making computers easier for older adults to use: area cursors and sticky icons. In *Proc. of CHI'97*, 266–271. ACM, 1997.

Integrality and Separability of Multitouch Interaction Techniques in 3D Manipulation Tasks

Anthony Martinet, Géry Casiez, and Laurent Grisoni

Abstract—Multitouch displays represent a promising technology for the display and manipulation of data. While the manipulation of 2D data has been widely explored, 3D manipulation with multitouch displays remains largely unexplored. Based on an analysis of the integration and separation of degrees of freedom, we propose a taxonomy for 3D manipulation techniques with multitouch displays. Using that taxonomy, we introduce *Depth-Separated Screen-Space* (*DS3*), a new 3D manipulation technique based on the separation of translation and rotation. In a controlled experiment, we compared *DS3* with *Sticky Tools* and *Screen-Space*. Results show that separating the control of translation and rotation significantly affects performance for 3D manipulation, with *DS3* performing faster than the two other techniques.

Index Terms—Multitouch displays, 3D manipulation task, direct manipulation, DOF separation.

1 Introduction

Three-dimensional (3D) manipulation is a challenge for 3D interface designers since it involves the control of six Degrees Of Freedom (DOF) : three for position (i.e., translation along x-, y-, and z-axes) and three for orientation (i.e., rotation around x-, y-, and z-axes). Using a mouse with a current desktop interface, rotating 3D objects can take from 10 to 30 seconds [1], much slower than real object manipulation which takes between 1 and 2 seconds [2].

Compared to the mouse, multitouch displays provide extra input bandwidth through multiple contact points and enable direct manipulation allowing users to directly *touch* data [3]. The Rotate-Scale-Translation gesture (RST) for manipulating 2D data is a typical example of this type of interaction paradigm [4]. While 2D manipulation on multitouch displays has been widely explored, 3D manipulation has not. This may be explained by the mapping difficulty of the inherently 2D input contact points to the 3D attributes which need to be controlled.

Jacob et al. [5] studied the impact of the input device control structure and the perceptual structure on task performance, with a focus on whether the structures were integral or separable. They found a strong relationship between the two structures with better performance when both match. While it has been shown that human fingers have separable DOF [6], 3D manipulation is inherently an integral task [7]. The thumb, index and middle fingers can be moved separately from one another while users perceive the attributes of 3D objects (position and orientation) as a whole. This mismatch between the separable input structure of multitouch devices and integral nature of a 3D manipulation task raises the question of how to optimize the mapping between the two structures.

While techniques like *Sticky Tools* [8] propose a way to separate the DOF of a 3D manipulation task, other techniques like *Screen-Space* [9] present a method to integrate them. However, the lack of user study makes it difficult to compare the two approaches.

After presenting the related work on DOF integration and separation, the existing metrics to measure DOF coordination, and the 3D manipulation techniques for multitouch displays, we introduce a taxonomy to compare 3D manipulation techniques for multitouch displays. Then, we introduce a new technique called *Depth-Separated Screen-Space* (*DS3*) based on the clear separation between translation and rotation, and present the results of a controlled experiment comparing this technique with *Sticky Tools* [8] and *Screen-Space* [9]. Finally, in the discussion we address the question of controlling a task with integrated DOF using a separable multitouch input device.

2 Related Work

2.1 Integration and Separation of DOF

According to the theory of the perceptual structure of visual information by Garner [7], a multidimensional object can be characterized by its attributes in two categories: integral structure and separable structure. Visual information has an integral structure if its attributes can be perceptually combined to form a unitary whole. If visual object attributes show perceptually distinct and identifiable dimensions, they are separable. According to this definition, the orientation and the position of a 3D object are two integral attributes, making 3D manipulation an integral task.

• *The authors are with the Bâtiment IRCICA, Parc scientifique de la Haute Borne 50, Avenue Halley, 59658 Villeneuve d'Ascq cedex, France. E-mail: {anthony.martinet, gery.casiez, laurent.grisoni}@lifl.fr.*

Manuscript received 3 Feb. 2011; revised 28 June 2011; accepted 6 July 2011; published online 21 July 2011.
Recommended for acceptance by T. Komura, Q. Peng, G. Baciu, R. Lau, and M. Lin.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCGSI-2011-02-023.
Digital Object Identifier no. 10.1109/TVCG.2011.129.

Jacob et al. [5] extended Garner's notion of integral and separable structure to interactive tasks by observing that manipulating a graphic object is simply the modification of the values of its attributes. They also extended integral and separable structure to describe the attributes of an input device, based on whether it is natural to move diagonally across all dimensions. With an integral device, the movement is in euclidean space and cuts across all the dimensions of control. A separable device constrains movement along one dimension at a time. They conducted an experiment in which participants performed two tasks that had different perceptual structures, using two input devices with correspondingly different control structures: an integral three-dimensional tracker and a separable mouse. Their results support their hypothesis: human performance increases when the perceptual structure of the task matches the control structure of the device. They concluded that the interplay between task and device was more important in determining performance than either task or device alone.

Wang et al. [2] extended this theory to extrinsic (i.e., orientation and position) properties of an object being manipulated by the human hand. They pointed out that human visual pathways (i.e., human processing chains) responsible for object perception are separated from those guiding the action. They ran an experiment that asked participants to dock a real wood cube using different visual feedback conditions. They reported that users had little difficulty in simultaneous control of object translation and orientation.

Considering an orientation task only, Veit et al. [10] studied the integration of DOF. They conducted an experiment in which users had to orient 3D objects using two interaction techniques, one integrating and the other separating the DOF of the orientation task. The results suggest that the simultaneous manipulation of all the DOF does not necessary lead to the best performance, leading to conclusions opposite to those of Jacob et al.

Regarding 2D manipulation with multitouch displays, Nacenta et al. [11] addressed the issue of manipulating a given subset of DOF. When using multitouch gestures, performing a subset of available operations may be difficult for users. For example, it can be hard to only scale and translate an object (without rotating it) because the object will also react to small variations of the angle between the contact points. They introduced an interaction technique that allows users to select a subset of DOF, reducing unwanted manipulation without negatively affecting performance. Separating the control of DOF like this improved the user's expectations.

2.2 Metrics for Quantifying DOF Coordination

While the concepts of integration and separation of DOF are useful to describe the attributes of tasks and input devices from a theoretical point of view, some metrics have been introduced to actually measure DOF coordination.

To measure the degree of DOF coordination for translation, Jacob et al. [5] first segmented the movement trajectory into equal units of time. Using distance thresholds, each time segment was then classified as Euclidian if the corresponding trajectory showed movement in more than one dimension or city-block if the trajectory only occurred along a single dimension. The degree of coordination was then defined as the ratio of euclidean to city-block movements. However, this metric is not defined when the number of city-block movements is equal to zero. The metric also does not provide any baseline for perfect coordination nor does it distinguish between movements which contribute toward the goal and movements which do not.

Zhai and Milgram proposed another metric for measuring DOF coordination in docking tasks [12]. The translation coordination is computed as the ratio of the lengths of the shortest path to the actual path. The rotation coordination is computed in a similar way using the ratio of the initial rotation mismatch to the amount of actual rotation. A ratio close to one corresponds to a high degree of coordination between the DOF of the input device while a ratio close to zero corresponds to a poor coordination between them. The translation and rotation coordination values provide a global measure of coordination independent of the temporal profile of the movement trajectory. As a result, it is not possible to know when the rotation and translation occurred as it is not possible to determine if translation and rotation occurred simultaneously. To address this, Zhai simply used a 2D visual representation [12] of the translation and rotation over time, where the x-axis represents the translation coordination and the y-axis represents the rotation coordination. We will refer to this metric as the *translation-rotation* coordination ratio. However, this visual representation does not provide any numerical value to represent the simultaneous coordination of the translation and rotation DOF.

To solve this problem, Masliah and Milgram [13] introduced the ***m***-metric. The goal of the ***m***-metric is to compare a tuple of n DOF (where $n \geq 2$) in a docking task that require the manipulation of m DOF ($n < m$). The metric measures the degree of simultaneous error reduction occurring in multiple DOFs. It takes into account both the simultaneity (i.e., simultaneous manipulation of multiple DOF) and the efficiency of control (i.e., manipulation of DOF that reduce the error) across the DOF. Error is defined as the distance between the current position and the target position. In other words, the ***m***-metric can be used to understand precisely which subsets of DOF are manipulated during a docking trial.

More recently, Veit et al. [14] introduced the *Number of Degrees of Freedom Combined* (NDC), a metric providing the number of DOF simultaneously manipulated during a multi-DOF docking task. They tested the metric on an orientation task and explained how it could be generalized for all six DOF tasks.

To sum up, Zhai's metric provides a global measure of the coordination while the NDC captures how many DOF are manipulated over time. Finally the ***m***-metric details which subset of DOF are manipulated together. From a qualitative point-of-view, the visual representation introduced with the *translation-rotation* coordination ratio helps to understand users' strategies to reach the target position and orientation.

2.3 3D Manipulation with Multitouch Displays

2.3.1 Limited 3D Manipulation

Using a standard vision-based tabletop, Wilson et al. [15] created a physics-enabled 3D environment with multitouch input manipulating the DOF of the task in an integral way. Their technique is able to model both multiple contact

points and more sophisticated shape information, such as the entire hand. They showed that their technique can be used to add real-world dynamics to interactive surfaces. While the underlying physical simulation can provide a number of convincing effects during the interaction (inertia, collision), integrating all the DOF in such an environment prevents users from lifting object (i.e., move the object along the z-axis).

Hilliges et al. [16] used depth-sensing camera to solve this issue so that users can "pickup" an object and manipulate it above the surface. This supports Jacob's conclusions since extra input information improves the match between the control structure and the task perceived structure, both integral in this case. However, those techniques require additional hardware making the compatibility with existing multitouch displays difficult.

Martinet et al. [17] proposed two techniques for 3D positioning. One technique, the Z-technique, presented 3D data in full screen while the other technique split the screen in four viewports. They conducted a docking task experiment, but were not able to draw conclusions on performance. From a qualitative point a view, they reported that users preferred the full-screen technique.

2.3.2 Full 3D Manipulation

Hancock et al. [18] presented one, two, and three touch input interaction techniques to manipulate 3D objects on multitouch displays. With three-touch interaction, users can perform simultaneous translation and rotation on the surface of the table. Depth positioning is activated as an option, by measuring the distance between two fingers. The three-touch technique, called *Shallow-Depth*, was found to be faster and more accurate and also preferred by their users. Nevertheless, the 3D task used in the experiment only required the manipulation of five DOF. To control all six DOF required for 3D manipulation, they introduced the *Sticky Tools* technique [8], allowing users to manipulate an object using three fingers. Each finger separately controls DOF that are integrated together. While the authors discussed the use of such a technique in a more general manner, the choice of DOF to integrate together is not addressed and no user study was carried out to measure the efficiency of *Sticky Tools*.

Reisman et al. [9] introduced a method to handle 3D manipulation in a direct way, integrating all the DOF needed to perform such an operation. Highlighting the fact that RST has become the *de facto* standard technique to handle 2D objects, they presented a technique to extend RST into 3D. The tool works by solving constraints fixed by users' fingers. A constraint solver minimizes the error between the screen-space projection of contact points (i.e., finger positions on the 2D screen) and their corresponding screen-space target positions (i.e., the 3D points *touched* by fingers). The paper discusses the use of the constraint solver and provides examples to use this tool to design interaction techniques—but no formal evaluation was performed.

Recently, Cohé et al. [19] introduced tBox, a technique that combines 3D widgets and direct manipulation. tBox separates all the DOF of the manipulation using the bounding box of an object. Scaling is also possible via this technique. However, the technique was not formally evaluated nor compared.

3 A Taxonomy of 3D Manipulation Techniques with Multitouch Displays

The 3D manipulation techniques mentioned above control different subsets of DOF depending on the number of fingers in contact with the surface. In addition a finger can be considered either *direct* or *indirect* depending on the euclidian physical distance between the finger position and the projection on screen of the virtual object being manipulated. When this distance is equal or close to zero, the finger is *direct* and turns *indirect* when this distance becomes greater. The number of fingers used for the interaction and the directness of each finger is referenced below as a mode.

To help comparing existing manipulation techniques, we chose to adapt the taxonomy introduced by Card et al. [20]. We wanted to represent the relationship between the number of fingers, their directness (whether *direct* or *indirect*) and the corresponding DOF controlled in the task. We also wanted to represent whether the DOF of the task is controlled in an integral or separable way. The DOF controlled in the manipulation task are represented in a cartesian direct framework where the x-axis belongs to the screen plane and is oriented toward the right and the z-axis is orthogonal to the screen and points toward the user. $Tx, Ty,$ and Tz represent the translations along the corresponding axis; Rx, Ry, and Rz the rotations around the corresponding axis. This taxonomy only takes into account the information available through the inputs provided by the current technology: the number of contact points (e.g., the number of fingers) and the directness of each finger. This taxonomy could be enriched by specifying the name and associated hand for each finger in contact, and also the order in which they have to enter in contact with the surface.

In Fig. 2, we represent an illustration of the use of the taxonomy with *Sticky Tools*. Each line represents a mode for the technique, annotated with the number of fingers associated. Indirect fingers are represented with an "i" in the corresponding circles, whereas direct fingers are left blank. Circles connected together with a single line represent the DOF of the task controlled in an integral way. Groups of circles disconnected represent the DOF of the task controlled in a separable way.

For the *Sticky Tools* technique represented in Fig. 2, the mode $1d$ represents the first finger in contact with the object to manipulate, which controls the object translation along the screen plane in a direct and integral way. When a second finger enters in contact with the same object (mode $2d$), translation and rotation around the z-axis are now possible in a direct and integral way, in addition to the DOF controlled by the first finger (i.e., each finger movement can now change four DOF at once). The second finger can also be used in an indirect way (mode $1d + 1i$) to control two DOF in rotation in an integral way but separately from the DOF controlled by the first finger. Last, the mode $2d + 1i$ shows the combination of the previous modes to control the six degrees of freedom at the same time, but with two DOF in rotation being controlled separately.

4 Screen-Space Technique

As mentioned previously, Reisman et al. [9] introduced a method to perform 3D manipulation with multitouch

displays. We refer to this technique as *Screen-Space*. This method uses a constrain solver to integrate all the DOF of the manipulation. The solver takes user's fingers as input and returns the updated values of the DOF. The calculation is a least-squares minimization of the error between the screen-space projection of contact points and their corresponding screen-space target positions. A simplified version of the algorithm can be described as follows:

1. When a finger touches a 3D object projected on screen:
   a. Record the 2D location of the finger on screen (point $F2d_1$).
   b. Record the 3D point corresponding to the ray-casting of the finger 2D position into the 3D scene (point $P3d_1$).
2. When a finger moves:
   a. Record the new position (point $F2d_2$).
   b. Use the constrain solver to adjust the position and orientation of the 3D object so that when $F2d_2$ is casted into the scene, it points to $P3d_1$.

The goal of the algorithm is to match user's fingers to 3D points and keep these 3D points *stuck under* user's fingers when they move. When it comes to scale and rotate a 2D picture using multitouch input, it is exactly the same process but instead of matching 2D points (i.e., fingers) with 3D points (i.e., 3D object), 2D points (i.e., fingers) are matched with 2D points (i.e., 2D picture).

To control the six DOF required for 3D manipulation, at least three fingers are required, as a single finger can only control two DOF at best (we consider here only the x, y positions of fingers). With less than three fingers, the interface designer has to choose the DOF controlled in the task. With one finger, the natural choice is to control the translation of the object in the camera plane, as illustrated with mode $1d$ in Fig. 3. With two fingers, up to four DOF among the six can be controlled. Reisman et al. do not recommend any particular mapping. The mode $2d$ presents one possible mapping chosen after a pilot study we discuss later.

5 Introducing *DS3*

According to Garner [7], the perceptual structure of 3D manipulation consists of six integrated DOF. Jacob et al. [5] recommend matching the perceptual structure of the task with the control structure of the input device. Strictly following these two recommendations leads to using only input devices with six integrated DOF such as 3D mice or 3D wands, and to interact with the whole hand instead of only interacting with fingers.

Considering the separable structure of fingers [6], it appears impossible to exactly match the perceptual structure of the task with the control structure of multitouch displays. The previous work above shows there is no clear answer to this problem. On the one hand *Screen-Space* proposes to control the six degrees of freedom in an integral way and on the other hand *Sticky Tools* proposes a separation between the degrees of freedom. As these two techniques were not evaluated nor compared, it is difficult to know which approach is the best. If the DOF separation appears better, it also addresses the question of the best way to separate DOF.

During an informal evaluation of *Sticky Tools*, we observed that the integral control of translation and rotation (modes $2d$ and $2d + 1i$ in Fig. 2) is indeed difficult. When DOF are controlled separately, our hypothesis is that a clear separation of translation and rotation improves user efficiency. As a consequence we designed a new technique clearly separating the control of rotation from the control of translation. We called this technique *DS3*. *DS3* combines the Z-technique [17] used to control the position; and the constraint solver described by Reisman et al. [9] to control the orientation.

With one direct finger, objects can be translated along the screen plane (mode $1d$ in Fig. 4). Depth translation is performed in an indirect way, with a second, indirect finger. When this finger is in contact with the surface we measure its relative motion on the surface and use backward and forward movement to control the depth position. Forward movement moves the object away from the user view and backward movement moves it closer to the user's view.

With at least two direct fingers, users can control only the orientation of the object in an integral way using the constrain solver previously described. Finger positions are used as inputs for the constrain solver, which provides us the appropriate rotation to perform.[1]

The number of fingers directly in contact with the object (one versus two or more) provides a clear separation between translation and rotation. In addition, when rotating the object, we also allow the manipulation of the object depth (i.e., translation along z-axis) with an indirect finger, as previously described. This is not a breach of the separation of position and orientation since depth position is handled with a additional separated finger.

6 Pilot Experiment

This pilot experiment was designed to pretest *Sticky Tools*, *Screen-Space*, and *DS3* on a real task. We also wanted to examine different mappings for the mode $2d$ of *Screen-Space*. In addition, this allowed us to tune technique parameters.

With *Screen-Space*, we can control up to four DOF with two fingers. To remove unintended translation as mentioned by Hancock et al. [18], we decided to remove the two DOF which were mapped to the one finger mode, leaving us with the four remaining DOF. Since we believe that separating rotation can improve efficiency, we decided to use two fingers mode for controlling rotation DOF only.

6.1 Apparatus

The experiment was conducted on an Immersion iLight[2] touch table based on the Diffused Illumination technique. The surface is a $100\text{ cm} \times 70$ cm (42 inches) monoscopic display positioned 105 cm above the floor. The video projector under the table was set at 60 Hz with a $1{,}400 \times 1{,}050$ pixel resolution giving a pixel density of 14 pixels per cm (36 DPI). A camera running at 120 Hz with a

1. http://www.youtube.com/watch?v=DnHvpyjgYik.
2. http://www.immersion.fr.

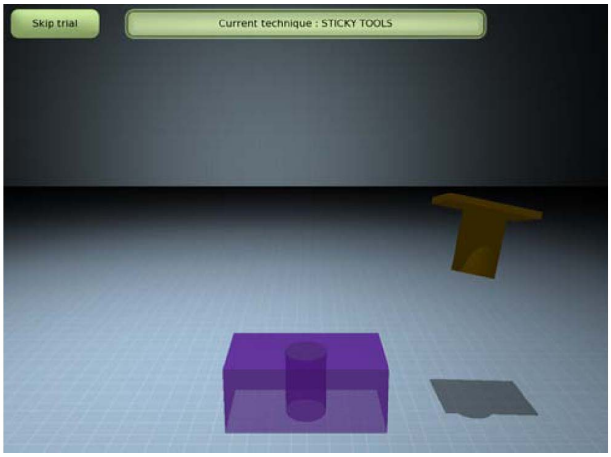


Fig. 1. Screen capture of the peg-in-hole task.

		Translation			Rotation		
	Mode	Tx	Ty	Tz	Rx	Ry	Rz
Sticky Tools	1*d*	○	○				
	2*d*	○	○	○			○
	1*d* + 1*i*	○	○		*i*	*i*	
	2*d* + 1*i*	○	○	○	*i*	*i*	○

Fig. 2. Description of the *Sticky Tools* technique using the taxonomy.

640×480 pixel resolution is positioned under the surface to capture finger movements. This gives a maximum resolution of 6.4 dots per cm (16.25 DPI) for finger tracking. We used the iLight framework version 1.6 for fingers detection and tracking. Finger data were then sent using TUIO messages[3] to a custom built 3D application based on the Ogre3D framework.[4] The source code of the Ogre3D listener implementing the different interaction techniques is available on `github`.[5]

6.2 Task and Participants

The task is a 3D peg-in-hole task similar to the one described by Unger et al. [21] (Fig. 1), but without collision detection enabled. Each experimental trial began after the previous peg was successfully positioned and ended with the successful positioning of the current peg. Participants were asked to position and orientate as quickly as possible a peg into a hole located at the middle of a 3D rectangular parallelepiped. The latter was made transparent to ease the fine positioning of the peg. The peg was made of a rectangular base on which a cylindrical shape was extruded. When both position and orientation were under a given threshold, the peg turned green to indicate it was successfully located. The trial was considered as fully completed when the peg stayed at the correct position for 0.8 s. The peg then moved to another position, selected randomly on a hemisphere (i.e., the center of the hemisphere was the center of the hole and the radius was defined to fit within the display space). The hole remained at the same place. In addition to perspective and occlusion, we also added a ground with shadows projection to improve depth perception. The virtual camera remained fixed during the whole experiment. We controlled for the presence of depth (whether translation along z-axis was required), the combination of axes required for the rotation and the amount of rotation required.

3. http://tuio.org.
4. http://www.ogre3d.org.
5. https://gist.github.com/764989.

		Translation			Rotation		
	Mode	Tx	Ty	Tz	Rx	Ry	Rz
Screen-Space	1*d*	○	○				
	2*d*	Added after pilot study		○	○	○	○
	≥ 3*d*	○	○	○	○	○	○

Fig. 3. Description of the *Screen-Space* technique using the taxonomy.

		Translation			Rotation		
	Mode	Tx	Ty	Tz	Rx	Ry	Rz
DS3	1*d*	○	○				
	1*d* + 1*i*	○	○	*i*	Z-Technique		
	≥ 2*d*				○	○	○
	≥ 2*d* + 1*i*			*i*	○	○	○

Fig. 4. Description of the *DS3* technique using the taxonomy.

Six males with a mean age of 25 participated. Participants had variable experience with virtual reality and multitouch displays. Two were experts, another had some experience, and the others were novices.

6.3 First Results and Discussion

Task completion time is defined as the time it takes to successfully position the current peg into the destination from the last successfully positioned peg. Results exhibited a strong learning effect indicating we should run more than three blocks in the final study.

The majority of users feedback concerned *Screen-Space*. They all complained about depth translation: they were frustrated by being unable to control the depth position with two fingers. They reported they were used to the *pinch-to-zoom* gesture available on commercial products and that handling depth translation with three fingers was tedious. As our mapping controlled orientation only (i.e., three DOF), one extra DOF remained available for the constraint solver (i.e., two fingers allow to control up to four DOF). We therefore decided to change our two fingers mapping and we added the control of depth position in addition to rotation (Fig. 3).

Based on these pilot results, we decided to increase the number of blocks to five in the controlled experiment. We also changed the mapping of two fingers with *Screen-Space* to control both depth position and orientation.

7 Controlled Experiment

7.1 Goals

The main goal of the experiment is to evaluate the effect of DOF separation on multitouch displays for 3D manipulation. A second objective is to compare *Sticky Tools* and *Screen-Space* which have never been compared or evaluated.

In designing the experiment, we formulated the following hypothesis:

- H1. Based on the results of the pilot study and user feedback, we hypothesize that separating the control of translation from rotation will increase performance since users will not get confused controlling both at the same time. Thus, *DS3* should be faster.
- H2. Separating the control of translation from rotation increases coordination (in translation or rotation): if users can manipulate DOF for translation and rotation separately in an efficient way, they will be able to improve the coordination of the DOF for the translation or the rotation. *DS3* should have a better coordination.
- H3. The presence of depth translation will affect performance and coordination, especially with *Sticky Tools* and *Screen-Space* that map depth translation and rotation together, highlighting the problem pointed out by Nacenta et al. [11].

7.2 Task

The task and hardware setup were the same as in the pilot study. However, we changed the design of our application by adding a button that allowed users to skip a trial if they think it was too difficult to complete.

7.3 Participants

Ten males and two females with a mean age of 24.8 (SD 0.83) participated. Nine were right-handed and three were left-handed and all had normal or corrected to normal vision. Participants had a variable experience with virtual reality and 3D applications but this is acceptable as we are observing a lower level physical behavior. Three were frequent users of multitouch displays. Six participants were familiar with tactile devices such as a tactile mobile phone or tablet-PC, but never worked for a long time on such devices. The other participants had never used a tabletop device.

7.4 Design

A repeated measures design was used. The independent variables were TECHNIQUE, PRESENCE OF DEPTH, ROTATION LEVEL, and ROTATION AMOUNT. There were three levels for TECHNIQUE: *DS3*, *Sticky Tools*, and *Screen-Space*. The presentation order of TECHNIQUE was counterbalanced across participants. The PRESENCE OF DEPTH variable had two levels whether depth position needed to be adjusted. The two levels were NODEPTH and DEPTH. There were also two levels for ROTATION LEVEL, influencing the type of rotation to be performed: SIMPLE and COMPLEX. SIMPLE sets the rotation only around one axis (x, y, or z) which was randomly chosen. COMPLEX sets the rotation to be a random mix of x-, y-, and z-axes. There were two levels as well for ROTATION AMOUNT, changing the angle of the rotation to perform: SMALL and LARGE. For SMALL, the total amount of rotation was set to 30 and 120 degrees for LARGE.

As suggested by our pilot experiment, we added extra blocks to the experiment. Participants thus completed five successive BLOCKS of trials. Each BLOCK consisted of 16 trials:

$$\begin{aligned}&2 \text{ repetitions of } 8 \text{ PRESENCE OF DEPTH}\\&\quad \times \text{ ROTATION LEVEL} \times \text{ ROTATION AMOUNT}\end{aligned}$$

combinations. The presentation order of TECHNIQUE was counter-balanced across participants. A break was encouraged after each set of 10 trials.

Before starting the experiment with a new technique, participants had a 5 minutes training period to get accustomed to the current technique. The experiment ended with a qualitative feedback from the participants. The experiment lasted approximately 100 minutes in total.

In summary, the experimental design was:

$$\begin{aligned}&12 \text{ participants} \times 3 \text{ TECHNIQUES} \times 5 \text{ BLOCKS}\\&\quad \times 2 \text{ PRESENCE OF DEPTH} \times 2 \text{ ROTATION LEVEL}\\&\quad \times 2 \text{ ROTATION AMOUNT} \times 2 \text{ repetitions}\\&= 2{,}880 \text{ total trials.}\end{aligned}$$

8 Results

8.1 Trial Completion and Number of Touches

Trial completion is defined as the ratio of the number of completed trials to the total number of trials. Trials that were skipped during the experiment were marked as invalid and removed from subsequent analysis.

Repeated measures analyses of variance (ANOVA) found a significant main effect for TECHNIQUE ($F_{2,22} = 7.95, p = 0.003$) on trial completion ratio. Pairwise comparisons showed significant differences ($p \leq 0.01$) between *Screen-Space* and the two other techniques. The mean completion ratio was 96.0 percent for *DS3*, 94.9 percent for *Sticky Tools*, and 86.8 percent for *Screen-Space*. These results highlight the difficulty of using *Screen-Space* to perform the task.

The number of touches is defined by the total number of *TouchDown* events.

The ANOVA found a significant main effect for TECHNIQUE ($F_{2,22} = 12.54, p = 0.002$) on the number of touches. Pairwise comparisons showed significant differences ($p \leq 0.02$) between *Screen-Space* and the two other techniques. The mean number of touches was 12.6 for *DS3*, 17.2 for *Sticky Tools*, and 30.0 for *Screen-Space*. We hypothesize that a higher number of touches may be related to a greater difficulty to perform the task as this number is related to the number of iterative steps to accomplish the task.

8.2 Task Completion Time

Task completion time represents the time it takes to successfully position the current peg into the hole from the time it appeared. Skipped trials were removed from the analysis.

The ANOVA found a significant main effect for BLOCK ($F_{4,44} = 4.27, p = 0.01$) on task completion time, showing the

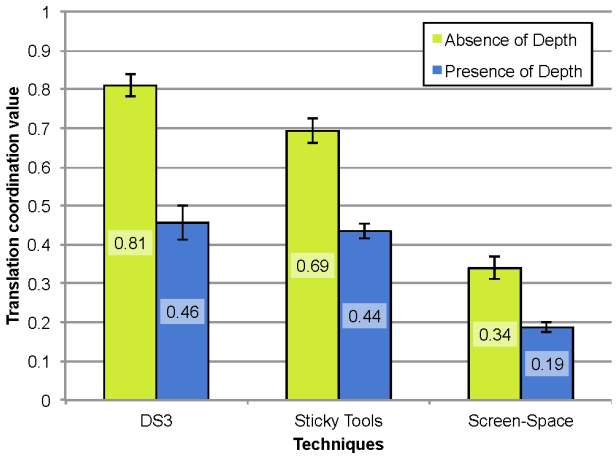


Fig. 5. Mean translation coordination for each technique under the different levels of PRESENCE OF DEPTH. Error bars represent 95 percent confidence interval.

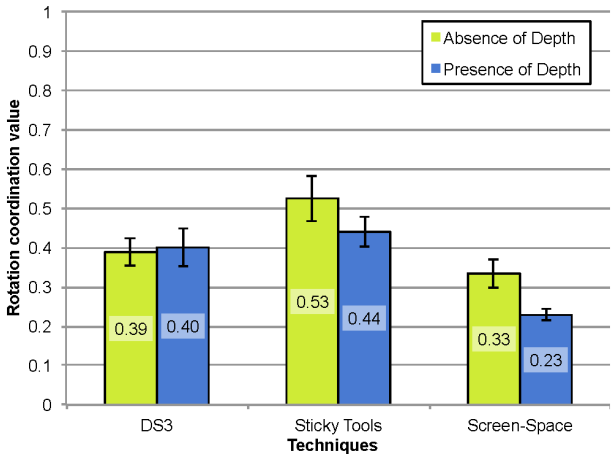


Fig. 6. Mean rotation coordination for each technique under the different levels of PRESENCE OF DEPTH. Error bars represent 95 percent confidence interval.

presence of a learning effect. Pairwise comparisons showed significant differences ($p < 0.05$) between the first block and the others. As a result we removed the first block for subsequent analysis.

The ANOVA also found a significant main effect for TECHNIQUE on task completion time ($F_{2,22} = 17.96$, $p < 0.001$). Pairwise comparisons showed significant differences ($p \leq 0.001$) between *DS3* (11.14 s) and the two other techniques, *Sticky Tools* (16.81 s), and *Screen-Space* (17.40 s). This supports **H1**: *DS3* is 36 percent faster than *Screen-Space* and 34 percent faster than *Sticky Tools*.

As expected, the ANOVA revealed a significant main effect of PRESENCE OF DEPTH ($F_{1,11} = 23.37, p = 0.001$) on task completion time, decreasing the mean completion time from 16.01 s with depth adjustment to 14.14 s with no depth adjustment. In addition, ROTATION LEVEL had a significant main effect ($F_{1,11} = 10.37, p < 0.01$), reducing the mean completion time from 16.33 s with COMPLEX to 13.90 s with SIMPLE level. Finally, the ANOVA found a significant main effect of ROTATION AMOUNT ($F_{1,11} = 5.98, p = 0.035$), diminishing the mean completion time from 17.40 s for LARGE to 12.86 s for SMALL level.

8.3 Translation Coordination

The translation coordination ratio is defined as the ratio of the length of the shortest path to the length of the actual path [12].

The ANOVA revealed a significant main effect for TECHNIQUE ($F_{2,22} = 112.19, p < 0.001$) on translation coordination. Pairwise comparisons showed significant differences ($p < 0.001$) between *Screen-Space* and the two other techniques. Pairwise comparisons also exposed a marginal difference ($p = 0.06$) between *DS3* and *Sticky Tools*. The mean translation coordination was 0.63 for *DS3*, 0.56 for *Sticky Tools* and 0.26 for *Screen-Space*. This result supports **H2**, *DS3* allowing more translation coordination.

As expected (**H3**), the ANOVA also revealed a significant main effect of PRESENCE OF DEPTH ($F_{1,11} = 1{,}286.73$, $p < 0.001$) and a significant TECHNIQUE × PRESENCE OF DEPTH interaction ($F_{2,22} = 19.81, p < 0.001$) on translation coordination. Under the NODEPTH level, *DS3* significantly outperformed *Sticky Tools* ($p = 0.01$) and *Screen-Space* ($p < 0.001$). For this level, the mean translation coordination was 0.81 for *DS3*, 0.69 for *Sticky Tools*, and 0.34 for *Screen-Space* (Fig. 5).

The ANOVA found a significant main effect for ROTATION AMOUNT ($F_{1,11} = 208.25, p < 0.001$), with the larger rotation level reducing the mean translation coordination from 0.56 to 0.41. Interestingly, the ANOVA revealed a significant TECHNIQUE × ROTATION AMOUNT interaction ($F_{2,22} = 17.32, p = 0.001$) on translation coordination. Under the SMALL level, *DS3* significantly outperformed *Sticky Tools* ($p < 0.01$) and *Screen-Space* ($p = 0.001$). For this level, mean translation coordination was 0.69 for *DS3*, 0.60 for *Sticky Tools*, and 0.40 for *Screen-Space*. The factor ROTATION LEVEL also revealed a significant main effect on translation coordination ($F_{1,11} = 16.49, p < 0.010$). Complex rotations reduced the mean translation coordination from 0.52 to 0.46.

Finally, the ANOVA found a significant ROTATION AMOUNT × PRESENCE OF DEPTH interaction ($F_{1,11} = 23.42, p = 0.005$) and a significant TECHNIQUE × ROTATION AMOUNT × PRESENCE OF DEPTH interaction ($F_{2,22} = 8.93, p = 0.006$) on translation coordination. Under the SMALL level of ROTATION AMOUNT and the NODEPTH level of PRESENCE OF DEPTH, *DS3* outperformed *Sticky Tools* ($p < 0.001$) and *Screen-Space* ($p = 0.002$). For these levels, mean translation coordination was 0.87 for *DS3*, 0.74 for *Sticky Tools*, and 0.55 for *Screen-Space*.

8.4 Rotation Coordination

The rotation coordination ratio is defined as the ratio of the initial rotation mismatch to the amount of actual rotation [12].

The ANOVA found a significant main effect for TECHNIQUE ($F_{2,22} = 11.71, p < 0.005$) on rotation coordination. Pairwise comparisons revealed significant differences between *Sticky Tools* and *Screen-Space* ($p = 0.02$), and between *DS3* and *Screen-Space* ($p = 0.02$) . The mean rotation coordination was 0.48 for *Sticky Tools*, 0.39 for *DS3*, and 0.28 for *Screen-Space*. This result supports **H2** but not the way we expected. Beyond separating translation DOF from rotation DOF, this indicates that the separation of rotation DOF themselves

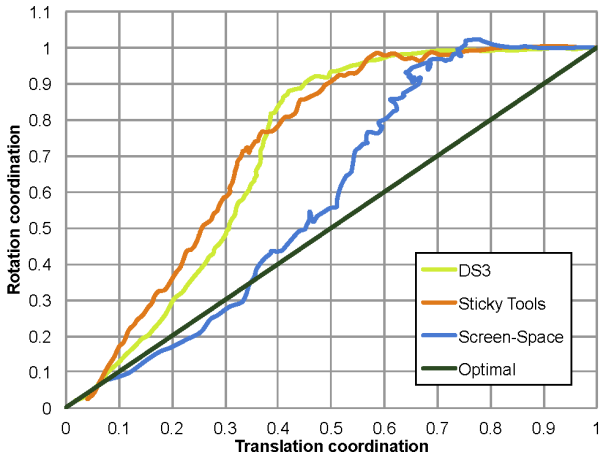


Fig. 7. Mean translation-rotation ratio for each technique. The curve begins at coordinates $(1, 1)$ and ends at coordinates $(0, 0)$. It is function of time (time not represented here).

might lead to better rotation coordination. However, this is just a preliminary result and further investigation would be required in order to conclude.

As hypothesized (**H3**), the ANOVA found a significant main effect of PRESENCE OF DEPTH on rotation coordination ($F_{1,11} = 12.57, p < 0.005$). The mean rotation coordination was 0.48 under the NODEPTH level and 0.39 under the DEPTH level. More interestingly, the ANOVA revealed a significant TECHNIQUE × PRESENCE OF DEPTH interaction ($F_{2,22} = 11.87, p < 0.005$) on rotation coordination (Fig. 6). Pairwise comparisons showed that both *Sticky Tools* and *Screen-Space* significantly reduced their rotation coordination ($p < 0.01$) with the DEPTH level whereas *DS3* was not affected ($p = 0.6$). The presence of depth reduced the mean rotation coordination from 0.53 to 0.44 for *Sticky Tools* and from 0.33 to 0.23 for *Screen-Space*. Interestingly, this result shows that PRESENCE OF DEPTH affects only coordination (both rotation and translation) as the ANOVA did not find any significant interaction on mean time.

In addition, the ANOVA found a significant main effect of ROTATION LEVEL on rotation coordination ($F_{1,11} = 33.17, p < 0.002$), with complex rotation reducing the mean rotation coordination from 0.44 to 0.33. Interestingly, the ANOVA also found a significant TECHNIQUE × ROTATION LEVEL interaction ($F_{2,22} = 17.97, p < 0.001$) on rotation coordination. Pairwise comparisons found that increasing the complexity of rotation significantly reduced the mean rotation coordination for *Sticky Tools* and *Screen-Space* ($p < 0.001$) whereas *DS3* was not affected. Mean rotation coordination decreased from 0.58 to 0.38 for *Sticky Tools* and from 0.34 to 0.22 for *Screen-Space*.

Finally, the ANOVA found a significant PRESENCE OF DEPTH × ROTATION AMOUNT interaction ($F_{1,11} = 37.91$, $p = 0.002$) and a significant TECHNIQUE × PRESENCE OF DEPTH × ROTATION AMOUNT interaction ($F_{2,22} = 7.82$, $p < 0.01$) on rotation coordination. Pairwise comparisons showed that under the SMALL level of ROTATION AMOUNT, changing the level of PRESENCE OF DEPTH significantly reduced the rotation coordination for *Sticky Tools* ($p < 0.01$) and *Screen-Space* ($p = 0.001$) whereas *DS3* was not affected ($p = 0.96$). When depth translation was required, mean rotation coordination decreased from 0.53 to 0.40 for *Sticky Tools* and from 0.41 to 0.19 for *Screen-Space*. This shows that separating translation DOF from rotation DOF helps increasing the rotation coordination (**H2**).

TABLE 1
Mean Time Spent in Each Mode

Title	Mean
Mode *1d*	38%
Mode *1d + 1i*	7%
Mode *2d*	50%
Mode *2d + 1i*	5%

(a) *DS3*

Title	Mean
Mode *1d*	42%
Mode *1d + 1i*	18%
Mode *2d*	34%
Mode *2d + 1i*	6%

(b) *Sticky Tools*

Title	Mean
Mode *1d*	35%
Mode *2d*	33%
Mode *3d*	32%

(c) *Screen-Space*

8.5 DOF Coordination

As explained in the *related work* section, several metrics exist to describe and quantify coordination in docking tasks. In this section, we want to understand how the participants used the different techniques to perform the evaluation task. For each technique we want to know how many DOF participants manipulated simultaneously. More precisely we want to determine which DOF were controlled in an integral way and which were controlled in a separable way. Finally, we want to investigate the strategies followed by the participants to adjust the orientation and position.

First, we computed the mean time that users spent in each mode of each technique. To investigate which DOF were manipulated together, we used the NDC [14]. This metric provides a percentage (Fig. 2) of time where participants manipulate n DOF ($1 \leq n \leq 6$). When a single DOF was manipulated, the NDC does not indicate which one actually changed. In this case, the corresponding DOF was identified as the one having the greatest absolute variation in translation or rotation. When several DOF were manipulated together, we used the ***m***-metric [13] to reveal which DOF were controlled simultaneously. Finally, we used the *translation-rotation* coordination ratio defined by Zhai and Milgram [12] to better understand the strategies employed by participants.

To compute the *translation-rotation* coordination ratio, we first resampled each trial into 100 evenly spaced time intervals using linear interpolation. For each interval we then computed the translation and rotation coordination [12], giving a corresponding 2D coordinate. These 100 points provided a representation of the curve starting at coordinate $(1, 1)$ (object not orientated nor positioned) and finishing at $(0, 0)$ (object well orientated and positioned) (Fig. 7).

8.5.1 Time Spent in Modes

The mean time spent in each mode of each technique can be found in Table 1. Statistical tests identify influences of independent variables.

For *Screen-Space*, the ANOVA found a significant effect of PRESENCE OF DEPTH for mode $1d$ ($F_{1,11} = 8.00, p = 0.03$) and $2d$ ($F_{1,11} = 15.40, p = 0.011$) but not for mode $3d$ ($p = 0.55$). When depth manipulation was required, the time spent in mode $1d$ dropped from 40 to 30 percent while time spent in mode $2d$ raised from 29 to 37 percent. This indicates a trend to use the pinch-to-zoom gesture for depth translation, as mentioned earlier in the pilot experiment.

TABLE 2
Mean NDC Values for Each Technique

Technique	MDS1	MDS2	MDS3	MDS4	MDS5	MDS6
DS3	31%	62%	6.5%	0.5%	0%	0%
Sticky Tools	55%	33%	6.5%	5%	0.5%	0%
Screen-Space	20.5%	14%	17%	42%	6.5%	0%

For *Sticky Tools*, the ANOVA revealed a significant effect of ROTATION LEVEL only for mode $1d$ ($F_{1,11} = 30.71$, $p < 0.001$) and $2d$ ($F_{1,11} = 13.05, p = 0.005$). Mode $1d + 1i$ and $2d + 1i$ were not affected (respectively, $p = 0.36$ and $p = 0.15$). When complex rotations were required, the use of mode $1d$ decreased from 46 to 38 percent while mode $2d$ increased from 31 to 37 percent. For *DS3*, we did not find a significant effect of ROTATION LEVEL on the time spent in the different modes. This, combined with rotation coordination result, indicates that an integration strategy for controlling rotational DOF is stable no matter the complexity of rotation.

8.5.2 Number of DOF Manipulated

The mean NDC for each technique can be found in Table 2. The results are consistent with the intended way of integrating and separating DOF for each technique. *Screen-Space* integrates several DOF more often than the two other techniques. To run statistical tests, we regrouped NDC3, 4, 5, and 6 into a global NDC that we named $NDC_{\geq 3}$. We then looked for interactions between TECHNIQUE and independent variables. A TECHNIQUE $\times$ PRESENCE OF DEPTH interaction was found for $NDC_{\geq 3}$ ($F_{2,22} = 6.62$, $p = 0.02$) showing only significant differences for *Sticky Tools* and *DS3* ($p < 0.01$). For *Sticky Tools*, when depth manipulation was required, $NDC_{\geq 3}$ raised from 10 to 16 percent. For *DS3* we noticed the opposite behavior: $NDC_{\geq 3}$ dropped from 8 to 6 percent. Combined with the time spent in mode $2d$ and the rotation coordination results, this illustrates a consequence of the integration strategy. When depth translation was needed with *Sticky Tools*, users spent significantly more time in mode $2d$, controlling significantly more DOF. We believe this conclusion highlights unwanted movements—*DS3*, which separates DOF, seems to avoid this issue.

The ANOVA also found a significant TECHNIQUE $\times$ ROTATION LEVEL interaction ($F_{2,22} = 11.75, p < 0.004$) which supports this conclusion. A significant difference was found only for *Sticky Tools* ($p = 0.03$) and *Screen-Space* ($p = 0.006$), not for *DS3* ($p = 0.92$). Complex rotations increased $NDC_{\geq 3}$ from 10 to 16 percent for *Sticky Tools* and from 61 to 71 percent for *Screen-Space*.

8.5.3 Single DOF Manipulation

Table 3 shows the percentage of time manipulating each individual DOF. Two relevant aspects emerge from the results. First, the manipulation of Tx and Ty is very high, regardless of the technique: the two DOF represent more than 50 percent of single DOF manipulation. Second, this highlights differences in the way the indirect finger is used for different techniques. For *Sticky Tools*, Rx and Ry (controlled by an indirect finger in mode $1d + 1i$ and $2d + 1i$) have high values, due to the use of the indirect finger in a separated way. For *DS3*, Tz (controlled in mode $1d + 1i$) presents a low value, since this is an integration of this DOF by the user.

TABLE 3
NDC1 Time Split for Every Single DOF

Technique	*Tx*	*Ty*	*Tz*	*Rx*	*Ry*	*Rz*
DS3	8.5%	15.5%	1%	3.5%	1%	1.5%
Sticky Tools	9.5%	20.5%	0%	10.5%	13.5%	1%
Screen-Space	6.5%	13%	0%	0.5%	0.5%	0%

TABLE 4
Time Spend Manipulating Couples of DOF

Couple	Mean
(*Ty, Tz*)	20.5%
(*Rx, Rz*)	13%
(*Ry, Rz*)	13%
(*Tx, Tz*)	9.5%
(*Rx, Ry, Rz*)	6.5%
(*Rx, Ry*)	6%
Other	≤ 2%

(a) *DS3*

Couple	Mean
(*Tx, Tz*)	11.5%
(*Ty, Tz*)	6%
(*Rx, Rz*)	5.5%
(*Ry, Rz*)	4%
(*Rx, Ry*)	3.5%
(*Ty, Tz, Rx, Rz*)	2.5%
Other	≤ 2%

(b) *Sticky Tools*

Couple	Mean
(*Ty, Tz, Ry, Rz*)	14.5%
(*Ty, Tz, Rx, Rz*)	14.5%
(*Tx, Tz*)	9.5%
(*Ty, Tz, Rx, Ry*)	6.5%
(*Ty, Tz, Rx*)	6%
(*Ty, Tz, Ry*)	6%
Other	≤ 2%

(c) *Screen-Space*

8.5.4 Multiple DOF Manipulation

We first analyzed subsets of DOF using the ***m***-metric, but were surprised by the results. All subsets presented very low values (<0.1), except for (Tx, Ty) which stood out (***m***-metric for $DS3 = 0.36$, for $Sticky\,Tools = 0.35$ and for $Screen\text{-}Space = 0.20$). This reflects the task difficulty which was much more difficult than real object manipulation. Nevertheless, we still wish to know which subsets of DOF were manipulated, and for how long. To calculate these results we used an approach similar to the single DOF measure. Using the results of NDC (e.g., $NDC = n$), we computed the ratio of times the n DOF were manipulated. Table 4 shows the mean results for each technique.

For *Screen-Space* the results are consistent with the taxonomy. The technique tends to integrate more DOF. Combined with the time performance and previous results, we now have insights on the users' strategies. By controlling several DOF at the same time, users cannot perform precise manipulation. Instead, they perform unwanted movements that need to be readjusted later.

For *Sticky Tools*, we can see that the first two subsets include Tz. In addition with the short time spent to manipulate Tz separately, this also supports our previous conclusion regarding unwanted movements. Also, we can see that Rx and Ry are not very integrated, highlighting once again the sequential use of the indirect finger.

For *DS3*, results show an integration of Tz, combined with Tx or Ty, 30 percent of the time. This gives us some insights regarding the use of the indirect finger. With *DS3*, users manipulate the same type of attribute with both direct and indirect finger (attribute of position), making it possible to combine the action of both fingers. With *Sticky Tools*, the controlled attributes are different (orientation with position) and the two fingers tend to perform more sequentially.

8.5.5 Translation-Rotation Ratio

Mean translation-rotation ratio can be found in Fig. 7. For all the techniques, we can see that users start to manipulate

the position first, then the orientation. The global shape of the curve is similar for *Sticky Tools* and *DS3*, and we can distinguish two phases. First, users primarily manipulate position DOF, reducing translation coordination by 0.6 (i.e., from 1 to 0.4) while reducing rotation coordination by 0.2 (i.e., from 1 to 0.8). Then, they switch to rotation DOF, reducing rotation coordination by 0.8 (i.e., from 0.8 to 0) while reducing translation coordination by 0.4 (i.e., from 0.4 to 0).

For *Screen-Space*, the global shape is different. The manipulation of rotational DOF starts sooner and more generally, the curve is closer to the optimal. This conclusion, together with poor performance, indicates that better coordination does not mean better performance, especially when working on multitouch table and 3D manipulation.

8.6 Qualitative Feedback

Among the participants, eight preferred *DS3*, two rated *DS3*, and *Sticky Tools* equally, and two preferred *Sticky Tools*. The participants who said they prefer *DS3* found the technique easier to use and appreciated the fact that rotation was decoupled from translation. Many summarized this saying, this allowed much more accurate control. They also mentioned that performing rotation was easy and felt *natural*. The participant who prefered *Sticky Tools* found the technique less difficult to use for manipulation with both hands. They also reported that they were able to do everything with only one hand. This is something they strongly preferred even if both hands were occasionally required. In contrast to the other participants, they did not like the sequentiality of *DS3*.

Regarding *Sticky Tools*, one user reported that the way to handle rotation was efficient but not intuitive. In addition, they did not like the fact that depth translation and rotation were linked together. This difficulty came from the coupling of translation and rotation—this was also pointed out by five other participants. Another user reported that the use of an indirect (i.e., not on the 3D object) finger to control orientation was easier to use, in contrast to *DS3* where the external finger controls the depth position.

Regarding the *Screen-Space* technique, all participants reported that this technique was very difficult to use to perform the task. Two participants reported that, although the technique was conceptually intuitive, it was very difficult in practice to perform specific operations. Four participants liked the technique when working on a planar surface such as the top of the peg. They successfully used a gesture highlighted by Reisman et al. [9], where four fingers manipulate a 3D plane. However, they also pointed out the limitation due to the object size: *Screen-Space* is difficult to use when the size of the object is reduced. Another participant commented that although complex movements were relatively easy to achieve, simple movements were difficult. This was supported by two participants who described the technique as *unpredictable*. Six reported that the integration of all DOF together made the technique difficult to use.

9 Discussion

We designed the experiment to compare three different techniques for performing 3D manipulation tasks on multitouch displays. *Screen-Space* controlled the six DOF of the task in an integral way, whereas *Sticky Tools* and *DS3* separated the DOF of the task using different strategies.

9.1 DOF Separation and Performance

Results show that, for the techniques studied, DOF separation improves performance compared to DOF integration for a docking task on multitouch displays: *DS3* showed significant lower task completion time compared to *Screen-Space* and *Sticky Tools*. *DS3* improves performance by 36 percent compared to *Screen-Space* and by 34 percent compared to *Sticky Tools*.

This result can be explained by the translation and rotation coordination values showing the effectiveness for controlling the DOF of the task. *Screen-Space*, which tightly couples rotation and translation, revealed the lowest translation and rotation coordination. *Sticky Tools* significantly improves translation coordination by 115 percent and rotation coordination by 71 percent compared to *Screen-Space*, while *DS3* improves translation coordination by 142 percent and rotation coordination by 39 percent compared to *Screen-Space*.

It appears that for the integral 3D manipulation task we considered, trying to use the separated DOF of a multitouch display in an integral way provides lower performance. Instead, separating the DOF of the task to match the separated structure of the input device leads to better results. This conclusion extends the work of Veit et al. [10] who found similar results for an orientation task.

Jacob's conclusion on integration of DOF was to match as closely as possible the structure of the task with the structure of the input device. But, when working with 2D devices interacting with 3D data, following Jacob's conclusions is not possible. When faced with a choice, our study suggests that performance is improved if the interaction technique follows the structure of the input device rather than the structure of the task.

9.2 DOF Separation Strategies

The experiment showed a significant lower task completion time for *DS3* compared to *Sticky Tools* with a 34 percent improvement for *DS3*. It shows that the strategy of separating DOF can have a severe impact on performance.

Garner [7] showed that orientation and position are two integral attributes of 3D manipulation, making the theory of Jacob et al. [5] difficult to apply to multitouch displays. The lower completion time for *DS3* suggests that orientation and position are still two *different* attributes which users can easily separate.

In designing *DS3*, we clearly separated translation from the rotation DOF, leading to an inferior completion time but also to a higher translation coordination. Providing the control of orientation and depth positioning at the same time, *Sticky Tools* has a significantly lower translation coordination compared to *DS3* when no depth translation was required, highlighting the fact that users had difficulties isolating the control of rotations with *Sticky Tools*. We illustrate this idea in Fig. 8, which shows the mean *translation-rotation* coordination ratio of a particular participant. For *Sticky Tools* and *Screen-Space* we can see that when manipulating rotation, users often changed, even reduced, the coordination in translation.

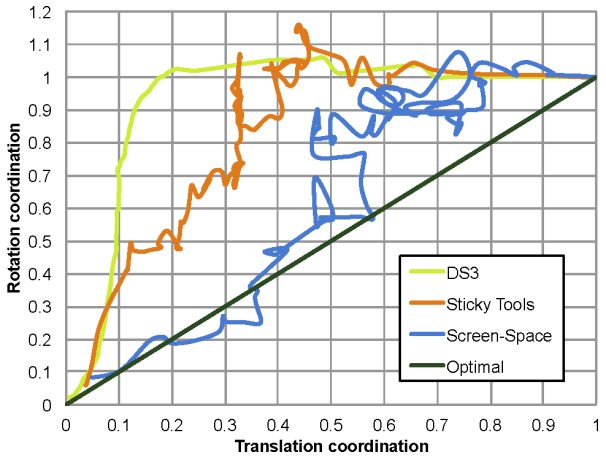


Fig. 8. Typical examples of mean *translation-rotation* coordination ratio from different participants for each technique. It illustrates the integrated control of translation and rotation with both *Sticky Tools* and *Screen-Space* and the separation of translation and rotation with *DS3*.

These conclusions extend the work of Nacenta et al. [11] to 3D manipulation. With techniques mapping depth translation and rotation together, it becomes more difficult to perform large rotations without also affecting position, resulting in poor performance and coordination. Introducing *DS3* with the separation of the control of rotation and translation, we increased performance, coordination, and user satisfaction.

These results suggest more separation of DOF. *DS3*, which separates translation DOF, shows the best translation coordination. *Sticky Tools*, which separates rotation DOF, shows the best rotation coordination. A new technique which combines the separation strategy of both *Sticky Tools* and *DS3* would likely improve performance.

The study of DOF coordination also revealed an interesting result, specific to bimanual interaction. Both *Sticky Tools* and *DS3* use an indirect finger, but the techniques differ in the nature of how the DOF are controlled. With *Sticky Tools*, the indirect finger controls an attribute of orientation, while the direct finger handles an attribute of position. For *DS3*, both the direct and indirect fingers control an attribute of position. Our coordination results indicate that controlling the same type of attributes can be parallelized with two hands, while the control of different types of attributes, if mapped to two hands, are performed more sequentially. Designers should consider this when creating two handed interaction techniques for multitouch displays.

9.3 Direct versus Indirect Control

One key point with *DS3* is the mode switching between the control of rotation and translation. In our case, we differentiate rotation from translation according to the number of finger contacts. However, this may have a major drawback when the size of the object being manipulated is small. For example, using three fingers to perform 3D manipulation with *Screen-Space* is difficult with small objects. Indirect control represents a way to solve this issue.

Indirect control provides users with the ability to clearly separate DOF, even with small objects. However, a drawback is the possibility of inadvertently selecting or manipulating another nearby object. When designing interaction techniques for multitouch displays, interface designers should either prevent interaction with multiple objects at once, or provide clear feedback to show which objects users can interact with.

Another drawback is the manipulation of multiple objects. While *Screen-Space* allows the manipulation of several objects at the same time due to its direct interaction, both *DS3* and *Sticky Tools* use an indirect finger which prevents manipulating multiple objects simultaneously. A way to solve this problem is to provide an area adjacent to each object, dedicated to indirect finger control. In its simple form, this area can be visualized as a circle which appears around or nearby the object being manipulated. Another solution could be to adapt gestures from Rock & Rails interactions [22].

10 Conclusion and Future Work

We have introduced a taxonomy to classify 3D manipulation techniques for multitouch displays. The study of 3D interaction techniques in relation to the structure of multitouch displays led us to introduce *DS3*, a 3D manipulation technique based on a total separation of the position and orientation control. Results of a controlled experiment show the strength of the new technique for both performance and user preference.

This relationship between the separation of DOF and performance confirms recent results showing that the simultaneous manipulation of all DOF does not necessary lead to the best performance [10]. Our study revealed that the integration of both translation and rotation reduce performance, coordination, and user satisfaction. A conclusion which extends the work of Nacenta et al. [11], who covered similar issues for 2D manipulation. As future work we wish to explore the design space introduced by the taxonomy presented in this paper.

Acknowledgments

This work was supported by the ANR project n° ANR-09-CORD-013 "InSTInCT." We thank the study participants for their time and Daniel Vogel for proof reading.

References

[1] K. Hinckley, J. Tullio, R. Pausch, D. Proffitt, and N. Kassell, "Usability Analysis of 3D Rotation Techniques," *Proc. 10th Ann. ACM Symp. User Interface Software and Technology*, pp. 1-10, 1997.

[2] Y. Wang, C.L. MacKenzie, V.A. Summers, and K.S. Booth, "The Structure of Object Transportation and Orientation in Human-Computer Interaction," *Proc. the SIGCHI Conf. Human Factors in Computing Systems*, pp. 312-319, 1998.

[3] J. Rekimoto, "SmartSkin: An Infrastructure for Freehand Manipulation on Interactive Surfaces," *Proc. the SIGCHI Conf. Human factors in Computing Systems: Changing Our World, Changing Ourselves*, pp. 113-120, 2002.

[4] M.S. Hancock, S. Carpendale, F.D. Vernier, D. Wigdor, and C. Shen, "Rotation and Translation Mechanisms for Tabletop Interaction," *Proc. the First IEEE Int'l Workshop Horizontal Interactive Human-Computer Systems*, 2006.

[5] R.J.K. Jacob, L.E. Sibert, D.C. McFarlane, and J.M.P. Mullen, "Integrality and Separability of Input Devices," *ACM Trans. Computer-Human Interaction*, vol. 1, no. 1, pp. 3-26, 1994.

[6] J.N. Ingram, K.P. Körding, I.S. Howard, and D.M. Wolpert, "The Statistics of Natural Hand Movements," *Experimental Brain Research.,* vol. 188, no. 2, pp. 223-236, 2008.
[7] W. Garner, *The Processing of Information and Structure.* L. Erlbaum Assoc., 1974.
[8] M. Hancock, T.T. Cate, and S. Carpendale, "Sticky Tools: Full 6dof Force-Based Interaction for Multi-Touch Tables," *Proc. ACM Int'l Conf. Interactive Tabletops and Surfaces '09,* 2009.
[9] J.L. Reisman, P.L. Davidson, and J.Y. Han, "A Screen-space Formulation for 2D and 3D Direct Manipulation," *Proc. the 22nd Ann. ACM Symp. User Interface Software and Technology,* pp. 69-78, 2009.
[10] M. Veit, A. Capobianco, and D. Bechmann, "Influence of Degrees of Freedom's Manipulation on Performances during Orientation Tasks in Virtual Reality Environments," *Proc. the 16th ACM Symp. Virtual Reality Software and Technology,* pp. 51-58, 2009.
[11] M.A. Nacenta, P. Baudisch, H. Benko, and A. Wilson, "Separability of Spatial Manipulations in Multi-Touch Interfaces," *Proc. Graphics Interface '09,* pp. 175-182, 2009.
[12] S. Zhai and P. Milgram, "Quantifying Coordination in Multiple DOF Movement and Its Application to Evaluating 6 DOF Input Devices," *Proc. SIGCHI Conf. Human Factors in Computing Systems,* pp. 320-327, 1998.
[13] M.R. Masliah and P. Milgram, "Measuring the Allocation of Control in a 6 Degree-of-Freedom Docking Experiment," *Proc. SIGCHI Conf. Human Factors in Computing Systems,* pp. 25-32, 2000.
[14] M. Veit, A. Capobianco, and D. Bechmann, "An Experimental Analysis of the Impact of Touch Screen Interaction Techniques for 3-D Positioning Tasks," *Proc. IEEE Conf. Virtual Reality,* 2011.
[15] A.D. Wilson, S. Izadi, O. Hilliges, A. Garcia-Mendoza, and D. Kirk, "Bringing Physics to the Surface," *Proc. 21st Ann. ACM Symp. User Interface Software and Technology,* pp. 67-76, 2008.
[16] O. Hilliges, S. Izadi, A.D. Wilson, S. Hodges, A. Garcia-Mendoza, and A. Butz, "Interactions in the Air: Adding Further Depth to Interactive Tabletops," *Proc. the 22nd Ann. ACM Symp. User Interface Software and Technology,* pp. 139-148, 2009.
[17] A. Martinet, G. Casiez, and L. Grisoni, "The Design and Evaluation of 3D Positioning Techniques for Multi-Touch Displays," *Proc. IEEE Symp. 3D User Interfaces,* pp. 115-118, 2010.
[18] M. Hancock, S. Carpendale, and A. Cockburn, "Shallow-Depth 3D Interaction: Design and Evaluation of One-, two- and Three-Touch Techniques," *Proc. SIGCHI Conf. Human factors in Computing Systems,* pp. 1147-1156, 2007.
[19] A. Cohé, F. Dècle, and M. Hachet, "Tbox: A 3D Transformation Widget Designed for Touch-Screens," *Proc. 2011 Ann. Conf. Human Factors in Computing Systems* pp. 3005-3008, 2011.
[20] S.K. Card, J.D. Mackinlay, and G.G. Robertson, "A Morphological Analysis of the Design Space of Input Devices," *ACM Trans. Information Systems,* vol. 9, no. 2, pp. 99-122, 1991.
[21] B.J. Unger, A. Nicolaidis, A. Thompson, R.L. Klatzky, R.L. Hollis, P.J. Berkelman, and S. Lederman, "Virtual Peg-in-Hole Performance Using a 6-Dof Magnetic Levitation Haptic Device: Comparison with Real Forces and with Visual Guidance Alone," *Proc. 10th Symp. Haptic Interfaces for Virtual Environment and Teleoperator Systems,* p. 263, 2002.
[22] D. Wigdor, H. Benko, J. Pella, J. Lombardo, and S. Williams, "Rock & Rails: Extending Multi-Touch Interactions with Shape Gestures to Enable Precise Spatial Manipulations," *Proc. 2011 Ann. Conf. Human Factors in Computing Systems,* pp. 1581-1590, 2011.

Anthony Martinet is working toward the PhD degree in the MINT research team. His thesis focuses on multitouch interactions, specifically 3-D interactions.

Géry Casiez received the PhD degree in computer science from the University of Lille 1. He is an assistant professor at the University of Lille 1 Computer Science Department and a member of the MINT research team, colabeled by CNRS and INRIA. His research interests include 2D and 3D interaction, haptic interaction, and the empirical evaluation of user interfaces, including associated metrics and predictive models of human performance.

Laurent Grisoni received the PhD degree in 1999 from the University of Bordeaux I. He is a professor at the University of Lille 1 Computer Science Department, head of MINT research team, colabeled by CNRS and INRIA. His research interests include geometric modeling, physical simulation, interactive 3D, and most recently computer-human interaction (gesture-based interfaces).

Conté: Multimodal Input Inspired by an Artist's Crayon

Daniel Vogel[1,2] *and Géry Casiez*[1]

[1]LIFL & INRIA Lille
University of Lille, FRANCE
gery.casiez@lifl.fr

[2]Cheriton School of Computer Science
University of Waterloo, CANADA
dvogel@uwaterloo.ca

ABSTRACT

Conté is a small input device inspired by the way artists manipulate a real Conté crayon. By changing which corner, edge, end, or side is contacting the display, the operator can switch interaction modes using a single hand. Conté's rectangular prism shape enables both precise pen-like input and tangible handle interaction. Conté also has a natural compatibility with multi-touch input: it can be tucked in the palm to interleave same-hand touch input, or used to expand the vocabulary of bimanual touch. Inspired by informal interviews with artists, we catalogue Conté's characteristics, and use these to outline a design space. We describe a prototype device using common materials and simple electronics. With this device, we demonstrate interaction techniques in a test-bed drawing application. Finally, we discuss alternate hardware designs and future human factors research to study this new class of input.

ACM Classification: H.5.2 Information Interfaces and Presentation: Input; Interaction styles

General terms: Human Factors

Keywords: pen, touch, gestures, tabletop, multimodal

INTRODUCTION

Touch interaction is arguably more immediate and natural in many situations, but fingers are imprecise [27] and difficult to write with [3,15]. Alternatively, using a pen (or stylus) makes writing more natural and pointing more precise [29]. Luckily, this does not need to be a unilateral choice; pen and touch can be used simultaneously [3,15,29,30]. However, without non-dominant hand coordination or graphical buttons, the pen *itself* supports few modes. This makes single-handed mobile usage difficult and reduces the number of combined touch and pen modes. When frequently switching between pen-oriented modes, such as drawing, handwriting, gestures, and lasso selection [29], this can hurt performance [18]. Inferring modes is difficult, and most users prefer explicit control [7]. Schemes for squeezing multiple *explicit* modes from a pen include adding barrel buttons and classifying pressure [22], tilt [26], barrel rotation [3], or grip [25]. But these can be error-prone and ambiguous. A simple way to add a second mode is by adding an "eraser," a second *contact* point. The pencil analogy lends intuition and users have explicit control.

UIST'11, October 16–19, 2011, Santa Barbara, CA, USA.

Our work extends this idea of switching modes using contact points by referencing how artists use a *real* Conté crayon, a square prism-shaped drawing stick. Its faceted shape creates a variety of marks according to how it contacts paper: corner, edge, end, or side (Figure 1). A *digital* Conté crayon could even sense *which* corner, *which* edge, or *which* side. Thus, Conté builds on ToolStone [23], but with notable differences: ToolStone is a larger block shape, uses only side contacts, and is designed for non-dominant hand usage on an indirect tablet. Conté can function single-handed, yet span multiple precise pen-like input modes and modes similar to tangible handles [16]. For example: one corner for drawing and another for handwriting recognition; one end edge for stroke-based gestures and another for lasso selection; one side to reveal a tool palette and another to create guidelines. Artists blend real Conté drawings with their fingers, so combining digital Conté with multi-touch input is natural. By tucking Conté in the palm, same-hand touch may be interleaved and, similar to Manual Deskterity [15], a Conté-enabled mode may be further manipulated with bimanual touch. For example: rotating a guideline with Conté while selecting angle snap with touch.

Inspired by informal interviews with artists, we catalogue digital Conté's characteristics, and use these to outline a design space. A principal challenge is the creation of a functional device. We describe the hardware and software for our digital Conté stick which is compatible with diffuse infrared illumination tabletops. Our solution uses common materials, basic tools, and simple electronics, making it possible for others to build their own Conté. As future work, we discuss alternate device implementations.

With our prototype Conté device, we implemented several compatible interaction techniques in a test-bed drawing application. These specific examples taken from the larger design space demonstrate the feasibility and highlight practical aspects of Conté techniques. A user evaluation would be premature for this nascent device [13], but we close with a discussion of avenues for formal evaluation with an emphasis on fundamental human factors questions raised by this new class of input.

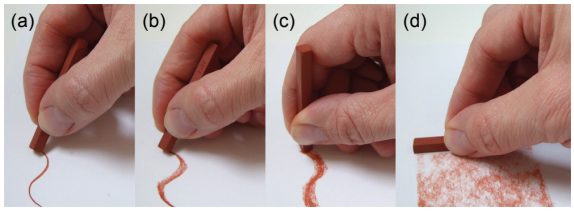


Figure 1. Drawing marks left by an artist's Conté crayon when using: (a) corner; (b) edge; (c) end; and (d) side.

RELATED WORK

Elegantly switching between multiple modes has been a challenge since the first pen input systems. The reader may recall the classic Sketchpad demonstration [31] where the operator is constantly "tickling" toggle switches. Later, DENIM [19] and Tivoli [20] state the problem clearly, and experiment with various techniques. In a study of "ink vs. gesturing" mode selection, Li et al. [18] found that pressing a hardware button with the non-dominant hand was fastest. However, each mode needs a button, and more than two modes may be necessary: Wu et al. [29] suggest 3 pen modes ("...write, annotate and sketch"); Brandl et al. add precise selection [4]; and Song et al. introduce 4 more [25].

Bimanual Touch and Pen Mode Selection. By combining a *single-touch* screen with a pen sensor Yee [30] demonstrates how non-dominant button touches, similar to Li et al.'s hardware button, can select among different pen tools. Wu et al. [29] and Brandl et al. [4] refine this further, by combining a pen with a *multi-touch* surface. Now modes may be selected according to the type of non-dominant hand contact. For example, touching two fingers changes from pen selection to pen inking [29], or a flat palm contact changes from free-form to poly-line pen drawing [4]. These combined touch and pen systems enable a broad continuum of interaction. On one extreme, pen and touch can act independently, like Yee's [30] and Brandl at al.'s [4] use of touch to pan the drawing surface while the pen writes. At the other end, pen and touch can coordinate very closely, such as Yee's [30] combined pen and touch zoom.

Hinckley et al.'s Manual Deskterity system [15] refines bimanual pen and touch coordination by considering the contextual relationship with direct manipulation objects (e.g. notes in a sketchbook application). When used alone, the pen always draws and the hand always manipulates, but when used in the context of an object, an implicit mode is entered. For example, depending on how an object is held with non-dominant touches and the object-relative motion of the pen, a pen stroke may cut like a knife, draw a straight line along an edge, or create a copy of the object. This greatly increases the number of modes in an elegant way, but various hand postures need to be recalled and performed unambiguously [2], and not all modes seem to have obvious bimanual contextual mappings. Consider how Hinckley et al. resort to a toolbar button to change the pen to a highlighter. Finally, along with Yee, Brandl et al., and Wu et al., two hands are required. Hinckley et al. do include redundant pen-only command access via radial menus, but acknowledge this adds syntactical complexity.

Unimanual Pen Mode Selection. Commercial pen input systems, such as the Wacom 6D Art Pen, provide multiple input dimensions which can be used for one-handed mode switching. Among one-handed techniques, Li et al. [18] find pressure and barrel button fastest, but with higher error rates (only beating conventional press-and-hold which was also slowest). Flipping an "eraser"-equipped pen is also slow, but with the lowest one-handed error rate. Li et al. only test binary mode selection, but by classifying continuous parameters like pressure [22], tilt [26], barrel roll [3], or even barrel grip [25], up to 6 modes can be supported. However, these can be ambiguous and error-prone due to interference from normal pen movements.

Mode Selection by Contact. Expanding on Fitzmaurice et al.'s flipbrick idea [8], Rekimoto and Sciammarella's ToolStone [23] is a block which is flipped and rotated to select between as many as 48 different modes. Although designed for non-dominant hand input on an indirect tablet, it illustrates how sensing the device-to-surface *contact* can be harnessed for mode selection. ToolStone focuses on face contacts; edge sensing is mentioned, but not explored. At $25 \times 40 \times 50$ mm, ToolStone is much larger than a pen but, in an elicitation study, Frisch et al. [10] observe some users using the side of a pen in a similar manner.

Summary. Conté builds on the ToolStone concept, but with notable differences. Conté has a much smaller form factor and is capable of detecting a more diverse set of contacts such as corners, short and long edges, sides, and ends. Although not a pen per se, when a corner or end edge contact is used, Conté approaches pen-like manipulation and precision. Thus, with an ability to switch modes based on contact, it directly addresses the unimanual pen mode selection problem. Compared to flipping an eraser-equipped pen, Conté has contact point adjacencies — clusters of different contact points at one end — with much shorter transition times compared to flipping a pen.

Also, unlike ToolStone, Conté is designed to work with multi-touch on a direct input display, a natural combination given how artists blend real Conté strokes with their finger. Dominant single-hand touch input can be interleaved by tucking Conté in the palm or setting Conté down. Bimanual pen and touch like Yee [30], Wu et al. [29], Brandl et al. [3], and Manual Deskterity [15] can also be achieved due to Conté's pen-like form factor. But, with Conté's multimodal capability, the design space is expanded.

EXPLORATORY INTERVIEWS

We conducted interviews with two practicing artists to gain a better understanding of real Conté and its potential digital counterpart. The specific questions and tasks were:

1. *How are real Conté sticks used*? Participants create simple Conté drawings on paper;
2. *How might digital Conté perform fundamental tasks?* Participants write, draw, tap, and trace paths;
3. *How might digital Conté perform basic manipulations?* Participants change contact points, set the stick down and pick it up, and "tuck" it in their palm.

For question 1, participants used a real Conté stick. For questions 2 - 4, they used solid acrylic mock-ups of different sticks with display-like feedback enabled by a "Magic Drawing Slate" novelty toy (Figure 2b). This is a grey sheet of film loosely laid over a black waxy under-layer. Pressing against the film creates a blackish mark and the marks are erased by lifting the film. We found this basic feedback essential to feel how digital Conté may work for mark-

making. We tested three different Conté mock-ups (Figure 2a): 6 × 6 × 64 mm (same as artist's Conté crayons), 8 × 8 × 81 mm (same as another type of square prism shaped artist's crayon), 8 × 11 × 81 mm (an extruded rectangle profile we thought might be advantageous). A tablet pen and the short pen from the toy slate were available for comparison.

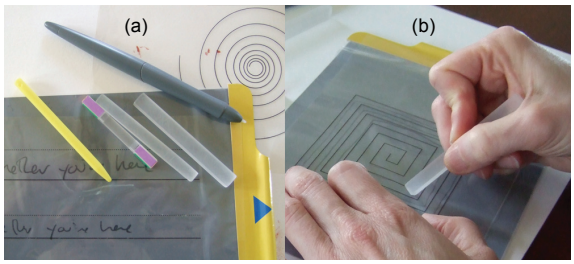


Figure 2. Interviews: (a) Conté mock-ups; (b) simulating fundamental tasks with the Magic Drawing Slate.

Primary observations are below (i.e. **1**a references task **1**):

1a. They use the corner, the end edge, and the side, and they blend with both non-dominant fingers and their dominant hand (by tucking the stick in their palm).

1b. Two different grips are used: a precise dynamic tripod grip, and a finger and thumb grip. The latter enables more contact surfaces to be accessed quickly.

1c. The tip becomes rounded with use, but artists can "find a corner" by feel as they rotate the shaft.

2a. Writing and tracing with a corner was similar to a pen, but the longer 81 mm mock-ups were preferred. Writing with Conté was less comfortable than a pen, but not especially uncomfortable either.

2b. The extruded rectangle profile could be held more securely, and did not reduce manipulative capability.

3a. "Tucking" Conté in the palm was easier than the tablet pen and up to three fingers could be used for touch.

3b. Transitioning contacts was fine, but took longer if the grip also had to be changed or re-adjusted. For example, changing from corner to side, or end to end.

3c. The most difficult contact to hold was a long side edge.

While clearly not a formal design study, these are useful general observations which we refer to later. Moreover, the open structure may have revealed aspects otherwise missed, and participants actually said it was fun.

CONTÉ CHARACTERISTICS AND DESIGN SPACE

In spite of its simple form factor, digital Conté has the potential to quickly switch between expressive input modes while providing additional parameters such as azimuth angle (barrel rotation), elevation angle (tilt), and pressure. After discussing the characteristics and capabilities of Conté, we outline a design space with includes using Conté alone and in combination with multi-touch.

Characteristics

The discussion here stems from our own observations supported by the artist interviews. Conducting formal quantitative experiments would be premature; we will discuss potential human factors studies as future work. To ground our discussion, we focus on the extruded rectangle Conté shape. The shape is pen-like given its crayon heritage, though compared to a pen it is shorter, faceted, and without a well-defined nib. Of course there will be some reduction in comfort and precision as the form factor deviates from a standard pen, and likewise a reduction in the number of stable contacts as the shape deviates from a ToolStone block. Our intention was to optimize this trade-off and we note that even efforts to establish an ideal pen shape are conflicted due to influences of individual preference and task [10,28]. The extruded rectangle shape was favoured by our interview participants and the slight irregularity should help users (and software) distinguish end edges and side faces. This form factor potentially supports 26 *different* contacts, classified into 7 *types* (Figure 3): 8 corners, 4 short end edges, 4 medium end edges, 4 long side edges, 2 end faces, 2 thick side faces, and 2 thin side faces.

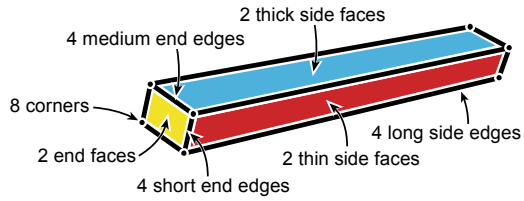


Figure 3. Seven types of contacts.

Contact Point Characteristics. For each type of contact, the combination of hand grip, contact shape, and equilibrium resulting from operating position dictate the availability of additional input parameters, level of precision, manipulation capability, and ability to maintain state when released (Figure 4). Song et al.'s [25] observation that people adopt different grips motivates a pen design which changes mode by sensing the current grip. With Conté, adopting different grips is a natural part of changing the desired contact.

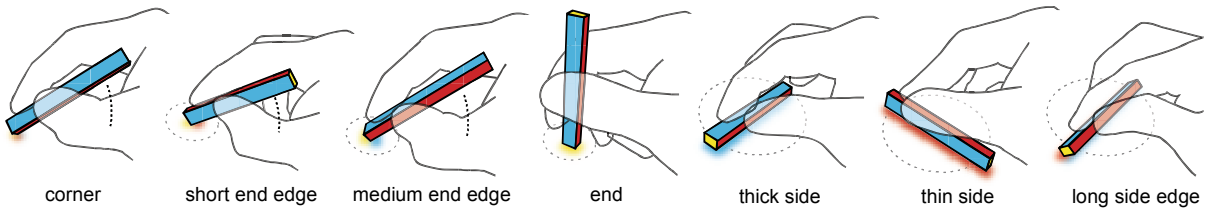


Figure 4. Typical hand grips used for different Conté contacts.

When contacting a corner, Conté is held like a pen using a precision grip, typically a dynamic tripod. The stick must be held at an angle to clearly disambiguate between adjacent contact points, so it has an unstable equilibrium (it must be supported to maintain a corner contact). Depending on the device thresholds for detecting adjacent contacts, usable elevation angles are less than a pen (well within 0 to 90°, clustered near 45°). There is no azimuth angle sensing limit, but in practice this is limited by wrist and elbow range-of-motion to about a 180° arc.

When contacting a short or medium edge, a slightly modified dynamic tripod grip [21] may be used and the increased contact area of the edge adds stability, though it must be held to maintain state. These factors reduce fine manipulation capability compared to the corner (or a pen), but the range of azimuth and elevation angles are similar.

In the exploratory interviews, we found that the long edge required a difficult thumb and index finger pinch grip. The small tolerance between the adjacent side faces reduces usable elevation angles to those near 45°. The instability of the grip and wrist and elbow range of motion reduce azimuth angles to approximately 180°, but rotating is difficult given the challenging grip.

The thick and thin side faces have a large contact surface area and low centre of gravity creating a very stable equilibrium: the grip can be loosened, or the stick set down, and the state maintained. The required pinch grip is much easier compared to the long edge. Also, the stability means azimuth angle covers the full 360° by manipulating with a clutching action. Elevation angle is invariant.

Contacting an end face suggests a further modified dynamic tripod grip which is moving towards a power grip [21]. Due to the high centre of gravity, the equilibrium is unstable and must be held to maintain state. Rolling the stick along the thumb enables 180° of azimuth rotation (like pen barrel rotation). Elevation angle is invariant.

Manipulation Characteristics. In our exploratory interview, participants commented on the extra time required to flip Conté end-over-end. This is similar to a pen nib-to-eraser transition, which takes 1.3 s on average [18]. With Conté, not all contact points are polar opposites; many nearby transitions should be quite fast: moving from a corner to an adjacent edge in a "rolling" adjacency *phrase* [5] for example. Hinckley et al. [15] also observed people "tucking" the pen in their palm to interleave dominant hand touch input. Based on our interviews, Conté's small size should make this even easier. The stable equilibrium of the side contacts also enable Conté to be set down and released (or grip partially loosened) yet maintain the same mode similar to ToolStone [23].

Cognitive Attributes. An open question is whether a user could recall command mappings for 26 different contact points, especially when coupled with rolling adjacency phrases and other parameter contexts such as azimuth angle. Similar strategies to those discussed by ToolStone's authors would increase usability, such as labelling faces for novices and adding tactile patterns for eye-free manipulation [23]. We have not explored this yet, but unlike ToolStone, Conté's greater variety of different contact points have diverse affordances which should assist recall. For example: the small corner size and tripod grip suggests drawing or writing; a medium edge is reminiscent of a highlighter and when oriented vertically, a text insertion point; a long side edge has a sharpness like a cutting blade; the flat ends act like stamps; and when laid on its side, Conté creates a tangible handle [16]. Also, unlike ToolStone, when Conté is used in the dominant hand on a direct input display it is the natural focus of attention. Thus, we can leverage proximal visual feedback to confirm the current mode. We display a rectangular "shadow" outlining the current contact and most activated modes have characteristic visual feedback such as displaying a guideline or rendering a dashed trail for lasso selection.

Design Space

The design space enabled by Conté is a super-set of the spaces demonstrated by commercial tablet pens, ToolStone [23], and Manual Deskterity [15]. At a base level, Conté can function like a pen, but also utilize the affordances and characteristics of different contacts to switch between modes (e.g. corner for drawing, end edge for highlighting).

Touch input can also be interleaved or coordinated with Conté input. For example, Conté could be used like ToolStone [14] in the non-dominant hand to set the mode for dominant hand touch – though contacts which use pen-like grips may be less effective [17]. Perhaps more interesting is *same-hand* touch input enabled by tucking Conté in the palm, setting it down on a stable side, or loosening the grip to free up a finger or thumb. The latter enables a hybrid style of "touch + handle" interaction, in which nearby touch input fine-tunes a mode enabled by Conté.

Conté can also realize the "pen + touch = new tools" design philosophy of Manual Deskterity. In addition to *touch context* techniques where the pen mode is inferred by the object context created by touch, Conté can be used for *pen context* techniques where the mode enabled by Conté contact creates the context for touch. Manual Deskterity's Tape Curve provides one example of this strategy where touch pins a pen stroke down to enter tape drawing mode. Since "the pen always writes" when used alone in Manual Deskterity, there are few possible *pen context* techniques — the only pen context is the stroke. This restrictive design rule makes sense with conventional pens. In contrast, Conté can enable multiple *"pen" context* techniques.

Below are primary qualities which define the total Conté design space covering these different functional styles:

- *Mode Mapping*. Consider precision and physical affordance when assigning contact point to mode (e.g. short edge is a highlighter, long side a ruler, end a stamp).
- *Group Modes*. Keep contact points of task related modes near each other (e.g. creation and revision at each end).
- *Use Adjacency Phrases*. Fast transitions among contacts can enable specialized modes (e.g. edge to corner roll).

- *Parameters for Direct Manipulation*. X-Y position and angle can be used for direct manipulation (e.g. position a guideline), to fine-tune a mode (e.g. pick default option), or to enable multiple modes per face like ToolStone.
- *Parameters at Invocation.* X-Y position and angle can fine-tune modes (e.g. object under contact to form context, angle on contact to pick sub mode).
- *Set It Down.* For stable sides, Conté will maintain the mode while freeing the hand for touch. Including nearby widgets revealed by the mode can further exploit this.
- *Use "Touch + Handle".* By loosening the grip on Conté, a finger or thumb can manipulate nearby widgets.
- *Leverage "Tucking."* Design techniques that interleave touch and Conté with one hand.
- *Set Context for Touch.* As discussed above, Conté's multimodal ability can enable "pen" context modes.
- *Non-Dominant Hand Usage.* Use more stable Conté contacts to set the mode for dominant hand touch.

In a later section, we demonstrate interaction techniques which encompass many of these qualities.

CONTÉ DEVICE PROTOTYPE

Creating a Conté device was challenging for multiple reasons. It was essential to make the device small, which ruled out some approaches. But we also added two more constraints: it must work with an unaltered multi-touch table, and it must be simple enough that researchers and tabletop hobbyists could create their own Conté sticks (we wanted to avoid complex electronics such as MTPen [25]). These constraints ruled out embedding multiple Anoto Pen sensors [4] or using magneto-electric coils like Toolstone [23] since they require too much space and tabletop alterations. Even if advanced magneto-electric pens such as the Wacom 6D Art Pen were compatible with direct touch input, two would have to be combined to support a two-ended Conté stick – and there is still the issue of altering the pressure-activated nib to work on corners, the limited range of elevation angles, and the size of the internal electronics. Instead, Conté emits and reflects infrared (IR) light which is captured by tabletop cameras, and translated by image processing algorithms into Conté input events. Emitting IR light uses simple and small electronics, and any diffuse infrared illumination multi-touch tabletop, such as the Microsoft Surface, can capture reflected IR patterns.

The simplicity of our design does make concessions: we detect and identify 10 out of the 26 possible contacts and we do not sense elevation angle. We detect a corner contact, but not *which* corner; likewise for short, medium, and long edges. Given the tiny surface area of corners and edges, there is little room to create a distinguishing ID pattern for image-based detection (consider that Surface ID tags are 19 × 19 mm). Since digital pens detect only one (or maybe two) contacts, this is still a significant increase and provides enough functionality to test Conté and begin exploring its potential. At the end of this paper, we discuss implementation ideas to detect all 26 contacts, but with added hardware complexity.

Hardware

Our prototype device is 9 × 11.5 × 84 mm and fits comfortably in the palm (Figure 5). It works by emitting IR light from each corner and reflecting IR light from paper labels affixed to the sides. Conceptually simple, we arrived at this design after much experimentation with materials and construction techniques. Two Osram SFH485 LEDs generate 880 nm near-IR light, matching the Surface IR pass band filter. Each LED is partially embedded in a highly polished 9 × 11.5 × 8 mm acrylic block using an open flame (Figure 5d). The acrylic block and LED are wrapped with foil tape to reflect IR light internally. Later, we cut small openings into each corner to let IR light escape (Figure 5c).

To house the electronics and fix the LED-embedded acrylic blocks at the tips, we cast everything in urethane resin using a custom mould (Figure 5b). This method creates a very solid feeling and durable stick, and since our mould was adjustable, it enabled fast prototyping of different shapes and versions. An alternative approach would be to mill a housing out of plastic, or print one using a solid printer, but these require more set up and would require high tolerances for the LED-embedded acrylic blocks.

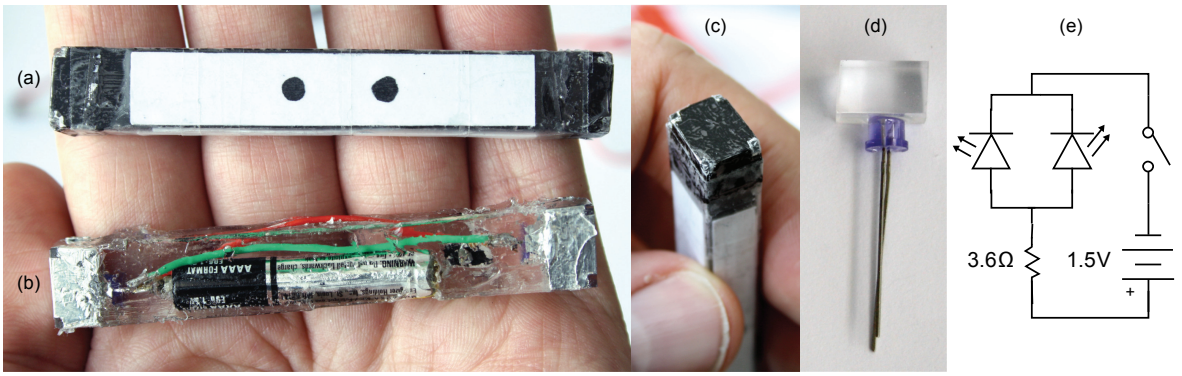


Figure 5. Infrared Conté prototype (shown actual size): (a) completed device; (b) after casting, showing battery and circuit inside; (c) end showing IR "windows"; (d) IR LED embedded in acrylic block; (e) circuit diagram.

After casting, the stick is sanded to fine-tune dimensions and painted matte black to increase the contrast of IR light. Then, small 2 × 2 mm openings are cut into each corner to emit IR light from the LEDs (Figure 5c). On one end, we paint a 3 mm white dot for identification. Finally, we glue foil-backed white paper labels to the side faces, and wrap everything in thin matte transparent tape. The foil backing increases IR reflectivity, and the tape prevents rubbing the paint off as well as protecting the Surface diffuser from scratches. Each label has 3 mm black dots: using the number of dots and size of the label, we can identify the side.

Our basic prototypes are powered by two external batteries supplying 3 V. In spite of being tethered by a thin wire, we found we could manipulate them surprisingly well. However, the protruding wire makes one side face unusable and does reduce manipulative capability somewhat. We found the Osram LEDs require a minimum voltage of 1.4 V, so they can be powered using a single alkaline AAAA battery. At just over 8 mm in diameter, we could embed one and keep a small form factor. With the LEDs wired in parallel and a 3.6 Ω resistor (Figure 5e), enough IR is emitted for tracking. But, compared to the 3 V tethered version, the emitted IR is lower making tracking high velocity movements less reliable (due to camera exposure blurring which we discuss in the next section). Our battery powered Contés operate for about 25 hours. The circuit draws 20 mA (due to battery voltage-current characteristics and LED forward current characteristics) and alkaline AAAA batteries are rated for 500 mAh. Rechargeable NiMH or NiCd AAAA batteries are only rated 1.2 V, so they would require an LED driver (such as the Zetex ZXSC300). We continue to investigate this rechargeable solution.

Unlike Hinckley et al.'s IR pen [15], we do not use tip switches to only activate the LEDs when touching the display. Attaching miniature switches to all contact points would be very difficult. Instead, we found that the combination of pinpoint IR light from the corners, and the Surface's high quality diffuser (Evonik ACRYLITE 7D006), resulted in only slight hover artefacts.

Software

Our software translates the IR light patterns of Conté into events describing the current contact point, position, and, if available, azimuth angle. It is written in C# using the Emgu 2.2.1 (www.emgu.com) OpenCV wrapper. Figure 6 illustrates the following image processing steps.

Pre-Processing. Using the Microsoft Surface SDK 1.0 SP1, we access the 768 × 576 px, 8 bit greyscale image taken by the internal IR cameras. A Top Hat morphological operation is applied using 3 iterations of a 3 × 3 structuring element. This brightens Conté's sharp, bright shapes and darkens the duller smooth contacts of fingers and palms. We found this initial operation to be extremely important.

Position Detection. Next, we locate the approximate size and position of the Conté contact. The pre-processed image is binarized using a *variable* threshold value and then 20 iterations of the Dilate morphological operation is applied, creating large connected blobs. The variable threshold is computed from the velocity of the Conté contact in the previous frame. Specifically, we linearly interpolate between a threshold of 20 when Conté is moving faster than 5 mm/s, and a threshold of 66 when Conté is moving slower than 1 mm/s. When at rest or moving slowly, the high threshold isolates Conté from everything else. With fast movements, the camera exposure blurs the image of the corners or sides, and the threshold must be reduced. Of course, as the threshold is reduced, the intensity of other types of surface contacts will be above the threshold. To address this, we use the binarized image blob closest to the last Conté position — a surprisingly effective and simple rule. Using this approximate contact position, we can update the velocity (which is low-pass filtered with a 0.03 Hz cut off). If the velocity is above 1.4 mm/s, we stop processing and the approximate contact position, together with the last known Conté contact point and azimuth angle, are used to construct a Conté movement event.

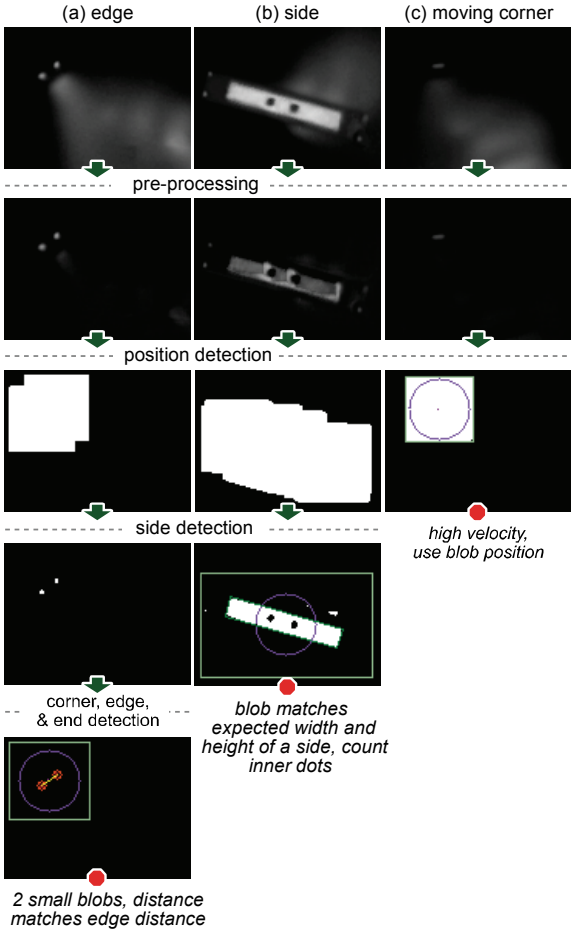


Figure 6. Image processing examples: (a) edge; (b) side; (c) moving corner. Rows show results of an image processing step: capture; pre-processing; position detection; side detection; corner, edge, and end detection.

Side Detection. If the velocity is lower, we continue processing and check whether a side face is touching. The pre-processed image is first binarized using a threshold of 66. Then, the minimum area rectangles of white outer connected component blobs are found (using Emgu `FindContours` and `GetMinAreaRect`). These are compared to the expected width and height of the thick or thin side face labels. If a match is found, the number of black connected component blobs within the matched blob is used to identify *which* side is touching. The minimum area rectangle also provides the azimuth angle of the side.

Corner, Edge, and End Detection. If no matching blobs were found in the side detection step, we search for corners. First, the pre-processed image is binarized using a threshold of 75. Any connected component blobs with an area greater than 16 pixels are removed. The number of remaining blobs and their relative distance determines the contact point: 1 blob is a corner; 2 blobs are an edge if their distance is ± 3 px of 9.5 px for short, 14 px for medium, 100 px for long; and 4 blobs is an end. Any other number of blobs is considered ambiguous and flagged as such. If an end is detected, we apply a lower threshold of 55 to the area between LEDs and determine if there is a white dot which identifies which end is contacting. An improved strategy would be to find the *best* match to all known contacts. For example, 3 blobs are either a corner with the stick held high, or an end with a slight tilt. These could be identified using intra-blob distances, rather than labelled ambiguous.

Performance. The main optimization we made was to restrict the processing area in the *Side Detection* and *Corner, Edge, and End Detection* steps to the contact blob bounding rectangle found in the *Rough Position Detection* step. Our algorithm runs at 45 Hz on a standard Microsoft Surface (Vista, 2.13 GHz dual core CPU, 2 GB RAM).

INTERACTION TECHNIQUE TEST-BED

An obvious application of digital Conté is an artist's sketching application which mimics how a real Conté crayon creates different line thicknesses. We go beyond this direct analogy and demonstrate the potential of Conté more fully in a sketching and drafting inspired application built using C#, WPF, and the Microsoft Surface SDK. Our application has enough complexity to force techniques to be mutually compatible and useful. It includes sketch-based drawing annotation like Manual Deskterity [15], but also demonstrates additional types of precise pen-like input and modal tools like guidelines. Earlier we discussed how Conté can support Manual Deskterity's *touch context* interactions, but we focus on techniques which set Conté apart: unimanual multi-modal input leveraging the characteristics of different contact points and *pen context* techniques where Conté sets the context for touch.

Pen-Like Input

The level of precision and affordance when contacting the corner or end edges suggest one-handed pen-like interactions. The corner is used for freehand drawing, since this is arguably the most precise pen-like contact (Figure 7a). Strokes made with the short end edge are interpreted as shape gestures, a simple solution enabled by Conté for the "ink vs. gesturing" problem [18]. These shape strokes are analyzed with the .NET 4 ink recognizer and replaced by a beautified version (Figure 7b). For example, a circular shape is recognized and replaced with a perfect circle; likewise for ellipses, rectangles, squares, triangles, etc. We use a dashed stroke pattern in the current stroke colour for visual feedback. The medium end edge performs a lasso selection (Figure 7d), a third pen-like input mode. Lasso selections are less precise than drawing and writing, but still require control over shape. A thick black dashed stroke provides visual feedback.

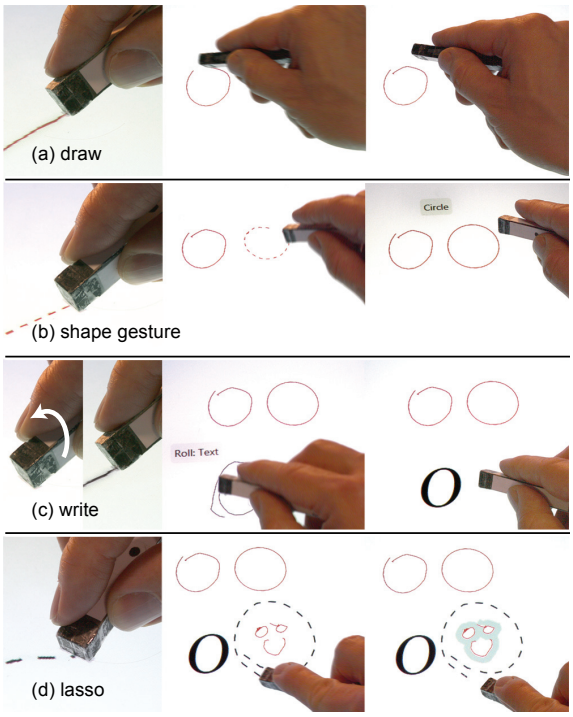


Figure 7. One handed pen-like input modes using corner and end edges: (a) freehand drawing and annotating with the corner; (b) making shape recognition gestures with the short edge; (c) a "roll" from short edge to corner changes corner mode to writing with text recognition; (d) medium edge for lasso selection.

In support of our drafting application scenario, users can enter typographic text by writing with Conté. Like drawing, writing is a precise task requiring fine grain manipulation, so a corner contact is most appropriate. Automatically distinguishing between drawing and writing is unreliable due to many ambiguous strokes (consider strokes resembling an 'O', 'l', or 'L' for example). With Conté, the user explicitly switches between writing and drawing mode. If our prototype Conté device could distinguish between corners, two different corners could be used. Instead, we exploit contact point adjacencies, and use a "roll" from a short end edge to a corner to enter a corner writing mode. A roll is recog-

nized when there is a change to a corner contact less than 1 s after a short end edge contact. We provide visual feedback with a little notification tab which says "Roll: Text" and the stroke colour changes to the current font colour. Corner strokes return to freehand drawing after a subsequent mode change, or when the user explicitly exits with another short end edge to a corner "roll."

Contextual Commands

Contextual commands are common in most applications. In Windows, these are accessed with a right-click on an object and selecting from a context menu (such as 'copy'). In Deskterity, copying is performed by dragging the pen off an object held by a finger. This is an elegant solution, but requires two hands and makes multiple and distant duplication difficult. Moreover, adding more contextual commands means adding more "touch + pen" gestures.

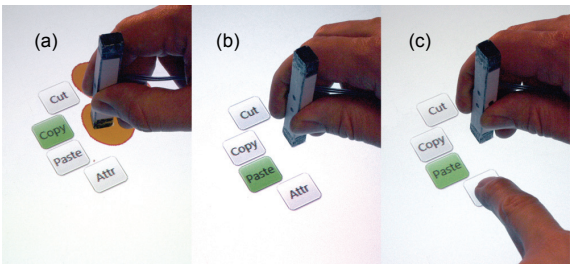


Figure 8. Contextual commands using the end contact: (a) a menu appears with 'copy' as default action since the thick side faces up-down; (b) 'paste' is default when the thick side faces left-right; (c) finger selects other commands which also become the new default for the current thick side orientation.

With Conté, an end contact on an object opens a contextual menu (Figure 8c) showing 'cut', 'copy', 'paste', and 'attr' (paste clipboard object *attributes* only, such as colours, typeface, etc.). The end stamping motion affordance seems to match these actions. To support one-handed operation, we use azimuth angle to *pre-select* a default command when the stick is immediately lifted. For example, when the thick side faces left-right, 'paste' is pre-selected, but when facing up-down, 'copy' (Figure 8a,b). The menu is only revealed after 200 ms, encouraging expert users to quickly access these default commands without visual clutter and enabling novice-to-expert transition. As a further refinement, when a command is selected by a non-dominant finger, it is used as the new default action for the current Conté azimuth orientation. Thus, 'attr' could become the new default when the thick side faces left-right, allowing the user to rapidly paste attributes to multiple drawing objects. Although not implemented, a different context menu could be associated with each end of the stick.

Attribute Palettes

Laying Conté down on one thick side opens one of four attribute palettes (fill colour, stroke colour, font colour, font properties). Since this a very stable contact, Conté can be released (Figure 9c) and an attribute selected with the same hand, or held as the non-dominant hand selects. We placed two touch sensitive buttons just above the stick to cycle through different palettes. With this placement, single-handed, simultaneous Conté and touch manipulation is possible by loosening the grip and using the middle and ring fingers to tap (Figure 9a). By touch-dragging on the palette bezel (Figure 9b), the palette may be "peeled" off Conté to remain visible after Conté is lifted. This can be done with non-dominant fingers or with the dominant hand by loosening the grip, dropping the index finger, and then tucking Conté in the hand. When cycling palettes, peeled palettes are temporarily brought to Conté's location with their peeled location shown as a dashed outline. Palettes are hidden by tapping the same thick side on or near the palette. This has the feeling of "picking up" the palette. We found that with palettes it can be natural to pass Conté to the non-dominant hand similar to ToolStone [23], and the dominant hand is free to touch for attribute selection.

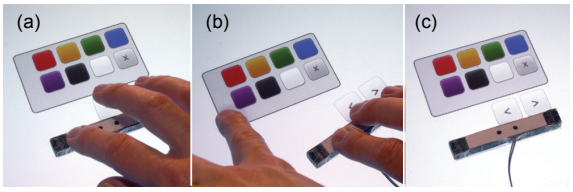


Figure 9. Attribute palettes using a thick side face: (a) one-handed cycling through palettes; (b) "peeling" a palette off to keep it visible; (c) side face is stable, so palette mode is held when Conté is released.

We explored alternate methods of cycling palettes such as using an up or down rolling action from side to adjacent long edge. Theoretically interesting, in practice it was difficult to maintain the necessary grip (partly because of the smooth sides created by wrapping Conté in tape).

Guidelines and Alignment

In commercial applications, guidelines are usually created by dragging off rulers anchored on the edge of the canvas. On a large tabletop this requires reaching, clutters the drawing area, and favours horizontal and vertical guidelines.

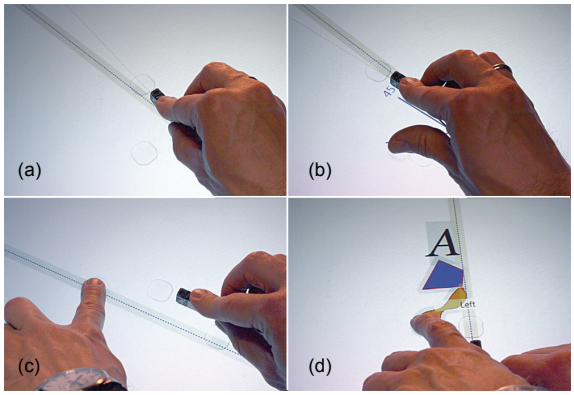


Figure 10. Guidelines using thin side face: (a) revealing and positioning temporary guideline with Conté; (b) adjusting snap angle with thumb; (c) "peeling" temporary guideline to make it permanent; (d) alignment tool, in this case left-aligning objects with a left-swipe.

With Conté, users can create guidelines at any angle by contacting the thin side face (Figure 10a). The stick translates and rotates the guideline, and similar to palettes, they may be peeled off (or perhaps "pinned down") with a dominant or non-dominant touch (Figure 10c). Guidelines are "picked up" by contacting the thin side nearby in the same orientation. Adjusting the guideline snap angle is another example of single-handed, simultaneous Conté and touch manipulation: with a loosened grip, the thumb is free to adjust snap angle by dragging (Figure 10b).

The guideline tool also aligns objects. Tapping or swiping a touch sensitive target just above Conté aligns currently selected objects. Swiping left left-aligns, swiping right right-aligns, and tapping centre-aligns (Figure 10d). While holding Conté, these are most comfortably done with the non-dominant hand, though centre taps are not too difficult with the dominant hand. However, given the stable thin side, the dominant hand can perform a complete alignment task: after positioning a temporary alignment guideline, Conté is released, and still in guideline mode the desired alignment command performed with a tap or swipe, and then the stick picked up to remove the temporary guideline.

Mouse-Like Pointing

Rekimoto and Sciammarella [23] pondered the idea of adding a button to ToolStone to use it as a mouse, and Forlines at al. [9] found that using a conventional mouse on a tabletop can outperform touch for some tasks. Inspired by this, when Conté is laid down on one thick side face it behaves like a standard mouse as it controls an arrow cursor. A stylized image of a mouse is rendered around Conté with its entire surface acting as a single touch sensitive button. This large button accommodates the restricted free finger movements when "clicking," while the remaining fingers continue to grip Conté. With a little practice, we could click and drag objects while maintaining a grip. We create a "mouse-aligned reference frame" [13] by mapping Conté movement vectors in display space to cursor movements. A pointer acceleration function is tuned for aggressive cursor movement with fast movements, but near 1:1 control-display gain when moved slowly. This enables precise selection and minimizes clutching over long distances [6].

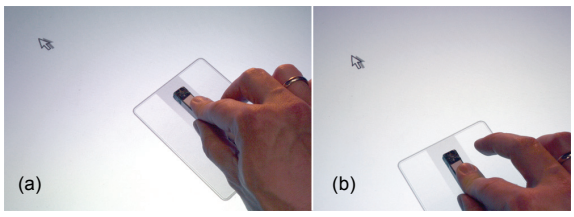


Figure 11. Mouse-like pointing: (a) point; (b) click.

CONCLUSION AND FUTURE WORK

Within the simple idea of recognizing different contacts of a faceted drawing stick lie hardware and software challenges, a new interaction design space, and new human factors questions to investigate. The work discussed in this paper focuses on the first two of these challenges to enable and motivate the third.

Focusing on IR hardware as we have has the advantage of fabrication tractability and tabletop compatibility, but removing these self-imposed constraints enables alternate strategies. One improvement we plan to test is adding a contact microphone to sense movement over the diffuser. If the sound profiles are distinct, this could act like a primitive tip switch. With a switchable diffuser [14] and colour-coded Conté faces, all 26 contacts can be recognized with basic image processing and a model of contact colour patterns. We have already implemented this strategy using a clear tabletop for a future quantitative manipulation experiment. By accepting more hardware complexity, accelerometers and/or a gyroscope could be embedded to accurately measure orientation, and the contact inferred similar to ToolStone. Diffuse or emitted IR light may continue to provide X-Y position information, or for a capacitive touch device, an outer layer of conductive material could be used.

Enabled by a hardware solution, and motivated by demonstrations of an interaction design space, we can turn to human factors. The many characteristics of Conté must be verified and quantified, and we are already planning an investigation of contact-to-contact transitions. Like Conté itself, this appears straightforward, but closer analysis reveals 54 different transitions (accounting for symmetry) which need to be contrasted and compared. This will complement Li et al.'s [18] work and help situate Conté among other mode switching techniques. Other investigations include precision, angular manipulation, grip styles, and recall capability: can people use 26 or more mappings? If not, how many? Related are ergonomic studies to test alternate form factors, construction materials, and tactile patterns.

The interaction techniques described here demonstrate key qualities of the Conté design space, but they may only hint at the potential. For example, some qualities can be pushed farther, such as leveraging "tucking," and others we have not addressed directly like switching Conté between hands. Art historians have argued that the invention of the versatile Conté crayon influenced a new style of art [24]. At the risk of sounding overly enthusiastic, perhaps digital Conté could have a similar effect on human-computer interaction.

ACKNOWLEDGEMENTS

This work was partially supported by the SHIVA project (Interreg IV A −2 seas) and FEDER.

REFERENCES

1. Benko, H., Wilson, A.D., and Baudisch, P. Precise selection techniques for multi-touch screens. *Proc. of the SIGCHI conference on Human Factors in computing systems*, ACM (2006), 1263-1272.
2. Bi, X., Moscovich, T., Ramos, G., Balakrishnan, R., and Hinckley, K. An exploration of pen rolling for pen-based interaction. *Proc. of the 21st annual ACM symposium on User interface software and technology*, ACM (2008), 191-200.
3. Brandl, P., Forlines, C., Wigdor, D., Haller, M., and Shen, C. Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces. *Proc. of the working conference on Advanced visual interfaces*, ACM (2008), 154-161.
4. Buxton, W.A.S. Chunking and phrasing and the design of human-computer dialogues. In *Human-computer interaction: toward the year 2000*. Morgan Kaufmann Publishers Inc, 1995, 494-499.
5. Casiez, G., Vogel, D., Balakrishnan, R., and Cockburn, A. The Impact of Control-Display Gain on User Performance in Pointing Tasks. *Human-Computer Interaction 23*, 3 (2008), 215-250.
6. Deming, K. and Lank, E. *Mode Selection Techniques for Pen Input Systems*. San Francisco State University, 2005.
7. Fitzmaurice, G., Baudel, T., Kurtenbach, G., and Buxton, B. A GUI paradigm using tablets, two-hands and transparency. *CHI '97 extended abstracts on Human factors in computing systems: looking to the future*, ACM (1997), 212–213.
8. Forlines, C., Wigdor, D., Shen, C., and Balakrishnan, R. Direct-touch vs. mouse input for tabletop displays. *Proc. of the SIGCHI conference on Human factors in computing systems*, ACM (2007), 647–656.
9. Frisch, M., Heydekorn, J., and Dachselt, R. Investigating multi-touch and pen gestures for diagram editing on interactive surfaces. *Proc. of the ACM International Conference on Interactive Tabletops and Surfaces*, ACM (2009), 149–156.
10. Goonetilleke, R.S., Hoffmann, E.R., and Luximon, A. Effects of pen design on drawing and writing performance. *Applied Ergonomics 40*, 2 (2009), 292-301.
11. Greenberg, S. and Buxton, B. Usability evaluation considered harmful (some of the time). *Proc. of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ACM (2008), 111–120.
12. Guiard, Y. Asymmetric division of labor in human skilled bimanual action: the kinematic chain as a model. *Journal of Motor Behavior 19*, 4 (1987), 486-517.
13. Hartmann, B., Morris, M.R., Benko, H., and Wilson, A.D. Augmenting interactive tables with mice & keyboards. *Proc. of the 22nd annual ACM symposium on User interface software and technology*, ACM (2009), 149–152.
14. Hilliges, O., Izadi, S., Wilson, A.D., Hodges, S., Garcia-Mendoza, A., and Butz, A. Interactions in the air: adding further depth to interactive tabletops. *Proc. of the 22nd annual ACM symposium on User interface software and technology*, ACM (2009), 139-148.
15. Hinckley, K., Yatani, K., Pahud, M., et al. Pen + touch = new tools. *Proc. of the 23nd annual ACM symposium on User interface software and technology*, ACM (2010), 27–36.
16. Jordà, S., Geiger, G., Alonso, M., and Kaltenbrunner, M. The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces. *Proc. of the 1st international conference on Tangible and embedded interaction*, ACM (2007), 139–146.
17. Kabbash, P., MacKenzie, I.S., and Buxton, W. Human performance using computer input devices in the preferred and non-preferred hands. *Proc. of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, ACM (1993), 474-481.
18. Li, Y., Hinckley, K., Guan, Z., and Landay, J.A. Experimental analysis of mode switching techniques in pen-based user interfaces. *Proc. of the SIGCHI conference on Human factors in computing systems*, ACM (2005), 461-470.
19. Lin, J., Newman, M.W., Hong, J.I., and Landay, J.A. DENIM: finding a tighter fit between tools and practice for Web site design. *Proc. of the SIGCHI conference on Human factors in computing systems*, ACM (2000), 510-517.
20. Moran, T.P., Chiu, P., and Melle, W. van. Pen-based interaction techniques for organizing material on an electronic whiteboard. *Proc. of the 10th annual ACM symposium on User interface software and technology*, ACM (1997), 45-54.
21. Napier, J.R. The prehensile movements of the human hand. *The Journal of Bone and Joint Surgery. British Volume 38-B*, 4 (1956), 902-913.
22. Ramos, G., Boulos, M., and Balakrishnan, R. Pressure widgets. *Proc. of the SIGCHI conference on Human factors in computing systems*, ACM (2004), 487-494.
23. Rekimoto, J. and Sciammarella, E. ToolStone: effective use of the physical manipulation vocabularies of input devices. *Proc. of the 13th annual ACM symposium on User interface software and technology*, ACM (2000), 109-117.
24. Scheyer, E. French Drawings of the Great Revolution and the Napoleonic Era. *The Art Quarterly IV*, 4 (1941), 187-204.
25. Song, H., Benko, H., Guimbretière, F., Izadi, S., and Cao, X. Grips and gestures on a multi-touch pen. *Proc. of the SIGCHI conference on Human factors in computing systems*, ACM (2011).
26. Tian, F., Xu, L., Wang, H., et al. Tilt menu: using the 3D orientation information of pen devices to extend the selection capability of pen-based user interfaces. *Proc. of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, ACM (2008), 1371-1380.
27. Vogel, D. and Baudisch, P. Shift: a technique for operating pen-based interfaces using touch. *Proc. of the SIGCHI conference on Human factors in computing systems*, ACM (2007), 657-666.
28. Wu, F. G. and Luo, S. Design and evaluation approach for increasing stability and performance of touch pens in screen handwriting tasks. *Applied Ergonomics [Kidlington] 37*, 3 (2006).
29. Wu, M., Shen, C., Ryall, K., Forlines, C., and Balakrishnan, R. Gesture Registration, Relaxation, and Reuse for Multi-Point Direct-Touch Surfaces. *Proc. of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, IEEE Computer Society (2006), 185–192.
30. Yee, K-P. Two-handed interaction on a tablet display. *CHI '04 extended abstracts on Human factors in computing systems*, ACM (2004), 1493-1496.
31. Computer Sketchpad. *Science Reporter (Television Series)*, 1964.

Estimating the Perceived Difficulty of Pen Gestures

Radu-Daniel Vatavu[1], Daniel Vogel[2,3], Géry Casiez[2], and Laurent Grisoni[2]

[1] University Stefan cel Mare of Suceava, Romania
[2] LIFL, INRIA Lille & University of Lille, France
[3] Mount Allison University, Canada
vatavu@eed.usv.ro, dvogel@mta.ca, gery.casiez@lifl.fr,
laurent.grisoni@lifl.fr

Abstract. Our empirical results show that users perceive the execution difficulty of single stroke gestures consistently, and execution difficulty is highly correlated with gesture production time. We use these results to design two simple rules for estimating execution difficulty: establishing the relative ranking of difficulty among multiple gestures; and classifying a single gesture into five levels of difficulty. We confirm that the CLC model does not provide an accurate prediction of production time magnitude, and instead show that a reasonably accurate estimate can be calculated using only a few gesture execution samples from a few people. Using this estimated production time, our rules, on average, rank gesture difficulty with 90% accuracy and rate gesture difficulty with 75% accuracy. Designers can use our results to choose application gestures, and researchers can build on our analysis in other gesture domains and for modeling gesture performance.

Keywords: gesture-based interfaces, pen input, gesture descriptors.

1 Introduction

There are three primary factors which contribute to a successful gesture-based interface: the acquisition technology, the recognizer, and the design of the gesture set. Technologies to acquire gestures [7,9,16,25], and gesture recognition algorithms [10,12,26], are now quite robust and widely available. However, developing techniques and criteria to help designers create an intuitive and easy-to-perform gesture set remain an active area of research. The challenge is that in order to successfully integrate into an application, a gesture has to satisfy multiple criteria: it must be unambiguously recognized [2,13,14]; fit well with its associated function [16,17,25]; be easy to learn and recall [17]; and be efficient to perform [16,17,25].

Researchers have offered two different strategies to assist designers. The first is to use predictive models to analytically evaluate candidate gestures. These have been successful for evaluating recognition ambiguity [2,14] and have made progress towards predicting actual performance time [4,8]. Unfortunately, creating accurate predictive models for non-recognition criteria such as performance time is difficult due to the complexity of gestural motion and criteria interdependencies — factors which are also influenced by an individual user's cognitive ability, physical skill, and cultural context. For these

P. Campos et al. (Eds.): INTERACT 2011, Part II, LNCS 6947, pp. 89–106, 2011.

reasons, researchers have proposed a second strategy using formal user studies for participatory design and gesture set evaluation [2,16,17,19,25]. Involving users in any design process is a good idea, but the effort to plan, run, and analyze these kinds of studies is large compared to using a predictive model.

We offer a practical solution in-between a model and a user study. Based on an estimate of actual production time, we found that designers can reasonably estimate user's perceived gesture *execution difficulty*. The notion of difficulty encompasses multiple criteria including the ease with which a gesture may be learned, remembered, and performed. This notion of difficulty has been mentioned in previous work [16,17,25], but there has been no previous attempt to examine it in detail or estimate it. In an experiment using single stroke pen gestures, we elicited a difficulty classification rating and a relative difficulty ranking from participants. Based on data from a second validation experiment, our results show that the difficulty ranking can be predicted with greater than 93% accuracy using measured production time and 87% using the Isokoski first-order predictive production time model [8]. Using a Bayes classification rule and measured production time, we can also classify the difficulty rating with 83% accuracy. Since the times predicted by the CLC predictive model [4] reduced the accuracy of our classification rule to 25%, we analyzed an alternative approach. We found that production time can be reasonably estimated by gathering a few samples of actual production time – a set of data which may already exist for the purpose of training a gesture recognizer. With three people supplying three gesture samples, our classification rule achieved 75% accuracy on average and increased the average accuracy of the estimated difficulty ranking to 90%.

Our findings that gesture difficulty can be predicted from production time, together with our results regarding the reasonable estimation of production time based on a very small set of data, provide designers with a general measurement encompassing multiple criteria to assess gesture sets without a full formal user study.

2 Previous Work

Creating a successful gesture-based interface is challenging. Once a vocabulary of gestures moves beyond a small set of directional strokes, it becomes more difficult to learn, remember, and use [11]. Techniques exist which assist with recall and help to transition users from novice to expert: examples include crib-sheet diagrams [11] and dynamic path guides [3]. While these techniques are effective, they assume that a good gesture set has already been created.

2.1 Gesture Design Tools

One way to make the designer's job easier is to use a gesture design tool. An example is Appert and Zhai's Stroke Shortcuts Toolkit [1] which includes a simple tool with a predefined dictionary of stroke primitives. The hope is that a designer's creativity is stimulated with a "structured design space that can be systematically explored". Long et al.'s Quill gesture design tool [13,14] goes further by providing metrics to help designers evaluate potential gesture sets. The metrics relate to recognition rate, and conveyed through values such as classification distance or visualized as confusion

matrices. Ashbrook and Starner's MAGIC tool [2] introduces gesture *goodness* as a metric. In an evaluation, this seemingly abstract metric was useful as a quantitative guideline compared to a specific breakdown of individual measures (such as inter-class variability graphs). However, goodness is also closely related to recognition rate. Although participants were also asked to design gestures that would be easy to remember, perform, and be socially acceptable, MAGIC, like Quill, does not provide any quantitative feedback for these criteria.

2.2 Models

Producing quantitative measurements to represent other criteria requires predictive models. For example, Long et al. [15] developed a model for predicting the perceived visual similarity of two gestures. Their model was generated by selecting a subset of geometric and dynamic features of gesture trajectories, and looking for a correlation with experimentally determined user rated visual similarity. The final model could predict visual similarity of two gestures reasonably well (correlated R^2=.56 with ground truth). One application is increasing recognition rate by avoiding ambiguous gestures, but the authors also argue that a visual similarity metric may be used to improve a gesture's fit with its function. For example, designers could assign visually similar gestures to similar operations (such as scroll up and scroll down), and dissimilar gestures to more abstract tasks such as cut and paste.

Isokoski [8] introduced a model to predict the relative ordering of gesture production times based on geometric complexity. The model sums the minimal number of straight segments needed to maintain a human recognizable shape in the gesture. This sum is interpreted as a complexity number and can be used as a first-order ranking of gesture production time: the model ranked production times of *Unistroke* characters with R^2=.85. Although there is ambiguity in the definition and calculation method, Isokoski's model has the advantage of being conceptually simple.

Cao and Zhai's [4] Curves, Lines and Corners (CLC) model goes beyond Isokoski by attempting to predict the actual production time of a single stroke gesture. After decomposing a gesture into curved and straight segments, the model calculates individual production times for curves based on Viviani's 2/3 power law of curvature [22] and a simple power term based on the length of straight lines (no time is calculated for corners, they are only used to segment lines and curves). The authors found that CLC works very well as a first order predictor (correlations with test data had R^2>.90), but over- or under-predicted arbitrary gestures times by 30% and over-predicted *Unistroke* and *Graffiti* gestures by more than 40%. Castellucci and MacKenzie also noted this type of performance for CLC [5]. Cao and Zhai attribute this behaviour to the model's inability to compensate for unfamiliar and little practiced gestures, or familiar and well-practiced gestures.

2.3 User Studies

Rather than rely on predictive models, researchers have suggested that user studies should be used to assist in the design and evaluation of gesture sets. For example, Nielsen et al. [17] provide a user-centered procedure to design whole-hand gestures. The procedure requires two user studies, an initial study to gather user input to inform

design and a subsequent study to evaluate. In a case study application, they report they were able to obtain a good gesture set, but the procedure was very time consuming. Also, key stages such as the generation of scenarios must be carefully prepared or else results may be substandard.

Wobbrock et al. [25] take a participatory design approach by eliciting a gesture set from users. Using wizard-of-oz techniques, they asked users to mimic the best multi-touch gesture to match a demonstrated action such as scale, rotate, move, etc. The study, as well as a follow-up [16], also gathered rankings for each candidate gesture's intuitiveness and ease-of-execution. Perhaps surprising, but the authors report that gestures which experienced designers propose are not always preferred by users [16].

2.4 Summary

Ideally, the best way to design an intuitive and easy-to-perform gesture set is to involve users like Nielsen et al. [17] and Wobbrock et al. [25] since even experienced designers cannot predict user preference [16]. But, faced with the large amount of effort required to plan, run, and analyze these studies [17], perhaps there is a way for designers to evaluate candidate designs using predictive models and/or minimal user data. Long et al.'s [15] visual similarity predictive model is interesting since it can guide designers with a gesture's fit with a function. Isokoski [8], and Cao and Zhai [4], have made progress towards estimating actual gesture production time, a measure which should directly relate to how efficient a gesture is to perform. However, Cao and Zhai [4] and Castellucci and MacKenzie [5] note that production time is a partial function of many factors and therefore an accurate predictive model remains elusive. Inspired by Long et al., as well as Ashbrook and Starner's success with a seemingly abstract post-hoc measure of goodness [2], we focus on a measure of *execution difficulty*.

3 Experiment 1: Measuring Execution Difficulty

The notion of execution difficulty (or the converse, ease-of-execution) is frequently mentioned [2,14,15,17] and has been measured for multi-touch gestures with post-experiment surveys [16,25], but there has been no attempt to estimate it a priori. Morris et al. associate difficulty with "carrying out the gesture's physical action" [16]. Carrying out an action refers directly to efficiency of performance, but also involves a cognitive process which relates to how easy a gesture is to learn and recall [4]. Thus, execution difficulty is a general quantitative measure which combines multiple design criteria: learn-ability, recall, and performance. More abstract measures, such as goodness [2] and general preference [26] may include additional criteria (such as social acceptability [19]), but the more general the measure, the more abstract it is due to more complex relationships of the underlying criteria. The challenge is how to estimate execution difficulty given a candidate gesture or gesture set, with the knowledge that it encompasses criteria which are known to be difficult to predict.

In the first experiment, we measure perceived execution difficulty for a set of single stroke gestures. If there is significant agreement across participants, then it is likely to be an intuitive measure suitable for a priori estimation. Using the participants'

movement logs and the geometric gesture shapes, we compute quantitative measures ("descriptors") and test these for correlations with the participants' responses. If well correlated descriptors exist, and they can be estimated or computed directly, then designers have a way to estimate perceived difficulty of candidate gesture designs.

Participants

14 right-handed people (3 females) participated in the experiment (mean age 21 years, SD 1). 11 out of 14 participants had no pen-based interface experience.

Apparatus

Gestures were entered using a 17 inch (431 mm) Wacom DTU-710 Interactive Pen Display running at a resolution of 1280 x 1024 px (pixel pitch 0.264 x 0.264 mm) and capable of capturing pen input at 133Hz. The display was positioned horizontally to approximate a physical pen and paper context. A 2.4GHz computer ran a C# full screen application. The participant entered gestures in a 420 x 420 px (110 x 110 mm) square box centered in the display (Fig 1).

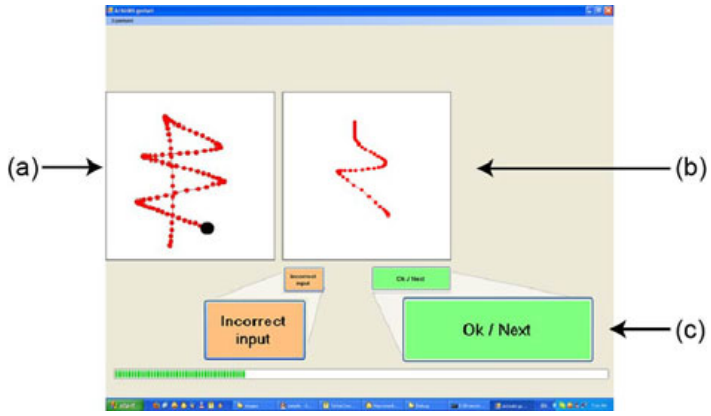


Fig. 1. Experiment Application: (a) current gesture to perform; (b) gesture input area; (c) post-entry choice buttons

Task

Each trial began with the path of the current gesture to be entered shown on the left side of the display (Fig 1a). Participants were instructed to enter a continuous stroke for the gesture and to balance speed and accuracy. After performing the gesture, two buttons were enabled representing a choice between flagging their input as incorrect or continuing to the next gesture. Participants were instructed to flag a stroke as incorrect if the shape they entered was different from the target gesture, or if some accidental input occurred such as the pen slipping or moving unevenly. This was logged as an input error and the participant was asked to re-execute the gesture. Like Wobbrock et al. [25], we wanted our participants to decide whether a gesture was similar to the template, avoiding any confounding effects due to the behavior of a recognizer. As an extra precaution, all participant executions were visually inspected by the authors and confirmed that they were correctly entered.

Gesture Set

There were 18 different single stroke gestures (Fig 2). The set contains 9 gestures designed to be *familiar* (i.e. letters and shapes used in everyday writing) and 9 gestures designed to be *unfamiliar* (e.g. the *twirl-omega* and *flower* shapes may appear familiar, but are unlikely to be practiced as a pen stroke, while *steep-hill* and *triangles-chain* are completely new shapes). As discussed earlier, Cao and Zhai [4] argue that familiarity affects actual performance time due to practice. The idea is that a more practiced gesture will result in a lower performance time in spite of high objective geometric complexity. For example, although the letter *g* is a rather complex series of twists and 180-degrees turns, it would be difficult to reproduce initially; but, with practice it can be executed very quickly. Since practice also relates to how easy a gesture is to learn and recall, *familiarity* is likely to relate to execution difficulty. We expected that more familiar gestures will be rated as easier to perform, even if they have high objective complexity.

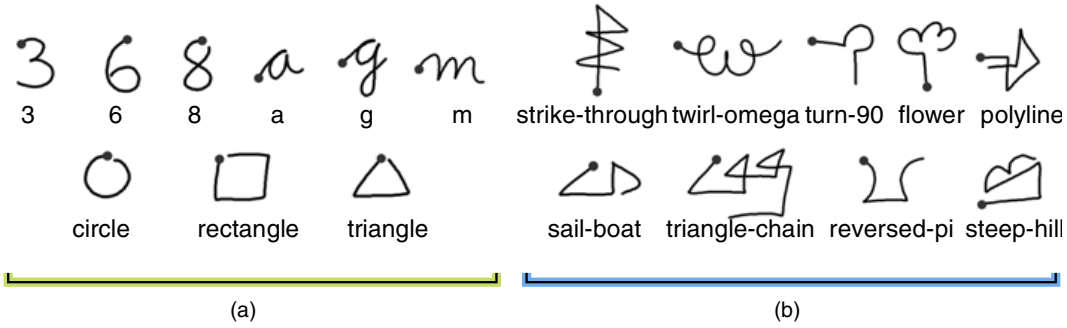


Fig. 2. The 18 gestures used in the experiment: (a) left 9 designed to be familiar; (b) right 9 designed to be unfamiliar

Design

Each participant executed each gesture 20 times, with the 18x20 = 360 gestures presented in random order. The number of repetitions (20) was chosen larger than the current practice when eliciting gestures from users, be it for training gesture recognizers [26] or even for deriving performance models [4]. We purposely did this to ensure motor learning for all gestures so that participants would reach execution automaticity. Participants were allowed to take as many breaks as they wished. The experiment took approximately 40 minutes.

Post-Experiment Questionnaire

After the experiment, participants answered a short questionnaire regarding their perceived execution difficulty when performing the gestures. We gathered this information in two different ways: an individual execution difficulty *Rating* for each gesture using a 5-point Likert scale; and an ordered *Ranking* of all gestures according to relative execution difficulty. The 5-point Likert scale rating question (Table 1) was presented as a 5 column table: participants entered ratings for the 18 gestures in any order they chose. Participants were asked to enter the rating by drawing the gesture in the column corresponding to the desired Likert rating. We hoped this would allow

them to re-enact the gesture performance and make visual inspection easy. They could modify previous ratings at any time until they were confident of their final choices.

Table 1. Likert questions used to elicit execution difficulty *Rating*

Likert rating	Associated explanation
1. very easy to execute	I executed these gestures immediately and effortlessly with absolutely no need to pay attention
2. easy to execute	I executed these easily, almost without paying attention
3. moderate difficulty	I occasionally paid special attention during execution
4. difficult to execute	I paid special attention with each execution
5. very difficult to execute	I had to concentrate for each execution. There were times when I did not get the right shape from the first attempt

The ordered ranking of all gestures according to ascending execution difficulty was completed after the Likert rating. This enabled participants to use the rating classes to assist with this otherwise difficult task. As before, we asked them to draw the gestures in order to revisit relative differences in difficulty as they completed the ranking.

We also asked participants to explain their perception of gesture difficulty: what they found difficult or easy for each gesture execution. Finally, we asked them to identify which shapes they found *Familiar* (they had seen and practiced before) in order to test our choice for familiar and unfamiliar gestures.

4 Results

We found a high degree of agreement between participant *Rating* of execution difficulty (Kendall's W=.78[1], $\chi^2(17)=185.60$, $p<.001$). The agreement was even stronger for *Ranking* which participants commented as being a difficult task (W=.82, $\chi^2(17)=195.17$, $p<.001$). Both coefficients are well above 0.5 indicating our sample size was appropriate with a large Cohen effect. Since *Rating* was designed to be used as a first approximation for *Ranking*, there was a significant correlation between their median ratings ($\rho_{(N=18)}=.97$, $p=.01$).

Fig 3 illustrates the median *Rating* and *Ranking* ratings for each gesture. A repeated-measures Friedman's ANOVA was used in order to test the influence of gesture type (nominal with 18 cases) over *Rating* and *Ranking*. The results showed a significant effect of *Gesture* over both *Rating* ($\chi^2(17)=185.60$) and *Ranking* ($\chi^2(17)=195.17$, at $p<.001$).

Across all 14 participants there were 17 deviations (6.7% of the total responses) from our gestures set's assumed *Familiarity*. 14 deviations were assumed unfamiliar gestures: 7 participants found the *twirl-omega* gesture familiar, 4 *reversed-pi*, 2 *flower*, and one participant said the *sail-boat* and *steep-hill* were also familiar. The

[1] Kendall's coefficient of concordance W in [0..1] where 0 denotes no agreement at all and 1 represents absolute agreement.

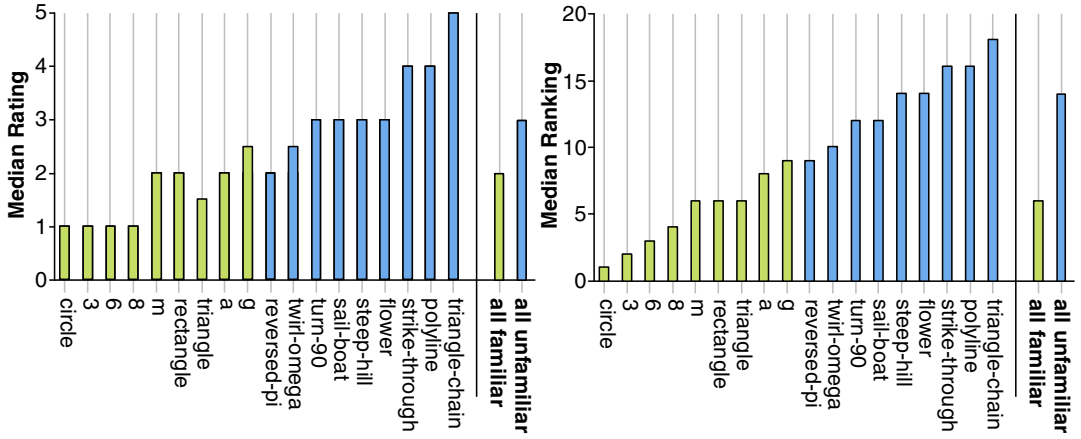


Fig. 3. Left: median gesture *Rating* (higher *Rating* values were perceived to be more difficult to execute). Right: median gesture *Ranking* (higher numerical *Ranking* for gestures perceived to be more difficult to execute). In both graphs, gestures are ordered by ascending *Ranking*.

latter also noted that the assumed-to-be-familiar gestures *a* and *g* were unfamiliar because the starting point was not in the same location where they usually start those letters. As part of their comments regarding their perception of gesture difficulty, three participants noted the same issue of starting position with *a* and *g* and one participant with *8*, but they did not feel this made them unfamiliar. This relates to the problem of allographic variation in handwriting where individual differences in the formation of character shapes pose problems for handwriting recognizers [21]. Aside from *twirl-omega* where *Familiarity* deviations occurred with half of our participants, our assumed gesture familiarity was reasonable. We could treat these deviations as outliers since they represent less than 4% out of the total responses, but when possible *Familiarity* related analysis is based on actual participant responses.

The median *Ranking* and *Rating* across *all familiar* and *all unfamiliar* gestures (Fig 3) are significantly different according to a Wilcoxon signed-rank test ($z_{(N=14)}$=-3.402, p=.001 for *Rating* and $z_{(N=14)}$=-3.400, p=.001 for *Ranking*, both with large effects, r=-.64). These 9 assumed familiar gestures are among the 11 gestures assigned to the easiest *Rating* levels**,** and are among the lowest 10 gestures in ascending difficulty *Ranking* (Fig 3). The *twirl-omega* and *reversed-pi* (two out of three contentiously unfamiliar gestures) also share the two easiest median *Rating* levels, and *reversed-pi* has the same median ranking as the familiar gesture *g*.

5 Towards Estimating Execution Difficulty

Given the high agreement of perceived execution difficulty *Rating* and *Ranking* in experiment 1, we can search for a way to estimate difficulty in the absence of a formal experiment. Essentially, if a correlation exists with one or more characteristic gesture descriptors, then those descriptors can be used to estimate execution difficulty. We examined many potential descriptors (Table 2): all of Rubine's static geometric descriptors and measured quantities [20], the additional geometric descriptors used by Long et al. [15], Hu invariant spatial curve moments commonly used in image processing for contours and shapes [18](p. 606), Isokoski's complexity measure [8], and the production time predicted by Cao and Zhai's CLC model [4].

Table 2. Descriptors (bold indicates significant correlation with *Rating* or *Ranking*)

Rubine's set [20]: Geometric	
1. Cosine of initial angle (cosine1)	7. Sine of angle between first and last points (sine2)
2. Sine of initial angle (sine1)	**8. Total length**
3. Size of bounding box (bbox size)	9. Total turning angle
4. Angle of bounding box (bbox angle)	**10. Total absolute turning angle (turn angle)**
5. Distance between first and last points	**11. Sharpness or (energy)**
6. Cosine of angle between first and last points (cosine2)	
Rubine's set [20]: Measured	
12. Production Time (time)	
13. Speed	
Long et al.'s visual similarity set [15]: Geometric	
14. Aspect	**18. Size of bounding box (density2)**
15. Total angle traversed / total length	19. Openness
16. Total angle / total absolute angle	**20. Area of bounding box (bbox area)**
17. Distance between first and last points (density1)	
Hu invariant spatial moments [18, p.606]: Geometric	
21 – 27. Hu1, **Hu2**, Hu3, Hu4, Hu5, Hu6, Hu7	
Model predictions	
28. CLC Predicted Production Time [4]	
29. Isokoski's complexity measure [8]	

The calculation of the Rubine, Long et al., and Hu descriptors are straightforward to apply to the geometric shape of the gesture, given the descriptions and equations in the cited works. We computed these measurements using two representations of geometric gesture shapes. To approximate a design scenario where the gestures have been drawn, but not performed, we used the target gesture shapes displayed in the left panel of our experimental application (i.e. the vector drawings in Fig 1). We will refer to these as geometric descriptors using *Drawn* representations. We also computed mean descriptors using the actual gesture geometries as performed by the participants in our experiment. Theoretically, this is a best case scenario for geometric descriptor performance, but with the potential issue of overfitting. We will refer to these geometric descriptors using *Performed* representations. Both *Drawn* and *Performed* representations were preprocessed similar to previous work [4,10,26] by normalizing without deformation, centering on the origin, and re-sampling uniformly into n=32 points. To calculate the CLC predicted production time, we used the PlayCLC program[2]. As noted earlier, the definition and calculation of Isokoski's complexity measure is ambiguous. By studying examples [8](p. 360) we developed quantitative guidelines to perform the necessary reduction of arcs into line segments: if the angle α inscribed by an arc was greater than 270° use 3 segments; if α<120° use 1 segment; otherwise use 2 segments. We could verify these guidelines with our *3* and *circle* shapes, also included in Isokoski's examples.

Note that all descriptors based on geometry are static and will not change with practice. For example, a geometrically complex, but familiar gesture such as *g* may have a lower *Rating* compared to a geometrically simple, but unfamiliar gesture such

[2] From http://www.cs.toronto.edu/~caox/PlayCLC/PlayCLC.htm

as *sail-boat*. Rubine's Production Time and Speed descriptors are measured, i.e. they are computed from data gathered during actual gesture performance, so they include effects for practice. Of course, using this type of post-hoc measure for a-priori prediction seems paradoxical. Our initial rationalization is that some future model may be able to accurately predict these measures (such as an improved CLC model for Production Time), and we show later that the relevant measure of Production Time can be approximated with a very small set of informally gathered user data.

All of the potential descriptors in Table 2 were tested for correlations with execution difficulty *Rating* and *Ranking*. This was done overall, as well as separately with familiar and unfamiliar gesture groups. Descriptors with at least one significant Spearman correlation coefficient are listed in Table 3 (for geometric descriptors using *Drawn* representations in Fig 1) and Table 4 (for geometric descriptors using participant *Performed* representations).

Table 3. Correlations of geometric descriptors using *Drawn* representations. Spearman correlation of descriptor with median *Rating* and *Ranking* in descending order of overall *Rating* coefficients; coefficients are reported at p = .01 (**) and p = .05 (*) significance levels; N = 18 for all, N = 9 for familiar and N = 8 for unfamiliar gestures (*twirl-omega* was excluded). The largest coefficient in each column is shown in bold text (two bold coefficients in the same column are not significantly different).

	all		**familiar**		**unfamiliar**	
	Rating	*Ranking*	*Rating*	*Ranking*	*Rating*	*Ranking*
bbox size	**.78****	**.75****	n.s.	n.s.	**.93****	**.98****
bbox area	.72**	.70**	n.s.	n.s.	**.93****	**.98****
length	.60**	.60**	n.s.	**.67***	**.79***	**.86****
cosine2	n.s.	n.s.	**.73***	**.81****	-.86*	-.88**
density1	n.s.	n.s.	n.s.	n.s.	**.73***	.81*
Hu2	n.s.	-.50*	n.s.	n.s.	n.s.	n.s.

Production time has the highest correlations with *Rating* and *Ranking* overall; and, in all but one case, it is among the highest correlations when tested separately with familiar and unfamiliar gesture groups. Speed had the second highest (negative) correlation when all gestures were considered together, but not significant when tested separately with familiar and unfamiliar.

Note that there is evidence that production time should be a scale invariant.Viviani and Terzuolo [23] found that execution times for single strokes in handwriting are scale invariant. If we accept that a single stroke gesture is similar, then scale invariance should not be problematic. Isokoski [8] also provides additional evidence with his observation that average velocity increases with longer strokes.

In many cases, descriptors based on geometry had significantly lower correlation coefficients compared to measured values. An exception is length, which has all significant coefficients in Table 4 and all but one in Table 3. In the case of familiar gestures in Table 4, coefficients for length, along with density2, and cosine2 are not significantly different from actual production time. Although not significantly highest, Isokoski's complexity and two bounding box descriptors in Table 4 correlate reasonably well when all gestures were considered together, but are not even

Table 4. Correlations of geometric descriptors using *Performed* representations. Correlations reported as in Table 3

	all		**familiar**		**unfamiliar**	
	Rating	*Ranking*	*Rating*	*Ranking*	*Rating*	*Ranking*
time	**.95****	**.96****	**.94****	**.84****	.79*	**.91****
-speed	**.87****	**.85****	n.s.	n.s.	n.s.	n.s.
length	.80**	.82**	**.94****	**.90****	.72*	**.81***
bbox size	.77**	.82**	n.s.	n.s.	n.s.	n.s.
Isokoski	.74**	.71**	**.70***	n.s.	n.s.	**.79***
bbox area	.70**	.75**	n.s.	n.s.	n.s.	n.s.
density2	.56*	.52*	**.90****	**.85****	.72*	**.76***
turn angle	.53*	.51*	n.s.	n.s.	.72*	**.83***
Hu2	-.48*	-.47*	n.s.	n.s.	n.s.	n.s.
CLC	.47*	n.s.	n.s.	n.s.	n.s.	**.79***
aspect	-.47*	n.s.	n.s.	n.s.	n.s.	n.s.
cosine2	n.s.	n.s.	**.86****	**.71***	**-.86****	-.88**
density1	n.s.	n.s.	n.s.	n.s.	.79*	**.86****
energy	n.s.	n.s.	n.s.	n.s.	n.s.	**.79***

significant when tested separately with familiar and unfamiliar. In Table 3, some geometric descriptors such as the two bounding box descriptors and cosine2 correlate very well with unfamiliar gestures. Intuitively, the larger gestures may be more complex, and thus be more difficult to execute, but the high correlation of cosine2 is surprising. With low N values (8 for unfamiliar and 9 for familiar), there will be fewer significant differences between descriptors. The tendency for geometric descriptors to exhibit higher coefficients in either familiar or unfamiliar gesture groups is most likely because they cannot adapt to the effect of practice. This is similar to reasons given for the under- or over-estimation behavior of the CLC model [4,5].

Visual inspection of the most promising descriptors provides some intuition for their relative performance in predicting difficulty (Fig 4). Gestures are listed by ascending median *Ranking* on the horizontal axis, so a monotonic trend would suggest it is a good candidate for estimating *Ranking*. Actual production time ascends almost monotonically with *Ranking* demonstrating that gestures rated as being more difficult to execute have a greater production time. The static geometric descriptors for the most part increase with difficulty overall, but irregularities are much more pronounced suggesting a weaker fit. For example, letters *a*, *g* and *m* have long lengths, yet they are rated as easy to execute. This again speaks to familiarity: despite objective complexity, practiced gestures are rated with lower execution difficulty.

There are also significant correlations between descriptors. Production time is correlated with length ($\rho_{(N=18)}$=.89, p=.01) and Isokoski's complexity and production time are correlated with length ($\rho_{(N=18)}$>.70, p=.01). This suggests a partial correlation between these three descriptors, so it is appropriate to test for shared variance. When controlling for production time, the other parameters are no longer significant (p>.05). When controlling for all but production time, production time was still found highly correlated with *Rating* ($\rho_{(N=18)}$=.73) and *Ranking* ($\rho_{(N=18)}$=.67) at p<.01. For familiar and unfamiliar groups, none of the correlations with *Rating* and *Ranking* were

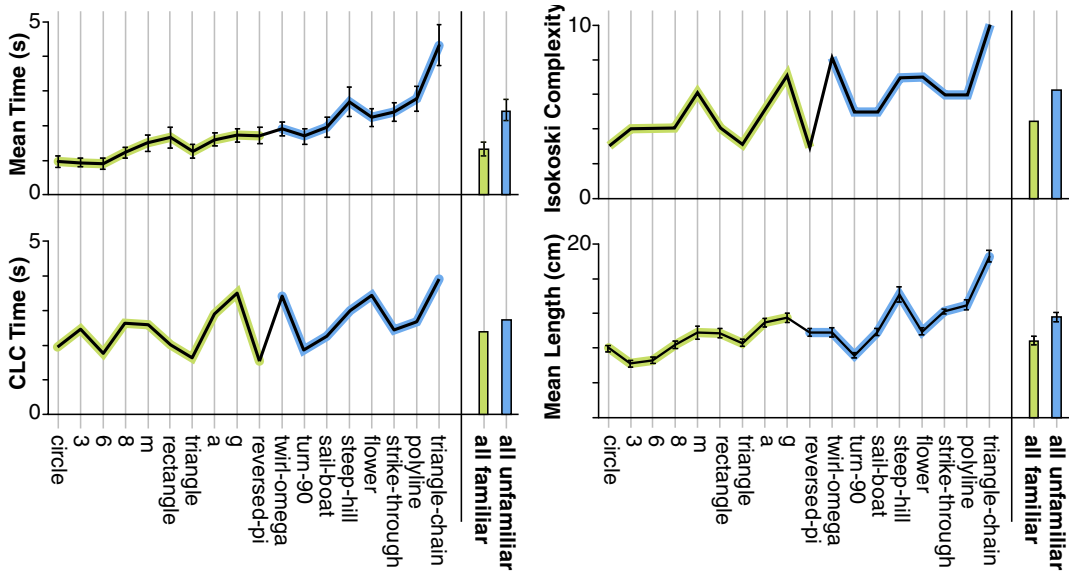


Fig. 4. Visual comparison of the four most promising measures and predictors (y-axes) with actual ascending median gesture *Ranking* (x-axis). A monotonic trend suggests the measure or predictor is a good candidate for estimating *Ranking* (e.g. mean measured time). NOTE: Error bars in all figures represent 95% CI.

significant when either variable was controlled during partial correlations. The t-statistic for comparing coefficients [6] showed a significant difference between coefficients for *Rating* ($t(15)=5.92$) and *Ranking* ($t(15)=4.02$) at $p<.01$.

The poor performance of the CLC predicted production time is somewhat surprising. Previous results found CLC to be highly accurate for first-order predictions when comparing relative ratios of gesture set production times [4,5]. So, we expected it would also perform well with a similar first-order prediction task for execution difficulty *Ranking*, but it has no significant correlations with *Ranking* at all. To investigate further, we directly compared the CLC predicted production times to actual production times. For magnitude, we found a significant, but low correlation ($R^2=.37$, $p=.01$). For relative ranking, we also found a significant, but low Spearman correlation ($\rho_{(N=18)}=.53$, $p=.05$).

Production time is the best indicator of execution difficulty, but the CLC model is not able to accurately predict performance time for our purposes. So, we continue the development of execution difficulty estimation rules based on actual production time, with the assumption (and caveat) that we are at the moment using a post hoc measured value. Later, we show that a small sample of data will provide suitable estimations of production time.

6 Difficulty Estimation Rules

We present two rules for estimating execution difficulty based on production time. The first is a simple rule which compares two candidate gestures according to relative

execution difficulty (as *Ranking* does), and the second uses Bayes' rule to classify a gesture into one of five categories of execution difficulty (such as those provided by the *Rating* measure).

Rule 1: Relative Difficulty Ranking
Gesture A is likely to be perceived as more difficult to execute than gesture B if the production time of A is greater than that of B:

$$time(\mathrm{A}) > time(\mathrm{B}) \text{ suggests } Ranking(\mathrm{A}) > Ranking(\mathrm{B})$$

To test this rule, we applied it to each pair of gestures (A,B) out of the (18x17)/2=153 possibilities in experiment 1 using the measured production time and counted how many times the rule was correct out of the total number of classification attempts (*Ranking* accuracy). The rule predicted the relative ranking correctly with 93% accuracy (11 errors out of 153 tests).

Rule 2: Classifying Difficulty Rating
Mapping from production time to one of our five difficulty classes (C_i, i=1..5: *very easy*, *easy*, *moderate*, *difficult*, and *very difficult*) is a pattern classification problem where each gesture is represented by a single feature, in our case production time. A common technique in statistical pattern recognition is Bayes' rule that minimizes classification error [24]. Bayes' rule uses each class-conditional density probability (i.e. the probability for a randomly chosen pattern x to lie in class C_i, denoted $p(x|C_i)$) together with the *a priori* probability of class C_i (or how likely it is to observe a pattern from this class, denoted $p(C_i)$). Using this data, Bayes' rule computes the *a posteriori* probability of x belonging to each class, $p(C_i|x)$, and assigns x to class C_j for which the a posteriori probability is maximum:

$$x \in C_j \Leftarrow j = \arg\max_{i=1,5} \{p(x \mid C_i) \cdot p(C_i)\} \quad (1)$$

In order to apply Bayes' rule, the conditional $p(x|C_i)$ and a priori $p(C_i)$ probabilities must be known for each of our 5 *Rating* classes. Normal parametric models are frequently assumed in practice (equation 2) for estimating the unknown conditional densities $p(x|C_i)$ [24](p.34) for which the parameters (mean μ_i and standard deviation σ_i) can be easily computed from the training set (in our case, data from experiment 1).

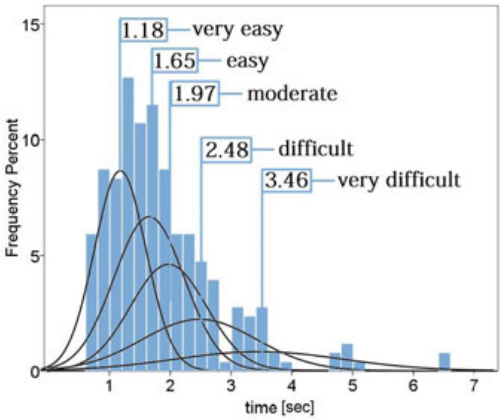
$$p(x \mid C_i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left(-\frac{(x - \mu_i)^2}{2\sigma_i^2} \right) \quad (2)$$

The a priori probabilities $p(C_i)$ are estimated from the training set as the percentage of samples falling into each class [24](p.34-39). In our case, μ_i are the mean production times for each *Rating* class (expressed in seconds); σ_i the standard deviations (seconds); and $p(C_i)$ the percentages of samples belonging to each *Rating* class. Table 5 lists these parameters as computed from our training data (experiment 1) with an illustration of each normal model superimposed over the production time histogram.

Table 5. Left: Bayes' Rule Parameters for the *Rating* Classification Rule. Right: Production time frequency histogram with superimposed time normal models for each *Rating*.

Rating levels	Bayes' Rule Parameters			
	μ_i	σ_i	$p(C_i)$	μ_i simplified[1]
very easy	1.18	0.43	27%	1.0
easy	1.65	0.58	28%	1.5
moderate	1.97	0.62	21%	2.0
difficult	2.48	0.95	16%	2.5
very difficult	3.46	1.38	8%	3.5

[1]simplified values represent *reasonable* approximations for the mean times μ_i; see the text for explanation and results.



We tested Bayes' rule in order to see how good it fits our data. We counted how many times the rule was correct out of 18 classification attemps (the *Rating* accuracy) by applying it to each gesture in our set. The rule achieved an accuracy rate of 83% on its own set (15 gestures were correctly classified to their *Rating* category as indicated by the participants). The three errors occurred for the *strike-through*, *turn-90*, and *sail-boat* gestures, all of which were misclassified to the next lower class. This confirms for now a good model fit for our data while Section 9 will show how the rule applies for new gestures in our validation experiment. The mean production times μ_i for each *Rating* level (see Table 5) could be approximated to more reasonable timestamps such as 1, 1.5, 2, 2.5 and 3.5 seconds (the μ_i simplified column in Table 5). These could represent more intuitive working estimates for each *Rating* class to be used by designers. When using these mean values with the computed standard deviations as before, we also obtained 83% classification accuracy.

7 Estimating Production Time

Applying our rules using measured production time works very well, but we would like designers to estimate production time without running such a formal experiment. Ideally, this could be done with predictive models. However, using times predicted by CLC, *Ranking* accuracy dropped to 67% and *Rating* down to 28%. Although Isokoski does not predict actual time, it can be used for relative *Ranking* where it managed a prediction accuracy of 82%.

Examining the data from experiment 1, we found that individual participant gesture production times are highly correlated with overall mean production times $\rho_{(N=18)}$=.96, p=.01 (min .92, max 1.0). This consistency made us wonder if a designer could estimate difficulty based on only few samples of measured production time. Instead of a long formal experiment, a few people could perform the candidate gestures a few times in a simple data gathering application. Even more, this data is likely to already exist for training the gesture recognizer [12,20,26].

We first consider the minimal case of gathering data from a single person. Again using data from experiment 1, for each participant, we randomly selected M out of 20 execution samples for each gesture to calculate a mean production time. Using these mean times, we apply our rules and compute the prediction accuracy for *Rating* and *Ranking*. The random selection was repeated 100 times as M varied from 1 to 20: thus 14 participants x 18 gestures x 20 M values x 100 repetitions = 504,000 predictions.

The mean accuracy for *Rating* begins to level out at M=3 near 53%, and *Ranking* also approaches 91% (Fig 5, left). The effect of M over *Rating* is significant ($\chi^2(19)=476.4$, $p<.001$). A Wilcoxon signed-rank test found significant effects between (1,20), (3,20) and (5,20) with a small Cohen effect ($r<.3$). The effect of M over *Ranking* was significant ($\chi^2(19)=4140.54$, $p<.001$) with significant differences between (1,20) ($r=.52$), (3,20) and (5,20) with medium effects ($r<.5$). With 3 samples, mean *Rating* accuracy was 53% (SD 18%) and mean *Ranking* accuracy 89% (SD 3%).

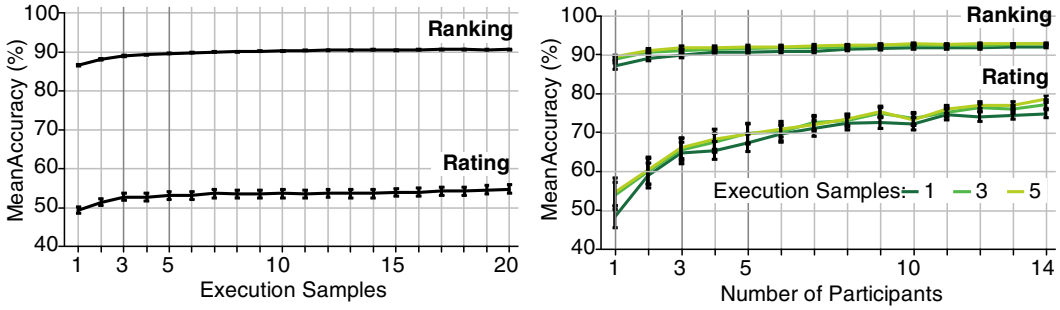


Fig. 5. Left: prediction accuracies for Rating and Ranking vs. number of execution samples. Right: Difficulty prediction accuracies vs. number of participants.

We continue our analysis by varying the number of participants N=1..14 given M=1,3,5 individual gesture execution samples from each. Similar to before, we randomly selected the gesture samples 100 times for each N: thus 14 participants x 18 gestures x 3 M values x 100 repetitions = 75,600 predictions.

The mean accuracy of *Rating* increases from 52% using one participant to 77% (significant, $\chi^2(13)=496.45$, $p<.001$) when data from all participants is used (Fig 5, right). The same trend is observed independently for M=1,3, and 5 executions from each participant. The accuracy of *Ranking* increases from 88% to 93% ($\chi^2(13)=715.1$, $p<.001$). The effect of M was found significant for both *Rating* and *Ranking* (at $p<.001$) but the Wilcoxon signed-rank test showed small Cohen effects between (1,3) and (1,5) $r<.3$ and very small between (3,5) $r<.15$. With 3 participants and 3 execution samples mean *Rating* accuracy was 66% (SD 14%) and mean *Ranking* accuracy 91% (SD 2%). With 5 participants and 3 execution samples mean *Rating* accuracy was 70% (SD 13%) and mean *Ranking* accuracy 92% (SD 2%).

In summary, on average, a designer could estimate a relative *Ranking* of execution difficulty with 89% using 3 gesture execution samples from a single person. To estimate *Rating*, 3 execution samples from 3 or 5 people are needed to achieve mean accuracies of 66% and 70% respectively.

8 Experiment 2: Validation of Difficulty Estimation Rules

A second experiment, similar to the first, was used to validate our execution difficulty rules as well as our simple production time estimation technique. The same apparatus, task, and design were used, but with 20 different gestures (Fig 6) and 11 new participants: 11 x 20 x 20 = 4,400 executions.

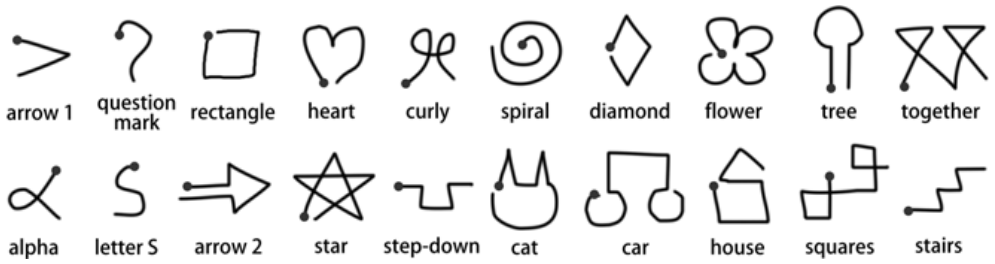


Fig. 6. The validation set of 20 gestures

Results
We found the same high level of correlation between participants' difficulty *Rating* (Kendall's W=.78, $\chi^2(19)=163.61$, p<.001) and *Ranking* (W=.80, $\chi^2(19)=166.79$, p<.001). *Rating* and *Ranking* were again highly correlated ($\rho_{(N=20)}=.94$, p=.01).

Estimates of Execution Difficulty
We first establish an accuracy upper bound using the actual measured production times logged in the experiment. To test the accuracy of estimating *Ranking* using Rule 1, we ordered the gestures in ascending order of production time, and correlated the resulting ranks with the median participant *Ranking*. Again, there was a strong correlation ($\rho_{(N=20)}=.94$, p=.01). Then, we applied Rule 1 for each pair of gestures (A,B) out of the (20x19)/2 = 190 possibilities, and calculated an accuracy rate (how many times the estimate was correct). In this way, estimating *Ranking* using Rule 1 attained 93% accuracy: 14 errors out of 190 tests. For Rule 2, we used the simplified Bayes parameters generated from Experiment 1 (Table 5). Estimating *Rating* using Rule 2 attained 90%: 18 gestures were correctly classified according to median participant *Rating*. The *rectangle* gesture was classified as *easy* instead of *very easy to execute*, and *tree* was classified as *easy* instead of *moderate* (both were shifted by one *Rating* class).

Next, we tested the accuracy of our rules using an estimate of production time generated from a small number of samples. Based on our analysis in the previous section, we tested N=1,3,5 participants and M=3 gesture execution samples. *Rating* accuracies varied from 66.9% to 79.8% while *Ranking* increased from 89.6% to 91.3%. Table 6 shows the accuracy rates obtained. We also re-tested using CLC and Isokoski for input to the model. CLC still produced a low *Rating* accuracy of 25%, but it performed better for *Ranking* with 75% accuracy. Isokoski did very well with 87% for *Ranking*, but cannot be used to estimate *Rating*. Overall, our rules to estimate difficulty performed well with our validation data, even when using only three samples from three participants as an estimate of production time.

Table 6. Validation experiments results: *Ranking* and *Rating* estimation accuracies using both measured and estimated production times

Production time		Estimation Accuracy	
		Ranking	*Rating*
Measured		93.0%	90.0%
Estimated (3 executions)	x 1 participant	89.6%	66.9%
	x 3 participants	90.5%	74.6%
	x 5 participants	91.3%	79.8%
Predicted	Isokoski	87.0%	n/a
	CLC	75.0%	25.0%

9 Conclusions and Future Work

Reducing gesture execution difficulty is an often mentioned goal of gesture set design. Our work provides support for this argument with empirical evidence showing that people tend to have similar perceptions of execution difficulty, that it is highly correlated with gesture production time, and that difficulty can be estimated using two simple rules for relative ranking and a classification rating. Because existing models cannot accurately predict the magnitude of production time necessary for our classification rule, we provide evidence that an estimate of production time using only a few execution samples from a few people is good enough. Moreover, this set of estimation data may already exist when designers train a recognizer.

Designers can use our quantitative rules as they are when selecting from candidate single stroke pen gestures. However, we plan to make this process more automatic, by incorporating our difficulty estimation into the popular $1 gesture recognizer [26]. As future work, we also plan to examine how execution difficulty relates to multi-stroke pen gestures and multi-touch gestures.

Acknowledgement. This paper was supported by the project "Progress and development through post-doctoral research and innovation in engineering and applied sciences- PRiDE - Contract no. POSDRU/89/1.5/S/57083", project co-funded from European Social Fund through Sectorial Operational Program Human Resources 2007-2013.

References

1. Appert, C., Zhai, S.: Using strokes as command shortcuts: cognitive benefits and toolkit support. In: Proceedings of CHI 2009, pp. 2289–2298. ACM Press, New York (2009)
2. Ashbrook, D., Starner, T.: Magic: a motion gesture design tool. In: Proceedings of the CHI 2010, pp. 2159–2168. ACM Press, New York (2010)
3. Bau, O., Mackay, W.E.: Octopocus: a dynamic guide for learning gesture-based command sets. In: Proceedings of UIST 2008, pp. 37–46. ACM Press, New York (2008)
4. Cao, X., Zhai, S.: Modeling human performance of pen stroke gestures. In: Proceedings of CHI 2007, pp. 1495–1504. ACM Press, New York (2007)

5. Castellucci, S.J., MacKenzie, I.S.: Graffiti vs. unistrokes: an empirical comparison. In: Proceedings of CHI 2008, pp. 305–308. ACM Press, New York (2008)
6. Chen, P., Popovich, P.: Correlation: Parametric and nonparametrized measures. Thousand Oaks, Sage (2002)
7. Grange, S., Fong, T., Baur, C.: Moris: a medical/operating room interaction system. In: Proceedings ICMI 2004, pp. 159–166. ACM Press, New York (2004)
8. Isokoski, P.: Model for unistroke writing time. In: Proceedings of CHI 2001, pp. 357–364. ACM Press, New York (2001)
9. Kratz, S., Rohs, M.: A $3 gesture recognizer: simple gesture recognition for devices equipped with 3d acceleration sensors. In: Proc. of IUI 2010, pp. 341–344. ACM Press, New York (2010)
10. Kristensson, P.-O., Zhai, S.: Shark2: a large vocabulary shorthand writing system for pen-based computers. In: Proceedings of UIST 2004, pp. 43–52. ACM Press, New York (2004)
11. Kurtenbach, G., Moran, T.P., Buxton, W.A.S.: Contextual animation of gestural commands. Computer Graphics Forum 13(5), 305–314 (1994)
12. Li, Y.: Protractor: a fast and accurate gesture recognizer. In: Proceedings of CHI 2010, pp. 2169–2172. ACM Press, New York (2010)
13. Long Jr., A.C., Landay, J.A., Rowe, L.A.: Helping designers create recognition-enabled interfaces. In: Multimodal Interface for Human-Machine Communication, pp. 121–146 (2002)
14. Long Jr., A.C., Landay, J.A., Rowe, L.A.: Implications for a gesture design tool. In: Proceedings of CHI 1999, pp. 40–47. ACM Press, New York (1999)
15. Long Jr., A.C., Landay, J.A., Rowe, L.A., Michiels, J.: Visual similarity of pen gestures. In: Proceedings of CHI 2000, pp. 360–367. ACM Press, New York (2000)
16. Morris, M.R., Wobbrock, J.O., Wilson, A.D.: Understanding users' preferences for surface gestures. In: Proceedings of GI 2010, pp. 261–268. Canadian Inf. Processing Society (2010)
17. Nielsen, M., Störring, M., Moeslund, T.B., Granum, E.: A procedure for developing intuitive and ergonomic gesture interfaces for HCI. In: Camurri, A., Volpe, G. (eds.) GW 2003. LNCS (LNAI), vol. 2915, pp. 409–420. Springer, Heidelberg (2004)
18. Pratt, W.: Digital Image Processing, 3rd edn. John Wiley & Sons, Inc., Chichester (2001)
19. Rico, J., Brewster, S.: Usable gestures for mobile interfaces: evaluating social acceptability. In: Proceedings of CHI 2010, pp. 887–896. ACM Press, New York (2010)
20. Rubine, D.: Specifying gestures by example. SIGGRAPH Computer Graphics 25(4), 329–337 (1991)
21. Schomaker, L.: From handwriting analysis to pen-computer applications. IEEE Electronics and Communications Engineering Journal 10(3), 93–102 (1998)
22. Viviani, P., Flash, T.: Minimum-jerk, two-thirds power law, and isochrony: converging approaches to movement planning. Journal of Experimental Psychology: Human Perception and Performance 21(1), 32–53 (1995)
23. Viviani, P., Terzuolo, C.: 32 space-time invariance in learned motor skills. In: Tutorials in Motor Behavior. Advances in Psychology, vol. 1, pp. 525–533. North-Holland, Amsterdam (1980)
24. Webb, A.: Statistical Pattern Recognition. John Wiley & Sons, Inc., Chichester (2002)
25. Wobbrock, J.O., Morris, M.R., Wilson, A.D.: User-defined gestures for surface computing. In: Proceedings of CHI 2009, pp. 1083–1092. ACM Press, New York (2009)
26. Wobbrock, J.O., Wilson, A.D., Li, Y.: Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In: Proceedings of UIST 2007, pp. 159–168. ACM Press, New York (2007)

Buttonless Clicking: Intuitive Select and Pick-release Through Gesture Analysis

Ali Choumane*
INRIA Lille
France

Géry Casiez †
LIFL & INRIA Lille
University of Lille, France

Laurent Grisoni ‡
LIFL & INRIA Lille
University of Lille, France

Abstract

Clicking is a key feature any interaction input system needs to provide. In the case of 3D input devices, such a feature is often difficult to provide (e.g. vision-based, or tracking systems for free-hand interaction do not natively provide any button). In this work, we show that it is actually possible to build an application that provides two classical interaction tasks (selection, and pick-release), without any button-like feature. Our method is based on trajectory and kinematic gesture analysis. In a preliminary study we exhibit the principle of the method. Then, we detail an algorithm to discriminate selection, pick and release tasks using kinematic criteria. We present a controlled experiment that validates our method with an average success rate equal to 90.1% across all conditions.

Index Terms: H.5.2 [User Interfaces]: Input devices and strategies

1 Introduction

Selection and positioning are two basic tasks of any 3D application that can be executed using buttons, gestures [3] or voice recognition [1]. Pinch to grab represents the most common use of glove for selection and positioning [2]. However most 3D input devices like 3D trackers have only three or six degrees of freedom (DOF) making them unsuitable for this interaction technique. Other devices, like the Gametrak™ and the EyeToy® have no button requiring the use of specific gestures or voice recognition to select options in 3D or move objects. Users are also reluctant to run the different calibration and training steps associated with gestures and voice recognition.

In this paper, we present a new approach based on trajectory and kinematic gesture analysis. We claim that selection, picking and releasing can be predicted based on the trajectory and kinematic patterns of the user's hand. The technique we propose requires no training or calibration and can be used with any 3D input device. In an initial study we analyze the trajectory and kinematic patterns of users using buttons to select and position 3D objects. Positioning starts with an object being picked and finishes with its release. We then propose an algorithm based on the invariants we found in the initial study to replace the use of buttons for selection, pick and release tasks. We last present a controlled experiment to evaluate our proposed algorithm.

2 Related Work

In 2D environment, the *dontclick*[1] project allows to navigate a website without using mouse buttons. This website principle is that when the cursor is on an item, it is selected. In 3D environment, Payne et al. [6] investigated issues affecting the usability and fun in the context of 3D gestures and video games. This work confirmed user benefits of 3D spatial gesture as a mean of interaction, such as intuitive movements linked to actions performed, as opposed to "button bashing". The shapewriter system [4] introduces an efficient way to combine keyboard stroke, using pen movement analysis. Stroke selection is identified by evaluating significant movement changes in pen movement. Although devoted to a specific task (stroke selection on a screen-displayed keyboard), this work can be seen as a major step in the direction of continuous gesture analysis for computer-human interaction simplification. Lank et al. provide estimators of gesture endpoint, using motion analysis [5].

None of the mentionned works proposed any method for combining selection and pick-release without button in the same application, which is the core contribution presented here. Next section presents the initial study we performed, that is at the origin of the proposed algorithm.

3 Initial Study

The purpose of this initial study is to investigate the trajectory and kinematic patterns associated to the select and pick-release tasks when they are performed using a button. We asked two participants

[1] http://www.dontclick.it/

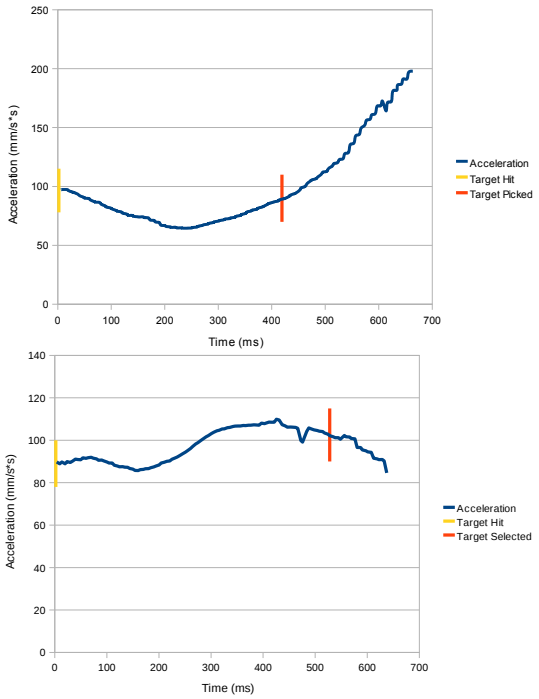


Figure 1: Acceleration profile for the pick (top) and selection (bottom) tasks when the cursor enters the target.

*e-mail: ali.choumane@inria.fr
†e-mail: gery.casiez@lifl.fr
‡e-mail: laurent.grisoni@lifl.fr

to select, and pick-and-release spheres randomly positioned in 3D. The hand was tracked using a Gametrak device[2] and we used the button of a wireless mouse hold by the hand to perform the task. The Gametrak has a 3m cube workspace with a resolution ranging from 0.01 mm to 7 mm in all directions and a 125 Hz sampling rate. We collected a total of 160 trials for the analysis. For each condition (select, pick, release), we computed the mean velocity and acceleration profiles. This gave us a total of six graphs.

The velocity and acceleration profiles for the whole movement measured from the cursor first move to the target selection or picking did not show any specific pattern. Instead we observed discriminant patterns for the end of each movement when the pointer reaches the target.

Among the different profiles we computed, the most specific pattern we observed was for the picking. For this task, the device acceleration when entering a target first decreases until reaching a minimum and increases before the user presses the button. The corresponding profile is illustrated in Figure 1 (top). The time when the pointer enters the target is represented by the yellow bar and the red bar represents the time the button is pressed. Figure 1 (bottom) represents the corresponding acceleration profile for the selection task. We could not find a similar profile for the acceleration between the time the target is entered and the time the button is pressed.

This characteristic profile for the picking task could be explained by a planning of the movement trajectory before pressing the button. The trajectory is planned to enter the target and then exit it in a direction given by the expected drop position. Trajectory analysis confirmed that the distance between the pointer position and the target center increases as soon as the button is pressed.

For the selection task, the velocity profile (Figure 2) represents the most characteristic profile. The velocity keeps decreasing once the target is entered (represented by the yellow bar) until the button is pressed down (red bar) and up (end of the blue curve). The average speed when the button is pressed down is equal to $21mm.s^{-1}$ (SD = $8mm.s^{-1}$) and the duration of the click (button down and up) was measured to be equal to 180ms.

For the release task, we observed that the average speed when the button up event occurs is equal to $14mm.s^{-1}$ (SD = $7mm.s^{-1}$). For both selection and picking tasks we observed that the average velocity when entering a target is equal to $75mm.s^{-1}$ (SD = $40mm.s^{-1}$).

In the next section we propose an algorithm taking into account the observations of the study presented in this section to predict the select, pick, and release tasks.

4 Proposed Algorithm

Based on the results of the preliminary study, we propose the state transition diagram as illustrated in Figure 3 to select, deselect, pick, and release objects.

As we can only discriminate the acceleration profiles when a target is entered, our algorithm first checks if the pointer is inside a 3D object that can be selected or picked (state 0). When the target is entered (state 1), the velocity, acceleration and trajectory profiles are analyzed.

In state 1, we check if the target is intended to be selected or picked by measuring the device speed. If it is above $V_{threshold1}$ we consider the user passing through a target without any intention of interacting with it. We used a value of $115mm.s^{-1}$ for $V_{threshold1}$ as the preliminary study showed that the mean velocity when entering a target is equal to $75mm.s^{-1}$ with a standard deviation equal to $40mm.s^{-1}$. If the device velocity remains below $V_{threshold1}$ we then discriminate between the select and pick tasks.

[2]http://www.pdp.com

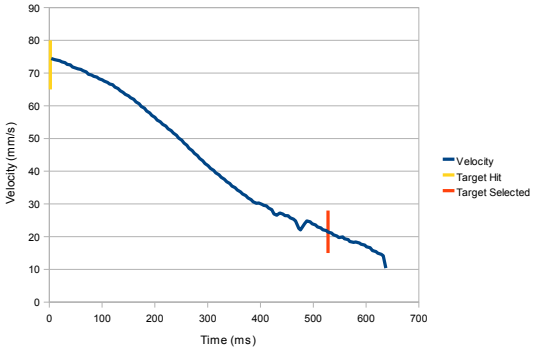


Figure 2: Velocity profile for the selection task when the cursor enters the target.

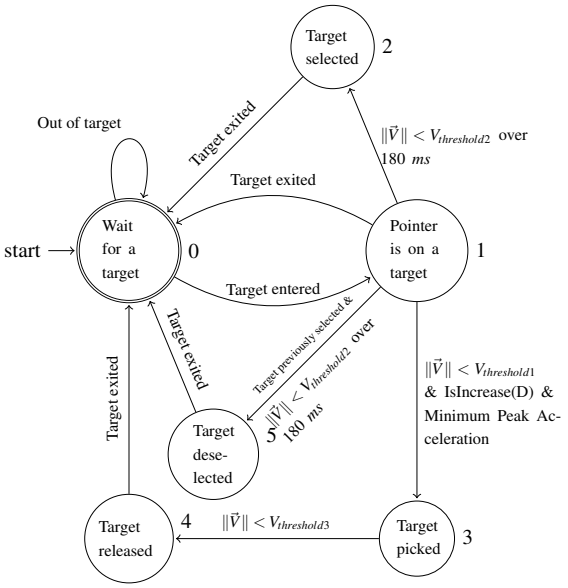


Figure 3: The state transition diagram representing the algorithm to predict select, deselect, pick, and release tasks. D is the distance between the current pointer position and the center of the entered target.

The picking task is conditioned by the two criteria found in the initial study: a minimum peak in the acceleration profile and an increase in the distance to the target center (*isIncreased(D)*). The increase in distance is checked after the minimum peak acceleration is detected. If the two conditions are fulfilled the target is picked (state 3).

For the selection task, we check if the device speed remains below a speed threshold $V_{threshold2}$ over a period of time. According to the initial study we found an average velocity equal to $21mm.s^{-1}$ (SD = $8mm.s^{-1}$) when the button is pressed down. We use a speed equal $29mm.s^{-1}$ for $V_{threshold2}$ and use 180ms for the time period. The target state (selected or deselected) is then toggled.

The target release occurs when the device speed drops below a third threshold speed $V_{threshold3}$.We used a value of $7mm.s^{-1}$ as the mean value found in the initial study with the corresponding standard deviation.

The detection of the pick task depends, in part, to the function *IsIncrease*(D). As mentioned above, this function allows to predict that the user is moving out from the target. In the formal experiment (cf. section 5) targets were spheres. Hence D is the distance between the sphere center to the pointer position. We assume that our algorithm is adapted for targets with a geometric form that can be easily enclosed in a sphere, for example, pyramid, cube, cylinder. For these forms, D is well defined.

5 Experiment

Our goal is to evaluate the effectiveness of our algorithm by measuring the percentage of correctly recognized actions for each subtask: selection, pick, and release. As our algorithm is based on kinematic gesture analysis, we also want to assess its robustness on parameters that can affect the kinematic profile: target size (the size of the object to select or pick), destination size (the size of the zone to release the picked target) and target orientation (measured as the angle formed by the movement starting point, the target position and the horizontal). The target size is known to affect the velocity profile with small sizes requiring more accuracy and thus reducing the corrective movement speed [7]. The target orientation affects the relative displacement of the hand which can make a difference in the velocity profile. Our goal is also to compare our technique to the button alternative in terms of performance (movement time and error rate) and subjective preference.

Participants

Four female and four male with a mean age of 26 (SD=1.8) participated. Participants had an average forearm length equal to 27 cm (SD=3), arm length equal to 29 cm (SD=2.5), and an average height equal to 173 cm (SD=9). All participants were right-handed and had normal or corrected-to-normal vision. Two participants had little experience and six had no experience with virtual reality and 3D applications but this is acceptable as we are observing a lower level physical behavior. Among the participants fours were computer scientists, three were electronic engineers and one was medical doctor. None of them participated in the initial study.

Apparatus

The experiment was conducted using a retroprojected 6500 mm (256 inch) large curved screen with stereoscopic display using a 2344 $\times$ 1050 pixels resolution (202 $\times$ 203 mm), 96 DPI pixel density, and 120 Hz refresh rate. The hand position was tracked using a DTrack device with an ARTtrack infrared-based optical tracking system[3]. The ARTrack system gives a 0.06 mm positional resolution in all directions, 0.4 mm positional accuracy and a 60 Hz sampling rate. The DTrack device was positioned on top of the hand. For the button condition, we used the same apparatus except participants hold a wireless mouse in the hand that was tracked. The left mouse button was used for interaction.

Task

We used two tasks in our experiment: a multi-directional pointing task and a multi-directional pick-and-release task. The two tasks were evaluated with and without button (figure 4). For each task we used four targets evenly distributed on a circle positioned at the center of the screen in a plane parallel to the screen. The pointer was represented as a pink sphere with a diameter equal to 2 mm measured at the center of the circle. We used a constant gain to map the hand position to the 3D cursor position.

For the selection task, the target to select appeared in red while the other targets remained grey. Upon successful selection of the target, it disappeared and the next target to select turned red. Picked targets were counted as errors and had to be released anywhere before the next target to select turned red.

[3]http://www.ar-tracking.de/

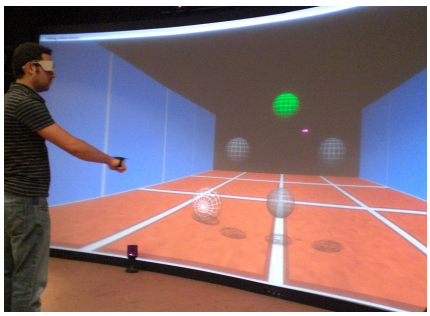


Figure 4: Picture of the experiment setup for the pick-and-release task showing our experimental hardware with the large screen, the DTrack device fixed to the hand of the participant, the targets to pick-and-release and the destination on the floor.

We used a similar scenario for the pick-and-release task except the target to pick appeared green and we used a sphere at a pseudo-random position on the floor representing the destination location to release it. The destination sphere appeared white and turned green when the target was fully inside. Targets that were inadvertently released were counted as error and had to be picked again until released at the correct destination. Targets inadvertently selected were counted as errors before the next target to pick turned green.

In addition to perspective and occlusion, we added shadow projection on ground for the pointer, targets and destination to improve depth perception. The camera remained fixed during the whole experiment. Participants were instructed to perform the tasks as accurately as possible. For all participants, the parameters of our model were set to the same values given by the initial study.

Design

A repeated measures within-subjects design was used. The independent variables were Technique (system used with or without button), Task (Select, Pick, Release), Orientation (0°, 90°, 180°, 270°), Target Size (6 mm, 12 mm and 24 mm - measured at the center of the circle) and Destination Size for the release task (10 mm and 20 mm greater than the target size). The orientation is measured from the horizontal axis counter-clockwise (the 0° is positioned at the right of the scene). The pick-and-release task is decomposed into two sub-tasks: the pick task and the release task.

Trials were organized in Blocks to measure the learning effect. We used six blocks for the no-button technique and two blocks for the button technique. Each Block was composed of the three Task evaluated each time with three Target size, four Orientation and the two destination Size for the Release task. Task and Technique were counter-balanced across participants. This gave us a total of 8 $\times$ 480 = 3,840 total trials. The experiment lasted approximately 60 minutes.

6 Results

The dependent variable are movement time and success rate.

6.1 Movement Time

Trials marked as error were removed from movement time analysis. Repeated measures analysis of variance found no significant effect ($F_{1,7}$ = 0.017, p = 0.9) for Technique, Task ($F_{2,14}$ = 3.426, p = 0.061) and no significant interaction between Technique and Task ($F_{2,14}$ = 2.91, p = 0.088) on movement time. These results

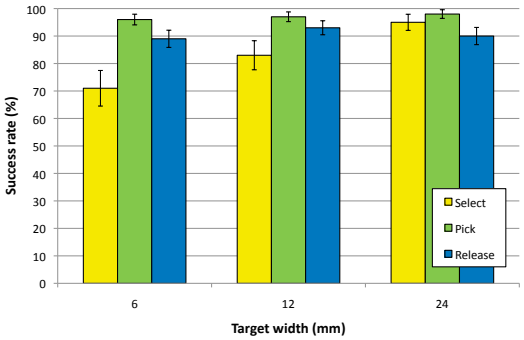


Figure 5: Mean success rate for each TASK and TARGET SIZE for the no-button technique. Error bars represent 95% CI.

lead us to conclude there is no degradation in the movement time for the no button technique. Further evaluations would be required to show that no small effect exists.

6.2 Success Rate

The success rate for the Release TASK is computed from the successfully picked targets. Repeated measures analysis of variance found no significant effect for BLOCK on the success rate showing no learning effect.

Repeated measures analysis of variance found a significant main effect for TECHNIQUE ($F_{1,7} = 67.5$, $p < 0.0001$) and a significant interaction between TECHNIQUE and TASK ($F_{2,14} = 9.0$, $p = 0.03$) on success rate. The overall success rate is 96.2% with button and 90.1% without button. Pairwise comparison show significant differences for the Pick TASK (p=0.004, 99.7% success rate with button and 96.7% without button) and Select TASK (p=0.024, 96.2% success rate with button and 82.9% without button). The Release TASK shows no significant difference ($p = 0.37$) with a 92.6% success rate with button and 90.7% without button. To better understand the factors influencing the success rate without button, we removed the button data for subsequent analysis.

Repeated measures analysis of variance found a significant main effect for TASK ($F_{2,14} = 41.2$, $p < 0.0001$), TARGET SIZE ($F_{2,14} = 17.4$, $p < 0.0001$) on success rate and significant interaction between TASK and TARGET SIZE ($F_{4,28} = 10.8$, $p < 0.0001$) on success rate. Pairwise comparison show significant differences ($p < 0.018$) between the different target sizes of the Select TASK with 70.5% success rate for the 6 mm target, 82.8% for the 12 mm target and 95.4% for the 24 mm target. Repeated measures analysis of variance found a significant main effect for DESTINATION SIZE ($F_{1,7} = 5.9$, $p = 0.045$) on success rate with 88.5% for the smallest width and 93% for the largest width.

7 SUBJECTIVE RESULTS

After each block, participants rated their fatigue level using a 5 points Likert scale (1:no fatigue, 5:high fatigue). The mean value across participants remained consistently around 3 throughout the blocks showing no increase in fatigue.

At the end of the experiment, we asked participants to give their favorite technique. Half the participants declared to prefer the technique without button and the other half with button. Participants who preferred the technique with button mentioned that it is closer to mouse-based interaction they are familiar with. From the informal qualitative feedbacks we noticed that participants were first surprised about the idea of selecting and moving object without button but they quickly adopted and enjoyed the technique. We observed that sometimes people intuitively closed their hand to pick the target. One participant said: "it's as if the computer reads my mind", another one said : "it's much more fun without button".

8 DISCUSSION

Overall the experiment validated our algorithm with an average success rate equal to 90.1% across all conditions. This result is lower than the 96.2% success rate with the button technique but we were pleased to see that our algorithm requires almost no learning and was robust to all participants with the settings defined in the initial study. We evaluated the influence of the orientation, target size and destination size as confounding parameters. The experimental results show that the orientation has no influence on the recognition rate while the target size significantly affect the Selection success rate with smaller target sizes decreasing the recognition rate. In addition the results on movement time indicate that the no button technique is equivalent to the button technique and the results on fatigue did not show any specific fatigue associated to the no button technique.

These results show that our method represents a valid alternative to the use of buttons for selection and pick-and-release tasks in applications where the targets size is sufficiently large. Our results also show there is space for improvement in our algorithm to increase the recognition rate. An area for improvement is to take into account the target size in our algorithm.

9 CONCLUSION AND FUTURE WORK

We presented a new approach for discriminating between selection and picking-and-releasing tasks based on trajectory and kinematic analysis, especially useful for tracking systems having no button. We detailed the algorithm and validated it in a controlled experiment that shows high recognition rates.

As future work we first plan to tune the algorithm for small target size. We then want to check if our algorithm can simply apply to object of any shape by taking their bounding sphere or if we need to refine the algorithm depending of the object shape.

ACKNOWLEDGMENTS

This work was supported by the ANR project n°ANR-07-TECSAN-020 "REACTIVE".

REFERENCES

[1] R. A. Bolt. "put-that-there": Voice and gesture at the graphics interface. In *SIGGRAPH*, pages 262–270, New York, USA, 1980. ACM.
[2] D. A. Bowman, C. A. Wingrave, J. M. Campbell, V. Q. Ly, and C. J. Rhoton. Novel uses of pinch gloves for virtual environment interaction techniques. *Virtual Reality*, 6(3):122–129, 2002.
[3] S. Carbini, J. E. Viallet, and L. DelphinPoulat. Context dependent interpretation of multimodal speechpointing gesture interface. In *ICMI Doctoral Spotlight &Demo Papers*, pages 1–4. ACM, 2005.
[4] P. O. Kristensson and S. Zhai. Command strokes with and without preview: using pen gestures on keyboard for command selection. In *CHI*, pages 1137–1146, New York, NY, USA, 2007. ACM.
[5] E. Lank, Y.-C. N. Cheng, and J. Ruiz. Endpoint prediction using motion kinematics. In *CHI*, pages 637–646, New York, NY, USA, 2007. ACM.
[6] J. Payne, P. Keir, J. Elgoyhen, M. McLundie, M. Naef, M. Horner, and P. Anderson. Gameplay issues in the design of spatial 3d gestures for video games. In *CHI*, pages 1217–1222, New York, USA, 2006. ACM.
[7] D. Wisleder and N. Dounskaia. The role of different submovement types during pointing to a target. *Experimental Brain Research*, 176:132–149, 2007.

Mockup Builder: Direct 3D Modeling On and Above the Surface in a Continuous Interaction Space

Bruno R. De Araùjo*, Géry Casiez† and Joaquim A. Jorge‡
*,‡INESC-ID, DEI IST, Technical University of Lisbon, Portugal
†LIFL, INRIA Lille, University of Lille, Villeneuve d'Ascq, France

ABSTRACT

Our work introduces a semi-immersive environment for conceptual design where virtual mockups are obtained from gestures we aim to get closer to the way people conceive, create and manipulate three-dimensional shapes. We present on-and-above-the-surface interaction techniques following Guiard's asymmetric bimanual model to take advantage of the continuous interaction space for creating and editing 3D models in a stereoscopic environment. To allow for more expressive interactions, our approach continuously combines hand and finger tracking in the space above the table with multi-touch on its surface. This combination brings forth an alternative design environment where users can seamlessly switch between interacting on the surface or in the space above it depending on the task. Our approach integrates continuous space usage with bimanual interaction to provide an expressive set of 3D modeling operations. Preliminary trials with our experimental setup show this as a very promising avenue for further work.

Index Terms: H.5.2 [User Interfaces]: Graphical user interfaces (GUI)—Input devices and strategies (e.g., mouse, touchscreen), Interaction styles (e.g. commands, menus, forms, direct manipulation);

1 INTRODUCTION

Despite the growing popularity of Virtual Environments, they have yet to replace desktop CAD systems when it comes to modeling 3D scenes. Traditional VR idioms are still umbilically connected to the desktop metaphor they aim to replace, by leveraging on the familiar Windows+Icons+Menus+Pointing (WIMP) metaphors. Worse, the command languages underlying many of these systems also do not map well to the way people learn to conceive, reason about and manipulate three-dimensional shapes. Another important obstacle, lies in that powerful modeling systems resort to constructive geometry and parametric formulations of a handful of primitives which run contrary to human perceptions and intuitions of space and physical models. As a result, users indirectly interact with models through widgets to control their parameters. However, new and affordable technologies such as depth cameras, multi-touch surfaces and multi-sensor devices motivate a fresh look at immersive interfaces. By providing more degrees of freedom, the new devices bear the promise of breaking from this mold by helping to develop interfaces that better support direct interaction. Indeed, these devices have the potential to support human modes of communication, such as sketching, gesturing and manipulating images and physical object as real-world proxies. Furthermore, many of these devices support a deeper use of human expression, such as two-handed manipulation, body posture, gaze and attention to name a few.

*e-mail: brar@vimmi.inesc-id.pt
†e-mail: gery.casiez@lifl.fr
‡e-mail: jorgej@acm.org

According to [9], immersive modeling brings three key advantages to the conceptual design process. First, they allow direct and real-time interaction. Second, users can work at full scale both in representation and interaction while being immersed. Finally in contrast to desktop systems, these attributes allow designers to get subjectively closer to their design ideas and work intuitively on their representation. Our strategy is to take full advantage of different interaction spaces and leverage their benefits for the tasks they are best designed for (e.g. using a flat surface for 2D sketching and 3D space for extruding an object). With these aims in mind, our goal is to develop a simple yet expressive system closer to the way people conceive, create and manipulate three-dimensional shapes. Thus, we devise a direct modeling approach taking advantage of sketching skills on 2D surface and gestures in 3D space operating seamlessly in the same immersive environment. These operations are fashioned following observations on how physical mock-ups are constructed manually and extend modeling operators from successful systems for fast prototyping such as Google Sketchup. Our direct modeling approach aims at interacting with the objects of interest without intermediate dialogues or gadgets, which promotes co-located interaction without sacrificing the expressivity power of the interface. Our immersive environment targets at supporting gestural and direct manipulation following the push and pull modeling paradigm to edit both topological and geometric representations of 3D models. By doing so, our goal is to propose plausible 3D gestures for modeling similar to physical mock-up interaction. Finally, we want to hide the underlying mathematical details associated to traditional CAD systems, thus bringing users into more intimate contact with virtual shapes without sacrificing their creativity. While we do not aim at working at full scale, the ability to control scale at will is an important feature to easily explore models. By using a god–like view, we intend to render virtual models as close as possible to physical mockup-ups without the associated physical constraints.

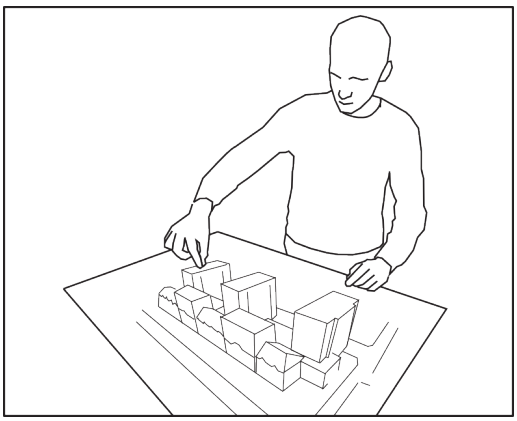


Figure 1: Mockup Builder Concept.

In this paper, we explore bi-manual and continuous interaction on and above multi-touch surfaces to bring direct modeling techniques to semi-immersive virtual environments. Our setup combines (1) stereoscopic visualization with (2) a multi-touch surface, (3) three-dimensional finger tracking and (4) a depth camera. In this way we can fuse four different but closely related human modalities to capture gaze, body posture, hand and finger position in synergistic ways. This rich sensing environment allows us to seamlessly pick and choose the sensing technique(s) most appropriate to each task. On this groundwork, we have developed an expressive set of modeling operations which build on user's abilities at creating and manipulating spatial objects. Indeed, from a small set of simple, yet powerful functions users are able to create moderately complex scenes with simple dialogues via direct manipulation of shapes. Noisy user input is continuously beautified to enable users to create smooth-looking forms by free-hand sketching. In this way, input gestures and strokes are continuously smoothed avoiding spurious artefacts and rendering shapes easier to create. Additionally, our setup affords continuous transitions between 3D (spatial) and 2D (planar surface) manipulations for modeling shapes extending the continuous space metaphor [26]. This allows users to issue gestures on and above the surface in an expected manner, e.g. extrusions of sketched shapes in contiguous, fluid gestures.

Another key feature of our approach lies in that it inherently supports bimanual asymmetric interactions. We adopt the Guiard's asymmetric model [15] for this purpose. This model proposes guidelines for designing bimanual operations based on observations of users sketching on paper. For these tasks, Guiard identifies different rules and actions for the preferred (also dominant-hand or DH) and non-preferred (also non-dominant hand, or NDH) hand. While the DH performs fine movements and manipulates tools, the NDH is used to set the spatial frame of reference and issue coarse movements. Moreover, people do not explicitly switch between defining the spatial frame of reference and manipulating tools.

We developed different interaction metaphors according to three main criteria: the location of the gesture, the participating hand(s) and the continuity of hand movements. This distinctive feature of our work combines continuous space and the Guiard asymmetric model harmoniously in a single definition of mode. Furthermore, it allows seamless and rapid mode switching in straight-forward ways, greatly contributing to the overall expressiveness of the interface using simpler dialogues.

The rest of the paper is organized as follows. After covering the related work we present our approach and describe in detail the experimental setup. We then explain the processing of input data and how the different modalities are fused. The remaining sections explain our apporaches to bimanual interaction and how we explore the continuous space to derive a simple yet very expressive set of modeling operations. Preliminary assessments and trials of our techniques show promise and encourage us to further pursue this avenue in the future.

2 Related Work

This section discusses research in three main areas relevant to our work. We first present non traditional modeling interfaces using gestures in the air, tangible objects, sketching and haptic feedback. We then cover bimanual interaction and follow with the on-and-above the surface related approaches.

Modeling System Interfaces. Schkolne *et al.* [32] introduced Surface Drawing using hand motion in the air to describe ribbon like shapes based on hand posture. Additionally, a set of tangible tracked artifacts were available, each with its own functionality. For example, kitchen tongs to pick objects, a magnet tool to deform objects or a squeezable object to delete parts of an object. While this approach allows creating free-form shapes, it appears inadequate to create rigorous manufactured shapes. FreeDrawer [34] alleviates this issue by providing a tracked stylus allowing the user to sketch networks of curves on top of a Responsive Workbench. These curves can then be used to define the boundary of free-form surfaces that can be deformed interactively. However more complex CAD editing and primitives are still out of the scope of such approach. Fleish *et al.* [11] support both freeform shape creation and regular CAD primitives by adapting traditional WIMP based interfaces to virtual immersive environment using a PIPSheet artifact. The PIPSheet is a tracked transparent glass where menus can be seen through, creating the illusion that the user interface is rendered on the glass surface. Items can be selected using a tracked stylus. Using head mounted displays, such system can be used by several users in a collaborative way to support the designing task as presented by Kaufmann [19]. Their system was used to introduce CAD modeling operations to students allowing creating simple regular primitives such as prisms, pyramids, cones and cylinders in the air. However the lack of physical support makes drawing in the air more adequate for free form modeling than to create CSG like regular objects [31]. Haptic devices can help sketching in the air although the working space is often restricted [20]. This provides an attractive solution for 3D modeling since users are able to easily learn how to use these systems and rigor improves rapidly with training as shown by recent studies [35]. Instead of only relying on gestures in the air, our approach takes advantage of both the surface and space above it, which aims at combining the benefits of both interaction spaces.

Sketching is a powerful communication tool of any real conceptual design task. However, it is still discarded by most of existing CAD modeling systems which rely primarily on single cursor based interaction and WIMP metaphor. Regarding traditional 2D environments, research on sketch based modeling interfaces has proposed several approaches to take advantage of designer drawing skills. Olsen presented a deep survey of most of the existing techniques [30]. These systems rely on gesture recognition (SKETCH), stroke beautification (Pegasus), line drawing reconstruction (SmartPaper), suggestive interfaces (Chateau), push pull sketching (Sesame [29]), freeform contour based inflation (Teddy or ShapeShop) to make sketching as a usable alternative to traditional CAD systems. Forsberg *et al.* [13] propose an adaptation of the SKETCH system to a stereoscopic ActiveDesk environment named ErgoDesk. However, they still rely exclusively on 2D gestures to create geometry using a light pen and the stereoscopic visualization is primary used for 3D exploration of shapes using a 6DoF tracker. Our approach adopts several of these techniques to go further than existing drawing-in-the-air approaches while mixing 2D sketch with 3D gestures continuously. We rely on the physical surface of a multi-touch device as a fixed drawing canvas, to free hands from holding devices used as a moving canvas in 3D space [37, 21]. We use sketch on the surface combined with gesture above the surface to define 3D trajectories while the user is experiencing a stereoscopic visualization more adequate to 3D perception. Alternatively, tangible interfaces have been used in space or on tabletop. Tangible interfaces offer natural manipulations and artifacts can correctly map tools functionality [32]. They can be as effective or even better than WIMP interfaces for 3D manipulation and edition as demonstrated by [28]. They can also be used to create 3D models such as Jota *et al.* [18] using wooden blocks of different shapes. Using a Kinect camera, the position and shape of the blocks can be captured and 3D simple scenes can be created by assembling blocks while the user is viewing the scene in stereo. Commands and plane height control are issued using an additional mobile device used as an operation console, stacking captured blocks on top of virtual content. In contrast with tangible interfaces, our approach is not limited to physical representations and provides an unconstrained designing environment regarding shape representation.

Bimanual Sketching. Bimanual interaction is a fundamental concept to our approach in that we expect to leverage the higher bandwidth provided by two–handed gestures. Hands can have an asymmetric or symmetric role [15]. Asymmetric bimanual interaction attributes different roles to each hand as presented by Balakrishnan and Kurtenbach [4] where the NDH controls a virtual camera, defining a frame of reference for the DH which manipulates objects. With symmetrical bimanual interaction, both hands have a similar role adapted to the task. While the symmetric model [3, 22, 23] has proved to be more adequate to support exclusive spatial tasks or describe shapes with hands, the asymmetric model makes it possible to take advantage of a natural task switching between hands. Initially, methods have been proposed that mimick existing asymmetric tasks such as the automotive tape drawing techniques [2, 14]. Using that approach, users create curves on a large scale display at a one to one scale using both hands. Other approaches associate different roles to each hand [4, 34]. Usually they use the NDH to manipulate objects or the view and the DH for editing as suggested by the Guiard asymmetric model [15]. To wit, our approach takes advantage of both asymmetrical and symmetrical hand operations. Most operations assign asymmetrical roles to each hand. However, for tasks such as scaling and rotating shapes on the surface or in the air, it is more natural to use symmetric assignments [33]. The IloveSketch system [1] adapts such a concept in traditional 2D sketch based modeling interfaces allowing users to control the virtual camera or 3D planes using a keyboard, while the other hand sketches on the 3D scene using a pen tablet to create curve wireframe models. While this approach is bimanual, it does not engage the hand directly – it operates two devices, the keyboard and the stylus pen. Other systems [5, 17, 24, 25] have explored the bimanual asymmetric model by combining finger- or hand- gestures with pen devices. Brandl *et al.* proposed a sketching system where the user selects options through touches using the NDH on a WIMP–based graphical interface, while the DH is used to sketch using a pen device [5]. Such a configuration allows to better explore hand gestures proposing richer interaction concepts to represent 2D editing operations such as demonstrated by Hinckley *et al.* [17]. Indeed, this makes switching between modalities easier and allows users to perform a wide range of 2D editing tasks without relying on gestures or GUI invocations. Lee combined hand gestures while sketching using a collapsible pen to define curve depth on a tabletop [24]. The NDH is tracked allowing users to seamlessly specify 3D modeling commands or modes such as the normal direction of an extrusion while specifying the displacement by interacting with the pen on the virtual scene. Contrary to their approach, we preferred to keep the surface for fast and accurate 2D drawing, while benefiting from the 3D input space for controlling depth directly. Lopes *et al.* adapted the ShapeShop sketch based free-form modeler to use both pen and multi-touch simultaneously [25]. They found out that the asymmetric bimanual model allows users to perform more manipulations in less time than conventional single interaction point interfaces, which increased the percentage of time spent on sketching and modeling tasks. By tracking the hands of the user, we adopt the asymmetric bimanual model to easily switch between sketching, model editing, navigation and spatial manipulation of objects. In addition, we do not need to rely on special input devices nor extra modalities to assign different roles to each hand.

"On" and "Above" Surface Interaction. With the widespread adoption of multi-touch devices and less expensive and intrusive tracking solutions such as the Microsoft Kinect, academic research on tabletop has refocused on "on" and "above" surface interaction techniques. Müller-Tomfelde *et al.* proposed different methods to use the space above the surface to provide ways of interacting with 2D tabletop content closer to reality [27]. While tangible devices complement the surface physically with a direct mapping to the GUI such as in the Photohelix system and StereoBlocks [18], finger gestures above the surface mimic physical interaction with real objects. Furthermore, instead of considering only finger touches, full hand posture on the surface can also be detected to provide richer interaction metaphors. Above the surface, the hand distance from the surface defines depth in 3D space giving a new dimension to the interactive region [27]. Wilson *et al.* proposed several metaphors to interact with different displays while capturing full body posture [36]. In this way, users can interact on or above the surface with 2D content or even between surfaces using the body to transfer virtual content to the hand or to another surface while moving their bodies in space. Users can also interact physically in space with projected GUI. In our system, we prefer to use the surface for GUI since it is more adequate for discrete selection and explore space gesture for modeling actions. Our approach explores the continuous space as presented by Marquardt *et al.* [26]; however we enrich their approach by combining it with the bimanual asymmetric model proposed by Guiard [15]. In addition, we rely on a stereoscopic visualization setup for architectural model visualization similar to [8]. While this system allows navigating or annotating the 3D scene mainly as if it was inside the table and use fingers as proxies over the scene, our interaction techniques focus on modeling and direct manipulation since 3D models are rendered as if they were lying atop the table. To avoid hands occlusions over the visualization, Toucheo [16] proposed a fish-tank like setup using a multi-touch surface and a stereoscopic display. However such as other setups relying on semi-transparent mirrors to create holographic illusion, it both reduces the working space and constrains the usage of the above surface space to hand gestures. Our stereoscopic visualization setup provides more freedom of movement allowing a continuous space of interaction. In addition, adopting a bimanual asymmetric model makes possible new interaction techniques which could benefit interaction with holographic display technologies when they become available.

3 Our Direct Modeling Approach

We propose a direct modeling approach to create, edit and manipulate 3D models using a small set of operations. Users interact through multi-touch gestures on a surface and gestures in space tracked by Gametrak[1] devices. Multi-touch gestures can also be used for sketching allowing to create 3D models by pushing and pulling existing content off the scene. Our models are represented using a boundary representation which decomposes the topology of objects into faces, edges and vertexes. Faces represent finite planar polygons or even surfaces delimited by edges. Edges are abstractions of segments or curves represented as 3D cubic Bézier parametric curves. These three kinds of topological features can be selected and edited by the user using direct manipulation in 3D space as explained below. Our push and pull approach proposes five operations. The simplest allows displacing topological features along a normal direction to change the geometry of the object without altering its topology. The second operation extrudes a face along the normal to extend the topology with new sided faces along the selected face. The third is a curvilinear extrusion which extends a shape by extruding a face along a path defined by a user gesture either in 3D space or on the surface. The fourth enables splitting faces by sketching linear or curvilinear strokes on them, subdividing those faces into more complex features. Finally, a snapping operation allows easily switching between surface and space editing when needed. This simple set of operations combined with modifiers (see Section 8) allows to create complex shapes through sketches and gestures using the same push and pull language as Google Sketchup or Sesame [29].

[1] See `http://en.wikipedia.org/wiki/Gametrak` for details.

3.1 User Inputs as Sketches or Gestures

We choose fingers tracking as our main input modality captured by the multi-touch surface when user touches it and by the Gametrak device once above. However to use such input data into sketches or gestures, we start by filtering the Gametrak data to remove the spatial jitter coming from the device and the user using the 1€ filter [6]. This data is then stored as an input gesture and updated continuously. While it is updated, input data is fitted incrementally to the best fit of lines and cubic Bézier curves. Thanks to this transformation, input gestures can be used as strokes creating shapes with sharp features or as gestures defining smooth trajectories. Our incremental fitting algorithm based on curve fitting tries to guarantee the continuity between curves and segments by adding tangency constraints during the fitting process without loosing fine details. This process also guarantees a maximal error distance of 7 millimeters between the raw and smoothed trajectories. This curve and line approximation is used for both sketches and gestures above the surface in place of the raw input data. While trajectory or 3D strokes could be defined directly using such representation, an additional beautification step is done on sketches to ease the creation of regular shapes. When a closed contour is created on the surface, further constraints are applied based on line segments to detect parallel and perpendicular line pairs and segment pairs with equal length. We use a threshold on angles between segments for parallelism and perpendicularity and a threshold ratio relationship between segments with similar length. An energy function is specified for each type of constraint and we perform an error minimization method to beautify user sketches. Regarding closed conic sections, we use a 2D shape recognizer [12] to detect circles and ellipses which are approximated by a closed piecewise curve using four cubic Bézier segments. This recognizer is also used to detect a simple erasing gesture used to delete shapes or strokes.

3.2 Selecting Modeling Parts

Selecting shapes or part of them is critical to any direct manipulation based approach. While this is done implicitly by touching a geometrical feature on the surface, we choose to use an explicit pinch gesture in space mimicking a grabbing gesture of physical objects. Visual feedback on shapes and geometrical features is provided based on their proximity with fingers.

Several selections can be performed with different granularity since any topological feature from our boundary representation can be edited. A whole shape can be selected by intersecting its bounding box with a finger. Intersecting a face, edge or vertex highlights it for selection. Since edges and vertices can be shared by more than one face or edge respectively, a continuous selection mechanism is provided to disambiguate the selection by analyzing the previously highlighted entity. For example, it is possible to highlight a particular edge of face shared by two faces by selecting it from the face the user is interested in. Empty selections, which are useful for scene manipulation, are possible both on the surface or in the space above it by simply selecting an empty area of the scene (i.e. one that does not intersect any bounding box of a shape).

3.3 Transitioning between Surface and Space

Creating 3D planar shapes in space remains an operation difficult to perform due to lack of physical constraints to guide the hand. We propose a snapping operator to easily switch between the surface and space allowing to use sketches on the surface or gestures in 3D space at convenience. Snapping is available through the contextual menu accessible on the NDH to snap on or back on any selected face. It works by computing a transformation matrix to align the 3D scene to the visible grid defined as a representation of the table surface. A simple linear animation between the two orientations is rendered to help the user understand the new orientation of the model. Furthermore, it allows sketching details on existing shapes or guaranteeing that new shapes are created on top of an existing shape. Additionally, since existing objects can occlude the selected face when snapping is performed, we give to the user the possibility to clip part of the scene using our menu. It is implemented using traditional OpenGL clipping planes defined as lying on the surface.

4 Hardware Modeling Setup

Our setup consists in a semi-immersive environment based on a stereoscopic multi-touch display 96×72 cm (42 inches) combined with a Kinect depth camera and two Gametraks used to identify and track the hands and fingers above the surface.

Head tracking is achieved in a non-intrusive way thanks to the Kinect using its skeleton detection algorithm. The skeleton is also used to track user hands allowing to locate the dominant hand according to the handedness of the user. Finger tracking is operated through multi-touch on the surface and using Gametrak devices in space (Figure 2). The visualization relies on a back-projection based system located under the table running at 120 Hz with a 1024 × 768 pixels resolution giving a pixel density of 10.6 pixels per cm (27 DPI). It is coupled with active shutter glasses from 3D Vision NVIDIA for the stereoscopic visualization.The 3D scene is rendered on top of the surface and the point of view is updated according to the position and orientation of the user's head to take into account motion parallax. The IR transmitter for the glasses uses an IR wavelength different from the multi-touch table which is based on the Diffuse Illumination technique. It is set at a position to cover the working volume around the table where the user interacts.

A camera running at 120 Hz with a 640×480 pixels resolution and positioned under the surface records finger movements on the surface, providing a maximum resolution of 6.4 dots per cm (16.25 DPI) for finger tracking. We use the iLight[2] framework version 1.6 for fingers detection and tracking. Fingers data are then sent using TUIO messages to our custom built application.

The two Gametraks are used to track the 3D position of the index and thumb of each hand when they are no longer in contact with the

[2]iliGHT Tactile Table product page: http://www.immersion.fr

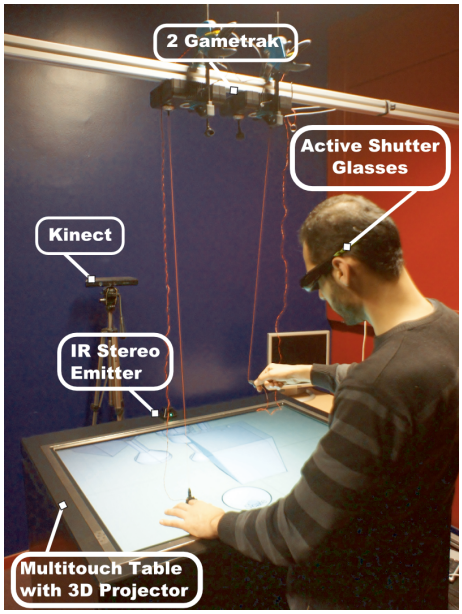


Figure 2: Overview of the setup.

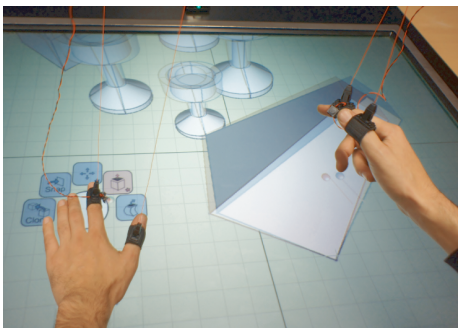


Figure 3: Detailed view of the Gametrak strings attached to the fingers with the buttons used for pinch gestures

multi-touch surface. These low cost gaming devices are placed in a reverse position centered above the table at a distance of 120 cm. The 3D position of each finger is computed from the two angles of rotation and the length of each cable, digitalized on 16 bits and reported at 125Hz to the host computer, resulting in a theoretical position resolution going from 500 dots per cm (1250 DPI) when the finger is close to the surface to 900 dots per cm (2250 DPI) when it is 50 cm above it. However the effective resolution is far lower (around 10 DPI) due to measurement noise. The retractable strings are attached to the fingers through a ring. Although strings introduce some visual clutter, they were not found to distract users from their task. The strings create a minor spring effect which reduces user hand tremor without adding fatigue. We added a 6mm diameter low profile momentary switch button on each index finger to detect pinch gestures without ambiguity (Figure 3). This simple solution provides a good trade-off regarding precision, cost and cumbersomeness compared to using a high end marker based optical tracking system or low sampling frequency (30 Hz) device such as the Kinect. The latter presents also a low tracking resolution (from 3 to 8 DPI) and is subject to finger occlusion.

The redundancy of information from the different input devices allows us to identify which finger of which hand is interacting on the surface or in the air or to choose the input source with the best tracking resolution.

5 Interpreting Input Data

Our setup relies on several input devices which should be on the same coordinate system to obtain a continuous interaction space. We chose the Kinect coordinate system as our primary coordinate system since it covers both the working and the user spaces. This section explains how we calibrate our continuous interaction space and how input data is fused into a single user model.

5.1 Calibrating Multi-touch Input Data

We provide a simple application for the user to pick the four corners of the multi-touch display in an image captured by the Kinect. These four points coupled with the 3D coordinate extracted from the Kinect depth map are used to compute the plane which minimizes the distance between them. The plane is then used to define two matrices converting touches on the surface into 3D positions and vice versa. Figure 4 presents a screenshot of our calibration application allowing the user to assess the correctness of the calibration thanks to a 3D preview of the plane and its mesh representation captured by the Kinect. The screen plane definition is used to define the frustum of the off-axis stereo perspective projection to render 3D content on top of the surface from the user point of view.

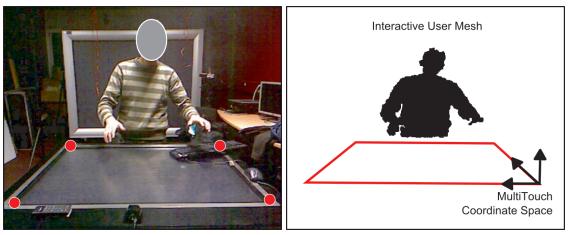


Figure 4: Calibrating 2D Touches: Kinect image camera with the four corner points selected by the user (red dots) on the left, 3D view of the user with the resulting screen plane on the right

5.2 Calibrating Gametrak Input Data

Gametrak input data is defined in a framework centered on the device base, requiring the computation of a transformation matrix into our primary coordinate system for each tracked finger. This is done using a set of one thousand matching 3D position pairs to compute the correspondence rigid transformation. The set is created by sampling the multi-touch surface screen and gathering the touch positions converted to our primary coordinate system using the matrix defined on the previous section. The rigid transformation is computed using a RANSAC algorithm [10], creating a matrix mapping Gametrak positions to our global coordinate system.

5.3 Fusing Inputs into a Single User Model

All input data that belong to the same finger are fused together as an input gesture. An input gesture might represent a stroke or gesture on or above the surface. Data coming from the multi-touch surface or the Gametraks has a unique identifier defined by the input device. After the coordinates have been converted into the same coordinate system, the fusing consists in determining when the identifiers from different sources correspond to the same finger. It also consists in adding the handedness information to each finger. A new input gesture is created when a finger touches the multi-touch surface without doing any pinch gesture, or when the finger performs the pinch and that finger was not touching the surface before. Input gestures are deleted when fingers are lifted from the surface without any pinching or when the pinch button is released above the surface. Otherwise the input gesture is updated. Multi-touch and Gametrak data are fused together based on close proximity. When a finger is on the multi-touch surface, we discard Gametrak data even if they are available as they were found to be less reliable. When a new input gesture is created, input handedness is determined by the closest hand position obtained from the Kinect skeleton.

6 Bimanual Interaction On the Surface

The multi-touch surface is primarily used as a sketching canvas where the user interacts using fingers. As previously explained, we followed the Guiard bimanual asymmetric model allowing the

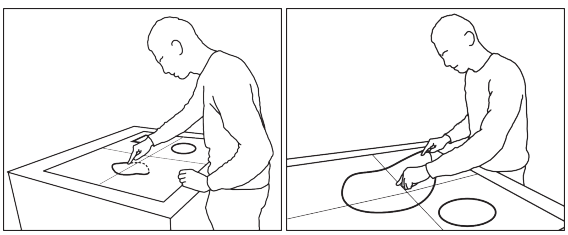


Figure 5: Bimanual Interaction on the Surface: Sketching using the DH (left) and scaling with both hands starting with the NDH (right).

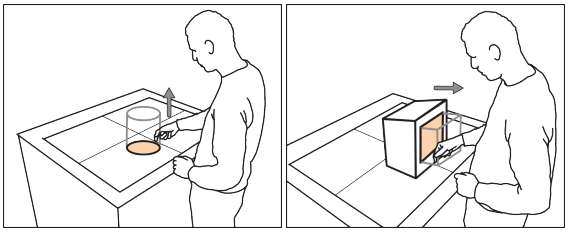


Figure 6: Face straight extrusion: along the surface normal direction (left), along a face normal direction (right).

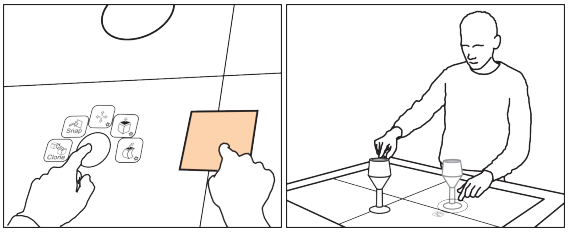


Figure 8: Example of menu presented under the NDH (left), cloning an object using both Hands (right)

user to implicitly switch between sketching tasks and object transformation / world manipulation (scale, rotate, translate operations on objects or on the world) depending on the hand used. Using the DH, user can sketch on the surface creating planar shapes from close contours. Contours might use lines, curves or both and can be sketched using multiple strokes. Open strokes whose extremities are close to each other are merged into a single stroke. Topological shape features are highlighted if a touch selection is performed nearby. Additionally, planar faces can be sub-divided into an arbitrary number of faces with different shapes if a face is overlapped by an open stroke starting and finishing outside that face. As explained in Section 3.1, strokes are automatically fitted into lines and curves ready to be used as sketch. However, we also use a 2D shape recognizer [12] allowing detecting simple gestures such as an erasing command by drawing a scribble. When an erasing gesture is recognized, if it overlaps open strokes, they are erased. However, if it overlaps only shapes and not open strokes, overlapped shapes are erased. This solution allows to use open strokes as construction lines while modeling.

When starting a gesture on the surface with the NDH, it is interpreted as object transformation if it is performed on an object, or world manipulation otherwise. Single touch gestures are interpreted as object or world translation. More than one finger gestures are interpreted as translation, rotation and scale operations on objects or world. 3D objects are constrained to movements along the plane parallel to the multi-touch surface. A gesture started with the NDH can be complemented by the DH allowing translation, rotation and scale with both hands (Figure 5).

Furthermore, bimanual interaction can be used to constrain drawing operations. In which case, the NDH defines constraints for the DH. For example, a user can sketch a straight line defining a plane of symmetry. First, the user selects the straight line using his NDH and sketches using the DH. As a result, the shapes sketched with the DH are mirrored by the plane of symmetry.

7 Continuous Interaction Above the Surface

Gestures with the DH above the surface are interpreted as 3D object creation or edition. Creation consists in extruding a planar shape previously sketched on the surface. The user first approaches the DH index finger near a shape on the surface to highlight it. He then performs a pinch gesture to extrude the shape along the normal of the surface (Figure 6). The height of the extruded object is then continuously updated and co-located with the finger position until the button is released. Planar shapes can also be extruded along the trajectory defined in the air after the user has selected this operation in a menu displayed on the NDH (Figure 7). While the user is defining the trajectory, the path is continuously re-evaluated and fitted into line segments and curve pieces to create a beautifulized freeform shape. Segments and curve pieces are created using the approach proposed by Coquillart [7] to offset the gesture from the centroid of the face to its vertexes and create a smooth free form extrusion of the profile. This method allows to extrude both poly-line and curvilinear profiles along linear or curvilinear paths.

Editing follows the push and pull modeling metaphor where topological features of the shape (vertexes, edges and faces) are moved in the air along the normal direction of the face it belongs to. As described in Section 3.2, our continuous selection method allows to distinguish which face an edge or a vertex belongs to if needed. The user first highlights the geometrical feature by moving his DH index finger close to it. He then selects it with a pinch gesture. The position of the geometrical feature is then updated according to the finger position until the pinch gesture is released. Alternatively faces can be extruded along to their normal or following the trajectory defined by the user after the corresponding operation has been selected in the menu displayed on the NDH. If no geometrical feature is selected while doing the pinch gesture with the DH, the user can sketch 3D poly-lines or curves in space.

The bimanual interaction used on the surface is also valid above the surface allowing to rotate, translate and scale objects using two fingers. As on the surface, the NDH begins the interaction using a pinch gesture. The NDH defines translations only while the DH adds rotation and scale operations using the method proposed by Wang *et al.* [33]. These direct 3D object manipulations appear much more efficient compared to indirect interactions on the multi-touch surface alone (e.g. changing the depth of an object while translating it along the surface plane).

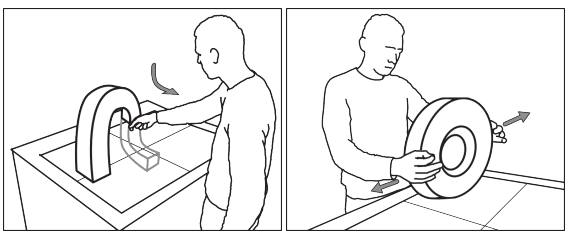


Figure 7: Extrusion along a curve gesture (left), 3D object scaling using both hands (right).

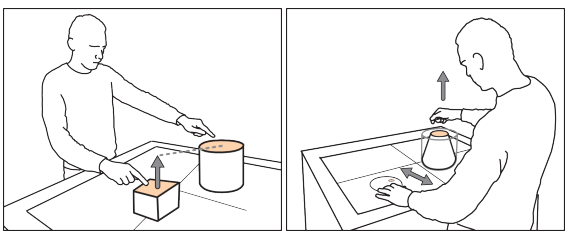


Figure 9: Defining an height constraint with the NDH (left), scaling with the NDH while extruding a shape (right).

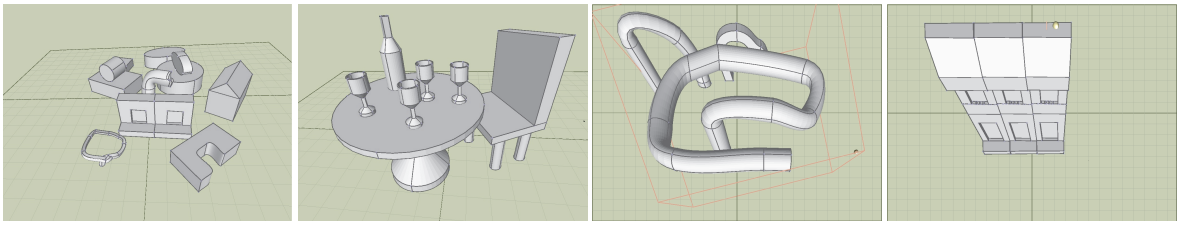


Figure 10: 3D models designed using Mockup Builder (from left to right): a set of shapes, a table with a chair, three different types of curved extruded profiles and a simple building façade. The last two images are rendered from the user point of view.

8 Exploring On and Above the Surface Interaction

We have previously used asymmetric hand operations to implicitly switch between sketching, object transformation and world manipulation. We now illustrates how the NDH can complement the operations performed by the DH with three types of operations.

First, the NDH can be used to select the mode used by the DH. Modes are presented through items shown in a contextual menu presented under the NDH. Modes presented in the contextual menu correspond to the ones available in the current mode associated to the operation performed by the DH (Figure 8). If the operation carried by the DH hand only supports a single mode, no contextual menu is shown under the NDH. To avoid visual clutter, the contextual menu transparency is adjusted based on the distance between the NDH and the surface. Above 15 cm, the menu is fully transparent and becomes progressively opaque as the NDH approaches the surface. To improve the accessibility, the contextual menu follows the NDH but its location is progressively fixed as the NDH comes closer to the surface to avoid spatial instabilities and reducing errors while selecting an item. This is simply done using the 1€ filter and adjusting its cutoff frequency based on the distance[6].

The discrete mode selection includes the type of extrusion (normal to a face or along a trajectory), the cloning operation and the snapping operation. Once in the cloning mode, discrete touches with the NDH define the location where clones appear. Snapping is available when a face is selected. It consists in rotating the world to align the face with the surface.

Instead of defining discrete operations through a contextual menu, the NDH can be used to select a geometrical feature that defines a constraint for the DH. The constraint is enabled as long as the NDH keeps his selection active. We use plane and line constraints in the extrusion and positioning operations. For example, the NDH can select a face of an object to define the maximum or minimum height for an object being extruded with the DH. Once the constraint is defined, the user continues to move his DH until the maximum or minimum height is reached. Further movements along the preceding direction do not continue to update the height of the object. This allows the user to also define that the height of an object should not be higher or lower that the height of another object. When translating an object, a plane constraint defines a limit beyond which an object cannot be moved further. While traditional modeling interfaces define constraints in a sequential way, we hypothesis that this definition of constraints on the fly allows to improve the flow of interaction.

Instead of defining discrete operations with the NDH, our last category of operations explores the usage of constrains continuously updated by the NDH. This is illustrated with the scale constraint that consists in scaling the profile while extruding a shape (Figure 7). This allows to create a cone or a frustum from a circle or a quadrilateral planar face respectively. The scaling factor can be controlled dynamically using a 2D overlay menu accessible by the NDH while extruding the shape.

9 Discussion and Conclusion

We have described an approach to model 3D scenes in a direct way using semi-immersive virtual environments through a synergistic combination of modalities afforded by novel input devices. Our system and experimental setup show that it is possible to enhance interaction by fusing data coming from different sensors. This provides a plausible environment combining benefits of multi-touch and stereo, using simple 3D operators, to model shapes using direct manipulation and simpler dialogues as compared to traditional and current systems. Combining the power of *bimanual interaction* with the flexibility of *continuous space*, we can provide effortless transition between modes and make it simple to switch between multi-touch 2D and spatial 3D gestures. This allows selecting the manipulations best suited to each task in non-obtrusive ways.

We implemented a prototype to demonstrate our modeling approach in C++ using OpenGL and OpenSG for stereoscopic visualization. Our system was deployed on an Intel I7 920 2.67 GHz processor with 3 Gb of memory RAM and an NVidia Quadro 4000 graphics card running Microsoft Windows 7 64-bit operating system. These first results are very encouraging and seemingly support further exploring our chosen avenue of work. Along the development, around 20 undergraduate and graduate students in Computer Science with variable experience with CAD applications and one Architectural researcher tested the system. They informally assessed the different design choices and iteratively improved the design of the interface. We plan to run formal evaluations with both novice and expert users to highlight and explore both the strengths and the weakness of our modeling interface. The remaining of the section discusses our achievements regarding our initial goal which was to provide a direct modeling solution.

Thanks to stereo, we provide co-location between user hands and virtual objects adapted to direct modeling methods. While the initial version used physics to detect collisions, this proved problematic while modeling. The feature was discarded instead of being activated on request. However it could be advantageous both for stacking and supporting 3D manipulations. While sketching is beneficial to surface–based interactions, beautification is a must to support creating more rigorous shapes for manufacturable objects. Figure 10 presents different models built using the interface by an expert user. As an example the second model from the left was built in 5'20" while the fourth took one of us 2'45" to complete. An expert user took 5'41" and 3'46" respectively for the same models using Rhino3D modeler. More rigorous tests should yield more exact measures, while direct editing of curves should be considered to reduce user retrials. On a positive note, the continuous interaction provides plausible gestures for extrusion and easy to define 3D trajectories leveraging the best features of the surface and space above it. While the surface invites users to sketch, the space above invites gestures and the snapping feature provides a suitable solution to transition between the two. In sum, *bimanual asymmetric interaction* provides an implicit switch between modeling and manipulation, letting the user focus on his design tasks. However

it might be confusing for some users, in particular when interacting with a large multi-touch surface. That is why we allow users to scale objects using both hands if they so do wish. Still, users should heed the precedence of the non–dominant hand. As in other sketching applications, menus could not be avoided altogether and are still required in particular when selecting from several modeling operations. However, providing a scaling widget while extruding provides an efficient separation of the degrees of freedom. We are considering to further explore multiple finger tracking as an alternative using non ambiguous start and end gestures. While speech as a modality could overcome such problems or alleviate the need for menus, on an interactive tabletop, button-like activation is likely more efficient and immune to recognition errors.

The system shows clear promise and provides a good case for augmenting interactive surfaces with gesturing gaze and body posture to support interactive modeling operations. The approach can be further extended by exploring combinations of different modalities and experimenting with mode-inferencing to further enhance the fluidity of our modeling techniques.

ACKNOWLEDGEMENTS This work was partially funded by the ANR InSTInCT project (ANR-09-CORD-013) , the Interreg IV-A 2 seas SHIVA project and by FCT (INESC-ID multiannual funding) through the PIDDAC Program funds, doctoral grant reference SFRH/BD/31020/2006 and MIVIS project PTDC/EIA-EIA/104031/2008.

REFERENCES

[1] S.-H. Bae, R. Balakrishnan, and K. Singh. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of UIST '08*, pages 151–160, NY, USA, 2008. ACM.

[2] R. Balakrishnan, G. Fitzmaurice, G. Kurtenbach, and W. Buxton. Digital tape drawing. In *Proc. of UIST '99*, pages 161–169, 1999.

[3] R. Balakrishnan and K. Hinckley. Symmetric bimanual interaction. In *Proceedings of CHI '00*, pages 33–40, NY, USA, 2000. ACM.

[4] R. Balakrishnan and G. Kurtenbach. Exploring bimanual camera control and object manipulation in 3d graphics interfaces. In *Proceedings of CHI '99*, pages 56–62, NY, USA, 1999. ACM.

[5] P. Brandl, C. Forlines, D. Wigdor, M. Haller, and C. Shen. Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces. In *Proc. of the working conference on Advanced visual interfaces*, AVI '08, pages 154–161, NY, USA, 2008.

[6] G. Casiez, N. Roussel, and D. Vogel. 1€ filter: A simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of CHI'12*. ACM, 2012.

[7] S. Coquillart. Computing offsets of b-spline curves. *Comput. Aided Des.*, 19:305–309, July 1987.

[8] J. B. De la Rivière, N. Dittlo, E. Orvain, C. Kervégant, M. Courtois, and T. Da Luz. iliGHT 3d touch: a multiview multitouch surface for 3d content visualization and viewpoint sharing. In *Proceedings of ITS '10*, pages 312–312, New York, NY, USA, 2010. ACM.

[9] J. Deisinger, R. Blach, M. Wesche, R. Breining, and A. Simon. Towards immersive modeling - challenges and recommendations: A workshop analyzing the needs of designers. In *Proc. of the 6th Eurographics Workshop on Virtual Environments*, pages 145–156, 2000.

[10] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395, June 1981.

[11] T. Fleisch, G. Brunetti, P. Santos, and A. Stork. Stroke-input methods for immersive styling environments. In *Shape Modeling and Applications*, pages 275–283, Los Alamitos, CA, USA, 2004. IEEE.

[12] M. Fonseca and J. Jorge. Using fuzzy logic to recognize geometric shapes interactively. In *The Ninth IEEE International Conference on Fuzzy Systems*, volume 1, pages 291 –296, 2000.

[13] A. S. Forsberg, J. J. L. Jr., and R. C. Zeleznik. Ergodesk: A framework for two- and three-dimensional interaction at the activedesk. In *Proc. of Immersive Projection Technology Workshop*, pages 11–12, 1998.

[14] T. Grossman, R. Balakrishnan, G. Kurtenbach, G. Fitzmaurice, A. Khan, and B. Buxton. Creating principal 3d curves with digital tape drawing. In *Proc. of CHI '02*, pages 121–128, NY, USA, 2002.

[15] Y. Guiard. Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. *Journal of Motor Behavior*, 19:486–517, 1987.

[16] M. Hachet, B. Bossavit, A. Cohé, and J.-B. de la Rivière. Toucheo: multitouch and stereo combined in a seamless workspace. In *Proceedings of UIST '11*, pages 587–592, NY, USA, 2011. ACM.

[17] K. Hinckley, K. Yatani, M. Pahud, N. Coddington, J. Rodenhouse, A. Wilson, H. Benko, and B. Buxton. Pen + touch = new tools. In *Proceedings of UIST '10*, pages 27–36, NY, USA, 2010.

[18] R. Jota and H. Benko. Constructing virtual 3d models with physical building blocks. In *Proceedings of CHI EA '11*, pages 2173–2178, New York, NY, USA, 2011. ACM.

[19] H. Kaufmann and D. Schmalstieg. Designing immersive virtual reality for geometry education. In *Proceedings of the IEEE conference on Virtual Reality*, VR '06, pages 51–58, DC, USA, 2006. IEEE.

[20] D. F. Keefe, R. C. Zeleznik, and D. H. Laidlaw. Drawing on air: Input techniques for controlled 3d line illustration. *IEEE Trans. Vis. Comput. Graph.*, 13(5):1067–1081, 2007.

[21] P. Lapides, E. Sharlin, M. C. Sousa, and L. Streit. The 3d tractus: A three-dimensional drawing board. In *Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 169–176, DC, USA, 2006. IEEE Computer Society.

[22] C. Latulipe, C. S. Kaplan, and C. L. A. Clarke. Bimanual and unimanual image alignment: an evaluation of mouse-based techniques. In *Proceedings of UIST '05*, pages 123–131, NY, USA, 2005.

[23] A. Leal, D. Bowman, L. Schaefer, F. Quek, and C. K. Stiles. 3d sketching using interactive fabric for tangible and bimanual input. In *Proceedings of Graphics Interface 2011*, GI '11, pages 49–56, 2011.

[24] J. Lee and H. Ishii. Beyond: collapsible tools and gestures for computational design. In *Proceedings of CHI EA '10*, pages 3931–3936, New York, NY, USA, 2010. ACM.

[25] P. Lopes, D. Mendes, B. Araújo, and J. A. Jorge. Combining bimanual manipulation and pen-based input for 3d modelling. In *Proceedings of SBIM '11*, pages 15–22, New York, NY, USA, 2011. ACM.

[26] N. Marquardt, R. Jota, S. Greenberg, and J. A. Jorge. The continuous interaction space: interaction techniques unifying touch and gesture on and above a digital surface. In *Proceedings of INTERACT'11*, pages 461–476, Berlin, Heidelberg, 2011. Springer-Verlag.

[27] C. Müller-Tomfelde, O. Hilliges, A. Butz, S. Izadi, and A. Wilson. *Tabletops - Horizontal Interactive Displays*. Human-Computer Interaction Series. Springer London, 2010.

[28] T. Novotny, I. Lindt, and W. Broll. A multi modal table-top 3d modeling tool in augmented environments. In *Proc. of the 12th Eurographics Symposium on Virtual Environments*, pages 45–52. EG, 2006.

[29] J.-Y. Oh, W. Stuerzlinger, and J. Danahy. Sesame: towards better 3d conceptual design systems. In *Proceedings of the 6th conference on Designing Interactive systems*, DIS '06, pages 80–89, NY, USA, 2006.

[30] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge. Technical section: Sketch-based modeling: A survey. *Comput. Graph.*, 33:85–103, February 2009.

[31] H. Perkunder, J. H. Israel, and M. Alexa. Shape modeling with sketched feature lines in immersive 3d environments. In *Proceedings SBIM '10*, pages 127–134, Aire-la-Ville, Switzerland, 2010. EG.

[32] S. Schkolne, M. Pruett, and P. Schröder. Surface drawing: creating organic 3d shapes with the hand and tangible tools. In *Proceedings of CHI '01*, pages 261–268, New York, NY, USA, 2001. ACM.

[33] R. Wang, S. Paris, and J. Popović. 6d hands: markerless hand-tracking for computer aided design. In *Proceedings of UIST '11*, pages 549–558, New York, NY, USA, 2011. ACM.

[34] G. Wesche and H.-P. Seidel. Freedrawer: a free-form sketching system on the responsive workbench. In *Proceedings of VRST '01*, pages 167–174, New York, NY, USA, 2001. ACM.

[35] E. Wiese, J. H. Israel, A. Meyer, and S. Bongartz. Investigating the learnability of immersive free-hand sketching. In *Proceedings of SBIM '10*, pages 135–142, Aire-la-Ville, Switzerland, 2010. EG.

[36] A. Wilson and H. Benko. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of UIST '10*, pages 273–282, NY, USA, 2010. ACM.

[37] M. Xin, E. Sharlin, and M. C. Sousa. Napkin sketch: handheld mixed reality 3d sketching. In *Proc. of VRST '08*, pages 223–226, 2008.

Bibliographie

Note : les publications référencées en caractères gras sont celles dont je suis coauteur.

[AG04] Georg Apitz and François Guimbretière. Crossy : a crossing-based drawing application. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, UIST '04, pages 3–12, New York, NY, USA, 2004. ACM.

[**AGS+11**] Michel Amberg, Frédéric Giraud, Betty Semail, Paolo Olivo, Géry Casiez, and Nicolas Roussel. Stimtac : a tactile input device with programmable friction. In *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology*, UIST '11 Adjunct, pages 7–8, New York, NY, USA, 2011. ACM.

[AM96] Motoyuki Akamatsu and I. Scott MacKenzie. Movement characteristics using a mouse with tactile and force feedback. *Int. J. Hum.-Comput. Stud.*, 45(4) :483–493, October 1996.

[AMH95] Motoyuki Akamatsu, I. Scott MacKenzie, and Thierry Hasbrouc. A comparison of tactile, auditory, and visual feedback in a pointing task using a mouse-type device. *Ergonomics*, 38(4) :816–827, 1995.

[AS10] Daniel Ashbrook and Thad Starner. Magic : a motion gesture design tool. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 2159–2168, New York, NY, USA, 2010. ACM.

[ASK+05] Takeshi Asano, Ehud Sharlin, Yoshifumi Kitamura, Kazuki Takashima, and Fumio Kishino. Predictive interaction using the delphian desktop. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, UIST '05, pages 133–141, New York, NY, USA, 2005. ACM.

[AZ01] Johnny Accot and Shumin Zhai. Scale effects in steering law tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '01, pages 1–8, New York, NY, USA, 2001. ACM.

[Bal04] Ravin Balakrishnan. "beating" fitts' law : virtual enhancements for pointing facilitation. *Int. J. Hum.-Comput. Stud.*, 61(6) :857–874, December 2004.

[**BCGLS08**] M. Biet, G. Casiez, F. Giraud, and B. Lemaire-Semail. Discrimination of virtual square gratings by dynamic touch on friction based tactile displays. In *Proceedings of the 2008 Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, HAPTICS '08, pages 41–48, Washington, DC, USA, 2008. IEEE Computer Society.

[BGBL04] Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. Semantic pointing : improving target acquisition with control-display ratio adaptation. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pages 519–526, New York, NY, USA, 2004. ACM.

[BGLS07] M. Biet, F. Giraud, and B. Lemaire-Semail. Squeeze film effect for the design of an ultrasonic tactile plate. *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, 54(12) :2678 –2688, december 2007.

[BH97] Doug A. Bowman and Larry F. Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, I3D '97, pages 35–ff., New York, NY, USA, 1997. ACM.

[BLFM05] Tim Berners-Lee, Roy Fielding, and Larry Masinter. Uniform Resource Identifier (URI) : generic syntax. RFC 3986, IETF, January 2005.

[BPIH10] Olivier Bau, Ivan Poupyrev, Ali Israr, and Chris Harrison. Teslatouch : electrovibration for touch surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 283–292, New York, NY, USA, 2010. ACM.

[Buc80] L. Buck. Motor performance in relation to control-display gain and target width. *Ergonomics*, 32 :579–589, 1980.

[BWC11] François Bérard, Guangyu Wang, and Jeremy R. Cooperstock. On the limits of the human motor control precision : the search for a device's human resolution. In *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part II*, INTERACT'11, pages 107–122, Berlin, Heidelberg, 2011. Springer-Verlag.

[**Cas04**] Géry Casiez. *Contribution à l'étude des interfaces haptiques. Le DigiHaptic : un périphérique haptique de bureau à degrés de liberté séparés*. PhD thesis, Université des Sciences et Technologies de Lille, 2004.

[CB05] Andy Cockburn and Stephen Brewster. Multimodal feedback for the acquisition of small targets. *Ergonomics*, 48(9) :1129–1150, 2005.

[**CCG10**] A. Choumane, G. Casiez, and L. Grisoni. Buttonless clicking : Intuitive select and pick-release through gesture analysis. In *Virtual Reality Conference (VR), 2010 IEEE*, pages 67 –70, march 2010.

[CEB78] C. Card, W. English, and B. Burr. Evaluation of mouse, rate controlled isometric joystick, step keys, and text keys for text selection on a crt. *Ergonomics*, 21 :601–613, 1978.

[CF03] Andy Cockburn and Andrew Firth. Improving the acquisition of small targets. In *Proceedings of HCI'03*, pages 77–80. BCS, 2003.

[CLP09] Olivier Chapuis, Jean-Baptiste Labrune, and Emmanuel Pietriga. DynaSpot : speed-dependent area cursor. In *Proceedings of CHI'09*, pages 1391–1400. ACM, 2009.

[CMR91] Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. A morphological analysis of the design space of input devices. *ACM Trans. Inf. Syst.*, 9(2) :99–122, April 1991.

[CQGF12] Andy Cockburn, Philip Quinn, Carl Gutwin, and Stephen Fitchett. Improving scrolling devices with document length dependent gain. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 267–276, New York, NY, USA, 2012. ACM.

[**CR11**] Géry Casiez and Nicolas Roussel. No more bricolage ! methods and tools to characterize, replicate and compare pointing transfer functions. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 603–614, New York, NY, USA, 2011. ACM.

[**CRV12**] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 1€ filter : A simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 2527–2530, New York, NY, USA, 2012. ACM.

[**CRVG10**] Géry Casiez, Nicolas Roussel, Romuald Vanbelleghem, and Frédéric Giraud. Efficacité et robustesse aux distracteurs d'un retour tactile pour faciliter le pointage. In *Conference Internationale Francophone sur I'Interaction Homme-Machine*, IHM '10, pages 25–32, New York, NY, USA, 2010. ACM.

[**CRVG11**] Géry Casiez, Nicolas Roussel, Romuald Vanbelleghem, and Frédéric Giraud. Surfpad : riding towards targets on a squeeze film effect. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 2491–2500, New York, NY, USA, 2011. ACM.

[**CV08**] Géry Casiez and Daniel Vogel. The effect of spring stiffness and control gain with an elastic rate control pointing device. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 1709–1718, New York, NY, USA, 2008. ACM.

[**CVBC08**] Géry Casiez, Daniel Vogel, Ravin Balakrishnan, and Andy Cockburn. The impact of control-display gain on user performance in pointing tasks. *Human-Computer Interaction*, 23(3) :215–250, 2008.

[**CVPC07**] Géry Casiez, Daniel Vogel, Qing Pan, and Christophe Chaillou. RubberEdge : reducing clutching by combining position and rate control with

elastic feedback. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 129–138, New York, NY, USA, 2007. ACM.

[CZ07] Xiang Cao and Shumin Zhai. Modeling human performance of pen stroke gestures. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 1495–1504, New York, NY, USA, 2007. ACM.

[CZMM99] Christopher S. Campbell, Shumin Zhai, Kim W. May, and Paul P. Maglio. What you feel must be what you see : adding tactile feedback to the trackpoint. In *Proc. of Interact'99*, pages 383–390. IOS Press, 1999.

[**DACJ12**] Bruno R. De Araùjo, Géry Casiez, and Joaquim A. Jorge. Mockup builder : direct 3d modeling on and above the surface in a continuous interaction space. In *Proceedings of the 2012 Graphics Interace Conference*, GI '12, pages 173–180, Toronto, Ont., Canada, Canada, 2012. Canadian Information Processing Society.

[**DACJH12**] Bruno De Araujo, Géry Casiez, Joaquim Jorge, and Martin Hachet. Modeling On and Above a Stereoscopic Multitouch Display. In *3DCHI - The 3rd Dimension of CHI*, Austin, United States, May 2012.

[**DCG10**] Jean-Philippe Deblonde, Géry Casiez, and Laurent Grisoni. Méthode de détermination des fonctions de gains. In *Conference Internationale Francophone sur I'Interaction Homme-Machine*, IHM '10, pages 141–144, New York, NY, USA, 2010. ACM.

[Deb12] Jean-Philippe Deblonde. *Exploitation de la dynamique du geste en IHM. Application aux fonctions de transfert pour le pointage et l'extraction d'événements discrets.* PhD thesis, Université de Lille, 2012.

[DLB+05] Lionel Dominjon, Anatole Lecuyer, Jean-Marie Burkhardt, Guillermo Andrade-Barroso, and Simon Richir. The "bubble" technique : Interacting with large virtual environments using haptic devices with limited workspace. In *Proceedings of the First Joint Eurohaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, WHC '05, pages 639–640, Washington, DC, USA, 2005. IEEE Computer Society.

[DM94] Sarah A. Douglas and Anant Kartik Mithal. The effect of reducing homing time on the speed of a finger-controlled isometric pointing device. In *Proceedings of the SIGCHI conference on Human factors in computing systems : celebrating interdependence*, CHI '94, pages 411–416, New York, NY, USA, 1994. ACM.

[DMH00] Jack Tigh Dennerlein, David B. Martin, and Christopher Hasser. Force-feedback improves performance for steering and combined steering-targeting tasks. In *Proc. of CHI'00*, pages 423–429. ACM, 2000.

[DY01] Jack Tigh Dennerlein and Maria C. Yang. Haptic force-feedback devices for the office computer : performance and musculoskeletal loading issues. *Human Factors*, 43(2) :278–286, 2001.

[Epp86] B.W. Epps. Comparison of six cursor control devices based on fitts' law models. In *In Proceedings of the 30th Annual Meeting of the Human Factors Society*, pages 327–331, 1986.

[FB08] Clifton Forlines and Ravin Balakrishnan. Evaluating tactile feedback and direct vs. indirect stylus input in pointing and crossing selection tasks. In *Proc. of CHI'08*, pages 1563–1572. ACM, 2008.

[FHSH06] Bernd Froehlich, Jan Hochstrate, Verena Skuk, and Anke Huckauf. The globefish and the globemouse : two new six degree of freedom input devices for graphics applications. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 191–199, New York, NY, USA, 2006. ACM.

[**GALSC12**] F. Giraud, M. Amberg, B. Lemaire-Semail, and G. Casiez. Design of a transparent tactile stimulator. In *Haptics Symposium (HAPTICS), 2012 IEEE*, pages 485 –489, march 2012.

[Gar74] W.R. Garner. *The Processing of Information and Structure*. 1974.

[GB05] Tovi Grossman and Ravin Balakrishnan. The bubble cursor : enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '05, pages 281–290, New York, NY, USA, 2005. ACM.

[GBLM99] Y. Guiard, M. Beaudouin-Lafon, and D. Mottet. Navigation as multiscale pointing : extending fitts' model to very high precision tasks. In *Proceedings of CHI'99*, pages 450–457. ACM, 1999.

[**GCR12**] Jérémie Gilliot, Géry Casiez, and Nicolas Roussel. Technique multi-points indirecte relative pour l'interaction avec des écrans de grandes dimensions. In *Conference Internationale Francophone sur I'Interaction Homme-Machine*, IHM '12, New York, NY, USA, 2012. ACM.

[GFB04] Sébastien Grange, Terrence Fong, and Charles Baur. M/oris : a medical/operating room interaction system. In *Proceedings of the 6th international conference on Multimodal interfaces*, ICMI '04, pages 159–166, New York, NY, USA, 2004. ACM.

[Gib62] C. B. Gibbs. Controller design : interactions of controlling limbs, time-lags and gains in positional and velocity systems. *Ergonomics*, 5(2) :385–402, April 1962.

[Gra96] Evan D. Graham. Virtual pointing on a computer display : non-linear control-display mappings. In *Proceedings of the conference on Graphics interface '96*, pages 39–46, Toronto, Ont., Canada, Canada, 1996. Canadian Information Processing Society.

[Gui87] Yves Guiard. Asymmetric division of labor in human skilled bimanual action : The kinematic chain as a model. *Journal of Motor Behavior*, 19 :486–517, 1987.

[HB04] Mark S. Hancock and Kellogg S. Booth. Improving menu placement strategies for pen input. In *Proceedings of Graphics Interface 2004*, GI '04, pages 221–230, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004. Canadian Human-Computer Communications Society.

[HCBM02] Ken Hinckley, Edward Cutrell, Steve Bathiche, and Tim Muss. Quantitative analysis of scrolling techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems : Changing our world, changing ourselves*, CHI '02, pages 65–72, New York, NY, USA, 2002. ACM.

[HCC07] Mark Hancock, Sheelagh Carpendale, and Andy Cockburn. Shallow-depth 3d interaction : design and evaluation of one-, two- and three-touch techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 1147–1156, New York, NY, USA, 2007. ACM.

[HCC09] Mark Hancock, Thomas T. Cate, and Sheelagh Carpendale. Sticky tools : Full 6dof force-based interaction for multi-touch tables. In *Proceedings of Interactive Tabletops and Surfaces 2009*, 2009.

[HCV+06] Mark S. Hancock, Sheelagh Carpendale, Frederic D. Vernier, Daniel Wigdor, and Chia Shen. Rotation and translation mechanisms for tabletop interaction. In *Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, 2006.

[HKLC03] Faustina Hwang, Simeon Keates, Patrick Langdon, and P. John Clarkson. Multiple haptic targets for motion-impaired computer users. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '03, pages 41–48, New York, NY, USA, 2003. ACM.

[**HLCC07**] Sébastien Hillaire, Anatole Lécuyer, Rémi Cozot, and Géry Casiez. Depth-of-field blur effects for first-person navigation in virtual environments. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, VRST '07, pages 203–206, New York, NY, USA, 2007. ACM.

[**HLCC08a**] S. Hillaire, A. Lecuyer, R. Cozot, and G. Casiez. Depth-of-field blur effects for first-person navigation in virtual environments. *Computer Graphics and Applications, IEEE*, 28(6) :47 –55, nov.-dec. 2008.

[**HLCC08b**] S. Hillaire, A. Lecuyer, R. Cozot, and G. Casiez. Using an eye-tracking system to improve camera motions and depth-of-field blur effects in virtual environments. In *Virtual Reality Conference, 2008. VR '08. IEEE*, pages 47 –50, march 2008.

[HTP+97] Ken Hinckley, Joe Tullio, Randy Pausch, Dennis Proffitt, and Neal Kassell. Usability analysis of 3d rotation techniques. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, UIST '97, pages 1–10, New York, NY, USA, 1997. ACM.

[HYP+10] Ken Hinckley, Koji Yatani, Michel Pahud, Nicole Coddington, Jenny Rodenhouse, Andy Wilson, Hrvoje Benko, and Bill Buxton. Manual deskterity :

an exploration of simultaneous pen + touch direct input. In *Proceedings of the 28th of the international conference extended abstracts on Human factors in computing systems*, CHI EA '10, pages 2793–2802, New York, NY, USA, 2010. ACM.

[IDA+06] Kori Inkpen, David Dearman, Ritchie Argue, Marc Comeau, Ching-Lung Fu, Sekhar Kolli, Jeremy Moses, Nick Pilon, and James R. Wallace. Left-handed scrolling for pen-based devices. *Int. J. Hum. Comput. Interaction*, 21(1) :91–108, 2006.

[IKHW08] James N Ingram, Konrad P Körding, Ian S Howard, and Daniel M Wolpert. The statistics of natural hand movements. *Experimental Brain Research.*, 188(2) :223–236, 2008.

[Iso01] Poika Isokoski. Model for unistroke writing time. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '01, pages 357–364, New York, NY, USA, 2001. ACM.

[JC90] Herbert D. Jellinek and Stuart K. Card. Powermice and user performance. In *Proceedings of the SIGCHI conference on Human factors in computing systems : Empowering people*, CHI '90, pages 213–220, New York, NY, USA, 1990. ACM.

[Joh94] T. Johnsgard. Fitts' law with a virtual reality glove and a mouse : Effects of gain. In *Proceedings of Graphics Interface Conference*, GI '94, pages 8–15, 1994.

[JSMM94] Robert J. K. Jacob, Linda E. Sibert, Daniel C. McFarlane, and M. Preston Mullen, Jr. Integrality and separability of input devices. *ACM Trans. Comput.-Hum. Interact.*, 1(1) :3–26, March 1994.

[KE88] B.H. Kantowitz and G.G. Elvers. Fitts' law with an isometric controller. effects of order of control and control-display gain. *Journal of Motor Behavior*, 20 :53–66, 1988.

[Key97] David V. Keyson. Dynamic cursor gain and tactual feedback in the capture of cursor movements. *Ergonomics*, 40(12) :1287 – 1298, 1997.

[KH90] Gordon Kurtenbach and Eric A Hulteen. *The Art of Human-Computer Interface Design*. Addison-Wesley Pub. Co., 1990.

[KI98] Kiyokuni Kawachiya and Hiroshi Ishikawa. Navipoint : an input device for mobile information browsing. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '98, pages 1–8, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[KI06] Masatomo Kobayashi and Takeo Igarashi. MoreWheel : Multimode scroll-wheeling depending on the cursor location. In *UIST 2006 Adjunct Proceedings : Demonstrations*, pages 57–58, 2006.

[KI08] Masatomo Kobayashi and Takeo Igarashi. Ninja cursors : using multiple cursors to assist target acquisition on large screens. In *Proceeding of CHI'08*, pages 949–958. ACM, 2008.

[KR10] Sven Kratz and Michael Rohs. A $3 gesture recognizer : simple gesture recognition for devices equipped with 3d acceleration sensors. In *Proceedings of the 15th international conference on Intelligent user interfaces*, IUI '10, pages 341–344, New York, NY, USA, 2010. ACM.

[KZ04] Per-Ola Kristensson and Shumin Zhai. Shark2 : a large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, UIST '04, pages 43–52, New York, NY, USA, 2004. ACM.

[KZ07] Per Ola Kristensson and Shumin Zhai. Command strokes with and without preview : using pen gestures on keyboard for command selection. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 1137–1146, New York, NY, USA, 2007. ACM.

[LCF76] G.D. Langolf, D.B. Chaffin, and J.A. Foulke. An investigation of fitts' law using a wide range of movement amplitudes. *Journal of Motor Behavior*, 8 :113–128, 1976.

[Li10] Yang Li. Protractor : a fast and accurate gesture recognizer. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 2169–2172, New York, NY, USA, 2010. ACM.

[LLR99] Allan Christian Long, Jr., James A. Landay, and Lawrence A. Rowe. Implications for a gesture design tool. In *Proceedings of the SIGCHI conference on Human factors in computing systems : the CHI is the limit*, CHI '99, pages 40–47, New York, NY, USA, 1999. ACM.

[LLR02] A. Chris Long, James A. Landay, and Lawrence A. Rowe. Multimodal interface for human-machine communication. chapter Helping designers create recognition-enabled interfaces, pages 121–146. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2002.

[LLRM00] A. Chris Long, Jr., James A. Landay, Lawrence A. Rowe, and Joseph Michiels. Visual similarity of pen gestures. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '00, pages 360–367, New York, NY, USA, 2000. ACM.

[Mar11] Anthony Martinet. *Etude de l'infuence de la séparation des degrés de liberté pour la manipulation 3-D à l'aide de surfaces tactiles multipoints*. These, Université des Sciences et Technologie de Lille - Lille I, October 2011.

[MB02] Michael McGuffin and Ravin Balakrishnan. Acquisition of expanding targets. In *Proceedings of the SIGCHI conference on Human factors in computing systems : Changing our world, changing ourselves*, CHI '02, pages 57–64, New York, NY, USA, 2002. ACM.

[**MCG09**] Anthony Martinet, Géry Casiez, and Laurent Grisoni. 3d positioning techniques for multi-touch displays. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, VRST '09, pages 227–228, New York, NY, USA, 2009. ACM.

[**MCG10a**] A. Martinet, G. Casiez, and L. Grisoni. The design and evaluation of 3d positioning techniques for multi-touch displays. In *3D User Interfaces (3DUI), 2010 IEEE Symposium on*, pages 115 –118, march 2010.

[**MCG10b**] Anthony Martinet, Géry Casiez, and Laurent Grisoni. The effect of dof separation in 3d manipulation tasks with multi-touch displays. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*, VRST '10, pages 111–118, New York, NY, USA, 2010. ACM.

[**MCG12**] Anthony Martinet, Gery Casiez, and Laurent Grisoni. Integrality and separability of multitouch interaction techniques in 3d manipulation tasks. *Visualization and Computer Graphics, IEEE Transactions on*, 18(3) :369 –380, march 2012.

[MD96] Anant Kartik Mithal and Sarah A. Douglas. Differences in movement microstructure of the mouse and the finger-controlled isometric joystick. In *Proceedings of the SIGCHI conference on Human factors in computing systems : common ground*, CHI '96, pages 300–307, New York, NY, USA, 1996. ACM.

[MG08] Regan L. Mandryk and Carl Gutwin. Perceptibility and utility of sticky targets. In *Proceedings of graphics interface 2008*, GI '08, pages 65–72. Canadian Information Processing Society, 2008.

[MGDS84] J.W. Morley, A.W. Goodwin, and I. Darian-Smith. Tactile discrimination of gratings. *Experimental Brain Research*, 49 :291–299, 1984.

[MJGJ11] Nicolai Marquardt, Ricardo Jota, Saul Greenberg, and Joaquim A. Jorge. The continuous interaction space : interaction techniques unifying touch and gesture on and above a digital surface. In *Proceedings of INTERACT'11*, pages 461–476, Berlin, Heidelberg, 2011. Springer-Verlag.

[MR94] I. S. MacKenzie and S. Riddersma. Effects of output display and control-display gain on human performance in interactive systems. *Behaviour and Information Tech- nology*, 1 :328–337, 1994.

[MSB91] I. Scott MacKenzie, Abigail Sellen, and William A. S. Buxton. A comparison of input devices in element pointing and dragging tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems : Reaching through technology*, CHI '91, pages 161–166, New York, NY, USA, 1991. ACM.

[MSK+88] D. E. Meyer, J. E. Smith, S. Kornblum, R. A. Abrams, and C. E. Wright. Optimality in human motor performance : Ideal control of rapid aimed movements. *Journal of Motor Behavior*, 95 :340–370, 1988.

[MSK+90] D. E. Meyer, J. E. Smith, S. Kornblum, R. A. Abrams, and C. E. Wright. Speed-accuracy tradeoffs in aimed movements : Toward a theory of rapid voluntary action. In *Proceedings of Attention and Performance XIII*, pages 173–226. Erlbaum, 1990.

[MTHB+10] Christian Müller-Tomfelde, Otmar Hilliges, Andreas Butz, Shahram Izadi, and Andy Wilson. *Tabletops - Horizontal Interactive Displays*. Human-Computer Interaction Series. Springer London, 2010.

[MWW10] Meredith Ringel Morris, Jacob O. Wobbrock, and Andrew D. Wilson. Understanding users' preferences for surface gestures. In *Proceedings of Graphics Interface 2010*, GI '10, pages 261–268, Toronto, Ont., Canada, Canada, 2010. Canadian Information Processing Society.

[NKK01] H.T. Nefs, A.M. Kappers, and J.J. Koenderink. Amplitude and spatial-period discrimination in sinusoidal gratings by dynamic touch. *Perception*, 30 :1263–1274, 2001.

[NMYT98] T. Nara, T. Maeda, Y. Yanagida, and S. Tachi. A tactile display using elastic waves in a "tapered plate". In *International Conference on Artificial Reality and Tele-existence*, pages 106–116, 1998.

[NSMG03] Michael Nielsen, Moritz Storring, Thomas B. Moeslund, and Erik Granum. A procedure for developing intuitive and ergonomic gesture interfaces for hci. In *GW'2003 : Gesture-Based Communication in Human-Computer Interaction, LNCS 2915*, pages 409–420. Springer Berlin / Heidelberg, 2003.

[OMBG00] Ian Oakley, Marilyn Rose McGee, Stephen Brewster, and Philip Gray. Putting the feel in 'look and feel'. In *Proc. of CHI'00*, pages 415–422. ACM, 2000.

[Pan08] Qing Pan. *Techniques d'interactions mixtes isotonique et élastique pour la sélection 2D et la navigation/manipulation 3D*. PhD thesis, Université de Lille, 2008.

[Poi02] Pointer ballistics for Windows XP. Archived white paper, Windows Hardware Developer Center, October 2002.

[**PPCR12**] Ludovic Potier, Thomas Pietrzak, Géry Casiez, and Nicolas Roussel. Méthodologie de conception de textures pour les interfaces tactiles à frottement programmable. In *Conference Internationale Francophone sur I'Interaction Homme-Machine*, IHM '12, New York, NY, USA, 2012. ACM.

[Pra01] W.K. Pratt. *Digital Image Processing, 3rd ed.* John Wiley & Sons, Inc., 2001.

[PTB98] J. G. Phillips, T. J. Triggs, and M. Bellgrove. Computer screen cursor trajectories as controlled by an accupoint : A kinematic analysis. In *Proceedings of the Australasian Conference on Computer Human Interaction*, OZCHI '98, pages 314–, Washington, DC, USA, 1998. IEEE Computer Society.

[**QCC$^+$12**] Philip Quinn, Andy Cockburn, Géry Casiez, Nicolas Roussel, and Carl Gutwin. Exposing and understanding scrolling transfer functions. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, New York, NY, USA, 2012. ACM.

[Qua10] Xu Quan. *Contribution à l'étude et au développement de techniques de gestion de fenêtres*. These, Université des Sciences et Technologie de Lille - Lille I, December 2010.

[RB03] Gonzalo Ramos and Ravin Balakrishnan. Fluid interaction techniques for the control and annotation of digital video. In *Proceedings of the 16th annual*

ACM symposium on User interface software and technology, UIST '03, pages 105–114, New York, NY, USA, 2003. ACM.

[RB10] Julie Rico and Stephen Brewster. Usable gestures for mobile interfaces : evaluating social acceptability. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 887–896, New York, NY, USA, 2010. ACM.

[RBB04] Gonzalo Ramos, Matthew Boulos, and Ravin Balakrishnan. Pressure widgets. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pages 487–494, New York, NY, USA, 2004. ACM.

[**RCAV12**] Nicolas Roussel, Géry Casiez, Jonathan Aceituno, and Daniel Vogel. Giving a hand to the eyes : Leveraging input accuracy for subpixel interaction. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, New York, NY, USA, 2012. ACM.

[RDH09] Jason L. Reisman, Philip L. Davidson, and Jefferson Y. Han. A screen-space formulation for 2d and 3d direct manipulation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pages 69–78, New York, NY, USA, 2009. ACM.

[RS90] Joseph D. Rutledge and Ted Selker. Force-to-motion functions for pointing. In *Proceedings of the IFIP TC13 Third Interational Conference on Human-Computer Interaction*, INTERACT '90, pages 701–706, Amsterdam, The Netherlands, The Netherlands, 1990. North-Holland Publishing Co.

[RS98] J.D. Rutledge and E.J. Selker. Controller for improved pointing devices. US Patent 5,764,219, 1998.

[RS00] Jun Rekimoto and Eduardo Sciammarella. Toolstone : effective use of the physical manipulation vocabularies of input devices. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, UIST '00, pages 109–117, New York, NY, USA, 2000. ACM.

[Rub91] Dean Rubine. Specifying gestures by example. *SIGGRAPH Comput. Graph.*, 25(4) :329–337, 1991.

[SBG+11] Hyunyoung Song, Hrvoje Benko, Francois Guimbretiere, Shahram Izadi, Xiang Cao, and Ken Hinckley. Grips and gestures on a multi-touch pen. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 1323–1332, New York, NY, USA, 2011. ACM.

[Sch41] E. Scheyer. French drawings of the great revolution and the napoleonic era. *The Art Quartely IV*, 4 :187–204, 1941.

[Shn96] Ben Shneiderman. The eyes have it : A task by data type taxonomy for information visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, VL '96, pages 336–, Washington, DC, USA, 1996. IEEE Computer Society.

[**SLLC09**] Jean-Baptiste Sauvan, Anatole Lécuyer, Fabien Lotte, and Géry Casiez. A performance model of selection techniques for p300-based brain-computer interfaces. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 2205–2208, New York, NY, USA, 2009. ACM.

[Syn] Synaptics. Edgemotion.

[TD91] U. Trankle and D. Deutschmann. Factors influencing speed and precision of cursor positioning using a mouse. *Ergonomics*, 34 :161–174, 1991.

[TFK+06] M. Takasaki, Y. Fujii, H. Kotani, T. Mizuno, and T. Nara. Proposal of tele-touch using active type saw tactile display. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1032–1037, October 2006.

[Tuc04] A.B. Tucker. *Computer Science Handbook, Second Edition*. 2004.

[TXW+08] Feng Tian, Lishuang Xu, Hongan Wang, Xiaolong Zhang, Yuanyuan Liu, Vidya Setlur, and Guozhong Dai. Tilt menu : using the 3d orientation information of pen devices to extend the selection capability of pen-based user interfaces. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 1371–1380, New York, NY, USA, 2008. ACM.

[VB05] Daniel Vogel and Ravin Balakrishnan. Distant freehand pointing and clicking on very large, high resolution displays. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, UIST '05, pages 33–42, New York, NY, USA, 2005. ACM.

[VB10] Daniel Vogel and Ravin Balakrishnan. Occlusion-aware interfaces. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 263–272, New York, NY, USA, 2010. ACM.

[**VC11**] Daniel Vogel and Géry Casiez. Conté : multimodal input inspired by an artist's crayon. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 357–366, New York, NY, USA, 2011. ACM.

[**VC12**] Daniel Vogel and Géry Casiez. Hand occlusion on a multi-touch tabletop. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 2307–2316, New York, NY, USA, 2012. ACM.

[**VCC+09**] Daniel Vogel, Matthew Cudmore, Géry Casiez, Ravin Balakrishnan, and Liam Keliher. Hand occlusion with tablet-sized direct pen input. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 557–566, New York, NY, USA, 2009. ACM.

[VIM12] *Vocabulaire international de métrologie - Concepts fondamentaux et généraux et termes associés (VIM)*. JCGM, 2012.

[**VVCG11**] Radu-Daniel Vatavu, Daniel Vogel, Géry Casiez, and Laurent Grisoni. Estimating the perceived difficulty of pen gestures. In Pedro Campos, Nicholas Graham, Joaquim Jorge, Nuno Nunes, Philippe Palanque, and Marco Winckler, editors, *Human-Computer Interaction - INTERACT 2011*, volume 6947 of *Lecture Notes in Computer Science*, pages 89–106. Springer Berlin / Heidelberg, 2011. 10.1007/978-3-642-23771-3_9.

[WB10] Andy Wilson and Hrvoje Benko. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of UIST '10*, pages 273–282, NY, USA, 2010. ACM.

[WF95] T. Watanabe and S. Fukui. A method for controlling tactile sensation of surface roughness using ultrasonic vibration. *IEEE Int. Conf. on Robotics and Automation*, pages 1134–1139, 1995.

[WFL+09] Jacob O. Wobbrock, James Fogarty, Shih-Yen (Sean) Liu, Shunichi Kimuro, and Susumu Harada. The Angle Mouse : target-agnostic dynamic gain adjustment based on angular deviation. In *Proceedings of CHI'09*, pages 1401–1410. ACM, 2009.

[WMSB98] Yanqing Wang, Christine L. MacKenzie, Valerie A. Summers, and Kellogg S. Booth. The structure of object transportation and orientation in human-computer interaction. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '98, pages 312–319, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[WMW09] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. User-defined gestures for surface computing. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 1083–1092, New York, NY, USA, 2009. ACM.

[WPP11] Robert Wang, Sylvain Paris, and Jovan Popović. 6d hands : markerless hand-tracking for computer aided design. In *Proceedings of UIST '11*, pages 549–558, New York, NY, USA, 2011. ACM.

[WWBH97] Aileen Worden, Nef Walker, Krishna Bharat, and Scott Hudson. Making computers easier for older adults to use : area cursors and sticky icons. In *Proceedings of CHI'97*, pages 266–271. ACM, 1997.

[WWL07] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures without libraries, toolkits or training : a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, UIST '07, pages 159–168, New York, NY, USA, 2007. ACM.

[**XC10**] Quan Xu and Géry Casiez. Push-and-pull switching : window switching based on window overlapping. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 1335–1338, New York, NY, USA, 2010. ACM.

[YNYH06] A. Yamamoto, S. Nagasawa, H. Yamamoto, and T. Higuchi. Electrostatic tactile display with thin film slider and its application to tactile telepresentation systems. *IEEE Transactions on Visualization and Computer Graphics*, 12(2) :168–177, 2006.

[Zha95] Shumin Zhai. *Human Performance in Six Degree of Freedom Input Control*. PhD thesis, University of Toronto, 1995.

[ZM06] Robert Zeleznik and Timothy Miller. Fluid inking : augmenting the medium of free-form inking with gestures. In *Proceedings of Graphics Interface 2006*, GI '06, pages 155–162, Toronto, Ont., Canada, Canada, 2006. Canadian Information Processing Society.

[ZSS97] Shumin Zhai, Barton A. Smith, and Ted Selker. Improving browsing performance : A study of four input devices for scrolling and pointing tasks. In *Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, INTERACT '97, pages 286–293, London, UK, UK, 1997. Chapman &amp ; Hall, Ltd.