Visualisation d'information : modélisation, interaction et nouveaux dispositifs

Mountaz Hascoët

UNIVERSITÉ MONTPELLIER II

— SCIENCES ET TECHNIQUES DU LANGUEDOC —

Visualisation d'information : modélisation, interaction et nouveaux dispositifs

Mountaz Hascoët

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, LIRMM

Département Informatique, Projet Visualisation Algorithmes et Graphes

Document présenté à l'Université des Sciences et Techniques du Languedoc
pour être autorisé à candidater au diplôme
d'habilitation à diriger des recherches
Version de relecture

SPÉCIALITÉ : **INFORMATIQUE**
Formation Doctorale : **Informatique**
École Doctorale : **Information, Structures, Systèmes**

TABLE DES MATIÈRES

Chapitre 1

INTRODUCTION

Mes travaux de recherche portent sur les problèmes d'interaction homme-machine liés à l'exploitation des ressources et des informations textuelles numérisées, qu'elles soient accessibles par des serveurs distants ou stockées localement.

Le fil conducteur de ces travaux de recherche se caractérise par une approche à la fois expérimentale et théorique des problèmes étudiés pour une meilleure prise en compte des capacités cognitives, perceptives et motrices de l'être humain. L'objectif étant de lui permettre au mieux d'exploiter et de contrôler les systèmes d'accès, de traitement et de manipulation des informations numériques.

Ce fil conducteur se retrouve dans tous mes travaux de recherches qui peuvent se structurer autour d'axes complémentaires : visualisation, interaction, modélisation, dispositifs matériels et évaluation.

Plusieurs thèses se sont inscrites dans ces différents axes. La thèse de Jérôme Thièvre relève de l'axe visualisation et interaction. Elle a consisté à développer des techniques de représentations multiples pour aider à l'indexation et à l'exploration de collections de documents multimédias [76]. La thèse de François Boutin [21] s'est plus particulièrement tournée sur les aspects modélisation et construction de modèles multi-échelles. Enfin la thèse de Maxime Collomb [30] porte davantage sur les nouveaux dispositifs matériels et en particulier sur les environments d'affichage distribués et hétérogènes.

Le but de ce document n'est pas d'être exhaustif sur ces domaines mais plutôt de donner notre vision et de présenter nos contributions. Le document s'organise de la manière suivante : nous abordons en premier les questions de visualisation d'information et d'interaction, nous poursuivons avec quelques éléments de modélisation et de construction automatique qui se situent en amont de la création des vues. Nous terminons par le chapitre sur l'impact de l'évolution des dispositifs matériels sur la visualisation et l'interaction et une conclusion sur nos perspectives de recherche. Dans certains chapitres nous avons intégré nos contributions dans la discussion générale c'est le cas

du chapitre 1 notamment. Dans les autres chapitres nous avons au contraire préféré séparer contexte et contributions.

Chapitre 2

VISUALISATION D'INFORMATION ET INTERACTION

Ce chapitre présente notre perspective sur le domaine de la visualisation et de l'interaction. Il inclut nos contributions. Les publications mises en annexe en fin de ce document ont été choisies de sorte à donner une vision plus approfondie de certaines de ces contributions.

1 Visualisation d'information

Le but de la visualisation d'information est d'exploiter les caractéristiques du système visuel humain pour faciliter la manipulation et l'interprétation de données informatiques variées.

Cette approche est assez bien caractérisée par la mantra de Ben Shneiderman : Overview first, zoom and filter, and details on demand

Elle se justifie pleinement par les travaux en perception visuelle qui ont montré que l'être humain a une perception d'abord globale d'une scène, avant de porter son attention aux détails [97].

D'autres travaux en conception graphique tels que ceux de Tufte [109] et plus particulièrement sur le codage de Bertin [14] ont montré comment exploiter, de façon intuitive ou ad hoc, ces caractéristiques de perception globale. La visualisation d'information cherche à exploiter ces mêmes caractéristiques de façon plus systématique afin de faciliter la réalisation de certaines tâches liées à la recherche d'information au sens large comme par exemple :

- ▷ exploration rapide d'ensembles d'informations inconnues ;
- ▷ mise en évidence de relations et de structures dans les informations ;
- ▷ mise en évidence de chemins d'accès à des informations pertinentes ;
- ▷ classification interactive des informations.

L'essor qu'a connu le domaine de la visualisation interactive d'information depuis plus de dix ans a conduit à l'émergence d'alternatives au modèle classique des interfaces graphiques dites WIMP – Window, Icon, Menu, Pointer.

La création automatique de vues d'ensemble pour des collections d'informations quelconques est très mal prise en charge par les interfaces de type WIMP. Dans ces interfaces, les composants classiques habituellement utilisés pour visualiser des informations tels que les listes déroulantes, les listes arborescentes ou même les tableaux cessent d'être efficaces lorsque la collection d'information à visualiser dépasse quelques centaines d'items. L'inefficacité de ces composants apparaît à différents niveaux. Pour commencer, la proportion des documents visibles par rapport aux documents présents dans la collection est particulièrement faible. Une liste déroulante qui occupe environ 100x200 pixels ne permet de visualiser simultanément qu'une vingtaine d'items. Ceci a non seulement pour effet de rendre difficile certaines tâches comme par exemple les comparaisons entre des éléments éloignés dans la liste qui ne peuvent pas être visibles simultanément, mais également de rendre le parcours de la liste beaucoup trop lent. En outre, ce type de composant s'avère également relativement coûteux en espace d'affichage, et inefficace quant à sa capacité à faire apparaître des relations entre différents items. Les alternatives ont donc émergés ces dernières années et sont présentées par rapport au type des collections pour lesquelles elles sont conçues. Parmi ces alternatives, nous nous interessons plus particulièrement aux éléments qui permettent la visualisation de données multidimensionnelles, d'arbres, de graphes ou de données plein-texte représentées dans un modèle vectoriel.

1.1 Visualisation de données multidimensionnelles

Les données multidimensionnelles correspondent aux informations issues de bases de données relationnelles par exemple, de tableaux ou d'autres modèles permettant de représenter chaque élément d'information selon ses valeurs dans n dimensions.

Dans ce domaine les travaux de Inselberg ([71]) ou d'autres travaux tels ceux de Keim ([79]) proposent des diagrammes dont l'objectif est de faire apparaître des tendances ou des relations entre des données. Les diagrammes par lignes d'Inselberg (cf Figure 1) sont créés selon un principe simple : il s'agit de représenter chaque élément de la base par une polyligne dans un système à n axes, n étant le nombre de dimensions de la donnée. Chacun des axes représente l'une des dimensions des données. Par exemple, si l'on considère des informations sur des voitures, un axe peut représenter le prix de vente de la voiture, l'autre l'année de première mise en circulation, etc. Dans cet exemple une voiture est entièrement représentée par une polyligne qui joint tous les points placés sur les axes pour représenter les valeurs que prend ladite voiture sur chaque dimension. Lorsque les dimensions de la donnée à représenter sont numériques, la correspondance entre une valeur pour

cette dimension et les coordonnées du point représentant cette valeur sur l'axe résulte d'un simple rapport de proportionnalité.
L'intérêt de ces diagrammes est immédiat : parce qu'il permettent de visualiser simultanément un nombre de données considérables, ils "dessinent" des modèles, des tendances, des corrélations ou encore des anomalies dans les données que l'œil humain peut repérer. Leur visée est essentiellement exploratoire : ils ont pour but de donner l'intuition ou de faire découvrir tel ou tel phénomène difficilement identifiable et qui se produirait sur des collections de données diverses. Des analyses plus poussées permettent ensuite de confirmer ou d'infirmer ces découvertes.

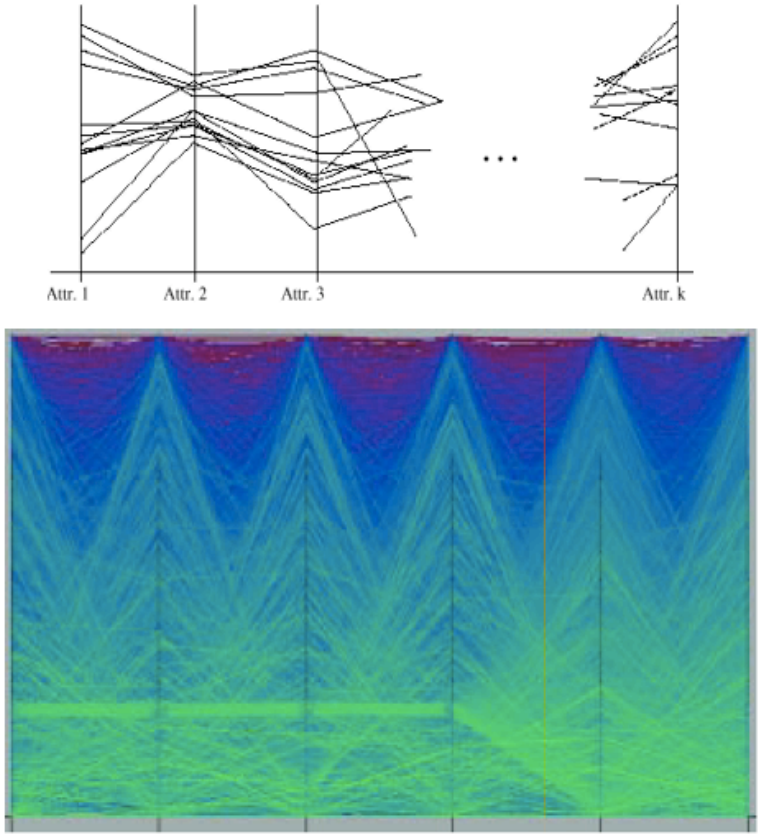

Fig. 1: *Diagrammes d'Inselberg : principe (en haut) et application à la visualisation de données boursières par Keim (en bas)*

Toute la difficulté dans la création de ces diagrammes réside dans la difficulté à faire apparaître des vues pertinentes. Pour ce faire, il est nécessaire

de jouer sur tous les paramètres du modèle et de faire en sorte que les visualisations qui en résultent soient interprétables, une partie de ces problèmes est abordée par Inselberg dans [71].

1.2 Visualisation d'arbres

Approche diagramme Généralement, des informations organisées hiérarchiquement (catalogues thématiques de sujets, documents organisés en chapitres, sections, etc.) sont visualisées sous forme de listes indentées. Ce type de représentation comporte un inconvénient majeur pour des arborescences de grande taille car elle ne parvient pas à donner une vue d'ensemble satisfaisante : l'arbre n'est jamais entièrement visible car le nombre de nœuds affichés simultanément est forcément limité par l'espace disponible. Comme pour les listes traditionnelles cela rend inefficace un certain nombre de tâches liées à la manipulation/exploitation de l'arbre.

D'autres techniques ont été développées pour les arbres de plus grande taille. Ces techniques privilégient la visualisation de l'ensemble de l'arbre aux dépens des détails de chaque élément. Ces détails pourront être obtenus par interaction sur l'arbre. Dans ce domaine, on peut distinguer quatre approches de visualisation très différentes :

- ▷ Approche diagramme ([8; 65])
- ▷ Approche "surfacique" : cartes d'arbres ([104], arbres circulaires ([4])
- ▷ Approche 3D : les arbres coniques ([98])
- ▷ Approche géométrie non-euclidienne : les arbres hyperboliques ([84])

Une première alternative à l'approche usuelle de listes arborescentes consiste à représenter l'arbre par un diagramme constitué de nœuds et de branches. Dans cette approche, le principal problème consiste à définir l'algorithme idéal de placement d'arbre. Pour cela, de nombreux algorithmes existent, surtout si l'on considère qu'un arbre étant un cas particulier de graphe, tous les algorithmes permettant de dessiner un graphe peuvent s'appliquer ([8; 65]).

Parmi les algorithmes les plus couramment utilisés dans le cadre de la visualisation d'information, on peut distinguer ceux pour lesquels les caractéristiques principales conduisent à mettre en valeur l'arborescence en utilisant une représentation usuelle horizontale ou verticale (cf. Figure 2) et ceux qui visent à mieux tirer parti de l'espace disponible en adoptant une représentation concentrique.

Dans le premier cas, les algorithmes de placement cherchent à satisfaire un certain nombre de contraintes : centrer un nœud au-dessus de ses enfants, respecter un espace inter-nœuds ou dessiner de façon similaire des sous-arbres isomorphes. Dans certains cas, les algorithmes permettent d'optimiser la lar-

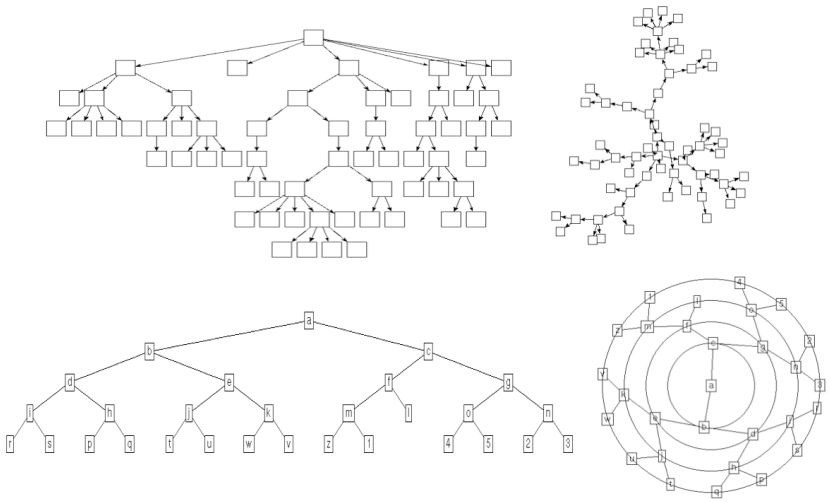

Fig. 2: *Diagramme d'arbres algorithme de Walker (gauche haut), binaire (gauche bas), algorithme général de placement de graphe de Kamada (droite haut), radial (droite bas)*

.

geur du graphe. Parmi les algorithmes utilisés dans la Figure 2 pour produire un dessin d'arbre vertical, le premier est une généralisation d'un algorithme de placement binaire et présente l'avantage d'être extrêmement simple à implémenter ([8]). Le deuxième ([110]) est moins simple mais donne de meilleurs résultats en minimisant entre autres la largeur utile pour le dessin.

Les algorithmes du deuxième type partagent une partie de ces contraintes mais intègrent une contrainte supplémentaire : tirer le meilleur parti de l'espace disponible. A cet effet, les algorithmes radiaux utilisent mieux l'espace que les représentations verticales ou horizontales : les nœuds de chaque niveau de l'arbre sont répartis sur des cercles ou ellipses concentriques dont le rayon augmente avec la profondeur dans l'arbre. Ce type de représentation est particulièrement bien adapté à des arbres relativement peu profonds pour lesquels il existe un grand nombre de feuilles (comme par exemple certains sites Web ou certains catalogues de sites). Nous avons remarqué deux effets contrastés avec l'utilisation de ce type de représentations : l'arborescence est moins bien perçue par une majorité d'utilisateurs, et pourtant ce type de vue est en général préféré ([55]).
L'utilisation des diagrammes ne résout pas tous les problèmes liés au volume de l'information à visualiser. Ainsi l'approche sous forme de diagramme peut-elle être envisagée avec succès pour des arbres contenant plusieurs centaines de nœuds mais elle l'est rarement pour des arbres de plusieurs milliers de

nœuds.

Approche surfacique Les cartes d'arbres ou "tree-maps" ([104]) ont justement pour but de traiter des arbres de plusieurs dizaines de milliers de noeuds. Leur principe consiste à visualiser un arbre en découpant une surface donnée (en général rectangulaire) proportionnellement à chaque sous-arbre. Chaque rectangle représente une feuille de l'arbre et sa surface est proportionnelle à son importance dans l'arbre. De même chaque regroupement horizontal ou vertical de rectangles représente un sous-arbre proportionnellement à son importance (cf. Figure 5). Cette approche permet de visualiser rapidement l'ensemble de l'arbre pour déceler certains éléments particuliers, la couleur ainsi que le motif de fond peuvent être utilisés pour représenter certaines caractéristiques des nœuds.

Les arbres représentés par des demi-disques (cf. Figure 3) relèvent d'une démarche analogue ([4]). Chaque niveau de l'arbre est représenté par un anneau que les nœuds se partagent. Par rapport aux arbres planaires, l'avantage de cette représentation est qu'elle rend l'arborescence beaucoup plus facilement perceptible. Par contre, l'affichage de texte à l'intérieur d'un nœud est quasiment impossible car les nœuds ont la forme de secteurs d'anneaux. A vrai dire ce problème se manifeste également avec les tree-maps lorsque les rectangles sont très étirés. Cependant, il existe des variantes des tree-maps permettant d'optimiser les proportions des rectangles représentant les nœuds de façon à pouvoir afficher des informations textuelles dans ces rectangles.

Dessin d'arbres en 3 dimensions Une autre approche consiste à avoir recours à la 3D pour visualiser un plus grand nombre de nœuds (cf. Figure 4). Les arbres coniques constituent le meilleur exemple de cette approche ([98; 99]). Ils permettent d'afficher des arbres de grande taille de façon intuitive. L'inconvénient majeur de ces techniques est lié à l'usage de la 3D et notamment aux problèmes d'occlusion, de manipulation et d'orientation qui lui sont inhérents.

Géométrie hyperbolique Le recours à la géométrie hyperbolique (Lamping, Rao et al., 1995) pour représenter des arbres (cf. Figure 6) peut être vu comme un cas particulier de l'approche diagramme. Les caractéristiques très particulières des transformations de la géométrie hyperbolique justifient néanmoins qu'on l'en distingue. Selon cette approche, les nœuds de l'arbre sont placés de manière analogue à l'approche radiale sur des cercles concentriques dont le rayon est calculé proportionnellement à la profondeur du nœud dans l'arbre. L'intérêt de la géométrie hyperbolique est qu'ici le périmètre des

cercles hyperboliques croit exponentiellement avec leur rayon ce qui permet de visualiser facilement des arbres contenant un grand nombre de feuilles.

Les transformations de la géométrie hyperbolique permettent d'interagir avec la vue et en particulier de déplacer des nœuds d'intérêts au centre. Ces transformations très particulières ont un effet un peu surprenant pour un utilisateur non entraîné, comparable à l'effet obtenu lors d'un placement radial avec application d'une transformation fisheye (cf. Figure 7 par exemple). Dans les deux cas, les utilisateurs non entraînés ont besoin de faire un effort cognitif non négligeable pour reconstituer l'effet des transformations sur le contexte qui se déplace de manière inhabituelle. Il reste à savoir si cette géométrie et les transformations qui en résultent peuvent être maîtrisées après quelques séances pour permettre d'exploiter pleinement le bénéfice de cette technique.

1.3 Visualisation de graphes

Très souvent, des graphes sont utilisés pour organiser des ensembles de données. C'est le cas par exemple des réseaux sociaux, des réseaux sémantiques, des diagrammes UML, etc.
Pour visualiser des données de ce type, il est possible d'utiliser des algorithmes issus de la recherche en placement automatique de graphes. Dans ce domaine, de nombreux algorithmes ont été développés. Cependant, ces algorithmes s'appliquent le plus souvent à des graphes ayant des caractéristiques particulières. De plus, ces algorithmes sont souvent coûteux et non-incrémentaux, ce qui les rend difficiles à utiliser dans un contexte interactif.

Multi-arbres Les structures hiérarchiques peuvent être étendues facilement pour donner une structure plus riche : la structure de multi-arbre, introduite par ([42]). Un multi-arbre est obtenu à partir d'un arbre en ajoutant un nœud supplémentaire comme “deuxième père" d'un nœud existant. Considérons par exemple l'arborescence formés par des pages Web classées dans un répertoire tel que celui de Yahoo. Cette hiérarchie a le mérite d'exister mais bien souvent, il serait intéressant d'ajouter des catégories transversales, sans toucher au reste de l'arbre. Dans ce contexte, cette catégorie transversale peut être introduite en ajoutant un nœud qui prend place dans la structure mais qui la transforme par sa présence en un graphe particulier appelé multi-arbre. Le multi-arbre possède deux propriétés intéressantes pour la visualisation : (1) l'ensemble des descendants d'un nœud quelconque du multi-arbre est un arbre appelé *arbre des contenus* et (2) l'ensemble des ascendants d'un nœud forme également un arbre appelé *arbre des contextes.* Les techniques de visualisation utilisées pour les multi-arbres peuvent donc

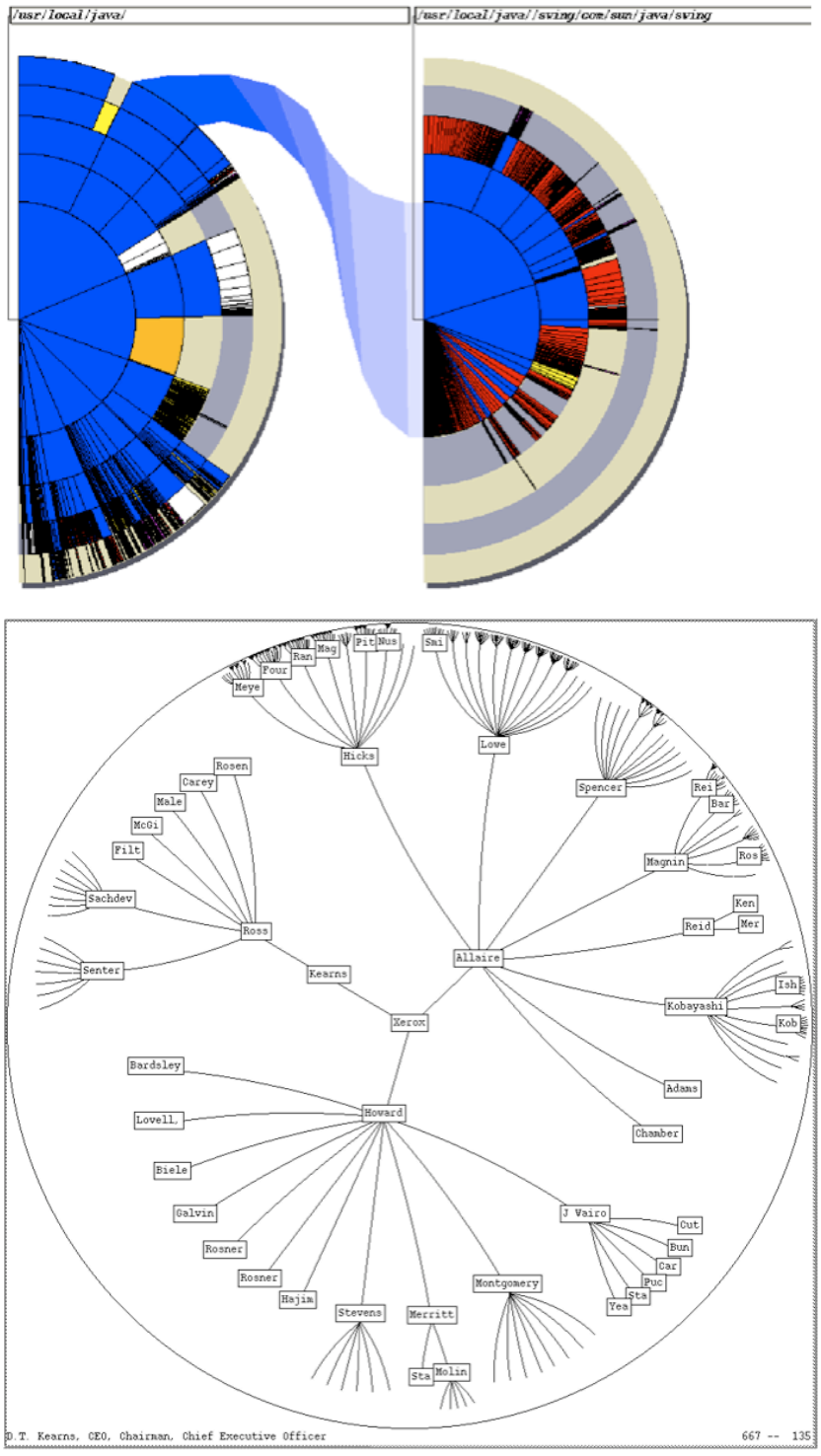

Fig. 3: *Disques d'arbre (en haut) et arbres hyperboliques (en bas)*

s'appuyer sur des techniques classiques de représentation d'arbre, c'est le cas notamment de la présentation centripète du multi-arbre qui consiste à afficher les nœuds d'un niveau de l'arbre au centre d'une fenêtre, et à utiliser un algorithme de placement d'arbre classique pour afficher d'un côté les arbres des contenus, de l'autre les arbres des contenants.

Graphes quelconques Pour visualiser des graphes quelconques, les algorithmes qui existent comme par exemple celui de ([78]) partagent le plus souvent un principe commun : le graphe à visualiser est représenté par un système de forces qui agissent sur les nœuds et les liens et l'algorithme permet de trouver un état d'énergie minimum du système (ou état d'équilibre) qui détermine les positions des nœuds.
Ces algorithmes sont difficilement utilisables pour visualiser des graphes de grande taille car ils sont particulièrement coûteux en temps de calcul, leur complexité étant en général au mieux polynomiale mais le plus souvent exponentielle (par rapport au nombre de nœuds). D'autres algorithmes existent et un état de l'art sur la visualisation de graphe dans le contexte de la visualisation d'information pourra être trouvé dans ([8; 65]).

Approche mixte : surfacique et diagramme Les arbres de silhouettes [Interact'2005] sont des structures qui permettent de modéliser les résultats d'un clustering à partir d'un focus. Ces arbres seront abordés plus en détail dans la section sur la modélisation. Nous avons proposé une visualisation des arbres de silhouettes qui s'appuie à la fois sur une approche surfacique et un algorithme de dessin radial.

Le principe du dessin consiste à associer à chaque nœud un secteur angulaire englobant tous ses descendants dans l'arbre couvrant. Jérôme Thièvre a proposé et implémenté une adaptation de l'algorithme de placement de Prefuse (Heer, Card et al. 2005), incluant la représentation automatique de clusters et de silhouettes (cf Figure 4).

Les nœuds sont initialement ordonnés par couche à partir d'un focus en tenant compte du niveau d'emboîtement des clusters (dans divers arbres de clusters T^p). Un arbre couvrant est ensuite extrait du graphe en utilisant l'ordre des nœuds et les relations entre clusters. Le choix adéquat de l'arbre couvrant conditionne la cohésion et le non recouvrement des clusters et silhouettes.
Une fois les nœuds placés, chaque cluster (ou silhouette) est dessiné comme forme englobante de ses nœuds, à l'exception des clusters (ou silhouettes) triviaux, pour ne pas surcharger la vue. A chaque silhouette est associée une couleur. Ainsi, les clusters ont la couleur de leur silhouette.

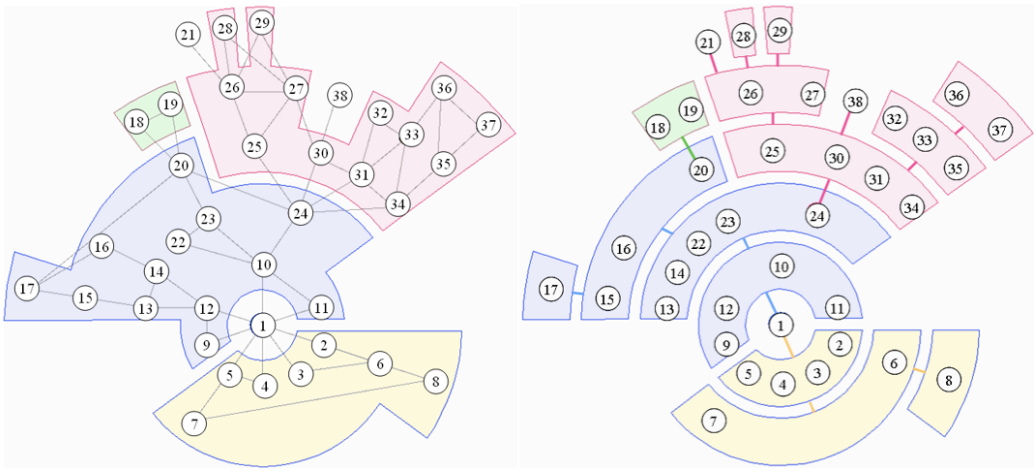

Fig. 4: *Arbre de silhouettes et arbre de clusters pour un graphe simple*

Nous représentons (Figure 4) l'arbre de clusters et l'arbre de silhouettes obtenus à partir du focus 1 dans le graphe école. Le passage de l'arbre de clusters à l'arbre de silhouettes se fait par simple « continuité » géométrique.

Par construction de l'arbre de silhouettes, il n'est pas indispensable de visualiser les arêtes du graphe pour comprendre sa structure. C'est un atout pour visualiser de gros graphes clusterisés.

L'API Grapho réalisée par Jérôme Thièvre dans le cadre de sa thèse permet la visualisation de silhouettes et clusters emboîtés (cf Figure 5). L'utilisation de couleurs transparentes permet de percevoir des niveaux d'emboîtement dans l'arbre d'inclusion de clusters ou silhouettes.

1.4 Visualisation pour le modèle vectoriel

Les modèles conventionnels de document issus de la recherche d'information consistent à représenter des documents par des vecteurs (sortes d'index). Les coordonnées de ces vecteurs dépendent du modèle utilisé ([100]). Généralement, les coordonnées sont calculées par rapport à une base de mots-clés et reflètent la présence ou l'absence de ces mots-clés dans un document.

Des procédés de calcul issus de ce modèle permettent de définir des mesures de similarités ou de pertinence entre différents “morceaux" d'information (documents, requêtes, thème, etc.). L'utilisation de ces structures de représentation a conduit au développement de techniques de visualisation spécifiques exploitant les caractéristiques de ces structures, en particulier les calculs de similarité.

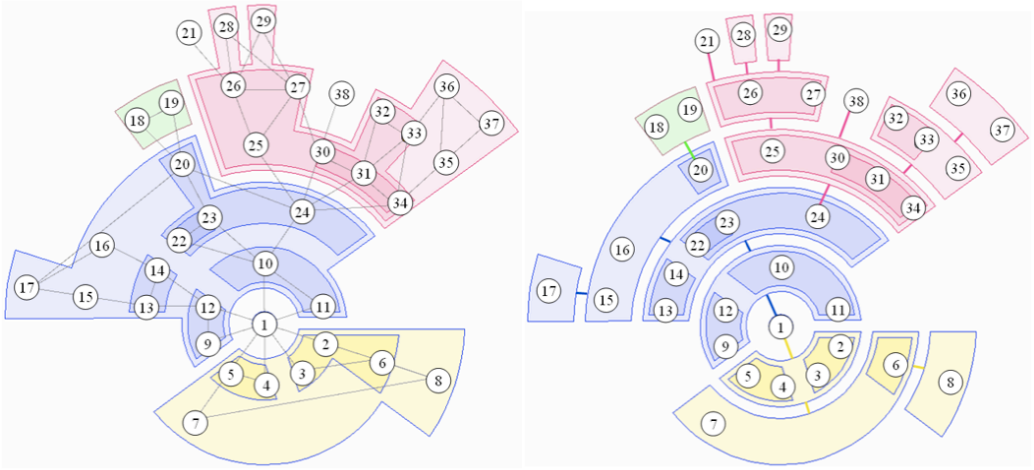

Fig. 5: *Arbre de silhouettes et arbre de clusters emboîtés du même graphe*

Métaphore du paysage Parmi ces techniques de visualisation, on trouve celles développées avec la métaphore du paysage ([25; 26]), où des documents sont placés dans un paysage virtuel en trois dimensions. Dans ce paysage, les documents similaires – au sens de la mesure de similarité – sont placés de telle sorte qu'ils soient voisins. Les algorithmes qui permettent ce type d'approche sont décrits dans ([25; 26]).

Utilisation d'un espace distance D'autres techniques de visualisation pour ce type de structures ont été développées dans le système *VIBE* ([93]). Leur principe consiste à utiliser les points d'un espace 2D pour représenter les informations (par exemple des documents), ces points sont placés dans l'espace en fonction de points de référence (représentant en général des mots-clés). Le placement de ces points est calculé de telle sorte que la distance euclidienne des points-documents aux points-références illustre la pertinence des documents par rapport aux mots-clés. Ce principe de placement a été repris en 3D dans le système *Lyberworld* développé au GMD ([64]).

Cartes interactives dynamiques Une *carte interactive dynamique* [IJHCS'95] est une vue d'ensemble sur des grandes collections de documents ou de grosses quantités d'informations. L'objectif est double : (1) cette vue d'ensemble doit servir d'interface homme machine permettant l'accès et la manipulation des informations sous-jacentes, (2) elle doit également servir de mémoire et être considérée comme un document. En ce sens l'interface peut être sauvegardée

en l'état, annotée, etc.

Il va de soit que ces cartes sont d'autant plus intéressantes qu'elles peuvent être générées automatiquement à partir de collections d'informations brutes. La mise en œuvre et la construction automatique des cartes pour des collections de documents variés a nécessité la conception et le développement d'algorithmes dans des domaines divers : analyse plein texte et filtrage, classification automatique et construction automatique de thesauri, codage graphique. Il faut souligner que l'acception de codage que nous retenons ici est celle de Bertin. Le codage est l'opération par laquelle une représentation graphique peut être construite à partir de données brutes pour en donner une vue de synthèse. Cette représentation graphique est construite de telle sorte que les attributs graphiques utilisés dans la représentation (forme, couleur, placement, texture, etc) représentent au mieux les attributs des données (par exemple, âge, genre, type, etc). Plusieurs types de cartes ont été développées pour des contextes divers. Les cartes de thesaurus par exemple sont des cartes obtenues par analyse plein-texte d'une collection de documents. Cette analyse permet l'extraction automatique de terminologie et les termes extraits sont ensuite soumis à une classification automatique. Il en résulte un thesaurus primaire qui est visualisé sous forme de carte. Les cartes de documents constituent un autre type de cartes dans lesquelles des documents sont représentés de telle sorte que les documents proches au sens de la mesure de similarité soient voisins sur la carte. Les algorithmes que nous avons développés dans ce domaine ont plus particulièrement porté sur la construction de pavages représentatifs de partitions, ainsi que sur le calcul de placement, de tailles et de couleurs en correspondance avec les caractéristiques des données.

L'enrichissement du concept de carte s'est fait par la suite en abordant des usages différents ou des collections de documents différentes. Ainsi, des travaux en collaboration avec la direction des études et recherches d'EDF sur des collections de documents fortement structurées m'ont permis d'ajouter au concept de cartes la dimension de perspective et par là même la prise en compte de perspectives multiples [62]. Dans le cadre de l'exploration de collections de pages Web le concept de carte s'est trouvé enrichi d'outils d'interaction supplémentaires [60]. Les travaux réalisés dans le cadre de cet axe ont demandé et demandent encore des développements importants qui ont permis de tester et d'enrichir le modèle d'interaction et les algorithmes de visualisation proposés jusqu'à maintenant. Nous avons effectué une grande partie de ces développements et encadré de nombreux stagiaires également.

2 Interaction

C'est l'interaction, qui rend possible l'exploitation réelle des vues d'ensembles une fois produite. En effet, l'être humain est particulièrement habile à extraire des informations d'un environnement qu'il contrôle directement et activement par rapport à un environnement qu'il ne peut qu'observer de manière passive. Selon l'approche écologique de la perception due au psychologue J.J. Gibson ([45]), la perception est indissociable de l'action : il faut agir pour percevoir et il faut percevoir pour agir. On parle de couplage (ou boucle) action-perception. De plus, la perception de notre environnement consiste à extraire des flux perçus (comme le flux visuel) des *invariants*. Par exemple, lorsque l'on se déplace, la direction du déplacement est donnée par le seul point immobile dans le flux visuel. Grâce à l'interaction sur les données, l'utilisateur peut agir sur ce qu'il perçoit et, par l'extraction d'invariants, mieux comprendre la nature des données ou de leur processus de représentation.
La visualisation interactive requiert des temps de réponse interactifs (de l'ordre du 1/10ème de seconde) afin d'exploiter le couplage action/perception. Dès que l'on traite des données de grande taille, cette contrainte exige l'utilisation d'algorithmes particulièrement efficaces. Pour les besoins de l'interaction, il est parfois préférable de dégrader la qualité de l'affichage au profit de la vitesse de réaction, quitte à raffiner l'affichage lorsque l'interaction s'arrête ou marque une pause. Ainsi, au couplage action-perception de l'utilisateur correspond un fort couplage affichage-interaction du système de visualisation.
Un certain nombre de techniques classiques d'interaction peuvent être utilisées pour la visualisation d'information, comme par exemple les barres de défilement pour naviguer le contenu d'une fenêtre, ou les boîtes de dialogues pour spécifier des paramètres de visualisation. Cependant, les meilleurs résultats sont obtenus en couplant plus fortement l'interaction avec la technique de visualisation. Trois catégories de techniques d'interaction peuvent être mises en œuvre avec un grand nombre de techniques de visualisation : (1) le filtrage dynamique, (2) les transformations sémantico-géométriques et (3) les zooms spécifiques.

2.1 Filtrage dynamique

Le filtrage dynamique est particulièrement utile pour interagir avec des vues globales, car l'information présente dans ces vues n'est pas définitivement pertinente pour un utilisateur. En effet, s'il est intéressant pour un utilisateur de voir dans un premier temps l'ensemble des informations qu'il explore pour mieux l'appréhender, il sera intéressant par la suite que l'utili-

sateur puisse recentrer son exploration, en réduisant son espace de recherche.

Les requêtes dynamiques ([1; 2; 3; 113]) développées à l'Université de Maryland puis à Ivee (système SpotFire) constituent un bon exemple de mode d'interaction permettant ce type de filtrage dynamique. Leur principe repose sur un fort couplage entre d'une part des éléments d'interaction (curseurs, menus de sélection), permettant de spécifier graphiquement des requêtes et d'autre part la vue globale sur les données qui se remet à jour immédiatement en fonction de la requête spécifiée. Ainsi, l'activation d'un curseur sur le panneau de contrôle permet de voir instantanément l'effet du changement de valeur correspondant sur la vue globale. Des données deviennent dynamiquement visibles ou invisibles sous l'action de l'utilisateur. Cette technique permet à l'utilisateur de réaliser le filtrage en fonction du résultat qu'il obtient et non pas en fonction du contrôle qu'il opère. Par exemple, il peut aisément détecter des situations où un faible changement d'une valeur de paramètre produit un effet important sur les données filtrées, et explorer plus en détail ces zones. Le succès de ce type de technique est fortement dépendant des temps de réponse du système.

Un autre type de filtrage dynamique a été développé par ([41]). Il s'agit d'une technique appelée *fisheye*. Cette technique a été appliquée à des visualisations d'arbres et permet de faire apparaître ou disparaître des nœuds de l'arbre en fonction de leur degré d'intérêt (Degree Of Interest ou DOI). Le DOI dépend (1) de l'importance a priori des nœuds (IAP) et (2) de leur distance au nœud pointé par l'utilisateur ou focus. L'importance a priori des nœuds est donnée, elle dépend des informations visualisées, tandis que la distance entre un nœud et le focus est calculée dynamiquement comme le plus court chemin entre le nœud considéré et le nœud focus.

Il est important de reconnaître que les techniques de fisheye existent dans deux variantes très différentes du point de vue de leur utilisation. En effet, la technique de fisheye "filtrant" telle qu'elle a été proposée par Furnas et décrite au paragraphe précédent, diffère des techniques de fisheye "déformants" proposés par exemple par ([101]) et décrites plus loin. Dans le premier cas, il s'agit bien de filtrage dynamique puisque des informations vont devenir visibles ou invisibles sous l'action de l'utilisateur, dans le deuxième cas il s'agit d'un mode d'interaction permettant de concilier détails et contexte puisque aucune information ne disparaît ou n'apparaît, les informations affichées sont seulement déformées. Les deux approches partagent néanmoins un concept commun : le calcul du degré d'intérêt (DOI) qui tient compte du centre d'intérêt courant de l'utilisateur pour effectuer le filtrage dans un cas et les déformations dans l'autre.

Enfin, notre approche du filtrage dynamique ([57]) a consisté à coupler une carte de documents avec une carte de thesaurus (cf. cartes interactives

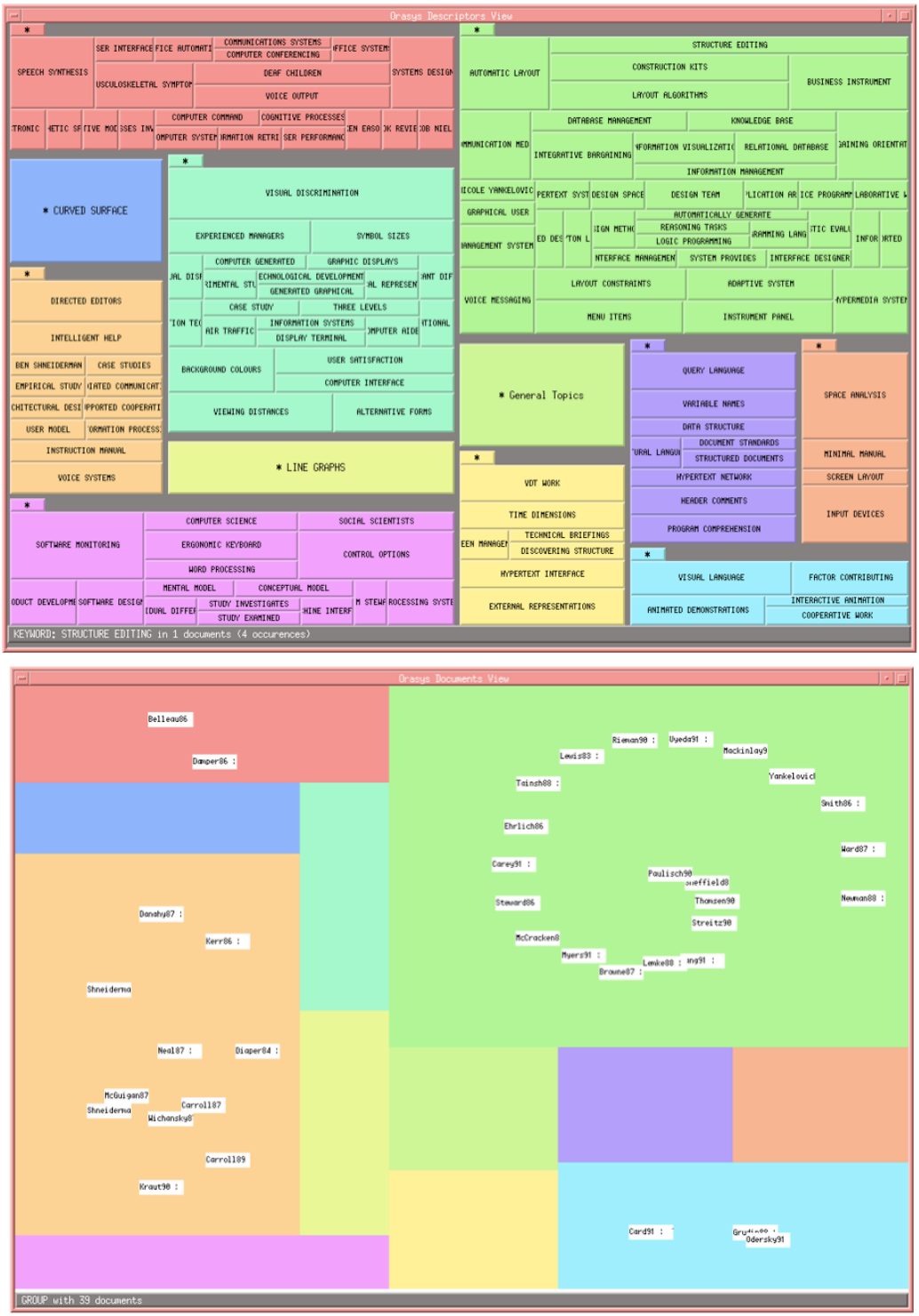

Fig. 6: *Filtrage dynamique de la vue de document(en bas) en manipulant la vue du thesaurus associé (en haut).*

dynamiques) et à interagir avec la carte de thesaurus pour filtrer les documents de la carte de documents (cf. Figure 6) : la sélection de termes et de relations dans la carte de thesaurus spécifie une requête dont le résultat s'affiche dans la carte de documents. Plus récemment ces travaux se poursuivent aujourd'hui mais avec des critères de filtrage différents, les résultats ne sont pas encore publiés.

2.2 Déformations : Intégration de différents niveaux de détails

Le filtrage déformant a pour effet de rendre visible ou invisible des données. Souvent, ces changements brusques de visibilité rendent l'exploration difficile. La notion de niveau de détail permet une approche plus graduée, en permettant d'amplifier dynamiquement les détails autour d'un centre d'intérêt tout en gardant le contexte global de la vue. Dans ce domaine, de nombreuses solutions ont été proposées. Nous présenterons ici les fisheye déformant de graphe ([101]) et fisheye déformant structurels ([7; 92; 92; 102]). De nombreuses autres techniques visent le même objectif : *Polyfocal display*, *Bifocal display* ou même *Mur Fuyant* ([98]) présenté plus haut. Pour une revue plus complète de ces techniques de déformation, le lecteur peut se référer à ([86]). Une approche unificatrice de ces techniques, appelée Scrolling Généralisé, est présentée dans ([50]).

La transformation fisheye proposée par Sarkar et Brown opère sur un graphe valué. Les poids des nœuds constituent leur importance a priori IAP. La transformation fisheye consiste à modifier les positions et les tailles des nœuds affichés à l'écran (cf. Figure 8). Les nouvelles tailles sont obtenues en fonction du degré d'intérêt DOI calculé pour chaque nœud. Les nouvelles positions sont obtenues en appliquant une fonction de déformation : si O est le nœud focus, et M est un nœud à transformer, le calcul de k tel que OM = k OS (S est l'intersection de (OM) avec les frontières de la zone d'affichage) permet de calculer l'image de M. Cette image M' est telle que OM' = f(k) OS, avec f : [0,1]->[0,1] et f(k)>k, et f' décroissante sur [0,1]. Ainsi les nœuds les plus proches du focus subissent un déplacement plus important que les nœuds plus éloignés.

Fisheye déformant Il existe plusieurs variantes de cette formulation de la transformation (voir par exemple ([101]). La Figure 8 montre l'effet de deux formulations différentes : l'une globale calculée en coordonnées cartésiennes et l'autre locale à une loupe calculée en coordonnées polaires. Le résultat de cette double déformation (modification des tailles et modification des posi-

tions) a ainsi pour effet d'amplifier les nœuds se trouvant dans la région du focus tout en repoussant et en diminuant l'importance des nœuds situés en dehors de cette région (cf. Figure 7). Comme avec l'utilisation de la géométrie hyperbolique, l'effet de déformation peut-être perturbant pour l'utilisateur, surtout si la distance et/ou la disposition des données a une signification particulière (comme pour des données géographiques). Une alternative consiste à utiliser une vue séparée pour représenter les détails, comme avec une loupe mais alors la vue détaillée masque le contexte. Le *DragMag* ([111]) évite cet effet de masquage en décalant la vue magnifiée. Dans tous les cas, c'est la qualité et la rapidité du couplage entre l'action sur la vue et son effet sur la représentation qui sont la clé de ces techniques.

Fisheye déformant et structurel La technique de transformation fisheye utilisée par Dill et testée par ([102]) est repose sur la transformation fisheye d'un graphe simple telle que celle proposée par Sarkar et Brown, mais elle va plus loin car elle profite de l'espace gagné sur certain nœuds pour les "développer". En effet, le fisheye déformant et structurel s'applique récursivement à un graphe organisé en nœud-nuées. Les nœuds-nuées peuvent être vus comme des nœuds composés qui représentent un ensemble de nœuds sur lesquels les déformations doivent être répercutées. Ainsi, le fisheye ([92]) s'applique-t-il récursivement pour que les nœuds-nuées grossis lors de la transformation répercutent ce grossissement sur les nœuds qui les composent. Les détails des calculs utilisés pour ces déformations ainsi que des expérimentations relatives à ces techniques peuvent être trouvées dans [102] et ([92].

2.3 Zoom infini et zoom sémantique

Une autre façon de concilier détails et vue globale est de permettre à l'utilisateur de zoomer facilement : un zoom avant révèle les détails tandis qu'un zoom arrière révèle le contexte. Le zoom est une transformation géométrique à part entière et peut être plus ou moins complexe. Le zoom bitmap simple, qui consiste seulement en un changement d'échelle est d'un intérêt limité quand il s'agit d'interagir avec un système de visualisation d'information car il n'est pas de nature à révéler suffisamment d'information. Ce qu'il faut au contraire, c'est un procédé qui permette une transformation géométrique (grossissement et recentrage) associée à une sémantique (chaque niveau de zoom correspond à un niveau différent de détails affichés) ([7; 12]). C'est l'approche choisie dans les interfaces zoomables avec le projet *Pad* ([94]) et ses successeurs *Pad++* ([12]) et plus récemment *Jazz* ([10]) et Piccolo ([11]). *Pad* offre une surface infinie sur laquelle sont présentées des informations et dans laquelle on peut zoomer infiniment. Chaque objet présent sur la surface

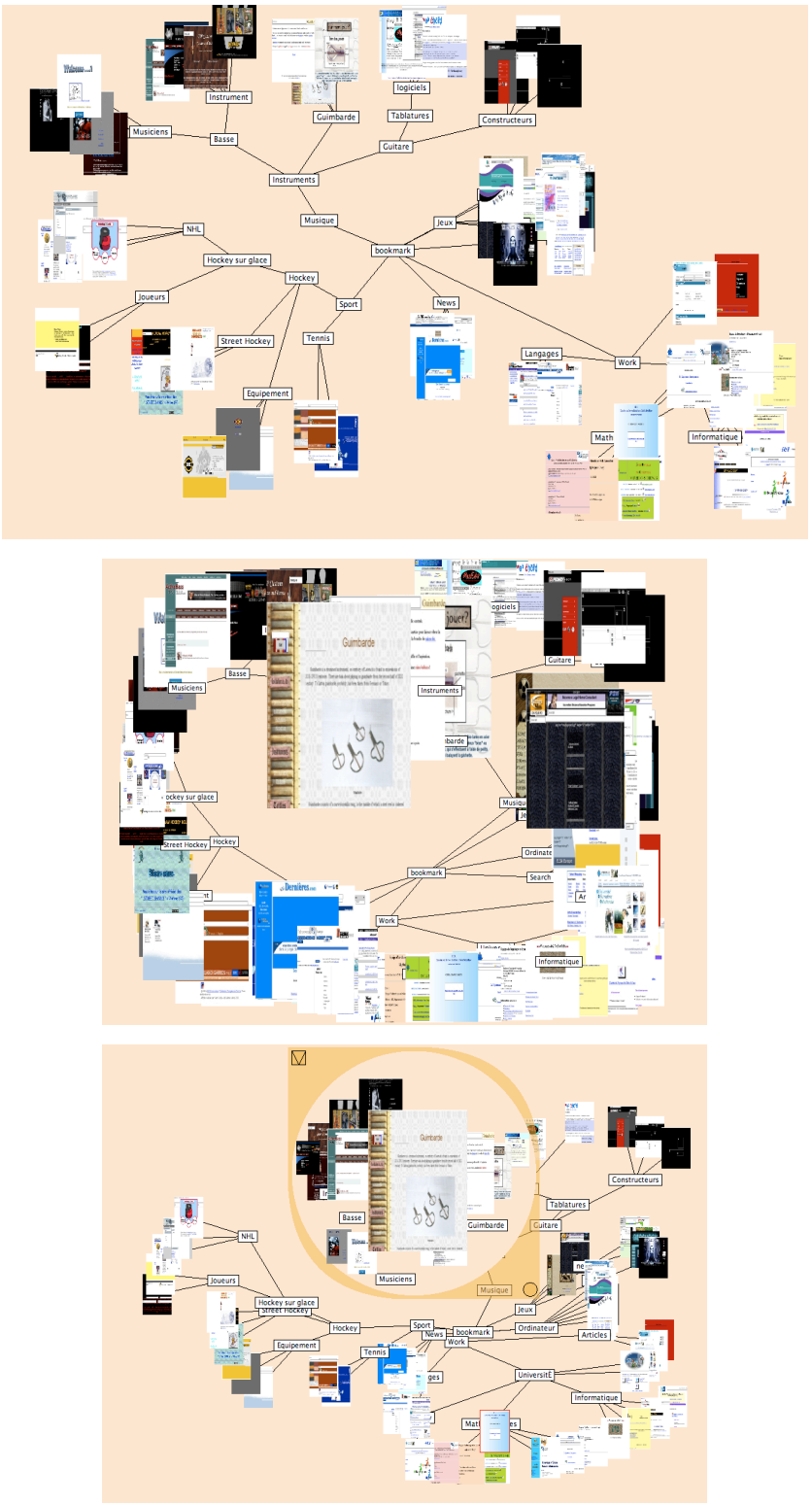

Fig. 7: *Vue globale (haut), déformation par fisheye global en coordonnées cartésiennes (centre) et déformation par fisheye local en coordonnées polaires (en bas).*

a une position donnée et occupe un espace donné à un niveau de zoom donné. Pour naviguer sur cette surface l'utilisateur doit faire des zooms avant ou arrière et des déplacements XY appelés pans (pour panoramique). Les objets soumis à un zoom peuvent générer des présentations différentes en fonction du niveau de zoom courant. Par exemple, un calendrier peut être représenté par le mot "Calendrier " à un faible niveau de zoom ; à mesure que l'on zoome dessus apparaissent les années, les mois puis les jours. De nombreuses applications à ce type d'interaction sont envisageables. L'utilisation de *Pad* pour la visualisation d'historiques de navigation sur le WWW en est un exemple ([66]). Ce travail a montré que la navigation multi-échelle dans un historique de navigation apportait des gains de performances significatifs en terme de temps passé pour accomplir une tâche donnée, ainsi qu'en terme de nombre de pages visitées en vain, pour certaines tâches de navigation.

Cependant la navigation multi-échelle reste encore un type de navigation mal connue à plusieurs titres : temps des déplacements physiques, mise en place de stratégies de navigation typiques et facilement appréhendées par tous, orientation, etc. Les diagrammes espace-échelle de [40] permettent de mieux comprendre la navigation dans une interface zoomable. Ils affectent à l'échelle une dimension à part entière. Dans cet espace 3D (deux dimensions d'espace et une d'échelle), le plus court chemin entre deux points n'est plus la ligne droite, mais une trajectoire en V constituée d'un zoom arrière jusqu'à ce que la cible du déplacement soit visible suivi d'une succession de zooms avant et de pans afin de se rapprocher de la cible. Pourtant les utilisateurs n'ont pas de difficulté à mettre en œuvre ces trajectoires. En fait, un résultat remarquable ([47]) est que le temps de pointage dans une interface zoomable suit exactement la même loi de Fitts que pour le pointage normal sur écran. La navigation dans un espace zoomable est donc très efficace.

La quantité d'information que l'on peut mettre dans une interface zoomable est gigantesque. Aussi dans la pratique une très petite partie de l'espace est utilisée. En conséquence, il est facile de se perdre, ou au moins de se trouver dans une partie d'espace où il n'y a rien, le "desert fog" ([77]). Aussi est-il nécessaire d'organiser l'espace de façon à limiter ces situations de désorientation.

2.4 Conclusion

L'avènement des postes de travail graphiques et l'augmentation de la puissance de calcul ont permis d'explorer et de développer des techniques interactives exploitant au mieux les capacités sensori-motrices et cognitives de l'être humain. Ainsi, la visualisation d'information cherche à présenter les données d'une façon telle qu'un utilisateur puisse en extraire un sens que la

machine n'aurait pu découvrir. L'interaction en temps réel permet de faciliter la navigation dans l'espace d'information et ainsi de mieux comprendre la structure et les caractéristiques de cet espace. Cette meilleure connaissance permet ensuite de mieux aborder les données sous-jacentes.
Ainsi, visualisation et interaction s'affirment comme des domaines complémentaires dont la combinaison offre plus que la somme des parties. Les travaux dans ce domaine, présentés ici, sont amenés à se développer notamment pour mieux explorer cette complémentarité.

Chapitre 3

MODÉLISATION ET CONSTRUCTION

Dans le chapitre précédent, nous avons souligné l'intérêt de la construction automatique de vues d'ensemble suivie de la mise en place de techniques d'interaction de type zoom ou déformation. Ces techniques de zoom et de déformation sont matures, néanmoins toutes se heurtent au problème de la génération automatique des niveaux d'échelle. En effet, en pratique, les informations à visualiser sont assez rarement structurées sous forme multi-échelle. Aussi, pour que des techniques telles que les déformations ou le zoom infini puissent être exploitées plus largement le travail doit donc être pris en amont et en particulier, il est indispensable que des modèles multi-échelle soient mis en oeuvre et que les procédures automatiques de construction soient à la fois efficaces et relativement génériques.

1 Contexte

Il existe maintes façons d'aborder le problème de modélisation multi-échelle des données. Les fractales ou les L-systems constituent par exemple deux approches matures. Néanmoins ces modèles sont le plus souvent adaptés à certains domaines précis : les L-systems ont été conçus et développés pour la représentation des plantes. Les fractales seront quant à elles utilisées dans des cas très particuliers de représentations d'images ou d'objets mathématiques. A l'inverse, les modèles de graphes clusterisés constituent de notre point de vue un modèle suffisamment ouvert pour permettre une représentation des informations qui nous intéressent ici : à savoir des informations de type qualitatif principalement exprimées sous forme de relations entre items. Ainsi, pour construire automatiquement des vues d'ensemble qui puissent ensuite se prêter à une exploration par zoom ou par déformation de type fisheye par exemple, les graphes clusterisés constituent un modèle privilégié.

La mise au point des modèles multi-échelle va de pair avec la mise au

point de procédures de construction de ces modèles à partir des modèles de base courants. Ces procédures sont à rapprocher des travaux effectués dans le cadre du clustering. Le clustering ou classification non supervisée est une procédure qui consiste à analyser une collection d'éléments et à regrouper des éléments de sorte à ce que la classification qui en résulte soit "pertinente". Ce problème a été étudié très largement par le passé [73] dans des contextes couvrant des disciplines scientifiques très différentes en allant de la physique jusqu'à la sociologie en passant par les mathématiques et l'informatique. Il continue néanmoins d'occuper de nombreux chercheurs aujourd'hui. En effet, alors que le domaine peut s'appuyer sur de nombreux résultats satisfaisants, il reste néanmoins des problèmes difficiles et ouverts pour lesquels les progrès sont lents.

Le but de cette section n'est pas d'être exhaustif, un état de l'art très complet et synthétique se trouve dans la thèse de F. Boutin. Au contraire, nous nous sommes attachés à présenter quelques uns des résultats les plus significatifs.

2 Résultats obtenus

Nous avons développé une famille d'algorithmes qui permettent à partir d'un graphe quelconque d'obtenir un modèle multi-échelle de ce graphe. Dans un premier temps, nous avons étendu des méthodes existantes d'optimisation de partition de sorte à obtenir des K-partitions des graphes initiaux. Par la suite et en particulier au travers des travaux de thèse de François Boutin, nous avons introduit les arbres de clusters emboîtés ainsi que des arbres de silhouettes emboîtés qui ont été définis pour permettre les représentations multi-échelles résultantes.

2.1 K-Partitions

Pour construire des structures multi-échelles utiles à nos représentations visuelles, nous avons commencé par utiliser des techniques classiques d'optimisation de partition. En effet, tout graphe peut être considéré comme un ensemble de sommets qui peut être partitionné en utilisant ces techniques. L'avantage de cette approche est d'être particulièrement peu contraignante. Elle peut s'appliquer à un graphe quelconque sitôt que l'on construit une mesure de similarité sur l'ensemble des sommets du graphe. Plusieurs variantes de ces algorithmes existent. Celles que nous avons utilisées passent par le calcul du noyau des parties, un noyau étant lui-même un sous-ensemble représentant de la partie. La figure 1 décrit succinctement cette version de

(1) Donner une partition initiale
$Pp = \{A1, A2, ...Ap\}$
(2) Calculer les noyaux G_i^n de chaque partie A_i^n
(3) Pour chaque i calculer
$A_i^{n+1} = \{x \in A_i^n | \forall j \neq i,\ s(x, G_i^n) < s(x, G_j^n\}$
(4) Retourner en (2) tant que il existe i tel que $A_i^n \neq A_i^{n+1}$

Fig. 1: *L'algorithme des nuées dynamiques*

l'algorithme.

Nous avons raffiné les techniques classiques afin d'obtenir des résultats qui permettent un plus grand contrôle sur la qualité des classes ou clusters obtenus. Ainsi, les résultats obtenus sont-ils plus adaptés aux représentations multi-échelles visées. Ces travaux ont été principalement effectués dans le contexte de la construction automatique de thesaurus. Dans ce contexte, le thesaurus peut être vu comme un graphe pour lequel les sommets sont les descripteurs et les liens correspondent aux relations. La construction automatique de thesaurus constitue ainsi un excellent terrain d'expérience car l'évaluation de la pertinence des résultats si controversée sous ses formes analytiques peut au moins être envisagée de manière expérimentale.

2.2 Définition et prise en compte de la granularité

Notre approche consiste à introduire la notion de grain pour contrôler la qualité des clusters obtenus. La notion de grain correspond à la moyenne des similarités entre éléments d'une même partie après traitement et est définie par :

Définition 2.2.1 *Grain*
Soit une partition $P = A_1, A_2, ...A_p$, le grain d'une partie A_j de P notée $Grain_j$ est défini par

$$Grain_j = \frac{2}{n(n-1)} \sum_{h=1,k>h}^{n} s(T_h, T_k)$$

où n est le nombre d'éléments dans la partie A_j et T_h et T_k sont des éléments de A_j et s une mesure de similarité donnée.

Pour que le thesaurus présente un intérêt maximal, il est nécessaire que la moyenne des grains des parties de la partition soit suffisamment élevée (c'est à dire que les éléments regroupés dans une même classe soient suffisamment

proches, du moins en moyenne). Pour ce faire nous introduisons la notion de grain minimal.

Grain minimal, grain total et grain moyen La mise en œuvre de stratégies visant à assurer que la moyenne des grains des parties dans les partitions obtenues soit suffisamment élevée repose sur la notion de grain minimal que nous définissons ici. Les stratégies mises en œuvre lorsqu'une partie a un grain inférieur au grain minimal choisi sont développées dans la section suivante.

Définition 2.2.2 *Grain total, Grain minimal et Grain moyen*
Tout d'abord, pour une partition P donnée de p éléments nous appelons Grain Total, noté $GrainTotal$, le grain de la partie faite à partir de tous les éléments c'est-à-dire :

$$GrainTotal = \frac{2}{p(p-1)} \sum_{i=1,j>i}^{p} s(T_i, T_j)$$

Le grain minimal noté $GrainMin$ est alors donné par :

$$GrainMin = a \times GrainTotal$$

où $a \in \mathbf{R}$.

Enfin, le grain moyen d'une partition de taille p ou d'un ensemble de p parties est la moyenne des grains des parties.

$$GrainMoyen = \frac{1}{p} \sum_{j=1}^{p} Grain_j$$

Le grain minimal peut donc être vu comme un seuil qui doit être fixé à l'avance. A la fin de la phase de classification, on comparera le grain de chacune des parties obtenues à ce seuil. Ainsi, lorsqu'au moins une partie a un grain inférieur au grain minimal, la partition sera reconsidérée.

Déterminer le grain minimal dans l'absolu n'est pas aisé car d'une mesure de similarité à l'autre, les similarités entre les éléments varient en amplitude. Par ailleurs, le grain minimal peut également varier en fonction des collections de documents et du vocabulaire d'indexation considérés. En effet, dans certains cas les collections sont telles que le seuil de similarité moyen entre les différents descripteurs est très faible auquel cas il faudra se contenter d'un seuil assez faible. Dans d'autres cas au contraire les similarités entre les descripteurs sont beaucoup plus importantes, par conséquent, le grain minimal choisi pourra être plus élevé.

C'est pourquoi nous avons choisi de définir le grain minimal relativement au grain total. Le coefficient a permettant de s'affranchir des variations en amplitude dues aux différentes mesures de similarité ou aux collections étudiées. Ainsi, plutôt que d'agir sur le grain minimal nous allons agir sur a qui peut être donné indépendemment de ces variations d'amplitude.

Respect du seuil minimal de similarité Pour un coefficient a donné, le grain minimal peut donc facilement être calculé et les grains parties résultant de la classification peuvent être comparés au grain minimal. Nous avons développé trois stratégies possibles dans le cas où le grain d'au moins une de ces parties se trouvait en dessous du grain minimal.

La première stratégie consiste à modifier la granularité du partitionnement (c'est-à-dire le nombre de parties de la partition). En effet, si pour un grand nombre d'éléments la similarité entre l'élément et la partie où il se trouve est en-dessous du grain minimal de la partie, c'est parce qu'il y a une grande dispersion entre les éléments à classer. Il y a donc lieu d'affiner le partitionnement en augmentant le nombre de classes créées. Nous avons donc modifié l'algorithme initial pour permettre de changer dynamiquement le nombre de parties dans certains cas de manière à augmenter le grain moyen des parties.

Notre deuxième solution consiste à créer une classe artibtraire nommée "divers" et à y mettre tous les éléments pour lesquels la similarité entre l'élément et la partie qu'il contient se situe en-dessous du grain minimal de la partie. Nous avons donc modifié l'algorithme des nuées dynamiques pour prendre en compte cette extension.

Les deux solutions que nous proposons ci-dessus présentent des avantages complémentaires. En effet, alors que la première solution agit sur le nombre de parties pour augmenter le grain moyen, la deuxième agit sur les éléments des parties, en enlevant les plus lointains. Dans la pratique, il se trouve que des parties disparaissent avec la deuxième stratégie, ce qui compense souvent l'augmentation de granularité liée à la première stratégie. Par exemple, dans les expériences que nous avons conduites, à partir d'un partitionnement en 15, la première stratégie nous avait conduit à augmenter la granularité jusqu'à 60. Le recours à la deuxième stratégie, à savoir la création de la classe "Divers", a eu pour effet de faire redescendre le nombre de parties à une quinzaine, avec un meilleur grain. Il en résulte le plus souvent une partie "sacrifiée" (la partie "Divers") dont le grain est nécessairement très faible. Par contre le grain des autres parties devient beaucoup plus élevé ce qui nous paraît plus satisfaisant.

La troisième stratégie consiste donc à combiner ces deux solutions pour

en tirer le meilleur parti. Nous commençons par augmenter le nombre de parties jusqu'à ce que le pourcentage de parties ayant un grain inférieur au grain minimal devienne suffisamment faible. Lorsque c'est le cas, la classe "divers" est créée et la deuxième stratégie est appliquée.

Nous avons pratiqué un certain nombre de tests dont les résultats sont résumés dans les tableaux suivants. L'objet de nos tests est de comparer le comportement des trois algorithmes en terme de grain minimal, de grain moyen et de nombre de classes finalement obtenu. Ils nous ont également permis de déterminer les valeurs de a qui ressortaient empiriquement comme des valeurs souhaitables.

Expériences et résultats Les expériences que nous présentons dans cette partie ont été réalisées sur une collection de près de 600 documents issus de la collection HCIBIB [43]. Le vocabulaire d'indexation sur lequel portent les tests a été extrait à l'aide des procédés d'extraction introduits dans [116]. Une épuration manuelle de ce vocabulaire nous a de plus permis de retirer les descripteurs non-pertinents parmi le vocabulaire extrait automatiquement.

Dans la première expérience nous avons appliqué simplement l'algorithme classique des nuées dynamiques en fixant le nombre de parties à 15. En conséquence, la moyenne des grains des parties résultantes est seulement de 0.029609.

Nous avons ensuite appliqué la première stratégie introduite précédemment qui consiste à augmenter le nombre de parties (c'est-à-dire la granularité) pour améliorer le grain.

Les résultats de cette expérience confirment le fait que l'augmentation de la granularité permet l'amélioration du grain. Ainsi, pour la granularité maximum (c'est-à-dire la partition de 120 parties) le grain moyen par parties s'élève à 0.213701 soit plus de 10 fois le grain obtenu pour seulement 15 parties.

A tous les stades de l'augmentation de la granularité, les parties qui ont les plus forts grains correspondent le plus souvent à des regroupements pertinents, c'est le cas par exemple des parties ci-dessous.

Numéro de partie	Grain	Similarité minimale	Descripteurs
17	0.275661	0.035714	HUMAN INTERACTION HUMAN COMPUTER COMPUTER INTERFACES COMPUTER INTERACTION COMPUTER INTERFACE
26	0.180831	0.000000	HYPERTEXT SYSTEMS HYPERTEXT SYSTEM HYPERTEXT NETWORK DESIGN TEAM
15	0.250000	0.000000	VISUAL DISPLAY VISUAL DISCRIMINATION VIEWING DISTANCE INFORMATION TECHNOLOGY
16	0.168763	0.000000	MANAGEMENT SYSTEM INTERFACE MANAGEMENT INFORMATION MANAGEMENT SPECIFICATION LANGUAGE
31	0.161941	0.000000	EXPERT SYSTEMS EXPERT SYSTEM KNOWLEDGE BASE KNOWLEDGE ACQUISITION

Cependant, il existe également quelques parties ayant des grains très inférieurs (voire nuls) et qui correspondent pourtant à des regroupements très pertinents. C'est le cas par exemple des deux parties suivantes :

Numero de partie	Grain	Similarité minimale	Descripteurs
6	0.088541	0.017544	INTERFACE SYSTEM INTERFACE SOFTWARE INTERFACE DESIGN
25	0.000000	0.000000	STRUCTURED DOCUMENTS AUTOMATIC TEXT NATURAL LANGUAGE

Les faiblesses constatées expérimentalement sur les jeux de tests réunis sont, de notre point de vue, à attribuer essentiellement à deux facteurs "externes" : la mesure de similarité et le vocabulaire extrait. Il est prévisible que les résultats plutôt encourageants que nous avons déjà obtenus se trouvent grandement améliorés par l'amélioration de la qualité des données en entrée. Soit en améliorant la phase d'extraction dans le cas par exemple du thésaurus soit en travaillant sur des données plus fiables. Le choix de la mesure de similarité est également crucial.

Notre approche actuelle consiste à étudier les éléments théoriques permettant d'exploiter davantage ces résultats expérimentaux préliminaires. Notre deuxième approche est beaucoup plus restrictive en terme de structure et s'appuie davantage sur les graphes. Elle est présentée succinctement dans la partie suivante.

2.3 Arbres de silhouettes emboîtées

Alors que la plupart des algorithmes de clustering fonctionne de manière non contextuel, nous avons abordé une approche qui nous permet de prendre en compte un point de vue dans la construction des clusters. Ce point de vue peut être réduit à la donnée par l'utilisateur d'un noeud du graphe. Mais

on peut aussi considérer un ensemble de noeuds pour représenter un centre d'intérêt plus large.

Nous donnons ici un résumé des résultats obtenus au travers des arbres de silhouettes emboîtés. Plus de détails se trouvent dans [21].

Soit un graphe connexe non orienté G = (V, E) où V décrit l'ensemble des sommets $\{X_0, \ldots, X_n\}$ et E l'ensemble des arêtes (X_i, X_j) et où X_0 représente le focus de l'utilisateur. Le niveau d'un nœud X_i est défini par la distance entre X_0 et X_i. Nous définissons G_n comme le sous graphe de G contenant les nœuds de niveau supérieur ou égal à n et les arêtes entre ces nœuds. La composante $C_{n,k}$ est définie comme la k^{me} composante connexe de G_n.

Rappelons en outre quelques définitions préliminaires :

Nœud d'articulation : un nœud d'articulation est un nœud qui, si supprimé, déconnecte le graphe.

Composante biconnexe : une composante biconnexe est une composante connexe qui ne peut être déconnectée en supprimant un seul nœud.

Composante triviale : une composante connexe est dite triviale si elle est formée uniquement de deux nœuds d'articulation interconnectés (ou d'un nœud d'articulation et d'une feuille).

Arbre biparti : Un graphe G = (V, E) est dit **biparti** si V est partitionné en deux ensembles V_1, V_2 tels que toute arête de E a une extrémité dans chaque ensemble.

A partir de ces définitions, nous introduisons une notion nouvelle : la notion de silhouette. Nous précisons également la notion de cluster pour éviter toute ambiguïté, en effet, le mot cluster a par ailleurs des acceptions très différentes selon les auteurs.

Cluster : Nous définissons un cluster comme le sous ensemble des nœuds de $C_{n,k}$ qui sont de niveau n. Les noeuds d'un cluster sont non seulement liés entre eux, ils sont également tous à la même distance du centre d'intérêt.

Silhouette : une silhouette est obtenue par construction à partir de l'ensemble des composantes biconnexes ou triviales extraites d'un graphe quelconque. Rappelons qu'un nœud d'articulation appartient à plusieurs composantes biconnexes ou triviales. Ainsi l'ensemble des composantes biconnexes et triviales ne constitue pas une partition du graphe mais un recouvrement du graphe.

Nous proposons une technique permettant de supprimer dans une composante biconnexe (ou triviale) les nœuds d'articulation déjà affectés à une autre composante. Les composantes résultantes sont appelées silhouettes et elles forment une partition du graphe initial.

Theorem 3.1 *Tout graphe peut être décomposé en arbre de silhouettes.*

Proposition 3.2 *Une silhouette est un arbre d'adjacence de clusters.*

Dans [Interact'2005], nous proposons une structuration multi-échelles de l'arbre de silhouettes. Pour cela il est nécessaire de définir G^p, le sous graphe de G contenant les nœuds de niveau au plus p. T'^p définit l'arbre de silhouettes associé à G^p et au focus. L'idée introduite consiste alors à « superposer » les arbres de silhouettes T'^p des sous graphes G^p.

Theorem 3.3 *Considérons les arbres de silhouettes T'^{p-1} et T'^p correspondant aux graphes :*
G^{p-1} et G^p. Toute silhouette de T'^{p-1} est contenue dans une silhouette de T'^p.

Proposition 3.4 *On peut ainsi définir une relation d'inclusion entre les arbres de silhouettes : $T'^0 \subset \ldots \subset T'^{p-1} \subset T'^p \subset \ldots \subset T'$.*

Proposition 3.5 *Chaque silhouette de T'^p est à l'origine d'un arbre d'inclusion de silhouettes.*
La structure complexe composée des divers arbres d'adjacence de silhouettes et des arbres d'inclusion est un arbre composé multi niveaux appelé arbre de silhouettes emboîtées.

Dans le chapitre précédent nous avons présenté la représentation graphique pour les arbres de silhouettes emboîtées T' et T'^2 obtenus à partir des composantes biconnexes et du focus 1. Le travail de thèse de F. Boutin a permis de concevoir et d'implémenter les algorithmes de construction automatique d'arbre de silhouettes emboîtées à partir d'un graphe quelconque.

Chapitre 4

DISPOSITIFS MULTI-SUPPORTS

La visualisation et l'interaction sont particulièrement sensibles aux dispositifs matériels utilisés pour mettre en oeuvre les systèmes développés. Dans ce domaine, les dispositifs type "écran, clavier, souris" sont, à bien des égards, inadaptés à certains types de visualisation et aux besoins croissants d'interactivité. Alors que les progrès des 30 dernières années en matière de dispositifs d'entrées/sorties ont été relativement lents, ils semblent s'accélérer ces dernières années et on peut espérer que les évolutions à court ou moyen terme soient beaucoup plus importantes.

1 Contexte

L'évolution des dispositifs d'entrées et sorties est d'autant plus intéressante qu'elle permet une évolution des modèles d'interaction. Les récents travaux sur les tables augmentées n'ont pas seulement été l'occasion d'innovations matérielles [34; 52; 53; 91], ils ont conduit à une reconception d'éléments de base des modèles d'interaction existants. Avec DiamondTouch [91] c'est toute l'interaction avec le bureau qui est repensée, avec la table multipoints de Jeff Han, c'est par exemple toute l'interaction liée au plaquage de texture 2D sur un modèle 3D qui peut être revue [46].

L'émergence de nouveaux dispositifs est très fortement liée aux avancées technologiques tant en matière de dispositifs d'affichage qu'en matière de dispositifs d'entrée. En matière d'affichage, le développement de nouvelles technologies appartient à d'autres domaines (micro-électronique, physique voire chimie). Les constructeurs tels que LG, Sony, Sharp, Hitachi, etc. sont nombreux à déployer des moyens considérables pour mûrir de nouveaux produits qui essuient bien des échecs avant de percer. Il est frappant de constater que les technologies telles que les technologies développées autour des LCD ont mis pratiquement un demi siècle à mûrir si l'on prend comme point de départ les premiers écran LCD produits [38].

Des technologies plus exotiques telles que les technologies OLED (organic

light emitting diode) ont plus de difficulté encore à percer. Cette technologie qui apparaissait comme le support idéal pour le papier électronique était annoncée sur le marché pour les années 2000 alors que les travaux dans ce domaine ont commencé dans les années 60 [38]. Longtemps cantonnés au noir et blanc, les premiers produits couleurs ont commencé à sortir. Universal Display Corporation, spécialiste des technologies OLED annonçait en 2006 la sortie imminente d'un écran couleur enroulable de près de 4 pouces. Sony annonçait à la même période la sortie d'un écran de ce type, souple et d'une épaisseur de 0,3 mm offrant 1,7 millions de couleurs. Même si les organismes de veille technologique tels que DisplaySearch annoncent que le marché des OLED dépassera les trois milliards de dollars avant la fin de 2008, pour l'instant dans les produits utilisant cette technologie la taille des écrans reste très petite (2,5 pouces dans le cas de l'écran de Sony) et les premiers usages de ce type de technologie visent donc en priorité des affichages de petite taille tels que ceux utiles pour des baladeurs, PDA, ou téléphones mobiles. D'autres produits plus originaux mais toujours de petite taille sont également annoncés tels que les claviers OLED ou les lunettes OLED. Pour les grands écrans, les perspectives sont moins claires. Néanmoins Sony annonce ce mois-ci qu'il commercialisera le premier écran OLED du monde à partir de décembre 2007. Baptisé XEL-1, cet écran doit faire 11 pouces pour une épaisseur de 3mm.

Aussi, les projets de papier électronique annoncés depuis déjà longtemps pourraient ainsi voir le jour même s'il est encore difficile de dire si la technologie sous-jacente sera alors OLED ou plutôt des technologies d'encre électronique issues des travaux du MIT et actuellement commercialisées sous le nom de e-Ink [35]. Ainsi, le livre électronique *iLiad* de iRex est-il sorti en 2006 avec un écran inédit de 8 pouces e-Ink d'une résolution 768 x 1024 avec 16 niveaux de gris. A la même période, Sony lance, après plusieurs essais infructueux, un nouvel eBook disposant d'un écran de 6 pouces avec une résolution de 800x600 et 4 niveaux de gris. Même si ces produits sont le résultat d'avancées technologiques très coûteuses, leur succès est encore loin d'être assuré et il est prévisible que la mise au point de ce type de matériel requiert encore du temps et des efforts.

En bref, les évolutions matérielles en matière d'affichage relèvent d'un domaine hautement technologique et requièrent des investissements très lourds sur de très longues années. Aussi ce domaine s'est-il resserré autour des industriels, principaux acteurs du marché. Par contre, l'exploitation des avancées en terme de matériel pour améliorer les systèmes de visualisation et d'interaction pose des problèmes d'un autre type auxquels nous nous intéressons particulièrement dans cette section. Ces problèmes sont nombreux et nous resserrons donc notre recherche sur les problèmes liés à l'introduction de sur-

faces d'affichage de grande taille qu'ils soient tactiles ou non et qu'ils soient distribués ou non. Nous nous intéressons en outre plus particulièrement aux systèmes d'affichage mixtes ie composés d'écrans de résolution et de définition différentes.

Le cas des dispositifs d'entrée est légèrement différent. William Buxton est sans doute le mieux placé pour faire justice à ce domaine par ses états de l'art complets et quasi exhaustifs sur le domaine [23; 24]. Aujourd'hui de nombreuses technologies relativement légères existent en partant de technologies totalement mobiles autour des stylos numériques tels par exemple les stylos IO Pen de Logitech ou le stylo Nokia SU-27W basé sur la technologie Anoto [5] ou bien des technologies plus fortement couplées au dispositif d'affichage comme dans le cas des tables ou mur tactiles. Dans ce domaine, les exemples les plus célèbres sont certainement la table DiamondTouch de Mitsubishi, [34; 91], la table de Jeff Han [52] ou bien plus récemment encore la *surface* de Microsoft [90]. Du côté des murs tactiles le nombre de projet est encore plus grand dans la mesure où les technologies sous-jacentes sont le plus souvent issues des tableaux interactifs, domaine désormais mature. A titre d'exemple on peut néanmoins citer le projet i-Land [105], l'un des précurseurs du domaine ainsi que d'autres projets plus récents tels que ceux de Stanford [39] ou le projet de Toronto [17]. Enfin un grand nombre d'efforts ont été faits pour exploiter les technologies basées sur la vision pour l'interaction sur des surfaces diverses. Des projets tels que celui de la table magique de F. Berard [22], ou les travaux d'Andy Wilson [114] de Microsoft sont des exemples de ce type d'approche.

Tous ces nouveaux dispositifs matériels qui battent en brèche le sempiternel dispositif clavier-souris-écran n'appartiennent pas non plus au paradigme courant de réalité virtuelle qui demande une immersion totale de l'individu dans un environnement virtuel. Le domaine de la réalité virtuelle, plus ancien et certainement plus mature repose sur du matériel souvent très lourd et particulièrement adapté à des interventions dans des domaines très spécifiques.

Le contexte de recherche dans lequel nous nous plaçons procède au contraire d'une démarche proche de celle de la réalité augmentée [87] où des objets réels (mur, table ou autre espace d'affichage) sont "augmentés" par des dispositifs informatiques légers et se voient ainsi transformés en instruments de visualisation et d'interaction.

Une illustration très représentative du contexte de recherche dans lequel nous nous plaçons est le *pick-and-drop* proposé par Rekimoto et al. [96] qui peut être vu comme une extension du *glisser-déplacer* (ou *drag-and-drop*). Le *pick-and-drop* permet un mode de transfert intuitif d'informations entre différents périphériques d'affichage. Dans le *pick-and-drop* la manipulation se fait avec un stylo, qui permet de prendre un objet (pick) sur un périphérique

d'affichage (un écran d'un ordinateur) et de le déposer (drop) sur un autre périphérique (l'écran d'un agenda électronique par exemple). Des alternatives à ce mode de transfert existent, par exemple, avec I-Land (Streitz, Geissler et al. 1999) c'est un objet physique personnel (stylo, lunette, etc) qui sert à véhiculer des informations prises sur des périphériques d'affichage différents.

D'autres travaux typiques de la réalité augmentée ont par contre une problématique plus éloignée de la notre. En effet, paradoxalement, les travaux sur les tableaux blancs numériques tels que Liveboard ([36]), qui pourraient a priori paraître pertinents en tant que nouveau dispositif d'entrée/sortie ont en fait une problématique qui s'avère assez différente. En effet, une grande partie des travaux effectués autour des tableaux blancs interactifs s'intéresse en premier lieu au tableau en tant que vecteur informel de communication entre différents individus lors de réunions de groupes. Ainsi, les modèles d'interaction et de visualisation élaborés visent-ils à permettre aux individus d'écrire, d'annoter, ou de dessiner en s'appuyant sur des algorithmes de reconnaissance de gestes, de reconnaissance d'écriture ou de segmentation pour que les objets du tableau puissent ensuite être sélectionnés et manipulés. C'est d'ailleurs cette dernière étape de segmentation qui demande le plus d'efforts car les informations susceptibles de se trouver sur les tableaux blancs sont complètement inconnues du système et donc plus difficiles à interpréter dans le cadre d'une sélection. Dans notre contexte, au contraire, les objets qui s'affichent sont connus du système qui a généré la vue et la segmentation est beaucoup moins un problème.

Face aux possibilités qu'offrent les nouveaux dispositifs, comment repense-t-on l'interaction et la visualisation d'informations ? Pour aborder cette question, nous commencerons par poser la problématique de ce domaine, nous présenterons les résultats que nous avons obtenus ces dernières années et nous finirons par une conclusion sur les perspectives.

2 Problématique

Pour mieux caractériser les problèmes rencontrés dans ce domaine, il est utile de préciser rapidement le vocabulaire que nous utilisons. En premier lieu, nous parlons de *display* ou *affichage* pour désigner un *dispositif physique capable d'afficher de l'information.* Une *fenêtre* est une sous-partie d'un affichage permettant à des programmes de gérer des entrées et des sorties. Un bureau ou espace de travail est un ensemble de fenêtres gérées par un gestionnaire de fenêtres ou système de fenêtrage unique. A partir de ces définitions de base, nous définissons une *surface* comme un ensemble de fenêtres potentiellement distribuées sur des displays gérés par des systèmes de fenê-

trage potentiellement différents. Nous appelons *surface partagée* une surface qui permet d'interagir de manière transparente avec les fenêtres de la surface de sorte que les personnes qui les manipulent interagissent aussi facilement que s'il s'agissait d'un même espace de travail (workspace). Les surfaces partagées auxquelles nous nous intéressons peuvent donc être vues comme un cas particulier de DDE (Distributed Display Environments)[70]. En outre, dans la mesure où une surface partagée est par définition distribuée, elle peut également être considérée comme un cas particulier de collecticiel (ou groupware).

Les problèmes auxquels nous nous intéressons plus particulièrement dans ce domaine s'organisent autour des axes suivants :

1. Utilisabilité des surfaces d'affichage et passage à l'échelle : dans quelle mesure les modèles d'interaction initialement conçus pour des dispositifs limités à un clavier-écran-souris sont-ils encore utilisables dans des environnements de surfaces partagées ? Comment assurer que des interactions génériques telles par exemple que le pointage, la sélection ou encore la manipulation directe d'objets se fassent dans des conditions optimales ?
2. Architecture et implémentation des surfaces partagées sur dispositifs potentiellement hétérogènes : Comment les surfaces partagées peuvent-elles dépasser les limites des systèmes de fenêtrage actuels. Comment la communication, la coordination et le couplage entre fenêtres gérées par des systèmes de fenêtrage différents peut-elle se faire de manière transparente ?
3. Dispositifs d'entrée et modèles d'interaction associés : Comment doit-on interagir avec les surfaces d'affichage émergentes ? Quel type de matériel ? Quelles révisions sur les modèles d'interaction classiques ?
4. Aspects analytiques : comment caractériser au mieux les surfaces partagées ? Quels sont les principales catégories de surfaces partagées et comment se comportent-elles par rapport aux problèmes précédemment évoqués ?

Nous ne pouvons pas faire ici un état de l'art exhaustif de toutes les avancées dans le domaine ces dernières années mais nous allons néanmoins tenter d'apporter quelques éléments marquants de sorte à caractériser un peu mieux la problématique sous-jacente. Les thèses relativement récentes de Lachenal [83], de Baralon [6] ou de Collomb [30] offrent des états de l'art à la fois plus ciblés et plus complets sur ces sujets.

Utilisabilité

L'avènement de nouveaux dispositifs d'entrée/sortie est très intéressant du point de vue de la visualisation et de l'interaction car elle offre de nouveaux espaces de conception. Néanmoins, il pose en premier lieu de nombreux problèmes d'utilisabilité. Ces problèmes sont de plusieurs types. En premier lieux un certain nombre de techniques d'interaction de base se heurtent au problème de passage à l'échelle, en second lieu les spécificités des nouveaux dispositifs offrent quelques opportunités pour de nouveaux styles d'interaction.

Passage à l'échelle

En 1997, la résolution d'un écran large tel que celui utilisé par Swaninathan et Sato [107] était de 2400x1200 pixels pour une taille de 1,8 m par 0,9m. Quelques années après le i-Wall [39] offrait une taille analogue mais avec une définition de 64 dpi, obtenue grâce à un cluster de 12 machines équipées de vidéo-projecteurs pour une résolution totale de 4608x2592. Récemment, le projet de Toronto [17] offre une résolution de 4730x1730 pixels obtenu par un cluster de 18 machines équipées de vidéo-projecteurs sur une surface d'une taille de 5m x 1,80m.

L'augmentation des surfaces d'affichage tant en taille qu'en résolution est certainement séduisante. Jusqu'où cette augmentation peut-elle aller ? La réponse est très certainement liée aux capacités perceptives humaines [115]. Néanmoins, les augmentations déjà enregistrées posent à elles seules des problèmes pour les techniques d'interaction classiques qui ne sont pas toutes susceptibles de réussir le passage à l'échelle sans des modifications majeures.

La problématique du pointage ou du déplacement d'objets en est la première illustration. En effet, dans la mesure où le pointage tel qu'il est pratiqué dans les environnements traditionnels est tenu par la loi de Fitt's [88], toute augmentation significative des distances à parcourir augmente la difficulté de ce type de tâche. Ces dernières années de nombreux travaux ont vu le jour pour permettre un pointage ou un déplacement des objets plus efficace en passant par des révisions majeures des paradigmes de base. Parmi ces travaux un certain nombre sont plus particulièrement conçus pour fonctionner avec des dispositifs de type pointage directs (doigt ou stylo par exemple) [9; 15; 18; 31; 48; 59; 59] alors que d'autres se sont intéressés au pointage distant [33; 39; 72; 74; 75; 89].

Vers une interaction plus directe

Alors que nous nous sommes accommodés du style WIMP sur les dispositifs

standards, il en va différemment des nouveaux dispositifs. En outre, les avancées post-wimp qui ont été relativement boudées dans un contexte "écran-clavier-souris" prouvent dans les dispositifs émergents tout leur intérêt. Car les nouveaux dispositifs conduisent de fait à une interaction plus proche de la manipulation directe. Les toolglass [16] sont un bon exemple. Alors que l'inadéquation du dispositif "écran-clavier-souris" a cantonné ce style d'interaction à une curiosité dans les années 90, on peut s'attendre à ce qu'avec avec une table multi-point, l'utilisation d'une toolglass paraît beaucoup plus pertinent. Il en va de même des pie-menus [68] ou autre style d'interaction analogue [81; 82] utile à l'activation de commandes [49]. Le concept de portail est un autre exemple. Ce type d'interaction multi-vue qui permet de sélectionner des zones d'écran et de les manipuler comme des objets graphiques tout en conservant leur propriété d'interactivité était déjà apparue dans les magic lenses [16] ou dans les portails de Pad++ [13]. Des approches plus récentes avec Wincuts [108] ou Façade[106] prennent tout leur intérêt sur des surfaces d'affichage de grande taille. La technique d'Alias appelée Frisbee [80] s'inscrit complètement dans cette logique.

Gestion de la topologie
Une grande spécificité des dispositifs d'affichage distribués est l'inadéquation totale de la topologie de référence : les dispositifs ont tous une position dans un espace 3D mais ce n'est pas cette localisation qui peut suffire à déterminer comment ils s'organisent. En effet, l'organisation des dispositifs nécessite que l'on définisse comment l'on passe de l'un à l'autre, comme ils se raccordent ou se recouvrent les uns par rapport aux autres. En bref, la topologie est entièrement à construire. Qu'elle soit construite à partir d'une analyse de la topologie réelle [67; 83] ou qu'elle s'appuie sur une approche plus conceptuelle [51], il commence à y avoir quelques solutions, mais le problème reste ouvert.

Architecture pour les surfaces partagées

Les problèmes d'architecture proviennent en premier lieux du manque crucial d'interopérabilité entre les principaux systèmes de fenêtrage existants. En effet, toute gestion distribuée d'un espace d'affichage hétérogène se heurte à l'incompatibilité entre les trois grandes familles de systèmes existants dans ce domaine : W-Windows, Windows, MacOS. Car, même si depuis plus de 10 ans des efforts sont faits pour obtenir une forme de cohérence du point de vue de l'utilisateur, cette cohérence se situe surtout au niveau des look-and-feel des systèmes.

Du point de vue du programmeur, les architectures et les implémentations divergent à bien des égards. Comment réussir l'intégration de fenêtres gérées

par des systèmes de fenêtrage différents de manière cohérente est une question encore ouverte. Cette problématique qui est au coeur de la problématique des environnements d'affichages multiples (ou MDE pour Multiple Display Environments) [70] est néanmoins une problématique qui a déjà été étudiée dans le cadre du Multiple Display Groupware (MDG). Elle apparaît plus complexe avec les MDE dans la mesure où alors que dans le cas du MDG les displays sont gérés par des personnes différentes dans des endroits différents, les situations d'usage des MDE sont des situations où les utilisateurs sont colocalisés, les affichages également et le même utilisateur peut interagir entre plusieurs displays. Les exigences sur l'intégration en sont d'autant plus fortes.

Caractérisation des dispositifs d'affichage distribués

S'agissant d'un domaine relativement jeune, il convient de mettre en évidence les principales caractéristiques des systèmes dédiés aux environnements multi-display. On peut commencer par distinguer deux grandes familles de dispositifs d'affichage distribués : d'un côté les murs d'images qui regroupent des dispositifs d'affichage identiques gérés de manière très rigide par un cluster de machines qui coopèrent et d'autre part les surfaces hétérogènes composées par exemple d'une table, d'un mur et d'un écran de mobile, et beaucoup plus ouvertes et reconfigurables. Les architectures pour les murs d'images sont assez matures et la problématique en terme d'implémentation est essentiellement tournée vers des problèmes de parallélisme pour l'augmentation des performances de rendu [69]. Les configurations hétérogènes n'ont pas les mêmes perspectives d'utilisation et les contraintes sur les performances de rendu sont moins fortes. Les exigences sont ailleurs : les affichages doivent pouvoir être connectés et déconnectés facilement, qu'il s'agissent du même système de fenêtrage ou pas, reconfigurés dynamiquement tant en terme de nombre qu'en terme de localisation. Ces dernières années des efforts ont été faits pour mieux caractériser l'ensemble de ces systèmes [6; 30; 32; 83; 85].

3 Résultats obtenus

Dans cette section nous parlons plus particulièrement des travaux très spécifiques que nous avons réalisés dans ce domaine.

3.1 Nouveaux dispositifs

Nous avons mis en place un dispositif appelé "mur-écran" qui est composé (1) d'une surface de projection – le "mur" - sur laquelle sont retro-projetées

des informations diverses susceptibles d'être utilisées à n'importe quel moment, et (2) d'un ou plusieurs écrans classiques servant plus particulièrement pour la tâche spécifique sur laquelle un individu travaille.
Le caractère hétérogène de cet affichage dans un contexte de visualisation interactive d'informations constitue en soit, la première originalité de nos travaux [61].
La surface de projection sur le mur est de plus dotée d'un détecteur de mouvements permettant aux individus d'interagir directement avec les éléments affichés sur la surface.
Le dispositif mur-écran présente ainsi au moins trois points forts :

→ **Augmentation et structuration de la surface d'affichage :** avec le mur-écran l'espace d'affichage augmente de manière significative en taille en même temps qu'il change de nature. En effet, cet espace d'affichage se décompose entre l'espace affiché sur le mur et l'espace d'affichage de l'écran. La différence de nature physique entre les deux espaces d'affichage (mur et écran) offre des possibilités de structuration : l'écran est considéré comme l'espace d'affichage principal, celui sur lequel se déroule la tâche en cours et l'affichage des informations utiles se fait donc a priori avec un niveau de détail maximal tandis que le mur est considéré comme une surface d'affichage secondaire qui permet d'afficher des vues d'ensemble de quantités importantes d'informations (avec un faible niveau de détail). En outre cette surface peut être de résolution et de taille variable en fonction des configurations.

→ **Rapidité d'accès à l'information :** les informations affichées sur le "mur" sont accessibles directement par le biais de dispositif de pointage direct sur le mur (ou par l'utilisation classique d'une souris).

→ **Coopération et partage des informations :** l'espace d'affichage mural offre des possibilités de coopération qui peuvent difficilement être mises en œuvre avec un affichage à l'écran. En effet, les informations affichées sur le mur sont susceptibles d'être utilisées, débattues, organisées par plusieurs soit de manière synchrone – lors de réunions s'appuyant sur des informations affichées à l'écran - de manière asynchrone – pour le partage d'informations importantes dans un groupe.

3.2 Architectures matérielles : vers des architectures distribuées

Toutes les architectures matérielles valides qui permettent un affichage multi-surfaces ont été envisagées et expérimentées, des plus simples aux plus difficiles à mettre en œuvre :

(A) configuration mono-machine, multi-écrans, mono-utilisateur ;
(B) configuration multi-machines, multi-écrans, mono-utilisateur et
(C) enfin une configuration multi-machines, multi-écrans, multi-utilisateurs (voir Figure 5) .

Il est important de souligner que dans l'état actuel des dispositifs de fenêtrage, il existe une différence colossale entre l'architecture A d'une part et l'architecture B (ou C) d'autre part. Dans le premier cas on peut parler d'affichage multi-écrans, la prise en charge par le système de fenêtrage de cette configuration facilite grandement la mise en place de techniques d'interaction spécifiques. Dans l'autre, on parlera d'affichage distribué : il s'agit de mixer des ressources d'affichage et d'interaction issues de plusieurs systèmes de fenêtrage. Actuellement aucun des trois systèmes de fenêtrage existants (Windows, X-Windows, MacOS) ne prend en charge ce type de configuration. Aussi, dans un tel contexte l'implantation de toute technique d'interaction spécifique est très lourde car elle doit commencer par mettre en place tous les mécanismes nécessaires à la gestion d'un environnement distribué (en particulier redirection et multiplexage des sorties et des entrées). Il existe pour ce faire des bribes de prises en charges implémentées de manière ad-hoc, et permettant redirection, réplication et parfois partage, mais la mise en œuvre de réels systèmes de fenêtrage distribués impose une refonte si colossale et si profonde des systèmes de fenêtrage actuels, qu'on peut penser qu'il s'écoulera encore un peu de temps avant qu'émergent des systèmes de fenêtrage capable de prendre en charge de manière transparente des dispositifs d'entrée/sortie distribués.

En attendant, l'architecture multi-machines et multi-utilisateurs peut être prise en charge de manière assez complète mais très spécifique au niveau applicatif, auquel cas, les modèles obtenus ne sont ni génériques, ni réutilisables. L'alternative consiste à prendre le problème au plus bas niveau ie au niveau système de fenêtrage comme dans I-AM [83]. Dans cette optique, le gain en généricité et plus réutilisabilité est énorme mais aussi plus coûteux. La prise en charge reste alors souvent plus partielle. Une alternative consiste à prendre le problème au niveau toolkit qui représente un bon compromis entre une perte de généricité pas trop importante et une couverture élargie permettant une intégration plus aisée.

Nous avons opté pour cette dernière approche de sorte à obtenir un modèle relativement générique et réutilisable. Nous avons limité notre champ d'étude à la prise en charge du drag-and-drop dans un contexte de systèmes d'affichage distribués. La configuration représentée par la Figure 5 caractérise bien ce type d'environnement. D'un côté un serveur d'affichage, le plus souvent muni de cartes graphiques musclées et d'écrans géants pour permettre de construire des vues d'ensemble. D'un autre côté des utilisateurs munis de

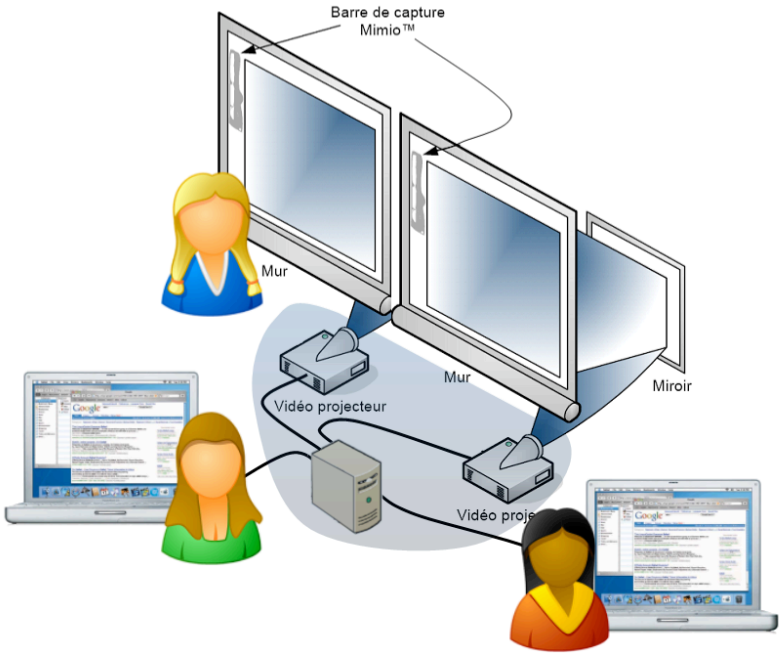

Fig. 1: *Exemple d'architecture d'affichage distribué*

portables et qui peuvent être amenés à interagir avec la surface gérée par le serveur. Cette configuration est un exemple typique de l'architecture de type (C) évoquée plus haut et c'est de loin l'architecture dans laquelle il est le plus difficile de mettre en œuvre des techniques d'interaction adaptées. C'est exactement le type d'architecture pour lequel le modèle de drag-and-drop a été conçu. Ce modèle est décrit dans la suite de ce document.

3.3 Extension du modèle classique de drag-and-drop : un nouveau modèle d'interaction

Dans les interfaces actuelles dites WIMP (Window-Icon-Menu-Pointer), le paradigme de manipulation directe introduit par Shneiderman [103] n'a jamais été aussi bien représenté que par le drag-and-drop. Cette technique d'interaction possède non seulement toutes les bonnes caractéristiques de la manipulation directe (représentation des objets d'intérêt et opérations sur ces objets par manipulation directe, opérations incrémentales, réparables) mais

elle a de plus la bonne propriété d'être générique. Elle est ainsi présente sous toutes sortes de formes pour toutes sortes d'applications et ce dans tous les systèmes de fenêtrage.

Paradoxalement, alors qu'il s'agit certainement là de la technique de manipulation directe la plus largement implémentée et utilisée aujourd'hui, les implémentations varient énormément : allant d'implémentations plutôt « archaïques » et non évolutives, à des implémentations plus élaborées permettant quelques extensions. Néanmoins, même dans les formes plus évoluées, les modèles d'implémentation actuels sont limités à 3 niveaux :

1. distribution : les systèmes actuels ne prennent pas en charge la distribution de l'affichage. Ainsi, une opération de drag-and-drop ne peut se faire entre deux affichages gérés par deux systèmes de fenêtrage distincts.
2. loi de Fitts : le modèle d'interaction sous-jacent au drag-and-drop est contraint par la loi de Fitts. En effet le temps de déplacement d'un objet par drag-and-drop classique est proportionnel à l'indice de difficulté associé à l'opération, cet ID correspondant au log du rapport de D sur L, où D est la distance à parcourir pour atteindre une cible et L est la largeur de la cible. Cette limite est importante pour les grands écrans et murs d'images. En effet dans des configurations de ce type l'ID des tâches de pointage peut atteindre des valeurs significativement plus élevées que sur les dispositifs d'affichage traditionnel. Dans le cas d'une configuration mur-écran où l'on couple ce type de surface avec d'autres écrans, l'augmentation de l'ID est inévitable.
3. pointage direct : dernière limitation, dans le cas d'un pointage direct, le drag and drop oblige que toutes les zones d'affichage puissent être atteintes. Dans sa version classique, il oblige également à exercer une pression continue sur le dispositif d'interaction tout le temps du déplacement. Là encore des problèmes se posent dans le cas d'écrans géants : certaines zones de l'affichage peuvent se trouver hors de portée, d'autre part si le dispositif de pointage est le doigt, ou même un stylo, maintenir la pression sur de longues distances est loin d'être une bonne solution !

3.4 Drag-and-throw, push-and-throw et push-and-pop.

Historiquement, la première technique proposée pour étendre le *drag-and-drop* proposée fut le *pick-and-drop*. Le *pick-and-drop* se propose de permettre à l'utilisateur de faire une chose impossible à faire avec *drag-and-drop* : transférer des données par manipulation directe d'une machine à une autre. Ceci

est fait en donnant l'impression à l'utilisateur de prendre physiquement un objet sur une surface et de le déposer sur une autre surface.
Cependant, le *pick-and-drop* est symboliquement plus proche du copier-coller que du *drag-and-drop* même s'il partage avec ce dernier l'avantage de dispenser l'utilisateur d'avoir conscience de l'existence d'un presse-papier et d'en connaître le fonctionnement. De la même manière que le *drag-and-drop* est plus naturel pour l'utilisateur que le copier-coller, le *pick-and-drop* est plus naturel qu'un copier-coller inter-machines utilisant le réseau.

Néanmoins, cette technique a plusieurs limitations :

- ▷ elle ne peut être utilisée qu'entre deux surfaces tactiles acceptant le même type de stylos. En particulier il est impossible de prendre un objet avec un stylo et de le déposer avec un autre stylo ou tout autre dispositif
- ▷ elle s'accompagne d'un modèle de gestion des évènements très sommaire : l'opération commence lors du pick, se termine lors du drop et rien ne peut se passer entre le pick et le drop.
- ▷ son implantation est ad-hoc et assez sommaire. Elle ne peut pas permettre d'intégrer d'autres extensions.

D'autres extensions ont été proposées (pour une revue de synthèse voire [30]), mais toutes ont des limitations, tant du point de vue du modèle d'interaction que du point de vue de l'implémentation.

Dans un premier temps, notre objectif a donc été de proposer un nouveau modèle d'interaction permettant d'étendre le modèle de drag-and-drop classique aux surfaces distribuées et de grande taille. Ce modèle s'inspire de la métaphore de lancer dans le monde réel et s'appuie sur des techniques permettant d'effectuer ce lancer de manière optimale dans un monde électronique. En particulier, l'accent est mis sur le calcul des trajectoires et les feedbacks visuels. Le but étant de trouver ceux qui permettent d'obtenir le meilleur confort d'utilisation, les meilleures performances en terme de temps de déplacement et les taux d'erreurs les plus faibles [59].
On peut distinguer trois types de retours visuels :

- → la trajectoire : affichage de la trajectoire de l'objet à déplacer comme une ligne joignant sa position actuelle et la position où il serait déplacé si le bouton de la souris était relâché. Notons que la trajectoire peut être aimantée par les cibles possibles (si on cherche à déplacer l'icône d'un document, la trajectoire sera aimantée par les objets compatibles : corbeille ou dossier par exemple).
- → la mire : affichage d'un point terminant la trajectoire.
- → la zone de « décollage » : c'est l'affichage sur la surface source d'une zone semi-transparente qui se projette sur la totalité de la surface cible.

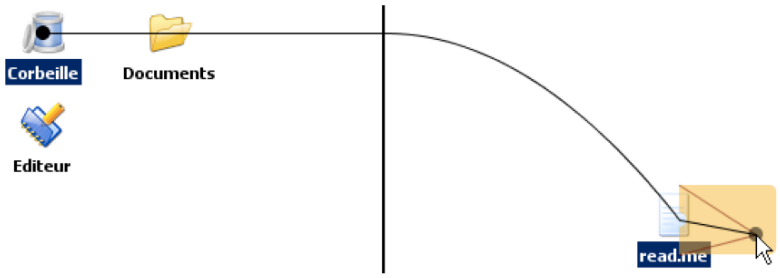

Fig. 2: *Exemple de drag-and-throw*

Cette aide visuelle aide l'utilisateur à comprendre où il peut déplacer le pointeur. On peut assimiler cette zone de décollage à une vue radar de la surface cible (stylisée par un simple rectangle de couleur unie) cf figures.

Le *drag-and-throw* et le *push-and-throw* sont des techniques de lancer d'icônes adaptées aux surfaces multiples. Ces techniques permettent de déplacer des icônes à partir d'une surface vers une autre (que ces deux surfaces soient reliées à la même machine ou non).

La différence entre le *drag-and-throw* et le *push-and-throw* se situe au niveau des trajectoires et de la méthode de projection de la zone de décollage sur la surface cible.

Le *drag-and-throw* utilise la métaphore du tir à l'arc (Figure 10), c'est-à-dire que l'utilisateur effectue un mouvement dans le sens opposé à la direction dans laquelle il veut lancer l'objet. S'il veut envoyer l'objet plus haut, il doit déplacer le curseur vers le bas. La trajectoire associée au *drag-and-throw* se compose d'une courbe de Bézier sur la surface source et d'une ligne droite sur la surface cible.

Le *push-and-throw,* quant à lui, illustre la métaphore du pantographe (Figure 11), c'est-à-dire que l'utilisateur fait un petit mouvement de souris qui est amplifié dans la même direction pour déterminer où lancer l'objet. La trajectoire est dessinée quant à elle par une simple ligne droite joignant le curseur de la souris au point où sera lancé l'objet.

L'utilisateur débute un *drag-and-drop* classique, et, après un petit mouvement (quelques pixels – suffisamment pour déterminer la direction du glisser-déposer), les aides visuelles apparaissent : une zone de décollage sur la surface source, une mire sur la surface cible, et une trajectoire joignant la mire au curseur de la souris. L'utilisateur ajuste ensuite son lancer puis relâche le bouton de la souris ce qui a pour effet de déplacer immédiatement l'objet.

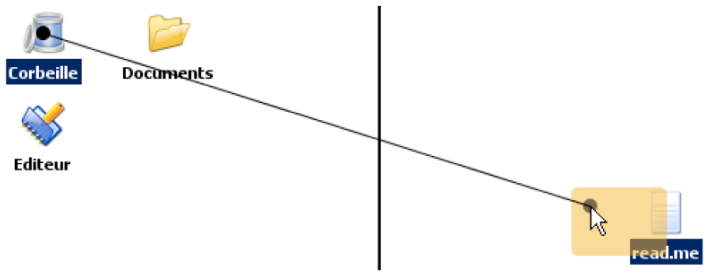

Fig. 3: *Exemple de push-and-throw*

L'avantage incontestable de ces techniques de lancer est qu'elles permettent un contrôle précis du lancer. En effet, le feedback visuel permet d'anticiper la destination du lancer et l'utilisateur est donc en mesure d'ajuster précisément son lancer. De plus, le calcul des trajectoires permet d'éviter le comportement chaotique du lancer résultant de trajectoires telles que celles proposées dans les travaux de Geissler et al. [44]

3.5 Expérimentations contrôlées sur dispositifs d'affichage distribué

Le modèle d'interaction élaboré a fait l'objet de plusieurs expérimentations. Ces expérimentations ont permis non seulement d'obtenir des résultats quantitatifs en terme de performances utilisateurs, mais elles ont également permis d'obtenir des résultats en terme d'analyse des dimensions de l'espace de conception.

En outre, ces études ont fait l'objet de collaboration. En effet, pour la comparaison entre les techniques de lancer et le reste des alternatives proposées récemment, nous avons monté une expérimentation en collaboration avec Patrick Baudisch de Microsoft research (Etats-Unis) et de Brian Lee de l'Université de Stanford (Etats-Unis). Les résultats de cette étude sont publiés dans [31].

3.6 Modèle générique d'implémentation de drag-and-drop

Le modèle d'interaction instrumentale [Beaudouin-Lafon, 2000] se prête très bien à la description du drag-and-drop et de ses évolutions. Le premier avantage de ce modèle est que son utilisation permet de clarifier les concepts qui entrent en jeu dans le modèle d'interaction. Le deuxième est que ce modèle, initialement introduit comme un modèle d'interaction, peut néanmoins également servir comme base pour un modèle d'implémentation.

Ainsi, le modèle d'implémentation générique [30; 54; SPE'2008] que nous proposons est directement dérivé du modèle d'interaction instrumentale. Nous y voyons l'assurance d'une implémentation cohérente et complète répondant aux besoins de l'interaction.
Comme nous avons pu le montrer dans notre étude approfondie des différents modèles d'implémentation du drag-and-drop classique, les variations sont importantes. L'absence d'un modèle d'implémentation unifié du drag-and-drop entre les différentes plates-formes entrave l'intégration des évolutions ainsi que le support des environnements distribués. Jusqu'a présent, le pick-and-drop, l'hyperdragging et le stitching ont été implémentés dans des environnements d'affichage distribués. L'objectif de notre travail est ici de proposer un modèle unifié et ouvert pour toutes les méthodes de type drag-and-drop. Ce qui inclut bien évidemment le drag-and-drop, mais aussi toutes les évolutions qui ont été récemment proposées pour étendre ses possibilités. Ces évolutions du drag-and-drop incluent des réactions supplémentaires ainsi que des feedbacks plus riches. Elles nécessitent également le support des environnements distribués. Notre volonté est de construire un modèle qui facilite l'intégration des évolutions récentes et futures tout en prenant en compte les implémentations existantes du drag-and-drop.

Chapitre 5

CONCLUSION ET PERSPECTIVES

Nous avons présenté séparément trois axes de recherche auxquels nous nous intéressons : visualisation interactive, modélisation/construction et dispositifs multi-supports. Leur complémentarité fait que les perspectives de recherche à venir sont nombreuses. Depuis mon arrivée au LIRMM j'ai essayé de développer une activité autour de ces thèmes. Les progrès sont lents mais conséquents. Ces dernières années nous avons en particulier construit deux outils que nous commençons à pouvoir exploiter : une salle avec du matériel dédié, un framework de développement pour la visualisation interactive de données. Par ailleurs nous avons commencé à mettre en place des éléments d'évaluation : métriques et protocoles expérimentaux. Aussi, nos perspectives de recherche actuelles s'inscrivent-elles dans la suite logique des travaux déjà réalisés. En premier lieu il s'agit de continuer à élargir notre recherche dans le domaine des nouveaux modèles de visualisation et d'interaction en adéquation avec les évolutions matérielles. En second lieu, il s'agit de poursuivre nos efforts en modélisation et construction automatique de vues d'ensemble. L'orientation que nous prenons étant toujours celle d'une représentation multi-échelle des informations sous-jacentes. Enfin, il s'agit de continuer à évaluer nos résultats tant par une approche expérimentale que par une approche analytique.

1 Visualisation et interaction sur nouveaux dispositifs d'entrée/sortie

Il nous a fallu quelques années pour faire équiper une salle de matériel approprié. Cette salle offre aujourd'hui un écran de 5mx1,35m en rétroprojection et un écran de 20 pouces. La totalité de la surface offre une résolution de 4 096x768. Cet équipement a été conçu pour être très modulaire. Pour le moment, la principale surface de projection est composée d'une toile, et rend un peu difficile toute interaction directe au stylo ou au doigt. A terme

nous aimerions remplacer cette toile par un écran rigide, mais les coûts des écrans rigides de cette dimension sont encore trop élevés. L'architecture modulaire de cette salle fait également que toute partie de l'écran peut être augmentée en résolution. Le matériel étant mis en commun pour tout le laboratoire et installé dans la seule salle de réunion du laboratoire de plus de 15 m2, il y a quelques pressions et l'évolution se fait par étape... En même temps, le fait que le matériel soit installé dans la salle la plus fréquentée du laboratoire offre une occasion unique d'expérimentations grandeur réelle.

Nous avons mis en évidence dans le chapitre 4 quelques problèmes ouverts de ce domaine. Nos perspectives de recherche s'inscrivent dans cette logique. Nous pouvons désormais nous appuyer sur un modèle assez complet pour la manipulation directe d'objets dans un environnement distribué et équipé d'affichages de grande taille. Nous avons de par notre expérience et les travaux nombreux de ce domaine une vision assez complète de ce que devrait être le pointage dans ces environnements. Il reste cependant des questions. Nos perspectives de recherche portent plus particulièrement sur la construction automatique de vues d'ensemble multi-échelle, ainsi que sur les problèmes de sélection, de gestion de la topologie et de d'activation de commande.

1.1 Pointage et sélection

Bien souvent pointage est synonyme de sélection, pourtant il s'agit de deux problèmes différents. Le pointage permet l'identification d'une ou plusieurs positions sur la surface. A partir de ces positions, le calcul de la sélection est à construire. Les environnements de visualisation interactives sont encore très pauvres de ce point de vue, et la plupart du temps la sélection n'est pas autre chose qu'un `pick()` (fonction généralement responsable du calcul des objets se trouvant sous le pointeur). Comme nous l'évoquions précédemment, les nouveaux dispositifs constituent un terrain privilégié de manipulation directe. Le premier pré-requis d'une vraie manipulation directe est un mécanisme de sélection plus développé. La sélection devrait permettre de saisir des objets numériques avec autant de possibilités que l'on a de saisir des objets dans le monde réel. Sélection d'une partie d'un objet, d'un sous-ensemble contigu ou discontigu d'objets, d'un ensemble autre que rectangulaire, etc. Les environnements de développement à l'usage des graphiques sont certainement à ce jour les environnements qui développent les mécanismes de sélection les plus évolués. Cependant, il reste encore beaucoup à faire et l'avènement de dispositifs de saisie multi-points ouvre des perspectives nouvelles.

1.2 Activation de commandes

L'activation de commandes peut-être vue comme un prolongement naturel de la sélection [49]. Nous avons évoqué rapidement dans les chapitres précédents les travaux autour des see-through tools, des pie-menus et des marking menus. Là encore l'avènement de dispositifs de saisie multi-point ouvre l'espace de conception. Ces techniques sont encore peu matures. Nos expériences préliminaires sur les pie-menus dans un contexte de visualisation et d'interaction directe montrent que des évolutions sont encore souhaitables [56].

1.3 Architectures et développement

L'état actuel des toolkits classiques rend tous les développements très lents. Aussi, une part importante de nos perspectives est dans le développement de toolkits adaptés aux problèmes de visualisation et d'interaction multi-supports.

Plusieurs API ont été développées ces dernières années notamment PoIP [28] pour la création de techniques de drag-and-drop multi-display en Java et PACOM pour la création de différentes modèles de Pie-Menus [56].

Nous avons travaillé également au développement d'un framework de visualisation interactive permettant de mettre en oeuvre les techniques de construction automatique, de clustering et de filtrage passées ou en cours de développement. Nous continuons ces travaux actuellement.

2 Modélisation, construction, filtrage

Les récents travaux que nous avons effectués nous ont permis de mettre en oeuvre des mécanismes de filtrage et de clustering qui donnent des résultats très intéressants sur des graphes du monde réels tels que les graphes de réseaux sociaux, les graphes de type UML, graphe d'appels ou graphes de visibilité. Les graphes tels que les interactomes issus de la biologie sont un autre cas d'étude auquel nous nous intéressons dans le cadre du projet ANR PlasmoExplore. Une particularité de tous ces graphes est leur caractère très dense résultant d'un certains nombre de caractéristiques dont deux identifiées récemment : un caractère petit monde et le caractère sans échelle. Ces caractéristiques ont été beaucoup étudiées dans le cadre de la thèse de F. Boutin [21]. Notre perspective de travail est désormais d'exploiter les résultats obtenus et de mieux les comparer aux autres approches en matière de clustering et d'optimisation de partition.

Au delà de la comparaison des résultats, cette analyse devrait permettre de mettre en évidence de manière précise les propriétés des graphes qui sont susceptibles de mettre en échec les techniques de classification courantes. Le but étant de fournir des outils de simplification automatique de structure qui agissent sur ces propriétés sans changer pour autant la nature profonde du graphe. Cette approche permet ensuite de coupler des techniques de filtrage pertinente aux algorithmes de clustering et offre des résultats prometteurs [21].

3 Evaluation

Pour aborder le problème de l'évaluation des stratégies de visualisation et d'interaction, nous nous intéressons à deux types d'approches : l'approche analytique (critères analytiques et quantitatifs) et l'approche empirique (expérimentations contrôlées auprès d'utilisateurs réels).

3.1 Approche Empirique

En matière d'approche empirique, nos travaux ont commencé par l'analyse comparative des effets des algorithmes de placement sur les performances des individus [63]. D'autres travaux que nous effectuons aujourd'hui en terme de modèle d'interaction ont également fait l'objet d'expérimentations contrôlées auprès d'utilisateurs réels. C'est le cas par exemple des modèles de trajectoires pour le lancer d'objets numériques définis dans [59] (primé du "best award short paper"). Même si ces évaluations sont souvent coûteuses en temps, elles représentent une occasion unique et indispensable de confronter la théorie à la pratique. Elles nous permettent non seulement d'évaluer nos modèles mais aussi de les faire évoluer. C'est dans ce contexte que nous avons effectué la collaboration informelle avec Patrick Baudisch de Microsof Research et Brian Lee de Stanford University . Les expérimentations contrôlées réalisées dans cette collaboration ont permis de mieux analyser et évaluer les évolutions du drag-and-drop pour dispositifs de pointage direct et écrans géants. Brian Lee a effectué l'expérimentation sur le iWall de Stanford, un écran géant homogène d'environ 4,50m sur 1,1 m tandis qu'avec Maxime Collomb, doctorant au LIRMM nous avons réalisé la même expérimentation sur un matériel hétérogène couvrant une surface d'environ 3m sur 1,1m. Les résultats de ces expérimentations ont été riches d'enseignements et ont donné lieu à la conception commune d'une nouvelle technique d'interaction publiée à la conférence Graphics Interface en 2005 [31].

3.2 Approche analytique

L'évaluation analytique des systèmes de visualisation pose quant à elle un problème crucial parce qu'il n'existe pas encore de consensus autour de mesures de référence qui permettraient d'évaluer les nouvelles techniques proposées.

Dans un cadre purement analytique nous nous sommes intéressés à la mise en place de critères d'évaluation des résultats des algorithmes de clustering. Une partie du travail de thèse de François Boutin a en particulier consisté à faire une analyse comparative des critères existants à partir de laquelle sont nés de nouveaux critères [IV'2004].

3.3 Approche Mixte

Si le cadre purement analytique peut s'envisager du point de vue des algorithmes de construction qui précèdent la production de toute visualisation, il est indispensable de les corroborer par une approche expérimentale quand il s'agit d'évaluation sur les vues produites. En effet, ce n'est pas la qualité de la vue en soit qui est intéressante mais plutôt la qualité de la vue interprétée par l'individu et de ce qu'il peut en faire. En d'autres termes il s'agit d'évaluer si la vue est optimale en terme de perception et d'action. L'évaluation du résultat d'algorithmes de placement est un bon exemple où nous avons mis en évidence les limites des critères analytiques purs et montré qu'ils n'étaient pas toujours corrélés aux performances d'individus qui utilisaient les vues [63]. Dans ce domaine nos perspectives consistent à continuer à poursuivre dans cette approche mixte en collaboration avec d'autres domaines tels que le domaine des sciences cognitives pour réussir à mettre en évidence des critères analytiques qui soient corrélés à des critères de performance évalués empiriquement. Une fois validée formellement de tels critères permettraient à terme d'obtenir des résultats prédictifs de performances de manière analytique.

3.4 Approche benchmarks

Nous avons participé à la mise en place de benchmarks au travers de différents projets notamment le projet CNRS As Evaluation et projet ministère – Action cognitique.

Ces travaux ont permis de collecter des jeux de données, des taxonomies de tâches pertinentes. Ces travaux ont également permis de commencer à mettre en place des tests préliminaires sur les individus intervenant dans les expériences [58]. Cette initiative originale est motivée par les observations de

Westerman et Cribbin [112] qui ont montré que le type de sujets choisis pour une expérimentation contrôlée introduisait un biais certain sur les résultats.

Ces travaux se sont concrétisés par la mise en place d'un site web coopératif en 2002 [58] qui a inspiré nos collègues Français et Américains pour la mise en place des défis "visualisation" de la conférence internationale de l'IEEE Infoviz [37; 95].

3.5 Approche complète

En matière de visualisation et d'interaction, l'un des axiomes de base que rappelle W. Buxton est toujours valide : "Everything is best for something and worst for something else". On peut donc regretter que les évaluations des systèmes de visualisation qui sont faites par les auteurs eux-mêmes restent souvent partielles. Nous avons essayé d'échapper à cette règle dans les expérimentations contrôlées que nous avons évoquées plus haut[31], mais ce type d'expérience ne peut pas toujours se faire de manière aussi complète. Le recours à des cadres d'évaluation plus globaux serait une alternative également intéressante. Des travaux dans ce sens ont déjà commencé à se développer [27]. C'est dans cette perspective que nous souhaitons poursuivre, en nous appuyant sur les approches que nous venons d'évoquer.

BIBLIOGRAPHIE

[1] C. Ahlberg, C. Williamson, and B. Shneiderman. Dynamic queries for information exploration : An implementation and evaluation. In *CHI'92*, 1992.

[2] C. Ahlberg and E. Wistrand. Ivee : An information visualization & exploration environment. In *Information Visualization'95*, 1995.

[3] Christopher Ahlberg and Ben Shneiderman. Visual information seeking : Tight coupling of dynamic query filters with starfield display. In *CHI'94*, pages 313–318. ACM, April 1994.

[4] Keith Andrews and Helmut Heidegger. Information slices :visualising and exploring large hierarchies using cascading, semi-circular discs. In *IEEE Symposium on Information Visualization (InfoVis'98)*, 1998.

[5] Anoto. Anoto pen. http://www.anoto.com., 2005.

[6] Nicolas Barralon. *Couplage de Ressources d'Interaction en Informatique Ambiante*. PhD thesis, 2006. 170 pages.

[7] L. Bartram and al. The continuous zoom : A constrained fisheye technique for viewing and navigating large information spaces. In *UIST'95*, 1995.

[8] G. Di Battista, P. Eades, and R. Tamassia. *Graph Drawing : algorithms for the visualization of graphs*. Prentice Hall, 1999.

[9] P. Baudisch, E. Cutrell, D. Robbins, M. Czerwinski, P. Tandler, B. Bederson, and Z. Zierlinger. Drag-and-pop and drag-and-pick : Techniques for accessing remote screen content on touch- and pen-operated systems. In *Human-Computer Interaction – INTERACT'03*, pages 57–64. IOS Press, (c) IFIP, 2003.

[10] B. Bederson, J. Meyer, and L. Good. Jazz : An extensible zoomable user interface graphics toolkit in java. In *ACM UIST 2000*, pages pp. 171–180, 2000.

[11] B. B. Bederson, J. Grosjean, and J Meyer. Toolkit design for interactive structured graphics. Technical Report HCIL-2003-01, CS-TR-4432, UMIACS-TR-2003-03, Univ. Maryland, 2003.

[12] B. H. Bederson and J. D. Hollan. Pad++ : a zooming graphical interface for exploring alternate interface physics. In *Proceedings Uist'94*, pages 17–26. ACM Press, 1994.

[13] Benjamin B. Bederson and James D. Hollan. Pad++ : a zooming graphical interface for exploring alternate interface physics. In *UIST '94 : Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 17–26, New York, NY, USA, 1994. ACM Press.

[14] Jacques Bertin. *La Graphique et le Traitement Graphique de l'information*. Flammarion, 1977.

[15] Anastasia Bezerianos and Ravin Balakrishnan. The vacuum : facilitating the manipulation of distant objects. In *CHI '05 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 361–370, New York, NY, USA, 2005. ACM Press.

[16] Eric A. Bier, Maureen C. Stone, Ken Pier, William Buxton, and Tony D. DeRose. Toolglass and magic lenses : the see-through interface. In *SIGGRAPH '93 : Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 73–80, New York, NY, USA, 1993. ACM Press.

[17] Jeremy P. Birnholtz, Tovi Grossman, Clarissa Mak, and Ravin Balakrishnan. An exploratory study of input configuration and group process in a negotiation task using a large display. In *CHI*, pages 91–100, 2007.

[18] Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. Semantic pointing : improving target acquisition with control-display ratio adaptation. In *CHI '04 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 519–526, New York, NY, USA, 2004. ACM Press.

[IV'2004] F. Boutin and M. Hascoët. Cluster Validity Indices for Graph Partitioning. In *IV'04 : 8th IEEE International Conference on Information Visualization*, pages 376–381, London (UK), 2004. IEEE.

[Interact'2005] F. Boutin, J. Thievre, and M. Hascoët. Multilevel Compound Tree Construction Visualization and Interaction. In *Interact'05*, Rome (Italie), 2005. Springer.

[21] François Boutin. *Filtrage, partitionnement et visualisation multi-échelles de graphes d'interactions à partir d'un focus.* PhD thesis, Université Montpellier II - Sciences et Techniques du Languedoc, 28/11/2005.

[22] François Bérard. The magic table : Computer-vision based augmentation of a whiteboard for creative meetings. 2003.

[23] William Buxton. Input devices. http://www.billbuxton.com/InputSources.html.

[24] William Buxton. Input theory. http://www.billbuxton.com/InputSources.html.

[25] Matthew Chalmers. Using landscape metaphor to represent a corpus of documents. In *LNCS*, pages 377–389. 1993.

[26] Matthew Chalmers and Paul Chitson. Bead : Explorations in information visualization. In *SIGIR'92*, pages 330–337. ACM, June 1992.

[27] Chaomei Chen and Yue Yu. Empirical studies of information visualization : a meta-analysis. *Int. J. Hum.-Comput. Stud.*, 53(5) :851–866, 2000.

[28] M. Collomb. Poip : an api for implementing advanced drag-and-drop techniques. http://www.lirmm.fr/edel/dragging/, 2005.

[SPE'2008] M. Collomb and M Hascoët. an open and unified implementation model for drag-and-drop extensions to large and distributed display environments. *International Journal on Software, Practice and Experiences*, Submitted and accepted with major modification, Resubmitted july 2007.

[30] Maxime Collomb. *Vers des systèmes de fenêtrage distribués : l'évolution du drag-and-drop.* PhD thesis, Université Montpellier II - Sciences et Techniques du Languedoc, 01/12/2006.

[31] Maxime Collomb, Mountaz Hascoët, Patrick Baudisch, and Brian Lee. Improving drag-and-drop on wall-size displays. In *GI '05 : Proceedings of Graphics Interface 2005*, pages 25–32, School of Computer Science,

University of Waterloo, Waterloo, Ontario, Canada, 2005. Canadian Human-Computer Communications Society.

[32] Joëlle Coutaz, Christophe Lachenal, and Sophie Dupuy-Chessa. Ontology for multi-surface interaction. In *Proceedings of Interact 2003*, 2003.

[33] Jr. Dan R. Olsen and Travis Nielsen. Laser pointer interaction. In *CHI '01 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 17–22, New York, NY, USA, 2001. ACM Press.

[34] Paul Dietz and Darren Leigh. Diamondtouch : a multi-user touch technology. In *UIST '01 : Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226, New York, NY, USA, 2001. ACM Press.

[35] E-Ink. E-ink website. http ://www.eink.com/, Accessed 2006.

[36] Scott Elrod, Richard Bruce, Rich Gold, David Goldberg, Frank Halasz, William Janssen, David Lee, Kim McCall, Elin Pedersen, Ken Pier, John Tang, and Brent Welch. Liveboard : a large interactive display supporting group meetings, presentations, and remote collaboration. In *CHI '92 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 599–607, New York, NY, USA, 1992. ACM Press.

[37] C. Plaisant et al. Infoviz contest. http://www.cs.umd.edu/hcil/iv04contest/, 2004.

[38] Society for Information Display. 40 years of sid symposia Ű nurturing progress in liquid crystal (lc) technology. http://www.sid.org/archives/0-LCpdfExhibit3.pdf, 1997.

[39] Armando Fox, Brad Johanson, Pat Hanrahan, and Terry Winograd. Integrating information appliances into an interactive workspace. *IEEE Computer Graphics and Applications*, 20(3) :54–65, 2000.

[40] G. Furnas and B. Bederson. Space scale diagrams : understanding multiscale interfaces. In *Chi'95*. ACM Press, 1995.

[41] George W. Furnas. Generalized fisheye views. In *CHI'86*. ACM SIGCHI, 1986.

[42] George W. Furnas and Jeff Zacks. Multitree : Enriching and reusing hierarchical structure. In *CHI'94*. ACM, 1994.

[43] Perlman Gary and al. The HCI bibliography project technical report. http://www.hcibib.org.

[44] Jörg Geißler. Shuffle, throw or take it ! working efficiently with an interactive wall. In *CHI '98 : CHI 98 conference summary on Human factors in computing systems*, pages 265–266. ACM Press, 1998.

[45] J. J. Gibson. *The Ecological Approach to Visual Perception*. Houghton-Mifflin (Boston), 1979.

[46] Yotam I. Gingold, Philip L. Davidson, Jefferson Y. Han, and Denis Zorin. A direct texture placement and editing interface. In *UIST '06 : Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 23–32, New York, NY, USA, 2006. ACM Press.

[47] Y. Guiard, F. Bourgeois, D. Mottet, and M. Beaudouin-Lafon. Beyond the 10-bit barrier. In *IHM-HCI 2001*, 2001.

[48] François Guimbretière, Maureen Stone, and Terry Winograd. Fluid interaction with high-resolution wall-size displays. In *UIST '01 : Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 21–30, New York, NY, USA, 2001. ACM Press.

[49] François Guimbretiére and Terry Winograd. Flowmenu : combining command, text, and data entry. In *UIST '00 : Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 213–216, New York, NY, USA, 2000. ACM Press.

[50] Carl Gutwin and Saul Greenberg. Focus and awareness in groupware. In *Proceedings of ACM CSCW'98 Conference on Computer-Supported Cooperative Work*, Videos, pages 425–426. 1998.

[51] Vicki Ha, Kori Inkpen, Jim Wallace, and Ryder Ziola. Swordfish : user tailored workspaces in multi-display environments. In *CHI '06 : CHI '06 extended abstracts on Human factors in computing systems*, pages 1487–1492, New York, NY, USA, 2006. ACM Press.

[52] Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05 : Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, New York, NY, USA, 2005. ACM Press.

[53] Jefferson Y. Han. Multi-touch sensing through frustrated total internal reflection. In *SIGGRAPH '05 : ACM SIGGRAPH 2005 Sketches*, New York, NY, USA, 2005. ACM Press.

[54] M. Hascoët, M. Collomb, and R. Blanch. Evolution du Drag-and-Drop : Du Modèle d'Interaction Classique aux Surfaces Multi-Supports. *Revue I3 : InformationInteractionIntelligence*, 4(2) :x, 2005.

[55] Mountaz Hascoët. Interaction and visualization supporting web browsing patterns. In *IV '01 : Proceedings of the Fifth International Conference on Information Visualisation*, page 413, Washington, DC, USA, 2001. IEEE Computer Society.

[56] Mountaz Hascoët, Maxime Collomb, and Jérôme Cance. Accelerating object-command transitions with pie menus. In *Enactive'06 : Conference on Enaction and Complexity*, pages 109–111, 2006.

[57] Mountaz Hascoët-Zizi and Nikos Pediotakis. Visual relevance analysis. In *DL '96 : Proceedings of the first ACM international conference on Digital libraries*, pages 54–62, New York, NY, USA, 1996. ACM Press.

[58] M. Hascoët. Voir site web : http ://www.lirmm.fr/ infoviz/aseval/, 2001-2002.

[59] M. Hascoët. Throwing models for large displays. In British HCI Group, editor, *Human Computer Interaction, HCI'2003*, Bath, UK, 2003.

[60] Mountaz Hascoët. A user interface combining navigationl aids. In *ACM Hypertext'00*, San Antonio, 2000. ACM Press.

[61] Mountaz Hascoët and Frederic Sackx. Exploring interaction strategies with wall-screen : A new dual-display device for managing collections of web pages. In *Information Visualization 2002*, pages p. 719–725. IEEE, 2002.

[62] Mountaz Hascoët and Xavier Soinard. Using maps as a user interface to a digital library. In Short paper, editor, *SIGIR*. ACM, 1998.

[63] Mountaz Hascoët-Zizi. Analytical versus empirical evaluation of spatial displays. In *ACM CHI'98*, Los Angeles, 1998. ACM Press.

[64] M. Hemmje, C. Kuntel, and A. Willett. Lyberworld - a visualization user interface supporting fulltext retrieval. In *SIGIR'94*, pages 249–259. Springer-Verlag, 1994.

[65] I. Herman, G. Melancon, and M.S. Marshall. Graph visualization and navigation in information visualisation. *IEEE trans. on Visualization and Computer Graphics*, 6(1) :p 24–43, 2000.

[66] R. Hightower and al. Graphical multiscale web histories : a study of padprints. In *Hypertext'98*. ACM Press, 1998.

[67] Ken Hinckley, Gonzalo Ramos, Francois Guimbretiere, Patrick Baudisch, and Marc Smith. Stitching : pen gestures that span multiple displays. In *AVI '04 : Proceedings of the working conference on Advanced visual interfaces*, pages 23–31, New York, NY, USA, 2004. ACM Press.

[68] Don Hopkins. The design and implementation of pie menus. *Dr. Dobb's J.*, 16(12) :16–26, 1991.

[69] Greg Humphreys, Mike Houston, Ren Ng, Randall Frank, Sean Ahern, Peter D. Kirchner, and James T. Klosowski. Chromium : a stream-processing framework for interactive rendering on clusters. In *SIGGRAPH '02 : Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 693–702, New York, NY, USA, 2002. ACM Press.

[70] Dugald Ralph Hutchings, John Stasko, and Mary Czerwinski. Distributed display environments. *interactions*, 12(6) :50–53, 2005.

[71] Inselberg. Multidimensional detective. In *Readings in information visualization*, pages 107–115. Morgan Kaufmann, 1999.

[72] Shahram Izadi, Harry Brignull, Tom Rodden, Yvonne Rogers, and Mia Underwood. Dynamo : a public interactive surface supporting the cooperative sharing and exchange of media. In *UIST '03 : Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 159–168, New York, NY, USA, 2003. ACM Press.

[73] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering : a review. *ACM Comput. Surv.*, 31(3) :264–323, 1999.

[74] Brad Johanson, Armando Fox, and Terry Winograd. The interactive workspaces project : Experiences with ubiquitous computing rooms. *IEEE Pervasive Computing*, 1(2) :67–74, 2002.

[75] Brad Johanson, Greg Hutchins, Terry Winograd, and Maureen Stone. Pointright : experience with flexible input redirection in interactive

workspaces. In *UIST '02 : Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 227–234, New York, NY, USA, 2002. ACM Press.

[76] Jérôme Thièvre. *Cartographies pour la Recherche et l'Exploration de Données Documentaires.* PhD thesis, Université Montpellier II - Sciences et Techniques du Languedoc, 12/10/2006.

[77] Susanne Jul and George W. Furnas. Critical zones in desert fog : Aids to multiscale navigation. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, Zoomable User Interfaces, pages 97–106. 1998. http ://www.acm.org/pubs/articles/proceedings/uist/288392/p97-jul/p97-jul.pdf.

[78] T. Kamada and S. Kawai. An algorithm for drawing general indirect graphs. *Information Processing Letters*, 31(1) :7–15, 1989.

[79] D.A. Keim and H.P. Kriegel. Visualization techniques for mining large databases : A comparison. In *IEEE Transactions on Knowledge and Data Engineering*, pages p. 1–29, 1996.

[80] Azam Khan, George Fitzmaurice, Don Almeida, Nicolas Burtnyk, and Gordon Kurtenbach. A remote control interface for large displays. In *UIST '04 : Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 127–136, New York, NY, USA, 2004. ACM Press.

[81] Gordon Kurtenbach and William Buxton. Issues in combining marking and direct manipulation techniques. In *UIST '91 : Proceedings of the 4th annual ACM symposium on User interface software and technology*, pages 137–144, New York, NY, USA, 1991. ACM Press.

[82] Gordon Paul Kurtenbach. *The design and evaluation of marking menus.* PhD thesis, Toronto, Ont., Canada, Canada, 1993.

[83] Christophe Lachenal. *Modèle et infrastructure logicielle pour lŠinteraction multi-instrument multisurface.* PhD thesis, 2004. 200 pages.

[84] J. Lamping, R. Rao, and P. Pirolli. A focus + context technique based on hyperbolic geometry for visualizing large hierarchies. In *CHI'95.* ACM Press, 1995.

[85] Eric Lecolinet. Multiple pointers : a study and an implementation. In *IHM 2003 : Proceedings of the 15th French-speaking conference on human-computer interaction on 15eme Conference Francophone sur l'Interaction Homme-Machine*, pages 134–141, New York, NY, USA, 2003. ACM Press.

[86] Y. K Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction*, pages p. 126–160., 1994.

[87] Wendy E. Mackay. Augmented reality : linking real and virtual worlds : a new paradigm for interacting with computers. In *AVI '98 : Proceedings of the working conference on Advanced visual interfaces*, pages 13–21. ACM Press, 1998.

[88] I. Scott MacKenzie and William Buxton. A tool for the rapid evaluation of input devices using fitts' law models. *SIGCHI Bull.*, 25(3) :58–63, 1993.

[89] Shahzad Malik, Abhishek Ranjan, and Ravin Balakrishnan. Interacting with large displays from a distance with vision-tracked multi-finger gestural input. In *UIST '05 : Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 43–52, New York, NY, USA, 2005. ACM Press.

[90] Microsoft. Surface. http://www.microsoft.com/surface/, 2007.

[91] Mitsubishi. Diamondtouch. http://www.merl.com/projects/DiamondTouch/, 2001.

[92] E.G. Noik. Layout-independent fisheye views of nested graphs. In *Visual Languages '93*, 1993.

[93] K. A. Olsen and al. Visualization of a document collection : The vibe system. *Information Processing & Management*, 29(1) :p. 69–81, 1993.

[94] Ken Perlin and David Fox. Pad : An alternative an alternative approach to the computer interface. In *Computer graphics Proceedings*, Annual Conference Series, pages 57–64. 1993.

[95] Catherine Plaisant. The challenge of information visualization evaluation. In *AVI '04 : Proceedings of the working conference on Advanced visual interfaces*, pages 109–116, New York, NY, USA, 2004. ACM Press.

[96] Jun Rekimoto. Pick-and-drop : a direct manipulation technique for multiple computer environments. In *UIST '97 : Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 31–39. ACM Press, 1997.

[97] George Reynolds. *Psychology today.* American psychology association, 1977.

[98] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone trees : Animated 3D visualizationz of hierarchical information. In *SIGCHI'91*, pages 189–194. ACM PRESS, April 1991.

[99] George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. The information visualizer, an information workspace. In *SIGCHI'91*, pages 181–188. ACM PRESS, April 1991.

[100] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval.* Mc Graw Hill, 1983.

[101] Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. In *CHI'92*, pages 83–91. Addison Wesley, May 1992.

[102] D. Schaffer. Navigating hierarchically clustered networks through fisheye and full-zoom methods. *ACM Transactions on Computer-Human-Interaction*, 3(2) :p. 162–188, 1996.

[103] Ben Shneiderman. Direct manipulation : A step beyond programming languages. *IEEE Computer*, 16(8) :57–69, August 1983.

[104] Ben Shneiderman. Tree visualization with tree-maps : 2-D space-filling approach. *acm Transactions on graphics*, 11(1) :92–99, January 1992.

[105] Norbert A. Streitz, Jörg Geissler, Torsten Holmer, Shin'ichi Konomi, Christian Müller-Tomfelde, Wolfgang Reischl, Petra Rexroth, Peter Seitz, and Ralf Steinmetz. i-land : an interactive landscape for creativity and innovation. In *CHI '99 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 120–127, New York, NY, USA, 1999. ACM Press.

[106] Wolfgang Stuerzlinger, Olivier Chapuis, Dusty Phillips, and Nicolas Roussel. User interface façades : towards fully adaptable user interfaces. In *UIST '06 : Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 309–318, New York, NY, USA, 2006. ACM Press.

[107] Kishore Swaminathan and Steve Sato. Interaction design for large displays. *interactions*, 4(1) :15–24, 1997.

[108] Desney S. Tan, Brian Meyers, and Mary Czerwinski. Wincuts : manipulating arbitrary window regions for more effective use of screen space. In *CHI '04 : CHI '04 extended abstracts on Human factors in computing systems*, pages 1525–1528, New York, NY, USA, 2004. ACM Press.

[109] E. R. Tufte. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, Connecticut, 1983.

[110] J. Walker. A node positioning algorithm for general trees. *Software Practice and Experience*, 20(7) :p 685–705, 1990.

[111] Colin Ware and Marlon Lewis. The dragmag image magnifier. In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 2 of *Videos*, pages 407–408. 1995.

[112] Steve J. Westerman and Timothy Cribbin. Mapping semantic information in virtual space : dimensions, variance and individual differences. *Int. J. Hum.-Comput. Stud.*, 53(5) :765–787, 2000.

[113] C. Williamson and B. Shneiderman. The dynamic homefinder : Evaluating dynamic queries in a real-estate information exploration system. In *SIGIR'92*, 1992.

[114] Andrew D. Wilson. Touchlight : an imaging touch screen and display for gesture-based interaction. In *SIGGRAPH '05 : ACM SIGGRAPH 2005 Emerging technologies*, page 25, New York, NY, USA, 2005. ACM Press.

[115] Beth Yost, Yonca Haciahmetoglu, and Chris North. Beyond visual acuity : the perceptual scalability of information visualizations for large displays. In *CHI '07 : Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 101–110, New York, NY, USA, 2007. ACM Press.

[116] Mountaz Zizi. *Cartes Dynamiques Interactives : une métaphore spatiale pour l'exploration des espaces informationnels.* PhD thesis, Université Paris-sud, Orsay, 1995.

[IJHCS'95] Mountaz Zizi and Michel Beaudouin-Lafon. Hypermedia exploration with interactive dynamic maps. *International Journal on Human Computer Studies*, 43 :441–464, 1995.

Chapitre 6

LISTE DE PUBLICATIONS

Revues internationales avec comité de sélection

[SPE'2008] M. Collomb and M Hascoët. an open and unified implementation model for drag-and-drop extensions to large and distributed display environments. *International Journal on Software, Practice and Experiences*, Submitted and accepted with major modification, Re-submitted july 2007. *International Journal on Software, Practice and Experiences*,

[IJHCS'95] Mountaz Zizi and Michel Beaudouin-Lafon. Hypermedia exploration with interactive dynamic maps. *International Journal on Human Computer Studies*, 43 :441–464, 1995.

Revues nationales avec comité de sélection

[RevueI3-2005] M. Hascoët, M. Collomb, and R. Blanch. Evolution du Drag-and-Drop : Du Modèle d'Interaction Classique aux Surfaces Multi-Supports. *Revue I3 : InformationInteractionIntelligence*, 4(2) :x, 2005.

[RevueI3-2001] M. Hascoët and M. Beaudouin-Lafon. Visualisation interactive d'information. *Revue I3*, 1(1) :77–108, 2001.

Conférences internationales avec comité de sélection

[Enactive'2006] Maxime Collomb and Mountaz Hascoët. Improving drag-and-drop on wall-size displays. In *ENACTIVE'06 : 3rd International Conference on Enactive Interfaces*, pages 115–116, Montpellier, France, 11 2006.

[Enactive'2006-2] Mountaz Hascoët, Maxime Collomb, and Jérôme Cance. Accelerating object-command transitions with pie menus. In *Enactive'06 : Conference on Enaction and Complexity*, pages 109–111, 2006.

[VDA'2006] Mountaz Hascoët, Francois Boutin, and Jérôme Thievre. Focus-based filtering + clustering technique for power-law networks with small world phenomenon. In *VDA'06 : Visual Data Analysis - SPIE-IS&T Electronic Imaging*, volume 6060, pages 001–012. SPIE P., U.S., 2006.

[GraphicsInterface'2005] M. Collomb, M. Hascoët, P. Baudisch, and B. Lee. Improving Drag-and-Drop on Wall-Size Displays. In *Graphics Interface'05*, Victoria (Canada), 2005.

[Interact'2005] F. Boutin, J. Thievre, and M. Hascoët. Multilevel Compound Tree Construction Visualization and Interaction. In *Interact'05*, Rome (Italie), 2005. Springer.

[HCI'2004] M. Collomb and M. Hascoët. Speed and Accuracy in Throwing Models for Large Displays. In *HCI'04 : 12th International Conference on Human Computer Interaction*, pages 21–24, Leeds (UK), 2004. British HCI Group.

[AVI'2004] F. Boutin and M. Hascoët. Focus Dependant Multi-Level Graph CLustering. In *AVI'04 : Advanced Visual Interface Conference*, pages 167–170, Gallipoli (Italy), 2004. ACM.

[ECDL'2004] F. Boutin and M. Hascoët. Multi-Level Exploration of Citation Graphs. In *ECDL'04 : European Conference on Digital Library*, pages 366–377, Bath (UK), 2004. Springer.

[IV'2004] F. Boutin and M. Hascoët. Cluster Validity Indices for Graph Partitioning. In *IV'04 : 8th IEEE International Conference on Information Visualization*, pages 376–381, London (UK), 2004. IEEE.

[IV'03] François Boutin and Mountaz Hascoët. Focus-based clustering for multiscale visualization. In *Information Visualization*, pages pp53–59, London, 2003. IEEE.

[HCI'2003] M. Hascoët. Throwing models for large displays. In British HCI Group, editor, *Human Computer Interaction, HCI'2003*, Bath, UK, 2003.

[IV'2002] Mountaz Hascoët and Frederic Sackx. Exploring interaction strategies with wall-screen : A new dual-display device for managing collections of web pages. In *Information Visualization 2002*, pages p. 719–725. IEEE, 2002.

[IV'2001] Mountaz Hascoët. Interaction and visualisation supporting web browsing patterns. In *Information Visualization, IV'2001*, pages 413–419, London, 2001. IEEE.

[Hypertext'00] Mountaz Hascoët. A user interface combining navigationl aids. In *ACM Hypertext'00*, San Antonio, 2000. ACM Press.

[SIGIR'98] Mountaz Hascoët and Xavier Soinard. Using maps as a user interface to a digital library. In proceedings of *SIGIR*. ACM, 1998.

[CHI'98] Mountaz Hascoët-Zizi. Analytical versus empirical evaluation of spatial displays. In *ACM CHI'98*, Los Angeles, 1998. ACM Press.

[DL'96] M. Hascoët-Zizi and N. Pediotakis. Visual relevance analysis. In *Digital Library'96*. ACM, 1996.

[UIST'96] Mountaz Hascoët-Zizi, Chris Ahlberg, Robert Korfhage, Catherine Plaisant, Matthew Chalmers, and Ramana Rao. Where is information technology going ? In *ACM Conference on User Interface Software Technology UIST'96*, 1996.

[ECHT'94] Mountaz Zizi and Michel Beaudouin-lafon. Accessing hyperdocuments through interactive dynamic maps. In *ACM Conference on Hypertext, ECHT'94*, Edinburgh, 1994. ACM Press.

Chapitres de Livre

[LivreHermes] Mountaz Hascoët. Visualisation d'Information et Interaction. In *Méthodes Avancées pour les Systèmes de Recherche d'Informations*. Hermès, 2004. Dir. M. Ihadjadene.

[KluwerBook96] Mountaz Zizi. Interactive dynamic maps for visualisation and retrieval from hypertext system. In A.Smeaton et A. Agosti, editor, *Information Retrieval and Hypermedia*. Kluwer Academics, 1996.

[Ecole'00] Mountaz Hascoët. Visualisation interactive de documents. In *Le Document Multimédia*, volume 1, pages 177–191. Cepaduès-Editions, Marseille, 2000.

Workshop internationaux avec comité de sélection

[CODATA'97] Mountaz Hascoët. Interactive dynamic maps. In *Codata Euro-American Workshop*. CODATA, 1997.

[FADIVA'96] Mountaz Hascoët. Visualisation of information spaces : evaluation issues. In *FADIVA*. TR 04-96 MAGGIO, Univ. DiRoma La Sapienza, 1996.

[IWHD'95] Mountaz Zizi. Providing maps to support the early stage of design of hypermerdia systems. In *Hypermedia design, Proceeding of IWHD'95*, pages 93–104, 1995. Springer-Verlag.

[BIWIT'94] Mountaz Zizi. Shadocs, the sharing documents system . dans, p. 15-26, france,. In *BIWIT, Information System Design and Hypermedia*. Pages 15–26. CEPADUES-Editions, 1994.

Colloques nationaux avec comité de sélection

[CORESA'2006] Maxime Collomb and Mountaz Hascoët. Affichages distribués et usage collaboratif. In *CORESA'06 : 11ième Conférence Nationale de COmpression et REprésentation des Signaux Audiovisuels*, page x, 11 2006.

[IHM'05] Maxime Collomb and Mountaz Hascoët. Drag-and-click : adapter les récentes améliorations du glisser-déposer pour l'activation à distance. In *IHM 2005 : Proceedings of the 17th conference on 17ème Conférence Francophone sur l'Interaction Homme-Machine*, New York, NY, USA, 2005. ACM Press.

[IHM'98] M. Hascoët, P. Janecek, C. Plaisant, C. Demarey, F. Foveau, and P. Plenacoste. Visualisation d'information. In *IHM'98*. Cépaduès-édition, 1998.

[IHM'93] Mountaz Zizi. Les cartes interactives dynamiques dans shadocs. In *Proc. Quatrièmes journées sur l'Ingénierie des Interfaces Homme-Machine (IHM*, pages pages 171–178, 1993.

Thèses

[TheseBoutin] François Boutin. *Filtrage, partitionnement et visualisation multi-échelles de graphes d'interactions à partir d'un focus.* PhD thesis, Université Montpellier II - Sciences et Techniques du Languedoc, 28/11/2005.

[TheseCollomb] Maxime Collomb. *Vers des systèmes de fenêtrage distribués : l'évolution du drag-and-drop.* PhD thesis, Université Montpellier II - Sciences et Techniques du Languedoc, 01/12/2006.

[TheseThievre] Jérôme Thièvre. *Cartographies pour la Recherche et l'Exploration de Données Documentaires.* PhD thesis, Université Montpellier II - Sciences et Techniques du Languedoc, 12/10/2006.

[TheseHascoet] Mountaz Zizi. *Cartes Dynamiques Interactives : une métaphore spatiale pour l'exploration des espaces informationnels.* PhD thesis, Université Paris-sud, Orsay, 1995.

Diffusion des connaissances

[PleinSud'2000] Mountaz Hascoët. L'université paris-sud en 70 sites. *Journal Plein Sud*, 2000.

[SIGLINK'94] Mountaz Zizi. Open systems, information structuring, and navigation : Echt94 trip report. *ACM SIGLINK Newsletter*, 1994.

[TSI'1994] Mountaz Zizi. Compte rendu de la conférence echt'94. *Technique et Science Informatique, TSI* , 1994.

Rapports internes et de fin de contrat et sites web de projets

[WEBASEVAL] Mountaz Hascoët. Site web de l'action spécifique sur l'évaluation.

[LRI1232'1999] Mountaz Zizi. Bookmap : a user interface supporting web navigation, technical report 1232. Technical report, Laboratoire de recherche en Informatique, Université Paris-sud, Orsay, 1999.

[CNET'1993] Mountaz Zizi. Filtrage et recherche de document, rapport technique 921b 178. Technical report, CNET France Telecom, 1993.

[LRI851'1993] Mountaz Zizi. Filtrage et recherche de document : le Rôle de l'interaction homme-machine technical report 851. Technical report, Laboratoire de recherche en Informatique, Université Paris-sud, Orsay, 1993.

Chapitre 7

PUBLICATIONS EN ANNEXE

[SPE'2008] M. Collomb and M Hascoët. an open and unified implementation model for drag-and-drop extensions to large and distributed display environments. *International Journal on Software, Practice and Experiences*, Submitted and accepted with major modification, Re-submitted july 2007. *International Journal on Software, Practice and Experiences*,

[IJHCS'95] Mountaz Zizi and Michel Beaudouin-Lafon. Hypermedia exploration with interactive dynamic maps. *International Journal on Human Computer Studies*, 43 :441–464, 1995.

[IV'2004] F. Boutin and M. Hascoët. Cluster Validity Indices for Graph Partitioning. In *IV'04 : 8th IEEE International Conference on Information Visualization*, pages 376–381, London (UK), 2004. IEEE.

[GraphicsInterface'2005] M. Collomb, M. Hascoët, P. Baudisch, and B. Lee. Improving Drag-and-Drop on Wall-Size Displays. In *Graphics Interface'05*, Victoria (Canada), 2005.

[Interact'2005] F. Boutin, J. Thievre, and M. Hascoët. Multilevel Compound Tree Construction Visualization and Interaction. In *Interact'05*, Rome (Italie), 2005. Springer.

SOFTWARE—PRACTICE AND EXPERIENCE
Softw. Pract. Exper. 2000; **00**:1–7 *Prepared using* speauth.cls *[Version: 2002/09/23 v2.2]*

A unified implementation model for drag-and-drop extensions to new emerging interactive environments

Maxime Collomb, Mountaz Hascoët

LIRMM - Univ. Montpellier II - CNRS
161 rue Ada, 34 392 Montpellier Cedex 5 - France.

SUMMARY

Drag-and-drop **is probably one of the mostly successful and generic representation of direct manipulation to be found in today's WIMP interfaces. At the same time, emerging new interactive environments such as distributed display environments or large display surface environments have shown the need for an evolution of *drag-and-drop* in order to address new challenges. In this context, several evolutions of *drag-and-drop* have been proposed over the past several years. However, implementations for these extensions are difficult to reproduce, integrate and extend. As a consequence, developing or integrating advanced *drag-and-drop* techniques in applications is still requiring more time and effort.**

The aim of this paper is to propose a unifying implementation model of *drag-and-drop* and of its extensions. This model aims at facilitating the implementation of advanced *drag-and-drop* support by offering solutions to problems typical of new emerging environments. The model builds upon a synthesis of *drag-and-drop* implementations, an analysis of requirements for meeting new challenges and a dedicated interaction model based on instrumental interaction. By using this model, a programmer will be able to implement advanced *drag-and-drop* supporting (1) multi-display environments, (2) large display surfaces and (3) multi-user systems. Furthermore by unifying the implementation of all existing *drag-and-drop* approaches, this model also provides flexibility by allowing users (or applications) to select the most appropriate *drag-and-drop* technique depending on the context of use. For example, a user might prefer to use *pick-and-drop* when interacting with multiple displays attached to multiple computers, *push-and-throw* or *drag-and-throw* when interacting with large displays and possibly standard *drag-and-drop* in more traditional context. Finally, in order to illustrate the various benefits of this model we provide a Java based implementation of the model that can be used with most Java based applications.

KEY WORDS: distributed display environment, augmented surfaces, drag-and-drop, interaction model, Java API

Contract/grant sponsor: Publishing Arts Research Council; contract/grant number: 98–1846389

1. INTRODUCTION

Drag-and-drop is probably one of the mostly successful and generic representation of direct manipulation to be found in today's WIMP interfaces. At the same time, emerging new interactive environments such as distributed display environments or large display surface environments have shown the need for an evolution of *drag-and-drop* in order to address new challenges. In this context, several evolutions of *drag-and-drop* have been proposed over the past several years. However, implementations for these extensions are difficult to reproduce, integrate and extend.

It is striking to see that from an implementation perspective, support for even traditional *drag-and-drop* operations, varies widely from one windowing system to another or from one toolkit to another [7]. It is not surprising then that the situation be worse for extensions of *drag-and-drop*: very little support if any is usually provided for *drag-and-drop* extensions that are usually implemented as prototypes or by ad-hoc solutions that can hardly be generalized nor reused as new needs arise.

As the number of *drag-and-drop* extensions is increasing it becomes important to propose a unified framework that clarifies the field and offers benefits from at least three perspectives: user perspective, design perspective and implementation perspective. From the user perspective such a unification will hopefully make it possible for users to choose the type of *drag-and-drop* they like best or that suits best some specific environment. From a designer perspective a unifying framework will help better understand the differences between the possible techniques and the design dimensions at stake. Lastly, from a programmer's perspective, a unifying implementation model should save a lot of time and efforts in the development of different extensions of *drag-and-drop* in new emerging and challenging environments such as large displays and distributed display environments. The aim of this paper is to propose the basis for building such a framework including a unified and open implementation model.

In the following section we discuss how emerging interactive environments bring new challenges for *drag-and-drop* paradigm, we review most extensions proposed so far and we propose a unified framework for comparing them. In the next section we present a new implementation model that not only builds upon analysis of requirements for adapting to new emerging interactive environments but also accounts for implementation models of existing solutions in most widespread windowing systems or toolkits [7]. This model is based on the definition of instruments that embody interaction technique. The presentation of the implementation model at a generic level is based on a generic type of instrument that embody basic drag-and-drop technique. We further discuss how specific emerging extensions of drag-and-drop can implemented as specific *instruments* that are integrated into the model smoothly.

2. NEW CHALLENGES FOR DRAG-AND-DROP PARADIGM

Most challenges that *drag-and-drop* has faced recently can be attributed to the emergence of new display environments such as wall-size displays or distributed display environments (DDE). DDE have been defined in [16] as

"Computer systems that present output to more than one physical display."

This general definition covers a broad range of systems. As a consequence, systems that can be studied in this field can exhibit huge differences. An example of such a huge difference is the difference between two typical types of DDE: (1) a configuration where multiple displays are attached to the same machine and (2) a configuration where multiple displays are attached to different machines. In both cases, it results in multiple displays but the degree of integration of the distinct displays is very different. In the first case the different displays are handled within the same windowing system, offering a unique workspace or desktop with full communication capacities between the windows of the different displays. In the second case, on the contrary, the two displays are handled by two distinct and potentially heterogeneous windowing systems making it much more difficult to offer similar unique workspace spreading over the different displays. Hence these two different configurations leads to different types of problems when it comes to the implementation of *drag-and-drop* in these environments.

2.1. Preliminary definitions

In order to better characterize the types of problems that are mostly challenging for *drag-and-drop* and its evolutions, preliminary definitions are useful. In this paper, we use a generally agreed definition of the term *display* : a physical device used to display information. Since we are interested in *drag-and-drop* we will always refer to displays associated to input devices therefore we consider that a display also allows input through its associated input devices. The term *window* is used to refer to an area of a display devoted to handle input and output from various programs. Based on these definitions, we define a *surface* as a set of windows that potentially appear on different displays attached to different machines. We further define the term *shared surface* as a *surface* supporting interactions between its windows in the same way as if they were part of the same workspace. For example, a set of windows spread over three different displays handled by three different machines such as, a laptop running MacOS, a PC running X-Windows and a tablet PC running Windows can be considered as a shared surface as soon as a system makes it possible to surpass the boundaries of each windowing system to support some interactions between displays. It is important to stress that systems handling shared surface environments are de-facto potentially multi-user systems. Indeed each machine involved in the surface can be controlled by a distinct user. To some extent they can be considered as a specific type of groupware environment.

In general, we use shared surface to describe a specific case of DDE where a surface integrates multiple displays, multiple users and output and input from multiple machines possibly associated with heterogeneous windowing systems. Support for shared surface environments is still at its very early stage and existing windowing systems boundaries are still difficult to surpass. The aim of our framework is to propose solutions for typical problems that can't be ignored when designing and implementing a drag-a-drop like interaction in a shared surface environment. The problems arising can be structured into three categories: (1) usability and scalability problems, (2) multi-computer and interoperability problems and (3) multi-user and concurrency problems.

2.2. Usability and scalability issues

Usability and scalability problems derive from the complexity of new environments and of their usage: heterogeneity of displays both in terms of size, number and nature, heterogeneity of users in terms of abilities, experience and style. The problem of scalability of *drag-and-drop* in these environments starts with the question: to what extent a technique originally designed to work on one unique and relatively low resolution display will adapt to increasing number, size and resolution of displays? When displays become large and are used with direct pointing devices for example as is the case in environments such as the iRoom [26] or DynaWall [9], *drag-and-drop* paradigm shows its limits [4, 6]. Indeed, performances decrease significantly as the size of displays increases: accessing icons located far away from the user may require users to physically walk over, requiring a target acquisition time that increases with distance [10].

Furthermore, interactions that involve dragging objects tend to be particularly tedious and error-prone [4, 6] and can be complicated further by the bezels separating screen units [1]. Lastly, in some contexts of use like large wall sized display with touch/pen input, *drag-and-drop* might fail when targets are out of reach e.g. located too high or too low or on a display.

In order to adapt to heterogeneous displays from very small displays up to large wall-size displays *drag-and-drop* basic paradigm has been extended with new interaction styles. Most of these extensions address specific needs for particular display environments. As a consequence, we now have interesting alternatives to *drag-and-drop* original style that best suits best some contexts. These alternatives will be discussed in section 2. The aim of our implementation is to support most styles in a unified implementation model so that shifting from one interaction style to another is facilitated. This feature can be seen as a particular support for plasticity: "the capacity of a user interface to withstand variations of both the system physical characteristics and the environment while preserving usability" [27].

2.3. Shared surfaces issues: integrating multiple computers and multiple windowing systems transparently

Multi-computer and multi-windowing systems problems are probably the most challenging problems that one has to address to handle shared surface properly. Because of the difficulty in surpassing the boundaries of windowing system associated with each display, there is still very little support for making windows of shared surface behave as if they were part of the same workspace. Some systems [23, 17, 18] aim to support communication between the displays based on redirection of input/output mechanisms but support is still at its early stage.

Other approaches like those found in distributed visualization environments provide multi-head support for multiple displays attached to different machines [28, 15]. These system provide advanced support for shared surfaces. However, they do not have the flexibility needed to handle heterogeneous nor dynamic shared surfaces. Indeed, they impose strict constraints on the architecture of the cluster of machines used and on the windowing systems or graphic toolkit that is run by these machines. Clearly they are not aimed at handling evolving sets of machines running heterogeneous windowing systems and graphical toolkits. Our model on the contrary, impose no particular constraints on the machines involved in the shared surface an

it also support evolving configurations so that windows from new machines can be added or removed from a shared surface dynamically.

2.4. Multi-user issues

Amongst all problems brought by new display environments, multi-user problems are probably the oldest. Most of these problems have been studied in the field of computer-supported computer supported collaborative work (CSCW) over the past decades. As far as *drag-and-drop* is concerned, the important aspect of multi-user support is that one is able to distinguish events streams from different users. By fulfilling this requirement, our model can also be used by a programmer willing to integrate advanced *drag-and-drop* features in a CSCW context.

3. DRAG-AND-DROP EVOLUTION

The problems listed in the previous section are partially addressed by a set of *drag-and-drop* extensions that have been proposed over the past 10 years. Table I) shows what types of problem these extensions primarily address. However, it is important to stress that these extensions have been implemented in ad-hoc ways for demonstration. We further review and compare these extensions in more detail. The implementation model we propose in further sections will provide a unified framework to support them all.

technique	**addressed limitations**
pick-and-drop [20]	shared surface, multi-user
hyperdragging [21]	shared surface, multi-user
stitching [14]	shared surface
throwing [11]	scalability
drag-and-pop [1], vacuum [3]	scalability, usability
drag-and-throw, push-and-throw [13]	scalability, usability
push-and-pop [6]	scalability, usability

Table I. Limitations originally addressed by *drag-and-drop* extensions.

3.1. Pick-and-drop

Pick-and-drop [20] has been developed to allow users to extend *drag-and-drop* to distributed environments. While *drag-and-drop* requires the user to remain on the same computer while dragging objects around, *pick-and-drop* let him move objects from one computer to another using direct manipulation. This is done by giving the user the impression of physically taking an object on a surface and laying it on another surface.

Pick-and-drop is closer to the *copy-paste* interaction technique than to *drag-and-drop*. Indeed like the copy/paste operation, it requires two different steps: one to select the object to transfer, and one to put the object somewhere else. But *pick-and-drop* and *drag-and-drop* share a common advantage over *copy-paste* techniques: they avoid the user having to deal with a hidden clipboard.

However, *pick-and-drop* is limited to interactive surfaces which accept the same type of touch-pen devices and which are part of the same network. Each pen has a unique ID and data is associated with this unique ID and stored on a *pick-and-drop* server.

3.2. Hyperdragging

Hyperdragging [21] (figure 1–a) is part of a computer augmented environment. It helps users smoothly interchange digital information among their laptops, table or wall displays, or other physical objects.

Using *hyperdragging* is transparent to the user: when the pointer reaches the border of a given display surface, it is sent to the closest shared surface. Hence, the user can continue its movement as if there was only one computer. To avoid confusion due to multiple simultaneous *hyperdragging*, the remote pointer is visually linked to the computer controlling the pointer (simply by drawing a line on the workspace).

3.3. Stitching

Stitching [14] (figure 1–b) is an interaction technique designed for pen-operated mobile devices that allow it to start a *drag-and-drop* gesture on a screen and to end the gesture on another screen. Devices have to support networking.

A user starts dragging an object on the source screen, reaches its border, then crosses the bezel and finishes the *drag-and-drop* on the target screen. The two parts of the strokes are synchronized at the end of the operation then bound devices are able to transfer data.

3.4. Shuffle, Throw or Take it

Geißler [11, 24] proposed three techniques to work more efficiently on interactive walls. The goal was to limit the physical displacement of the user on a 4.5 x 1.1 meters triple display (the DynaWall [9]).

The first technique is *shuffling*. It is a way of re-arranging objects within a medium-sized area. Objects move by one length of their dimensions in a direction given by a short stroke of users on the appropriate widget.

Next, the author proposes a *throwing* technique. To throw an object, the user has to achieve a short stroke in the opposite direction that the object should be moving, followed by a longer stroke in the correct direction. Length ratio between the two strokes determines the distance to which the object will be thrown. According to the author, this technique requires training to be used in an efficient way.

The third technique, *taking*, is an application of the previously described *pick-and-drop* to the DynaWall.

3.5. Drag-and-pop and vacuum

Drag-and-pop [1] is intended to help *drag-and-drop* operations when the target is impossible or hard to reach, e.g., because it is located behind a bezel or far away from the user.

The principle of *drag-and-pop* is to detect the beginning of a *drag-and-drop* and to move potential targets toward the user's current pointer location. Thus, the user can interact with these icons using small movements.

As an example, the case of putting a file in the recycle bin, the user starts the drag gesture toward the recycle bin (fig. 1–c). After a few pixels, each valid target on the drag motion direction creates a linked tip icon that approaches the dragged object. Users can then drop the object on a tip icon. When the operation is complete, tip icons and rubber bands disappear.

If the initial drag gesture has not the right direction and thus the target icon is not part of the tip icons set, tip icons can be cleared by moving the pointer away from them but the whole operation has to be restarted to get a new set of tip icons.

The vacuum [3] (figure 1–d), a variant of *drag-and-pop*, is a circular widget with a user controllable arc of influence that is centred at the widget's point of invocation and spans out to the edges of the display. Far away objects standing inside this influence arc are brought closer to the widget's centre in the form of proxies that can be manipulated in lieu of the original.

3.6. Drag-and-Throw & Push-and-Throw

Drag-and-throw and *push-and-throw* [13, 4] are throwing techniques designed for multiple displays (one or more computers). They address the limitation of throwing techniques [11, 24] providing users with a real-time preview of where the dragged object will come down if thrown. These techniques are based on visual feedbacks, metaphors and the explicit definition of trajectories (fig. 1–e). Three types of visual feedback are used: trajectory, target and take-off area (area that matches to the complete display).

Drag-and-throw and *push-and-throw* have different trajectories: *drag-and-throw* uses the archery metaphor (user performs a reversed gesture - to throw an object on the right, pointer has to be moved to the left) while *push-and-pop* uses the pantograph metaphor (user's movements are amplified).

The main strength of these techniques is to visualize and control the trajectory of the object before actually sending the object. So users can adjust their gesture before validating it. Therefore, contrary to other throwing techniques, *drag-and-throw* and *push-and-throw* have very low error rates [4].

3.7. Push-and-Pop

Push-and-pop [6] was created to combine the strengths of *drag-and-pop* and *push-and-throw* techniques. It uses the take-off area feedback from *push-and-throw* while optimizing the use of this area (fig. 1–f): it contains full-size tip icons for each valid target. The notion of valid target and the grid-like arrangement of tip icons are directly inherited from the *drag-and-pop*'s layout algorithm. The advantage over *drag-and-pop* is that it eliminates the risk of invoking a wrong

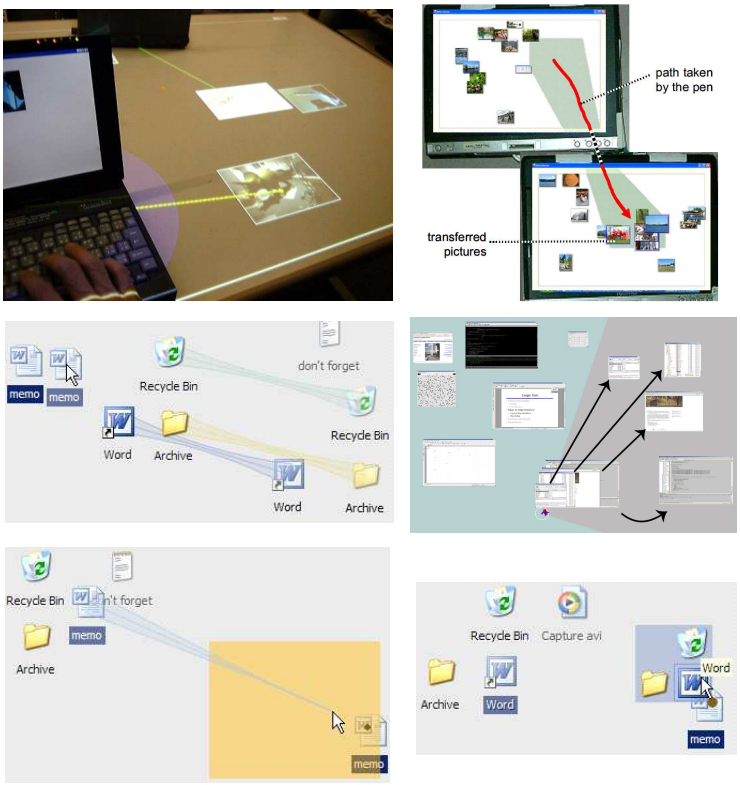

Figure 1. (Left to right, top to bottom) Examples of (a) *hyperdragging*, (b) *stitching*, (c) *drag-and-pop*, (d) *push-and-throw*, (e) *vacuum* (black arrows are added) and (f) *push-and-pop*. (reproductions with authors permission)

set of targets. And the advantage over *push-and-throw* is that it offers a better readability (icons are part of the take-off area), target acquisition is easier [6] and users can focus on the take-off area.

All these extensions have been developed individually as prototypes with *ad-hoc* event models. To represent them in a unified way, we propose to use the instrumental interaction model.

4. COMPARISON OF *DRAG-AND-DROP* EXTENSIONS

The instrumental interaction model [2] can be used to facilitate the comparison of *drag-and-drop* extensions. In this model, interaction is described through instruments. An *instrument* can be considered as a mediator between the user and domain objects. The user acts on

the instrument, which transforms the user's *actions* into *commands* affecting relevant target objects. Instruments have *reactions* enabling users to control their *actions* on the instrument, and provide *feedback* as the command is carried out on target objects (see figure 2).

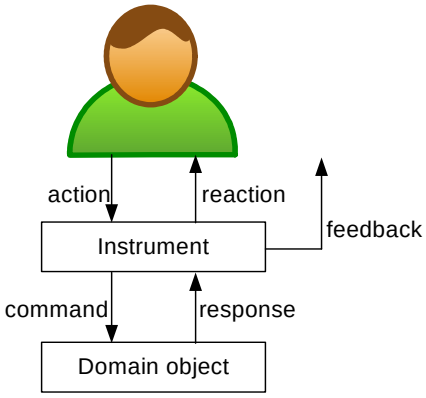

Figure 2. Interaction instrument mediates the interaction between user and domain objects [2].

Different extensions of *drag-and-drop* can be embodied through different instruments. In this context, the interactions between all instruments and domain objects (commands/responses) are all the same for all *drag-and-drop* like instruments. Indeed, all instruments support typical commands associated to *drag-and-drop*: source selection, target selection, specification of type of action, data transfer (validation of the selected target) and cancellation. The most important part of *drag-and-drop* typical interaction is described in the interactions between the user and the instrument (principally reactions and feedback). Reactions and feedback of instruments can be decomposed into three types of feedback: drag-under feedback, drag-over feedback and instrument reaction. When a user needs to change from one instrument to another one, drag-under and drag-over visual effects might roughly be preserved, but instrument reactions will vary significantly.

4.1. Drag-under and drag-over feedback

For regular *drag-and-drop* operations, feedback is usually referred to as *drag-under feedback* and *drag-over feedback*.

Drag-over feedback consists mainly in feedback that occurs on a source object. Typically, during a regular drag on a source, the pointer shape changes into a drag icon or ghost that represents the data being dragged. This icon can change during a drag to indicate the current action (copy/move/alias). Hence drag-over feedback mainly consists of shape and color of source ghost changes when the user changes the type of action, or when drop becomes possible / impossible. Some windowing systems may go a step beyond by providing animation: for example to indicate that the action was canceled, they may animate ghosts back to their

original location. It is interesting to note that even if the *drag-and-drop* model is mature, not all windowing systems offer this feature. When no animation is provided, it is significantly more difficult for the user to follow the effect of cancel operation.

Drag-under visual effects represent feedback provided on the target side. It conveys information when a potential target has a drag icon passing through it. The target can respond in many ways: by modifying its shape and color or even in more sophisticated cases by performing actions. For example, when moving a file over a set of folders, when the file remains above a particular folder a sufficient amount of time, the folder might open up to let the user recursively explore the file hierarchy to the desired target folder.

If drag-over and drag-under visual effects are sufficient to describe feedback in the case of regular *drag-and-drop* operation, they are not for most of its recent extensions. In the latter case, more feedback is needed. This additional feedback will be referred to as *instrument reaction* and will be described in the next section.

4.2. Instruments reaction

The reactions provided by *drag-and-drop* instruments consist mostly in *instrument feedback* e.g. specific feedback provided by the instrument so that users can better control their *actions*. Instruments reaction or feedback can be considered as a specific type of recognition feedback. As it was suggested by [19], "Recognition feedback allows users to adapt to the noise, error, and miss-recognition found in all recognizer-based interactions". Such feedback includes for example, the rubber bands that are used in the case of *drag-and-pop* to help user with locating/identifying potential targets. Another example is the case of throwing, where take-off areas as well as trajectories are displayed to help user adjust target selection, etc. Such feedback is used in other extensions and vary significantly from one particular instrument to another.

4.2.1. Different visual effects

Firstly, instrument feedback varies in terms of visual effects. *Hyperdragging* adds a trajectory feedback that links the remote pointer to its original display. *Throwing initial technique* does not provide any instrument feedback. This is probably the critical point that yields high error rates and low precision. *Drag-and-throw* and *push-and-throw* introduce two different instrument feedbacks: the take-off area and the trajectory. *Accelerated push-and-throw* can not offer the take-off area feedback so only trajectory is used. *Drag-and-pop* offer two kinds of instrument feedbacks: the tip icons and the rubber bands that link the tip icons to the actual target they represent. Finally, *Push-and-pop* combines strengths of *push-and-throw* and *drag-and-pop*. Therefore, it uses the take-off area **and** the tip icons feedbacks.

4.2.2. Different focuses: source-oriented, target-oriented and undirected instruments

Secondly, instrument feedback varies in terms of focus, e.g. the area of the surface where most visual effects occur. We define as *source-oriented instruments* the instruments where the focus remains around the source object original location. On the contrary, with *Target-oriented*

instruments the focus is located around the potential targets locations. *Drag-and-throw*, *push-and-throw* or *accelerated push-and-throw*, are typical example of target oriented instruments. As the user acts on the instrument, the instrument moves the source all the way to the target and further feedback occurs around the target location. In contrast, *drag-and-pop* and *push-and-pop* are good examples of source-oriented instruments: such instruments move all potential targets as ghosts around the source object and further feedback occurs around the original source location.

In most cases, with target-oriented instruments, users have to monitor the instrument to adjust their move around potential target locations to finally acquire the right target at its real location. This continuous adjustments on the instrument have strong implications on the systems. Implementation of the reactions must support high refresh rates : a little lag between the user hand movement and the reaction/feedback provided by the instrument would significantly decreases the usability of such instruments. On the contrary, source-oriented instruments usually involve a single fairly dramatic movement from the user and no constant monitoring as in the previous case. However, in some cases such as, for example, in push-and-pop, different drag directions cause the tip cluster to be a little different every time, users might need to re-orient once or twice to identify the correct tip icons. For such instruments, additional feedback is usually provided to help in identifying the real location of the objects that correspond to the ghost. In *drag-and-pop*, the rubber bands have been designed to help users in that task.

Furthermore, to be effective, target oriented instruments should be used with displays where targets are all roughly equally within sight. In very large wall-size displays it might be difficult to distinguish very well targets very far away from source location. In such environments, target oriented instruments would fail and source oriented instruments would be more suitable.

Lastly, some instruments are neither source-oriented nor target-oriented. We call these instruments *undirected instruments* since feedback will not be concentrated in one particular area of the display. *Drag-and-drop* instruments and *pick-and-drop* instrument are the two main examples of this category of instruments. They provide no particular reaction of the instrument, the only feedback associated to such instruments occurs through drag-under and drag-over feedback.

4.3. Instruments coverage

All instruments described above do not support full coverage. By coverage, we mean: areas of a surface where an instrument can drop an object. Some instruments (mainly source oriented instruments) have limited coverage.

For example, with *push-and-pop*, only potential targets can be reached. With such instruments, source objects can not be dropped to any other area of the surface. There are many contexts of *drag-and-drop* situations where no specific target is aimed and where such instrument would fail so it is important to consider this issue. Limited coverage is mostly typical of source oriented instruments even though there is no strong reason that prevents the development of a source oriented instrument with full coverage. This is probably a portion of the design space yet to be explored.

For undirected instruments, such as *drag-and-drop* or *pick-and-drop*, coverage will be larger but still limited to areas that can be physically reached by the user. In most situations, this will result in full coverage as all areas of a given surface can be reached. However, in a wall-size displays with touch/pen input coverage might be partial since some areas might be out of reach (situated too high for example).

5. IMPLEMENTATION MODEL AND POIP API

In this section we propose an implementation model that addresses the different issues discussed in section 2. This model also takes into account the pros and cons of the recent extensions discussed in previous section and offer a unified framework, that makes it possible to implement them all and shift from one interaction style to another on the fly.

In order to illustrate this model we provide a Java-based implementation named *PoIP*. *PoIP* is built as an API (application programming interface) that supports drag-an-drop manipulations in different environments. Hence it can be further used in the development of most Java application. *PoIP* is implemented in Java 5.0 and is based on some Java features. For example, RMI (Remote Method Invocation) is used for network communication and the events are managed by AWT. Even though we used *PoIP* to illustrate our model and to provide more details when useful, we believe that our model is general enough to be implemented by others in other languages or at other levels. *PoIP* is available for download with an example of use [5].

The implementation model described here relies on several key elements from most basic elements (drag-and-drop manager, source and object components and appropriate event model) to more advanced elements (instruments, shared surface management and input stream management). In the next section we first introduce basic elements on which the model further builds. We further introduce more advanced key elements of the models and show how the main issues discussed in section 2 are addressed.

5.1. Basic Elements of the Model

5.1.1. Drag-and-drop manager

The drag-and-drop manager plays a central part in the model. It is specified and implemented through the class *DmManager* that manages the registration process for source and target components. It is further used to coordinate the main entities of the model, therefore the *DmManager* interacts with:

- The associated shared window: interactions with the associated shared window occur when receiving input events, when displaying visual feedbacks and when initializing connections with remote shared windows and their *DmManager*s. Shared window management is described in more details in section 5.3.1.
- Instruments: the *DmManager* handles a set of instruments. One master instrument and a variable number of slave instruments. It redirects input event to the master instrument

and it dispatches repaint order to all instruments (master and slaves). Both master and slave instruments are described in more details in section 5.2.

- Source and target components: the *DmManager* maintains a list of registered source and target components. This list is mainly used by instruments.
- Remote *DmManager*: When a master instrument asks its *DmManager* for the creation or the destruction of slave instruments, the *DmManager* redirects this query to all other remote *DmManagers*.

The *DmManager* is the unique interface to a set of instruments. Indeed, a master instrument can change depending of the context, and slave instruments are created and destroyed upon users activities. The *DmManager* provide a static interface for this set of instruments and therefore simplifies the communication between the different entities: shared windows, slave instruments, master instruments, source components and target components. Fig. 3 summarizes the interfaces and abstract classes that constitute the core entities of the model. It also shows their interactions.

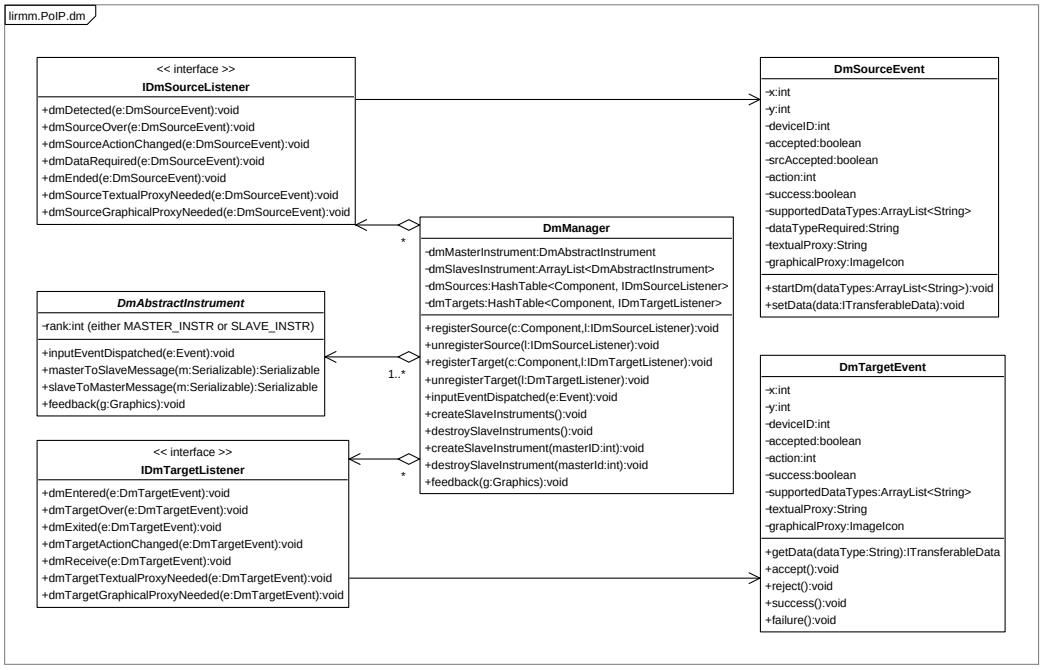

Figure 3. Class diagram of *drag-and-drop* model.

5.1.2. Source component, Target component and Event Model

Events are at the origin of all types of feedback and actions. Events and events handlers are specified through four interfaces in *PoIP*. As depicted by figure 3, two interfaces (*DmTargetListener* and *DmSourceEvent*) are introduced to specify event types and two interfaces (*DmSourceListener* and *DmSourceListener*) are introduced to specify event processing. Event processing interfaces have to be implemented respectively by the source and the target components.

A *source component* is the component from which the *drag-and-drop* operation begins. It provides the data to transfer. A *target component* is a component capable of receiving data from a *drag-and-drop* operation. During a *drag-and-drop*, the target component is the component under the pointer and it can change while the pointer moves. Note that, for a given operation, the source and the target can be the same component. In PoIP, *DmSourceListener* interface defines all the methods needed by the source component to process events. The *DmTargetListener* interface has to be implemented by a component that wants to be a potential target for *drag-and-drop* like techniques.

5.2. Multi-instrument support and genericity

Our approach to usability and scalability problems mentioned in previous section (see 2) consists in providing a unified implementation model that makes it possible to handle drag-and-drop like operation through different extensions depending on the context. Since different contexts exhibit different requirements it seems very important to facilitate the implementation of different drag-and-drop styles and to leave the choice of which one to use to the designer, the application, the context or the user.

Even though the instrumental interaction [2] model was primarily devoted to describe interaction, some aspects of the model are well suited to structure implementation. In particular, to support various types of drag-and-drop extensions, instruments are very useful to embody drag-and-drop like interaction techniques. Different types of instruments embodying different types of techniques.

Hence our approach consists in proposing a multi-instrument model – i.e. a model which meets the following requirements:

- Instruments act upon objects transparently: objects are notified of operations when they occur but they are not aware of the type of instrument in use so that the effort needed to introduce new instruments is minimal.
- Each user can choose the instrument he wants to use, depending on his preferences and on the physical constraints (touch display, large display, small display). The choice of the instrument can be part of user profile but it can also be made on the fly to adapt to an evolving context.
- Several users can manipulate objects at the same time with different instruments.

5.2.1. *Generic instruments*

Practically, instruments are defined through a hierarchy of classes all inheriting from a very generic instrument class in the same way that in most UI toolkits all widgets or graphical components usually inherit from a generic window class. In PoIP, generic instruments are specified through an abstract class *DmAbstractInstrument* (see figure 3) that encapsulates the instrument behavior.

An instrument can be decomposed into two parts: a master instrument and slave instruments. A given *DmManager* handles one and only one master instrument and a variable number of slave instruments. Slave instruments were introduced mainly for multi-computer purposes. However, for consistency purposes they are also used in a single computer environment. A master instrument is mainly devoted to dispatching input events and handling associated slave instruments, while slave instruments do the real job, e.g. find which component is at a given location or perform adequate feedback. Master instrument asks for creation of slave instruments when a *drag-and-drop* like manipulation is detected and asks for their destruction at the end of the manipulation. Thus, a master instrument handles n slave instruments during manipulation where n is the total number of shared windows in the shared surface.

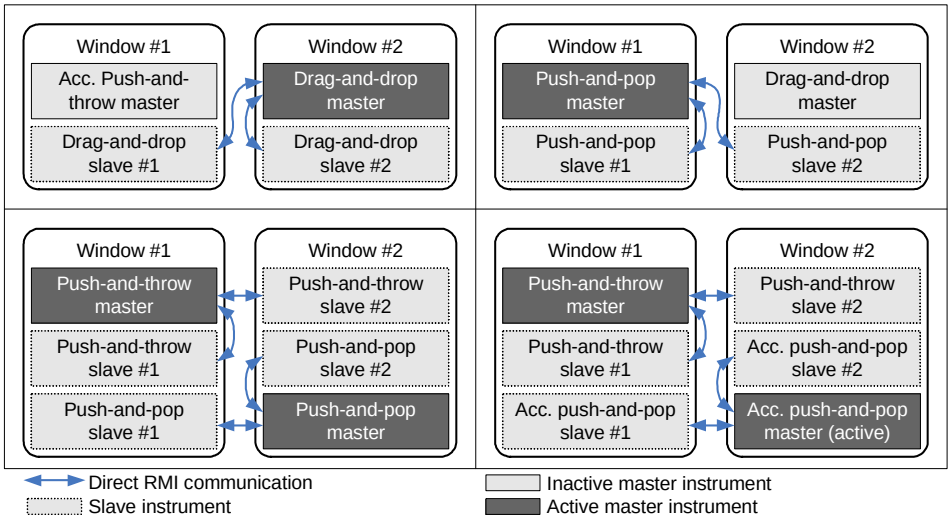

Figure 4. Master/slave communication corresponding to the four screenshots of figure 8.

The figure 4 presents the communications between master and slave instruments in four different contexts corresponding to the four screenshots of figure 8.

For the example C (bottom left example of figure 8 and 4), a shared surface spreads over two computers (A and B). A *push-and-throw* instrument is used on computer A while a *push-and-pop* instrument is in use on computer B. In this example, *DmManagers* running on A

and B respectively handle a master *push-and-throw* instrument and a master *push-and-pop* instrument. When the user of computer A starts a *push-and-throw*, the *DmManager* of A will create an additional slave *push-and-throw* instrument on A and ask *DmManager* of B to create another slave *push-and-throw* on B. If, at the same time, a user of B starts a *push-and-pop* on B then a slave *push-and-pop* instrument will be added to both *DmManager* of A and *DmManager* of B. As a consequence, in this particular case, *DmManagers* handle three instruments each: one master and two slaves per *DmManager*. This example also illustrates how several types of instruments can be used without any problem. More generally, if more windows appear on a shared surface, each *DmManager* owns exactly one master instrument and as many slave instruments as *drag-and-drop* like manipulations are running. Another example of master/slave instruments in use can be found in section 6.2.

5.3. Multi-computer support and interoperability

In order to address multi-computer issues discussed in section 2, one preliminary requirement is to support some sort of interoperability between windowing systems. In our model, interoperability is based on (1) implementation of shared surfaces and (2) slave instruments described in the previous section. We now describe the shared surface implementation.

As defined previously, a shared surface is a set of windows possibly displayed by different windowing systems and behaving as if they were part of the same workspace. In particular, a drag-and-drop interaction can transparently start with one window of the shared surface handled in one windowing system and ends on another window operated on another windowing system.

5.3.1. Shared windows

Until now we used the term window in a general way. We now need to refine the concept and introduce the term *shared window* to provide more details on implementation. A shared window is used to make it possible for a given common window or graphic component to be part of a shared surface. A shared window and its associated window are so tightly coupled that we will often use the term window to refer to both. A shared window owns a name and a unique ID. Shared windows act on their associate windows or components in two ways: they capture and redirect input events received in the associated windows and render visual feedback on top of them.

In *PoIP*, shared windows are implemented through *LayeredPane*. The *LayeredPane* is used to display the fake pointers. It is also made available for instruments to implement different types of feedbacks. Note that drag-under feedbacks are mostly managed by targets and will not make extensive use of the *LayeredPane*. On the contrary, the *LayeredPane* will be very helpful to instruments to perform drag-over feedback and instrument feedback. The layered pane is located on top of all other layered panes. So fake pointers are always displayed on top of the workspace.

5.3.2. Shared surface management: client-server architecture and topology management

A shared surface is built by registering each shared window on a server named shared surface server. The server is useful to establish connections between the different shared windows independently of their associated windowing system. This client-server architecture is used to make transparent the boundaries of windowing systems. Hence, shared windows make a continuous workspace using a given topology. This workspace can be distributed between multiple computers, used by multiple users, each one with different input devices (e.g. multi-computer, multi-user).

The shared surface server is used to :

- manage windows IDs. An ID identifies both windows and associated input devices. An ID is assigned to a window when the window is registered on the server.
- maintain a list of shared windows and ensure that each time a shared window registers or unregisters all other windows are notified.
- handle the topology of windows within the surface according to a topology manager (detailed below).

Default topology management is based on a 2D description of shared windows contained in the surface. Performing more sophisticated topology management would certainly be useful in general. However, since it would go beyond the scope of this paper to address this issue completely, we provided a rather simple but modular topology management. So that eventually, integrating more sophisticated topology management when useful is not an issue. Figure 5 is an example of the topology described in our model where three windows are shared. In this example, when the pointer reaches the right border of window 1 it calls the topology manager to determine where it should go next. In *PoIP*, the method *getContiguousWindow(sw: ISharedWindow, x: int, y: int)* is called. The *sw* parameter is the window on which was the pointer, and *x* and *y* represent the location of the pointer relatively to the origin of this window – note that *x* and *y* are out of the bounds of the window. In this example, the result of the method will be window 3. Then the method *teleportPointer(r1: Rectangle, r2: Rectangle, x: int, y: int)* is invoked. Rectangles *r1* and *r2* represent respectively the bounds of the windows 1 and 3 while *x* and *y* represent the location of the pointer relatively to the origin of window 1. The result of this method call will be the new location of the pointer on window 3, relatively to the origin – upper left corner – of window 3.

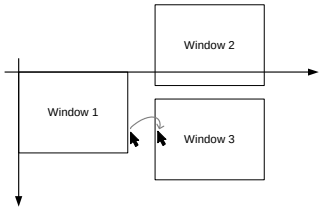

Figure 5. An example of windows topology.

5.3.2.1. Events tagging and redirecting *Shared Windows* process input events in two steps: first, events are captured and second, they are augmented by the ID of window where the event occurred originally, applied to a fake pointer and redirected. Note that these events are eventually reproduced on a remote shared window. Tagging events with IDs is necessary to let several user interact simultaneously.

In *PoIP*, event capture is handled through a *AWTEventListener* attached to the default toolkit. This implies that all the events of the Java Vitual Machine are captured and processed. It is satisfactory to share windows across different computers but it might need more work to share different pointing devices of the same machine. This implementation makes the hypothesis that when *several* pointing devices are used on the *same* computer (for example, a touchpad and a mouse on a laptop), input events are not distinguished. Whether they come from the touchpad or the mouse, they are treated in a unique input stream identified by the ID of the window they were first associated to. In contrast, when input streams are attached to different windows to begin with, they will further be considered as distinct input streams.

5.4. Multi-user support and concurrency

Some multi-user issues are addressed by the underlying mechanisms of the *PoIP* API. Indeed, windows of a shared surface are potentially controlled by different users. Drag-and-drop like operations can be processed simultaneously offering some support for concurrency. Each user is able to interact with any window thanks to a identified pointer. However, many other concurrency issues typical of CSCW applications are not adressed by this model devoted to drag-and-drop like operations. For example, in order to work appropriately, windows' component (button, textfield, etc.) would have to support concurrent interactions. Designing a multi-user interface requires to redesign each component and event handling at a more general level so they support different input events streams and take account of IDs identifying these input streams. Such a work goes beyond the scope of this paper. Hence using our model using regular Swing components is fine as long as no more than one user interact with a component at a given time. If two users click on the same button at the same time, result may be unexpected.

In this section we quickly describe how input streams are processed independently for different users and associated each to a different pointer. These pointers are named fake pointers as opposed to system pointer. Fake pointers have a semi-transparent background bubble which has different colors depending on the ID of the pointer (see figure 8). In the current implementation of *PoIP*, each window provides its ID to only one pointer: the pointer associated with one user and the pointing device handled by the window's windowing system. This ID will never change even though the pointer can further move to other windows. Hence several pointers can be simultaneously on the same window but there is at most one that is originally representative of the pointing devices directly associated with the window, the others come from other windows of other machines.

In *PoIP*, system pointer is only used to fire appropriate events coming from a device. Fired events are directly captured, processed and redirected appropriately by the shared window. Fake pointer are used for feedback and system pointer is kept hidden from the user and relocated at the center (x_0, y_0) of the window after events are fired. For example, when the user moves a fake pointer of (x, y) pixels, the new location of the "invisible" system pointer

is $(x_0 + x, y_0 + y)$ and corresponding move event is captured. Processing of the event results in moving the fake pointer consequently and moving the invisible system pointer back to the center (x_0, y_0). Centering system pointer is just a trick used to keep system pointer within the boundaries of the window while the fake pointer might go beyond to reach all possible windows of the shared surface. In *PoIP*, centering system pointer is performed thanks to the class *Robot* of the AWT.

The Robot class is used in most situations but there are some exceptions. For some direct pointing devices such as pen devices of Mimio or Smart BOARD the class Robot conflicts with the drivers of the devices when they both take control over system pointer. In such cases, the class Robot will not be used. As a result, with such pen-like interface, a given pen will be limited to its original display. It does not prevent the user from using such device for dragging-dropping objects from this display to another but it might prevent the user from moving the pointer to another display. Since, such devices are usually physically bound to a unique display we consider this limitation as minor and consistent with the type of device used.

5.5. A complete example of a typical drag-and-drop operation

To illustrate a typical drag-and-drop operation that successfully moves one object from a source component to a target component, lets consider an example where two computers are involved in the operation: computer A and computer B. We further consider that the operation starts on computer B with the input device associated with computer A and that the target component is handled by computer B. This example of a typical drag-and-drop operation is depicted in figure 6. This figure details typical communication between entities of the model. Overall, the details can be summarized in 5 steps: initialization, drag detection, drag, drop, finalization.

5.5.0.2. Initialization When the user interface is created, a few elements needed for *drag-and-drop* like operation are created once: the source listener, the target listener, and the master instrument. While these operations are required only once, it is still possible to change these elements later – for example, the master instrument can be changed according to the context or the user preferences. Note that computer B also owns a master instrument but that it is not involved in the current process since the input device of computer A is used and therefore does not appear in the sequence diagram. Then, source and target components have to register themselves on the *DmManager*s. All needed elements are now created in order to detect drag gestures.

5.5.0.3. Drag detection The beginning of a *drag-and-drop* like manipulation is detected by the master instrument of computer A (which receives all input events of computer A). When a drag gesture is detected, the source listener is notified by the *dmDetected(e)* method (cf method 1 of figure 6). The listener accepts the operation by calling the *startDm()* method of the event *e* and provides a list of formats in which the data can be provided. Once the source has accepted the operation, the master instrument asks for the creation of all the slave instruments (cf method 2 of figure 6). This query is made to the *DmManager* which creates its own slave instrument and transfer the query to all other *DmManager*s (cf method 3 of

figure 6). At this point, each *DmManager* owns a slave instrument for the master instrument of computer A.

5.5.0.4. Drag During the drag process, the master instrument notifies the slave instrument that the pointer is moving (cf figure 6, method 4). In this example, the slave instrument of computer B is notified. This means that the pointer that corresponds to the input device of computer A is currently moving over the shared window of computer B. When needed, the slave instrument responds by notifying the master instrument that the target component changed – which means that the pointer moved over a new component. If it exists, the previous target is notified by the *dmExited()* method. The new target is notified by the *dmEntered(e)* method (cf method 5 of figure 6). Depending on the data that can be provided by the source component, the target component has accepted the drag by calling the *accept()* method of the event *e*. While the pointer moves over the target component, both source and target component are notified by, respectively, *dmSourceOver(e)* and *dmTargetOver(e)* (cf figure 6 methods 6 and 7). Note that the target listener is notified before the source component so the source can know if the target accepts the drag operation.

5.5.0.5. Drop At the end of the operation – when, for example, the user releases its mouse button – the target is notified by the *dmReceive(e)* method (cf figure 6 method 8). At this point, no data has been transfered from the source to the target. Therefore, the target requires the data by calling the *getData()* method (cf figure 6 method 9) which trigger the *dmDataRequired(x)* method of the source listener (cf figure 6 method 10), *x* being the type of data needed by the target component. If the data is successfully received by the target, the *success()* method of the event *e* should be called by the target listener. Then, the source component is notified of the end of the operation by the *dmEnded(e)* method (cf figure 6 method 11). The source then knows if the transfer has been a success and can therefore acts appropriately by deleting the transfered data. Finally, the master instrument requires the destruction of all its slave instruments which are not needed anymore. The principle is the same that for their creation (cf figure 6 methods 12 and 13).

5.5.0.6. Finalization When the user interface is destroyed, the source and target components unregister themselves from the *DmManager*s.

6. IMPLEMENTATION OF SPECIFIC INSTRUMENTS

In this section we explain how recent extensions of drag-and-drop can be implemented as specific instruments and integrated in the overall model at little cost. We first introduce common principles used in implementing a new instrument, we further provide more details for the implementation of *push-and-throw* which uses a large scope of feedbacks. We further quickly review other instruments behavior.

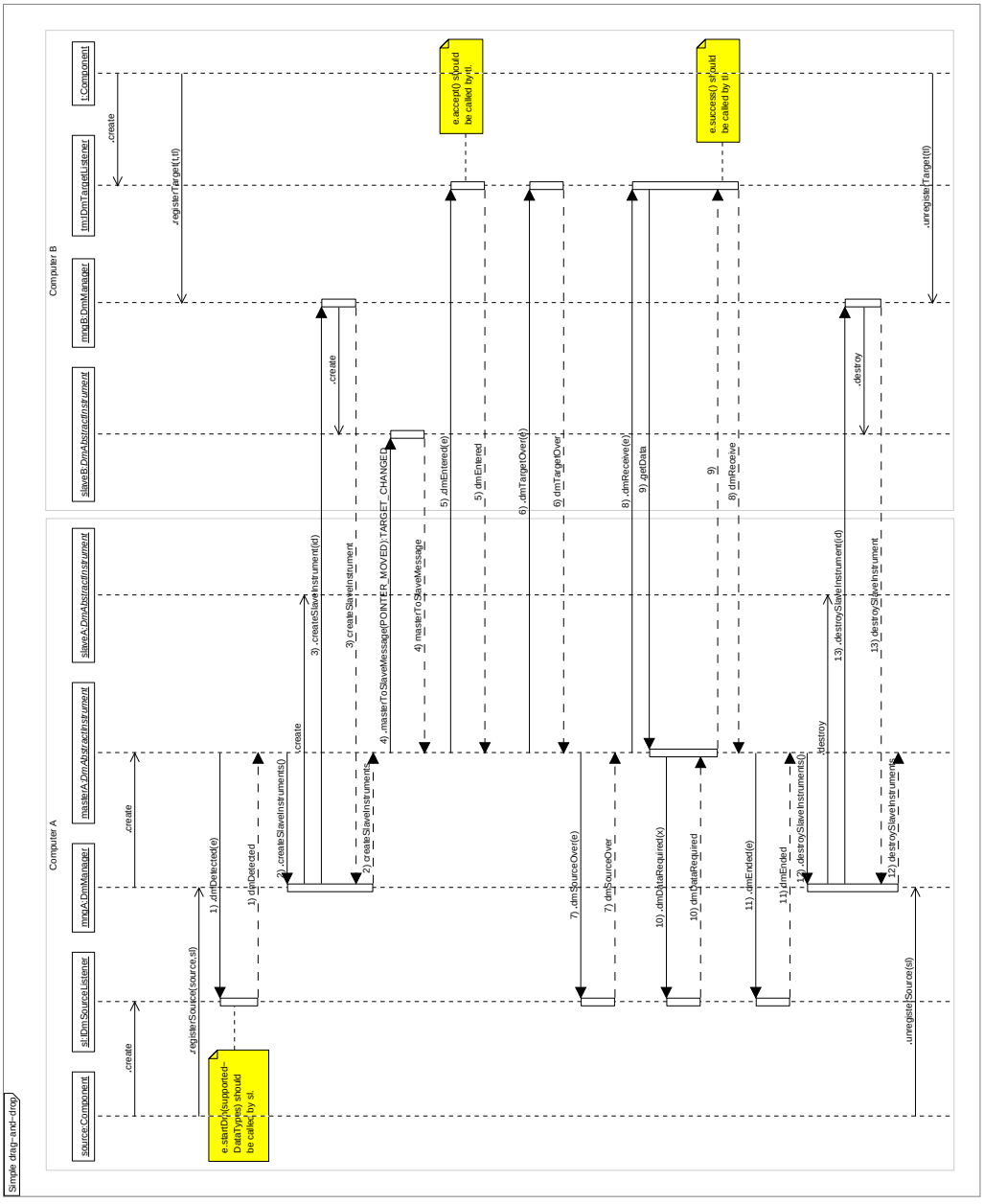

Figure 6. Sequence diagram of *drag-and-drop* model.

6.1. Implementing an instrument: common principles

Implementing a new instrument implies writing a new class which inherits directly or indirectly from the *DmAbstractInstrument* abstract class. Both master and slave instruments are instances of this same class. This new class needs to implement five methods that can be split into three categories:

- Input: the method *inputEventDispatched(event)* is called when an input event occur (mouse or keyboard). The instrument behavior depends of the input event received through this method. Only the master instrument manage input events.
- Output: the method *feedback(graphics)* is called when the workspace is repainted. Instrument feedback are displayed through this method. Usually, only slave instruments manage feedbacks.
- Communication: The communication between master and slave instruments is done through the methods *masterToSlaveMessage(slaveID, message)* and *slaveToMasterMessage(masterID, message)*. Note that an additional class needs to be written in order to implement messages. This class should be very simple: a type and a list of properties. Messages are only described as *Serializable* in the *DmAbstractInstrument* class. And finally, as shown on figure 6, when the target component needs the data at the end of the operation, it calls the *getData()* method of the event which itself calls the *getData()* method of the master instrument. This method needs to be implemented and its behavior should be the same for any instrument: asking the data to the source component (*dmDatarequired()* method) and returning it to the target component.

6.2. Implementing an instrument: the case of push-and-throw

Push-and-throw [13, 4] (see section 6.2) uses the pantograph metaphor, which means that user's movements are amplified.

The figure 7 shows the state diagram for *push-and-throw*.

6.2.1. States

The *push-and-throw* instrument involves four states :

- **Stopped**: when the instrument is inactive (begin and end states in the figure 7)
- **Idle**: when the mouse button is pressed and the pointer has not moved. Indeed, every *drag-and-drop* like manipulation is recognized when the pointer is moved while a mouse button is pressed. When a user simply click on an object, the mouse button is pressed and released while the pointer does not move. However, like every implementation of *drag-and-drop* or *drag-and-drop* like techniques, a threshold is used to avoid accidental drags. This means that while the mouse button is pressed and the pointer stay in few pixels area, the instrument stays in the *idle* state.
- **Inactive**: when a drag movement has been detected and the source component has refused to start a drag operation.

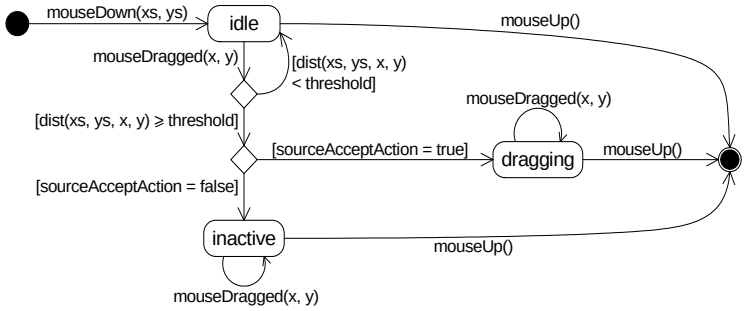

Figure 7. State diagram for the *push-and-throw*.

- **Dragging**: when the instrument is actually in use. This implies that the source component accepted to start a drag operation. Every movement of the pointer is amplified and reported to a crosshair that represents the target point of the *push-and-throw*.

6.2.2. Transitions

An instrument receives every input event. *Push-and-throw* mainly uses the *mouseDown*, *mouseDragged*, and *mouseUp* events.

6.2.2.1. mouseDown The *mouseDown* event is the transition between the *stopped* and *idle* states. Note that the *mouseDown* event occurs necessarily inside the JVM since source components that support *drag-and-drop* needs to be registered in the *DmManager*. It is not possible to drop an object that does not come from a shared window.

6.2.2.2. mouseUp The *mouseUp* event ends the *push-and-throw*:

- if the state is *idle* or *inactive*, nothing happens and the state changes to *stopped*.
- if the state is *dragging*, the operation is ended (notification of source and target components, data transfer) and the state changes to *stopped*.

6.2.2.3. mouseDragged The *mouseDragged* event has different consequences depending of the instrument state:

- if the state is *idle*, the location of the pointer is checked: if the distance between the pointer and its initial location (i.e. when mouseDown occured) is lower than a given threshold (usually 4 pixels), the state stays *idle*. Otherwise, the source component is notified that a *push-and-throw* gesture is recognized. If the source component ask for a *push-and-throw*, the state becomes *dragging*, otherwise the state becomes *inactive*.
- if the state is *inactive*, *mouseDragged* has no effect.
- if the state is *dragging*, for each *mouseDragged* event, the crosshair location is re-computed, the instrument feedbacks are updated and target component is notified

of the *dmTargetOver()* event or eventually of the *dmEntered()* or *dmExited()* events if the target changes. Drag-under feedbacks are managed by the target components when receiving these events (an example of drag-under feedbacks is the highlight of the instrument). Drag-over feedbacks are managed by the instrument (an example of drag-over feedback is the modification of the pointer to reflect the type of action: copy, move or link).

Note that when an indirect device is used (e.g. mouse), the pointer is constrained in the take-off area (see section 6.2).

6.2.3. Master and slave instruments: an example

In order to illustrate the master and slave instruments behavior, we will detail an example involving three shared windows in which the pointer that belongs to window 1 successfully moves an object from window 2 to window 3. In this example, window 1 owns a *push-and-throw* instrument while window 2 and 3 own *drag-and-drop* instruments.

6.2.3.1. Before the activation of the instrument While the pointer that belongs to window 1 moves over window 2, input events are captured by window 1 and are:

- augmented with the ID of window 1.
- redirected to window 2. Then, components of window 2 are notified of these input events.
- transmitted to the *push-and-throw* master instrument of window 1 (via the *DmManager* of window 1).

At this point, there is no slave instrument since there is no *drag-and-drop* or *push-and-throw* operation in progress. There are only: a *push-and-throw* master instrument on window 1, and 2 master *drag-and-drop* instruments on window 2 and 3.

6.2.3.2. Activation of the instrument In order to activate the instrument, the user has to press the button of the mouse (or any other input device) and move the pointer. When the pointer movement is higher than a given threshold, three slave instruments are created: one on each window of the shared surface. Since the operation begins on window 2, the slave *push-and-throw* instrument of window 2 will be considered as the main slave until the end of the operation. The first task of the main slave instrument is to find the source component (e.g. the component under the pointer) – this task can not be made remotely using RMI – and to send the corresponding *DmSourceListener* to the master instrument. Once the main slave has found the source component, the master notifies it that a *push-and-throw* gesture has been recognized. In response, if the source component chooses to actually start the operation and provides the supported data type (e.g. the formats in which the data can be provided), the instrument state changes to *dragging*. Otherwise, if the source component refuse to start the operation, the instrument state changes to *inactive*.

6.2.3.3. Use of the instrument Once the instrument is activated (state *dragging*), the master instrument computes the crosshair location depending of the location of the pointer in the take-

off area. The pointer stays in the take-off area until the end of the operation, thus the pointer stays on window 2. On the other hand, the crosshair can move all over the shared surface.

For each movement of the pointer, the master notifies every slave that the crosshair location changed. If the window on which the crosshair is finds that the target component changed, the master is notified (and receives the new corresponding *dmSourceListener*). Then, the old and the new target are notified by the master instrument of *dmExited()*/*dmEntered()* events.

Each time the crosshair changes of window, slaves instruments are notified. For example, when the crosshair moves from window 2 to window 3, the slave instrument of window 2 is notified that the crosshair exited and the slave instrument of window 3 is notified that the crosshair entered.

When a slave instrument receives movement notifications, the window is asked for a repaint. Therefore, the instrument feedbacks are repainted. Only the slave instruments manage feedbacks (not the master). There are three cases for *Push-and-throw* feedbacks:

- The main slave displays the take-off area and the beginning of the trajectory
- The slave of the window on which is the crosshair displays the end of the trajectory and the crosshair. This can be the main slave.
- Other slaves display the trajectory

6.2.3.4. Finalization When the operation is terminated (i.e. when the user release the mouse button), The master instrument already knows the source and the target component. Therefore, the data is transfered as explained in the section 5 and all the slave instruments for *push-and-throw* are destroyed.

6.2.3.5. Concurrency In the previous example, there was only one operation: user of window 1 was moving an object from window 2 to window 3. But it is possible to have several concurrent operations. For example, if in the same time the user of window 2 move an object (*drag-and-drop*) from window 2 to window 3, then the following instrument will exists:

- window 1: one master *push-and-throw* instrument, one slave *push-and-throw* instrument, one slave *drag-and-drop* instrument.
- window 2: one master *drag-and-drop* instrument, one slave *push-and-throw* instrument, one slave *drag-and-drop* instrument.
- window 1: one master *drag-and-drop* instrument, one slave *push-and-throw* instrument, one slave *drag-and-drop* instrument.

A slave instrument is linked to its master and is identified by the ID of the master instrument (i.e. the ID the window that owns the master instrument).

6.3. Other instruments

We now present the other instruments. All instruments share many mechanisms so we will only talk about the particularities of each instrument.

Figure 8. (Left to right, top to bottom) (a) *drag-and-drop* (b) *push-and-pop* (c) two concurrent manipulations: a *push-and-throw* and a *push-and-pop* (d) two concurrent manipulations: a *push-and-throw* and an *accelerated push-and-pop*.

6.3.1. Drag-and-drop

The only feedback of *drag-and-drop* instrument is a *ghost*, provided by the source component, which is displayed at the pointer location by a slave instrument (fig. 8 - (a)). Note that drag-under feedbacks are managed by components and are not depending of the instrument.

Communications between the master instrument and its slaves occur at different stages:

- when the source component listener is needed (at the beginning of a *drag-and-drop*),
- when the location of the fake pointer changes (so the ghost should also move),
- when the target component changes,
- and when the ghost image changes (occurs only once).

Globally, the *drag-and-drop* instrument is very similar to the *push-and-throw* instrument except that feedbacks are different and that no crosshair is managed.

6.3.2. accelerated push-and-throw

The implementation of *accelerated push-and-throw* is very similar to *push-and-throw*. The differences are:

- Computation of the crosshair location is different since acceleration enters in consideration,
- A new *pause* state is added (figure 9). Indeed, due to the use of acceleration, the pointer may reach a window border while the crosshair has not reach its target [6]. Therefore,

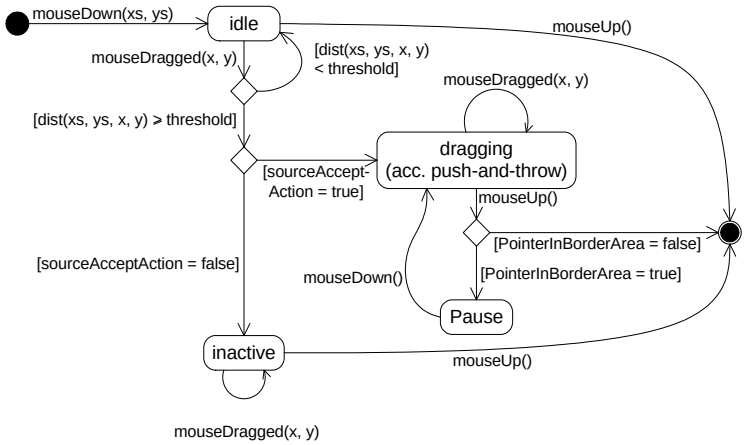

Figure 9. State diagram for the *accelerated push-and-throw*.

the input device can be released near a border (in an area of 20 pixels wide) without terminating the operation. The instrument enters in the *pause* state and gets back to *dragging* state when the input device is putted down again.

6.3.3. push-and-pop

The *push-and-pop* instrument (fig. 8 - (b) and (d)) creates an array of targets around the pointer [6, 1].

The set of targets only contains targets that accept the manipulated data. Indeed, when creating the set, each target's *dmEntered()* and *dmExited()* methods are called. If the event's *accept()* method is called, the potential target component is included in the array. Otherwise (if *reject()* method is called), the component is ignored.

The array of targets is laid out with the mechanism described by authors of [1]. The original algorithm was enhanced by the ability to manage a set of icons representing targets of different sizes. If a target provides a graphical proxy which is too large, the image is reduced. This can be done either by scaling the entire image or by cropping it. The best method has still to be specified. Examples of too large images are returned by components like text areas or text fields.

As shown on the state diagram (figure 10), *push-and-pop* instrument includes the *accelerated push-and-throw* instrument. Indeed, while *push-and-pop* is a powerful technique [6], it has a limitation: a set of targets can be reached but it is not possible to reach an empty area (which is needed when re-arranging a desktop for example).

For this reason, the *push-and-pop* instrument can be deactivated to an *accelerated push-and-throw* instrument which is the most efficient technique among the ones that allow to reach any part of the workspace [6]. This deactivation is done by moving the pointer back to its initial

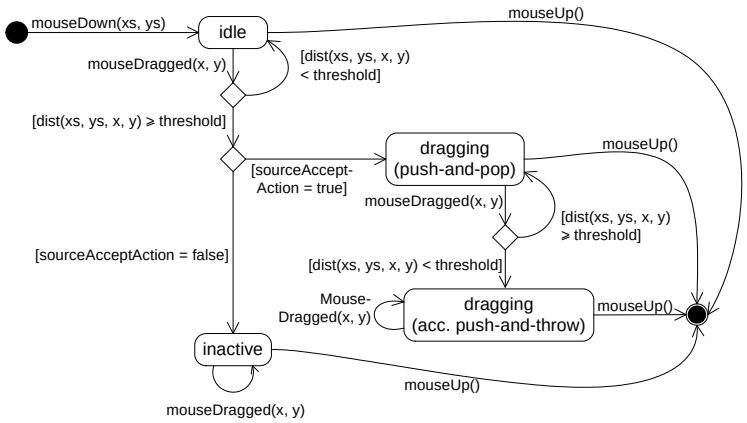

Figure 10. State diagram for the *push-and-pop*.

location (i.e. the distance between the activation point and the pointer is lower than a given threshold).

7. CONCLUSION

In this paper we have shown the necessary evolution of drag-and-drop to meet requirements of new emerging interactive environments. We have pointed out issues that are brought by these new environments. We have further reviewed, compared and discussed recent drag-and-drop extensions that partially address these issues. Finally, we have proposed an implementation model that builds upon these analysis to meet the challenging requirements of new emerging interactive environments and to make it possible for a programmer to support most extensions of drag-and-drop in a single unified framework.

We provide an API called *PoIP* that implements our model that can be used in the development of most java based applications. The API has been tested in the development of several test applications and in the development of a collaborative bookmarking system. Overall *PoIP* was found to be robust, it did not show any weaknesses during these developments. Even though *PoIP* uses a layered pane to display pointers and feedbacks, we didn't notice any significant drop in performances with the Java applications tested.

However, the question of the level at which the model should be implemented is left open. Our approach with *PoIP* was to implement the model at the toolkit level. However, considering lower levels (windowing system level, window manager level or middleware level) would certainly be costly but also most likely to provide great benefits. Even though our model is implemented at the toolkit level we made it general enough to be implemented at other levels as well.

Our model proposes a multi-instrument approach which is important to address problems of usability and scalability mentioned in section 2. In that context, modularity and genericity were used to minimize the cost of introducing new drag-and-drop extensions as new needs arise. Our model further supports multi-computer environments transparently. This is useful to make it possible for drag and drop operations to occur on shared surfaces displayed over several distinct computers possibly running different windowing systems. Multi-computer support is achieved thanks to the combination of shared surface management and slave instruments. Finally, our model supports multiplexing of input streams thanks to input device ID. Hence, we make it possible for multiple users to perform different types of drag-and-drop operations simultaneously.

8. ACKNOWLEDGEMENTS

Thanks to our reviewers for the thorough review of this paper and many helpful comments and suggestions. Thanks to Peter King and Martine Hornby for helping us in improving the readability of this paper.

REFERENCES

1. P. Baudisch, E. Cutrell, D. Robbins, M. Czerwinski, P. Tandler, 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 *Proceedings of Interact 2003*, Sep. 1–5 2003.
2. Michel Beaudouin-Lafon. Instrumental interaction: an interaction model for designing post-wimp user interfaces. In *ACM CHI '00 proceedings*, pages 446–453, ACM Press, 2000.
3. A. Bezerianos and R. Balakrishnan. The vacuum: facilitating the manipulation of distant objects. In *ACM CHI '05 Proceedings*, pages 361–370, ACM PRESS, 2005.
4. M. Collomb and M. Hascoët. Speed and accuracy in throwing models. In *HCI2004, Design for life, Volume 2*, pages 21–24. British HCI Gr oup, 2004.
5. M. Collomb PoIP: an API for implementing advanced drag-and-drop techniques In http://edel.lirmm.fr/dragging/.
6. M. Collomb, M. Hascoët, P. Baudisch, and B. Lee. Improving drag-and-drop on wall-size displays. In *Proceedings of Graphics Interface 2005*, Victoria, BC, 2005.
7. M. Collomb, M. Hascoët. Comparing drag-and-drop implementations. Technical Report RR-LIRMM-05003, LIRMM, University of Montpellier, France, 2005.
8. Joëlle Coutaz, Stanislaw Borkowski, and Nicolas Barralon Coupling Interaction Resources: an Analytical Model. In *EUSAI'2005*, pages 183–188, 2005.
9. Fraunhofer institute. The dynawall project. *http://www.ipsi.fraunhofer.de/ambiente/english/projekte/projekte/dynawall.html*.
10. Paul M. Fitts The information capacity of the human motor system in controlling the amplitude of movement. In *Journal of Experimental Psychology*, volume 47, number 6, June 1954, pp. 381-391. (Reprinted in Journal of Experimental Psychology: General, 121(3):262–269, 1992).
11. J. Geißler. Shuffle, throw or take it! working efficiently with an interactive wall. In *ACM CHI '98 proceedings*, pages 265–266, ACM Press, 1998.
12. M. Hascoët, M. Collomb, and R. Blanch. Evolution du drag-and-drop : du modèle d'interaction classique aux surfaces multi-supports. *revue I3, 4(2)*, 2004.
13. M. Hascoët. Throwing models for large displays. In *HCI2003, Designing for society, Volume 2*, pages 73–77. British HCI Group, 2003.
14. K. Hinckley, G. Ramos, F. Guimbretiere, P. Baudisch and M. Smith. Stitching: Pen Gestures that Span Multiple Displays. In *Proceedings of AVI'04*, pages 23–31, ACM PRESS, 2004.

15. G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner, and J. T. Klosowski. Chromium: a stream-processing framework for interactive rendering on clusters. In *Proceedings SIGGRAPH '02*. Pages 693–702, ACM Press, 2002.
16. D. R. Hutchings, J. Stasko, and M. Czerwinski. Distributed display environments. In *interactions 12*, 6 Nov. 2005, pages 50–53.
17. Brad Johanson, Greg Hutchins, Terry Winograd, and Maureen Stone. Pointright: experience with flexible input redirection in interactive workspaces. In *ACM UIST '02 proceedings*, pages 227–234, ACM Press, 2002.
18. C. Lachenal. Modèle et infrastructure logicielle pour l'interaction multi-instrument multisurface. PhD these, Université Joseph Fourier, Grenoble, France, 2004. *http://iihm.imag.fr/publs/2004/theseLachenal.pdf*.
19. Dan R. Olsen and S. Travis Nielsen. Laser pointer interaction. In *ACM CHI'2001 proceedings*, pages 17–22, ACM PRESS, 2001.
20. Jun Rekimoto. Pick-and-drop: a direct manipulation technique for multiple computer environments. In *ACM UIST '97 proceedings*, pages 31–39, ACM Press, 1997.
21. Jun Rekimoto and Masanori Saitoh. Augmented surfaces: a spatially continuous work space for hybrid computing environments. In *ACM CHI '99 proceedings*, pages 378–385, ACM Press, 1999.
22. B. Shneiderman. Direct manipulation: A step beyond programming languages. In *Proceedings of the Joint Conference on Easier and More Productive Use of Computer Systems*. (Part - II): Human interface and the User interface - Volume 1981, ACM Press, 1981.
23. C. Shoeneman. Synergy. *http://synergy2.sourceforge.net/*.
24. N. A. Streitz, J. Geißler, T. Holmer, S. Konomi, C. Mller-Tomfelde, W. Reischl, P. Rexroth, P. Seitz, R. and Steinmetz. i-LAND: an interactive landscape for creativity and innovation. In *ACM CHI '99 proceedings*, pages 120–127, ACM PRESS, 1999.
25. Sony Japan. Flyingpointer. *http://www.sony.jp/products/consumer/pcom/software_02q1/flyingpointer*.
26. Stanford University ComputerScience. The stanford interactive workspaces project. *http://iwork.stanford.edu/*.
27. D. Thevenin and J. Coutaz. Adaptation and Plasticity of User Interfaces. In *Workshop on Adaptive Design of Interactive Multimedia Presentations for Mobile Users*, 1999.
28. Xdmx project. Distributed multihead X. *http://dmx.sourceforge.net/*.

Hypermedia Exploration with Interactive Dynamic Maps[1]

MOUNTAZ ZIZI AND MICHEL BEAUDOUIN-LAFON
Laboratoire de Recherche en Informatique, CNRS URA 410, Bâtiment 490, Université de Paris-Sud,91 405 Orsay Cedex, France. E-mail: {mountaz,mbl}@lri.fr

Interactive Dynamic Maps (IDMs) help users interactively explore webs of hypermedia documents. IDMs provide automatically-generated abstract graphical views at different levels of granularity. Visual cues give users a better understanding of the content of the web, which results in better navigation control and more accurate and effective expression of queries. IDMs consist of: *topic maps*, which provide visual abstractions of the semantic content of a web of documents and *document maps*, which provide visual abstractions of subsets of documents.

The major contributions of this work include 1) automatic techniques for building maps directly from a web of documents, including extraction of semantic content and use of a spatial metaphor for generating layout and filling space, 2) a direct manipulation interaction paradigm for exploring webs of documents, using maps and an integrated graphical query language, and 3) the ability to use the maps themselves as documents that can be customized, stored in a library and shared among users.

1. Introduction

Hypermedia systems have become increasingly popular. However, the early vision of hypermedia [Bush45] as a panacea for easy access to large quantities of information has proven elusive. Many users complain of being 'lost in hyperspace' and most current hypermedia systems must supplement their basic navigation strategies with search mechanisms. Even so, looking for information in a hypermedia system can be frustrating. Traditional interaction styles provide users with limited views of the information space being explored and hide most of the information (and its organization). Users often have no idea of what information is contained within a web of documents until they actually find it. This leads to both navigation errors and badly formulated queries.

We propose a possible solution to this problem that consists of creating abstract graphical views of webs of documents. We call these views *interactive dynamic maps* or IDMs, and combine them with an interaction style that facilitates navigation and expression of queries. Users can take advantage of visual cues that give them a better understanding of the content of a web of documents. Two types of IDMs have proven particularly useful: *topic maps*, which provide visual abstractions of the semantic content of a large web of documents and *document maps*, which provide visual abstractions of subsets of documents.

Many researchers have argued in favor of providing graphical overviews for users (e.g., [Nielsen90]). However, few guidelines have been proposed for their design. We provide an original approach that offers three key contributions: First, we provide automatic techniques for building maps (IDMs) directly from any web of documents. Some techniques extract semantic

[1] This is a revised and expanded version of a paper presented at the European Conference on Hypermedia Technology (ECHT'94, Edinburgh, Scotland, 1994), entitled "Accessing Hyperdocuments through Interactive Dynamic Maps".

content from a web, while others implement layout and space-filling strategies according to a spatial metaphor that we have defined. Second, we provide an interaction paradigm that enables users to explore webs of documents through direct manipulation of maps. Interaction is based on graphical transformations of various views based on their semantic content and on an integrated graphical query language. Third, our maps are themselves documents, like maps in the real world. They can be customized and stored in a library of maps that can be shared among several users. This allows for both re-use of past search efforts and cooperation among users.

The paper is organized as follows. We begin with a brief overview of related work and then introduce the concept of Interactive Dynamic Maps. Subsequent sections present the techniques we have developed for the computation of topic maps, including mechanisms for extracting semantic information from a web of documents using full-text analysis, a description of our strategies for laying out objects on maps, space-filling of map backgrounds and link visibility. We then describe the interaction techniques that enable users to access these maps, which combine transformations of views with graphical query formulation and expansion. We discuss customization, re-usability and sharing and describe our prototype system, SHADOCS, followed by a brief conclusion and description of future work.

2. Related work

Many researchers have investigated the navigation problem in hypermedia systems. Early research led to techniques and design strategies for finding the *proper structure* of a hyperdocument. Most attempted to reduce the graph structure to a hierarchical or semi-hierarchical structure and differ primarily in the stage at which the structure of hyperdocuments is made hierarchical (or semi-hierarchical). While some authors suggest that hypertext designers choose this structure at the design stage of the hypertext [Meyrowitz86, Acksyn88], others suggest that existing hypertext documents with complex underlying structures can be transformed into hierarchical hypertext. Experimental techniques that provide the necessary automatic transformation have been proposed in [Rivlin94], based on some critical measures of hypertext systems [Rodrigo92]. Other techniques are based on the aggregation of nodes [Crouch89].

Another body of research led to *visual tools* such as overview diagrams or maps that help users navigate through complex hypermedia structures [Halasz87, Yankelovich88]. Other works have extended maps by providing footprints [Bernstein88], which help users know where they are, where they come from and where they can go.

Although the purpose of overview diagrams was to help users navigate a complex hypertext, most research was conducted on hierarchical structures or on restricted substructures. For example, Notecards [Halasz87] automatically provides tree diagrams while Intermedia [Yankelovich88] automatically provides local overview diagrams of the current node and the links adjacent to it. In other systems, e.g. [Nielsen90], overview diagrams that display more complex graph structures are hardcoded. In a user study of the navigation problems in Nielsen's hypertext, Foss [Foss89] concluded that the overview diagrams would have been much more helpful if they had been designed more carefully.

Unfortunately, designing good overview diagrams for complex structures has proven to be very difficult. According to [Bernstein91]:

> *[...] while visualization tools have made great progress toward representing hierarchical structures, the representation of more complex webs has proven to be much more challenging. Visualization approaches depict regular hierarchies well but are much less effective at revealing regular structures in more complex hypertext webs.*

Furthermore, most overview diagrams do not share conventions or common features. In fact, to our knowledge, few guidelines have been developed for designing maps for complex hypertext structures.

Other researchers have been investigating methods and techniques for visualizing information, taking advantage of the interactive facilities provided by modern workstations. Among these, the Information Visualizer project [Card91] has introduced new interactive display types such as the Cone Tree [Robertson91] and the Perspective Wall [Mackinlay91]. Other authors have introduced Tree Maps [Shneiderman92] and Starfield Displays [Ahlberg94]. Interactive Dynamic Maps build upon the information visualization paradigm by providing a dynamic and interactive display for the exploration of hypermedia documents.

3. Interactive Dynamic maps (IDM)

The interaction model we have defined is based on a metaphor of navigation in the real world. A collection of documents is considered to be a territory that contains resources (documents and topics) and maps of these territories can be drawn. An Interactive Dynamic Map is a document that provides a global view of either the semantic content of a set of documents or the set of documents themselves. The semantic content reflects the topics contained within the set of documents and the way they are organized or related to each other. It is represented by a thesaurus that is built automatically from the full-text analysis of the documents.

Since IDMs are computer artifacts, they offer more possibilities than geographical maps of the real world:

- users lost in hyperspace can always find a map showing where they are, where they came from and where they can go;
- maps can change dynamically as new documents are added to the web;
- maps support navigation, querying and browsing through direct manipulation;
- maps can be customized, stored, and reused, which is useful for extensive search tasks that are often interrupted: the map can be stored at any point in the navigation and reused at a later time when the user wants to resume;
- maps can be shared among a group of users, which is valuable when several users share a common interest, since they can customize a common, shared map.

We have adapted real-world conventions about the design of geographical maps to meet our needs. Maps are made up of regions, cities and roads which are used to convey information about the structure of the web. For example, a region represents a set of documents and the size of a region reflects the number of documents in that region. Similarly, the distance between two cities reflects the similarity relationship between those two cities. Thus, in maps where cities represent documents, if two cities are close to each other, then the documents have similar or strongly related contents.

Time invariance is another important aspect of real-world maps. Although a map of a given territory may change from year to year, the changes are local and the modifications are minor. Because objects in an IDM do not change location much over time, users can take advantage of their spatial memory to track and store information efficiently.

We have defined two kinds of IDMs: Topic Maps represent the semantic content of a large collection of documents, and Document Maps display and allow access to *subsets* of documents.

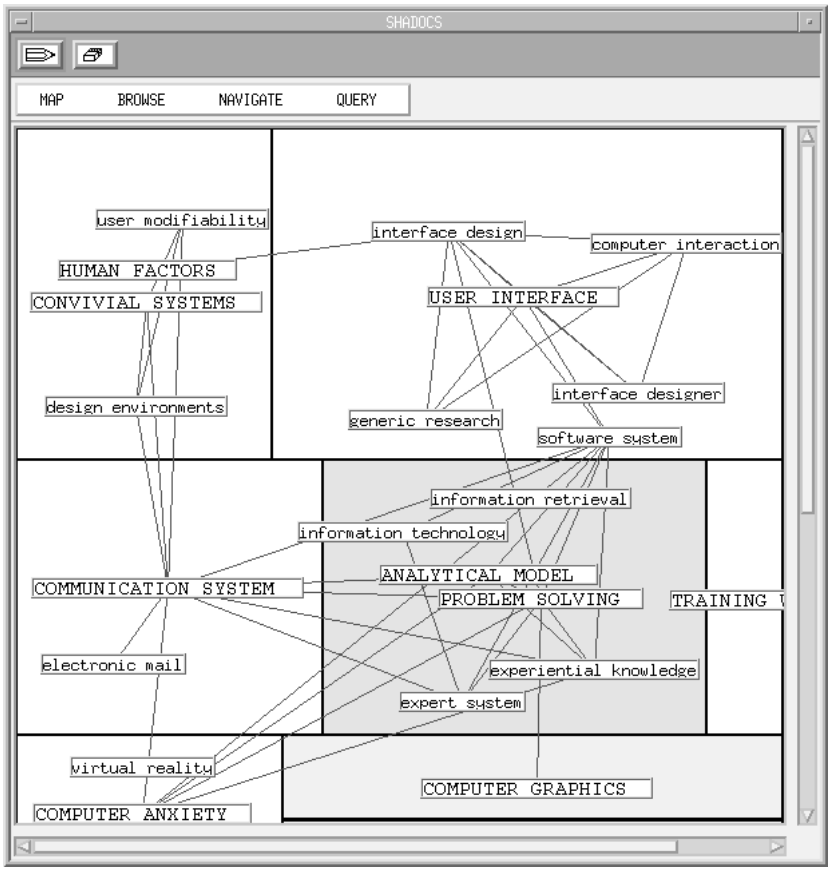

Figure 1: Sample Topic Map

Topic Maps (figure 1) provide an overview of the topics represented in a collection, their importance, and similarities or correlations among them. The areas of the map are the classes of an intermediate thesaurus. Each class contains a set of topics represented by cities on the map and depicted by icons. Roads between cities represent relationships between topics. Topic Maps provide an overview of a large number of documents by extracting semantic information from them rather than displaying the documents themselves. Users can issue queries by selecting regions, cities and roads of the map. The result of the query is displayed in a Document Map.

Document Maps (figure 2) represent collections of documents either generated automatically from a user query or gathered manually by a user or designer. Unlike Topic Maps, the cities of these maps are documents. Cities are laid out such that similar or highly correlated documents are placed close to each other. The size of a city reflects its importance with respect to a given criterion, which varies according to how the map was created. If the map gathers documents resulting from a query, document importance (and therefore city size) is computed as the relevance of the document to the given query. If the map contains documents that were gathered manually by a user or designer, document importance is the subjective importance given by the users.

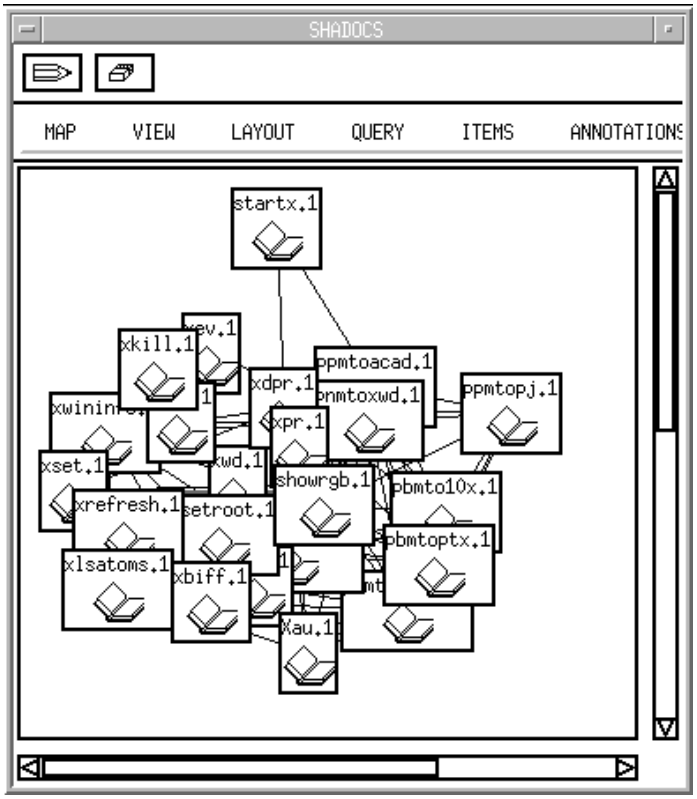

Figure 2: Sample Document Map

One aspect of IDMs is the abstract visualization of the semantic content of a set of documents, as described above. Another aspect of IDMs is the support for interactive document retrieval tasks: IDMs support two forms of interactive navigation, *browsing* and *querying*, through direct manipulation of the maps. A third aspect of IDMs is customization: maps can be easily edited by users. Customization aids navigation tasks and increases the users' sense of engagement, helping them to better understand and remember the underlying information space.

4. Automatic extraction of the web semantics

4.1. INFORMATION RETRIEVAL TECHNIQUES

The computation of IDMs is based on a set of information retrieval techniques known as *full-text analysis*. Most of these techniques consist of representing documents and keywords by vectors. The coordinates of document vectors are computed from the set of keywords and the coordinates of keyword vectors are computed from the set of documents, as follows: coordinate x_i of document D is 1 if keyword T_i was found in D and 0 otherwise. Conversely, coordinate x_i of the vector representing a keyword T is 1 if document D_i contains T and 0 otherwise. More complex formulas for computing these vectors can be used (see for example [Salton83] or [Larson92]).

Correlations among documents, correlations among keywords and the relevance of documents to keywords can be computed from these vectors. The basic idea is that if two documents are similar, their vectors are similar. One measure of the similarity between two vectors is the cosine of the angle between those two vectors: co-linear vectors have an angle of zero and thus a cosine of 1, while orthogonal vectors have an angle of 90 degrees and therefore a cosine of 0. The cosine in turn can be computed from the scalar product. Various measures and computation techniques exist [Salton83, Larson92].

4.2. REPRESENTING THE WEB SEMANTICS WITH A THESAURUS

The first step in generating a Topic Map consists of creating a *thesaurus*. The thesaurus is a set of topics or descriptors (i.e. terms and expressions selected to describe the semantic content of the documents). Some descriptors have related meanings and can therefore be organized into classes.

Most real-world thesauruses are built manually and use descriptors according to their particular goals. Some, such as Roget's thesaurus, contain general descriptors, while others, such as the INSPEC thesaurus, contain more specific terms. They also differ according to the types of relationships they support, the granularity of the classes and the depth of the hierarchy. In most cases, thesauruses provide concepts within a single area although some attempt to be more exhaustive. The latter requires development of complex semantic relations built by human experts. In an automatic construction context such as ours, the goal is slightly different. We are only interested in terms relevant to the description of the semantics of the underlying web of documents. Since we want to apply our technique to various types of documents, we use as little linguistic knowledge as we can. The semantic relations are constructed automatically based on a co-occurrence analysis, in which words and expressions that appear in a subset of the documents (i.e., not all of the documents, but at least some of them) are selected as potential descriptors.

Automatic construction of the thesaurus begins with the extraction of terms and the computation of correlations between them. Classes of the thesaurus are then obtained by applying clustering techniques to the resulting set of descriptors. In this section we briefly review our approach, the results of our experiments in descriptor extraction (see [Zizi95] for a detailed description), and the construction of classes.

4.3. EXTRACTING THE DESCRIPTORS

Our extraction process (see figure 3) extracts two-word expressions rather than single words[2] for constructing the thesauruses and lists of keywords. We experimented with extracting single words, like [Padmini92] who extracted terms for use by other computer programs. However, we found that single words were not expressive enough to be presented visually to end users. Longer expressions (from two to five words) were much more meaningful. Studies of the contents of manual thesauruses or lists of keywords show that more than 50 % of the keywords are expressions.

We have focused on two-word expressions because (1) they give reasonable keywords and represent a fair proportion of the keywords found in manual thesauruses and (2) they can be extracted efficiently at relatively low cost, e.g. by using statistical filters. Extracting 3, 4 or 5-

[2] Two additionnal common operations such as substituting synonyms and normalizing plural forms can be added to improve performance. The experiments reported in this article were conducted without these two operations.

word expressions is probably useful but requires additional linguistic text analysis since syntactical information has to be considered to filter such expressions efficiently [Bourigault92].

Our approach to two-word extraction consists of building an adaptive dictionary while parsing the documents. The dictionary contains two-word expressions together with information about the frequency and inverse frequency of the expression. Every two successive words of a document defines a two-word expression that is added or updated in the dictionary. A small set of rules filter out a large number of such expressions, such as those containing one small word or one stop word (see figure 3).

When a complete set of documents is parsed, statistical filters are used to determine potential descriptors among the expressions stored in the dictionary. This is done by adapting traditional single word frequency filters to handle expressions. In our experiment, we used the average frequency filter and the inverse frequency filter (see figure 3). In the first case, expressions that have a total number of occurrences below a threshold *s0* or over a threshold *s1* are removed because they are not appropriate to efficiently represent the semantic content of documents [Salton83]. In the second case, descriptors are removed when the inverse number of documents in which they occur is below a threshold *s2*. Thus descriptors that occur in a high proportion of the documents are ignored because they are inefficient at discriminating the documents. Other more accurate statistical filters exist but they often require additional computational cost [Salton83].

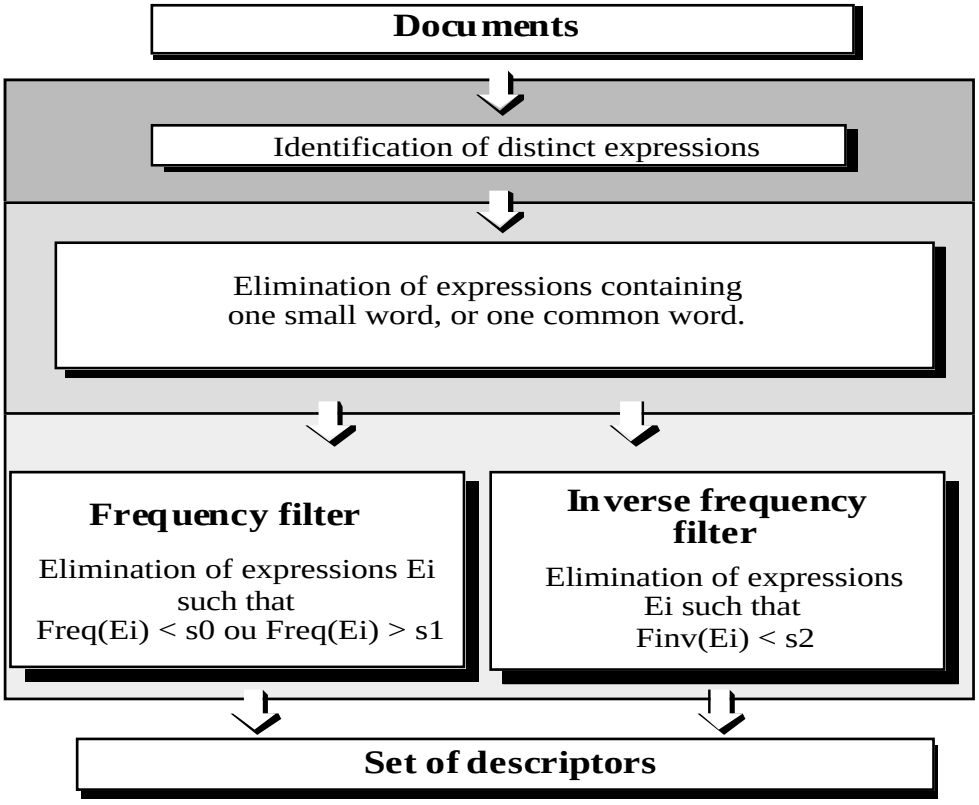

Figure 3: The extraction process

In order to use the average and inverse frequency filters mentioned above, the thresholds *s0*, *s1*, and *s2* have to be defined. We do this with the heuristic developed by [Zizi95], which is

based on a user-defined parameter (the extraction factor) and a corpus-dependent parameter (the theoretical average frequency). The extraction factor influences the degree of specificity of the descriptors obtained: the smaller the extraction factor, the more specific the descriptors. For example, table 1 contains the subset of descriptors relative to the concept of *software* that were extracted from a set of 12,000 documents. The left column shows the result of using 1139 as the extraction factor and the right column shows the result of using 55.

Extraction factor set to 1139	Extraction factor set to 55		
SOFTWARE TOOLS	SOFTWARE USER	SOFTWARE PROGRAMS	SOFTWARE ERGONOMICS
SOFTWARE SYSTEM	SOFTWARE USABILITY	SOFTWARE PRODUCT	SOFTWARE ENVIRONMENT
SOFTWARE ENGINEERING	SOFTWARE TOOL	SOFTWARE PROCESS	SOFTWARE ENGINEERS
SOFTWARE DEVELOPMENT	SOFTWARE TECHNOLOGY	SOFTWARE PACKAGE	SOFTWARE ENGINEERING
SOFTWARE DESIGN	SOFTWARE SYSTEM	SOFTWARE OBJECTS	SOFTWARE DEVELOPMENT
	SOFTWARE SUPPORT	SOFTWARE MANAGEMENT	SOFTWARE DEVELOPERS
	SOFTWARE REUSE	SOFTWARE MAINTENANCE	SOFTWARE DESIGNERS
	SOFTWARE QUALITY	SOFTWARE LIBRARIES	SOFTWARE DESIGN
	SOFTWARE PSYCHOLOGY	SOFTWARE INTERFACE	SOFTWARE COMPONENTS
	SOFTWARE PROJECT	SOFTWARE FEATURES	SOFTWARE APPLICATION

Table 1: Setting the extraction factor to influence the degree of specificity of descriptors.

We conducted an experimental evaluation of the extraction process on several subsets of documents from the HCI Bibliography Project, including the entire bibliography. The HCI Bibliography [Perlman94] is a free-access on-line corpus containing approximately 12,000 paper abstracts from the field of Human-Computer Interaction. The experiment showed that when the thresholds *s0*, *s1* and *s2* are fixed, the set of extracted descriptors tends to specialize as the number of documents grows. In other words, the description of the topics found in the web gets more accurate, but the thesaurus grows endlessly. When using the heuristic described above, the experiment showed that the number of extracted descriptors tends to stabilize. More details about the experiments can be found in [Zizi95].

4.4. CLASSIFYING THE DESCRIPTORS

The classification of the descriptors is based on a dynamic clustering technique [Diday82]. This technique requires the computation of a similarity index between pairs of topic descriptors in the web. The similarity between two descriptors T_k and T_h is computed as follows [Salton83]:

$$Sim(T_h, T_k) = \frac{\sum_{i=1}^{n} t_{ik} t_{ih}}{\sum_{i=1}^{n} t_{ik}^2 + \sum_{i=1}^{n} t_{ih}^2 - \sum_{i=1}^{n} t_{ik} t_{ih}}$$

where t_{ik} is 1 if descriptor T_k appears in document D_i and 0 otherwise and *n* is the number of documents in the web.

The dynamic clustering computes a set of classes A_1, A_2,...,A_p of the thesaurus that define a partition $\{A_1, A_2,...,A_p\}$ where each A_i contains a set of descriptors (*p* is a user-settable parameter that defaults to 12). The computation starts with an arbitrary partition $\{A_1, A_2,...,A_p\}$ and improves it incrementally until it reaches stability. At each step, the partition is improved by

computing the distance for each element of each class A_i (based on the correlation formula above) between the document and the centroid of its class and by comparing it with the distance between the document and the centroids of the other classes. If the element is found to be closer to the centroid of another class, it is moved to that class and the centroid of this class is re-computed.

In order to evaluate the quality of the classification, we have defined the *grain* of a class as the mean similarity between elements of the class: a good class has a high grain. There is a trade-off between the number of classes and the grain: a higher grain can be achieved by increasing the number of classes. Another way to obtain higher grains involves introducing a special class "Miscellaneous" to gather terms that are dissimilar with every other term. We obtained good results by combining an adaptive algorithm to compute the number of classes and the use of the Miscellaneous class [Zizi95].

Creating the thesaurus is computationally expensive. However, the thesaurus need only be created once from scratch. It can then be updated incrementally when documents are added to the initial web. When the number of documents is large, it is unlikely that a new document will change the classes of the thesaurus. Hence the cost of adding a new document can be reduced to the cost of re-computing the correlations. A full re-computation of the classes can be done after adding a large number of documents to the web. Experiments with various bodies of documents show that, in most cases, the dynamic clustering converges in a few iterations (typically 5 or fewer).

5. Visualization

5.1. LAYING OUT TOPIC MAPS

A Topic Map consists of *regions* corresponding to the descriptor classes A_i and *cities* corresponding to the descriptors. Regions and cities are laid out to reflect the importance of the topics they represent within the collection. The size of a region is proportional to the importance of the descriptors it contains and also depends on its relative importance in the collection. The importance of a descriptor is defined as the number of documents that contain this descriptor.

The regions of a Topic Map are rectangles $\{R_1, R_2,...,R_n\}$ that represent the classes $\{A_1, A_2,...,A_n\}$ of the thesaurus. Each class A_i has an importance u_i that is the sum of the importance p_k of the descriptors it contains.

Each rectangle R_i is identified by the coordinates of its upper-left corner (X_i, Y_i) and its width and height (W_i, H_i). The rectangle that represents the whole map is referred to as R. X_{top} and Y_{top} are the coordinates of its upper-left corner and W and H are its width and height.

Our algorithm successively lays out the rectangles R_i in the rectangle R. At each step the algorithm computes X_i, Y_i, W_i, H_i such that:

- the area covered by R_i, that is $A(R_i) = W_i H_i$, is proportional to its importance u_i. In other words, we have
  $$u_i / S = W_i H_i / W H$$
  where $S = \sum u_k$ for $k = 1 ... n$
- the *aspect ratio* of R_i defined as
  $$AR(R_i) = \max(W_i, H_i) / \min(W_i, H_i)$$

is as close to 1 as possible, i.e. R_i looks more like a square than a thin strip. This is because we have to lay out cities inside R_i. We have determined experimentally that an acceptable ratio is under 3/2.

At each step, the part of the rectangle R that is still uncovered is a rectangle R_{rest}. The algorithm computes the width, height and top-left coordinates for the rectangle R_i that represents the next class A_i that has not yet been processed. This layout is obtained by cutting either the width or height of R_{rest} proportionally to the size of A_i. If the aspect ratio of the resulting rectangle is not acceptable, A_i is gathered with A_{i+1}, A_{i+2}... A_{i+k} until the aspect ratio of the corresponding rectangle is satisfying. In this case the algorithm is called recursively on rectangle R_i with classes $\{A_{i+1}, A_{i+2}... A_{i+k}\}$.

Although this algorithm can lead to rectangles that do not match the aspect ratio, our evaluation showed that when the classes are processed in decreasing order of importance, the average frequency of such rectangles is under 2% [Zizi95].

Once the regions are laid out, we can lay out icons representing the cities in the regions. In a Topic Map, each city represents a descriptor of the thesaurus; the importance of the city reflects the number of documents to which the descriptor is assigned. A threshold value *e* can be defined by the user in order to control the importance of the descriptors to be displayed. The set V_i of icons representing descriptors T_k that appear on the map in a region A_i is defined as the set of T_k in A_i such that $u_k > e$.

The icons representing descriptors in V_i are laid out in the rectangle R_i such that the most important city is at the center of the rectangle and other cities are laid out on ellipses according to their importance. This layout offers a convenient way to spread out the cities in each region according to their importance, with the "capital" of the region at its center. The coordinates of the center of rectangle R_i and hence the position of the biggest city are:

$X_o = X_i + W_i / 2$

$Y_o = Y_i + H_i / 2$

The other cities are laid out on ellipses centered at (X_o, Y_o) with axis:

$a_k = ((m - p_k) / m) (W_i / 2)$

$b_k = ((m - p_k) / m) (H_i / 2)$

with $m = \max (p_k)$ for *k* such that v_k is in V_i

Hence the coordinates of city v_k are:

$X_k = X_o + a_k \cos (q_k)$

$Y_k = Y_o + b_k \sin (q_k)$

with q_k varying according to the number of cities so as to obtain a spatial distribution.

Finally, the roads of the map represent pairs of descriptors whose similarity is higher than a given threshold.

Figure 1 is a hard-copy of a Topic Map computed by the algorithm outlined above. Each class corresponds to a rectangle. The figure clearly shows that descriptors in the same class are more correlated than descriptors in different classes, which illustrates the effectiveness of the dynamic clustering.

The layout of the IDM is computationally efficient. Therefore, the overall cost of the generation of a Topic Map is the generation of the thesaurus and, more precisely, the dynamic clustering part.

5.2. LAYING OUT DOCUMENT MAPS

In a Document Map, cities are documents that are represented by icons. The documents may be the result of a query or they may have been selected manually by users or designers. The map is drawn according to the following conventions:

- Roads between cities correspond to special relationships between the documents represented by the cities, e.g. *comment*, *contains reference to*, *to be read before*. In particular, they can be the links of a traditional hypertext system.
- Cities representing similar documents are laid out close to each other on the map while cities representing very different documents are laid out far away from each other.
- The size of the icon depicting a document reflects its importance according to either its relevance to a given query or its subjective importance as set by users or designers (depending on the origin of the map).

The main difficulty in achieving this layout is to reflect the similarity between documents as distances on the map. We have adapted a general algorithm for laying out graphs [Kamada89]. This algorithm maps the graph into a physical system made of particles and springs. Each node is mapped to a particle and a spring is set between each pair of particles. The strength and default length of each spring depends on the length of the shortest path between the two nodes corresponding to the particles it is attached to. A spring corresponding to nodes that are directly connected in the original graph has a length of one and a strong strength. Springs corresponding to non-directly connected nodes in the original graph have a longer default length and a weaker strength. The layout of the nodes corresponds to the positions of the particles when the spring system is in a stable state. The algorithm first initializes the positions of the particles and then computes the energy of the system. It then decreases the energy step-by-step, by moving the particle with highest energy to a stable position. When the energy of the system reaches a minimum, the algorithm stops. We have added an optional final pass that moves the nodes so as to avoid or minimize the overlap of the icons representing the cities.

The algorithm does not guarantee that the distances in the layout match the similarity relationship, but in practice it generally gives good results: overall, similarities are roughly reflected in the final display. Furthermore, customization and re-usability enable users to have final control of the layout. Thus if a misleading layout were to occur, the map could be edited and saved with a more appropriate layout. This possibility illustrates the benefit of treating maps as documents.

Figure 2 shows a Document Map computed with this algorithm. While the layout of a Topic Map is computationally efficient, the algorithm used here to lay out Document Maps is expensive, especially for the first layout. The time to complete the layout is polynomial in the number of nodes in the graph. This is not a problem since we mostly use Document Maps to display relatively small sets of documents. We have not yet focused on performance issues concerning the layout of very large Document Maps. However, we expect to improve performance by giving better initial positions to the nodes, since this dramatically improves the efficiency of the layout algorithm.

5.3. LINKS VISIBILITY

Both Topic Maps and Document Maps use links to display the relationships between their items. As shown in figure 4, displaying all possible links is not satisfactory. We have

experimented with two strategies to solve this problem. One strategy uses the types and weights of the links to filter out irrelevant or non-important links. As show in figure 5, this strategy can reduce the number of links dramatically; however it is still difficult to follow the relationships between the documents.

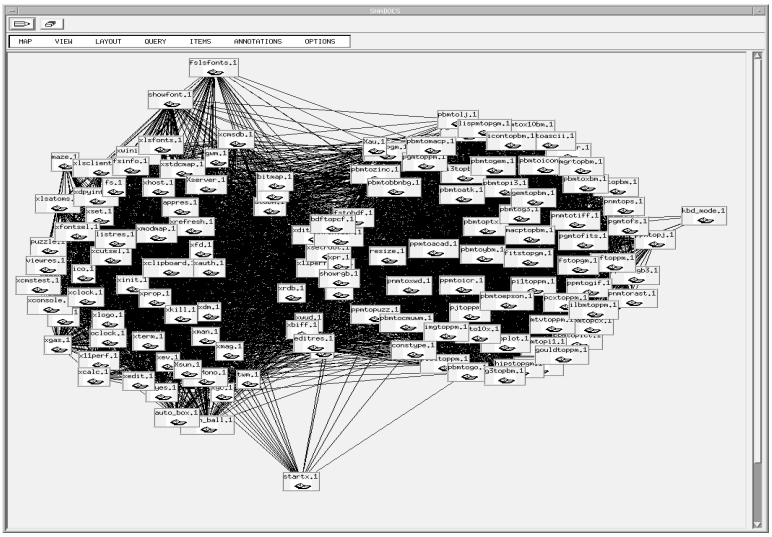

Figure 4: The problem when showing all the links in a Document Map

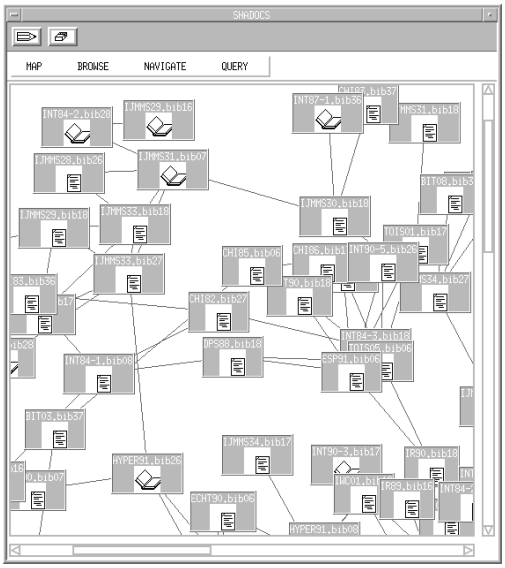

Figure 5: Showing links according to their type and weight

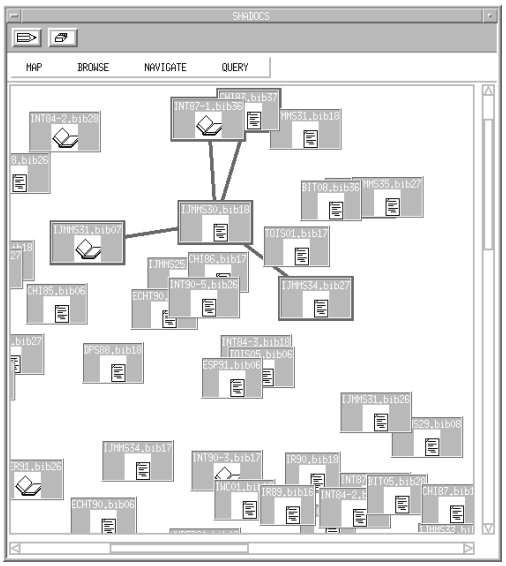

Figure 6: Showing links around the user's focus

The other strategy uses a dynamic approach: all links are hidden, except those around the user's focus (figure 6). This strategy is much more promising, because the dynamic aspect of the display (link appearing and disappearing as the user's focus changes) makes it easier to

understand the relationships between the nodes by transferring part of the cognitive load to the perceptual system.I

6. Navigation

DMs support two forms of interactive navigation, *browsing* and *querying*, which are achieved by direct manipulation of the maps. IDMs also support customization, which aids navigation tasks and increases the sense of engagement by the users.

Browsing is used to explore an unknown set of documents or when users do not know how to describe what they are looking for. We have defined several levels of browsing, akin to using maps at various scales. Large-scale maps provide an overview at the expense of detail, while smaller scale maps provide increased detail at the expense of context. Fortunately, computer-generated maps provide more facilities than traditional paper maps. The scale can be defined by the user, and it can vary for a given map over time (e.g. zoom) and space (e.g. fisheye views [Furnas86]).

We have defined three levels of browsing: the wide browsing level uses *semantic fisheye views*, the medium browsing level uses a *semantic zoom*, and the narrow browsing level uses *multiple windows*. Using multiple windows consists of displaying full documents in separate windows, a very traditional approach. Therefore in the next sections we focus on the semantic fisheye and semantic zoom.

6.1. SEMANTIC FISHEYE VIEWS

The wide browsing level is intended for fast scanning over a web of documents. Few details are shown about the items except under the focus area, which displays more information. To achieve this, we use fisheye views (figure 7) and scaling.

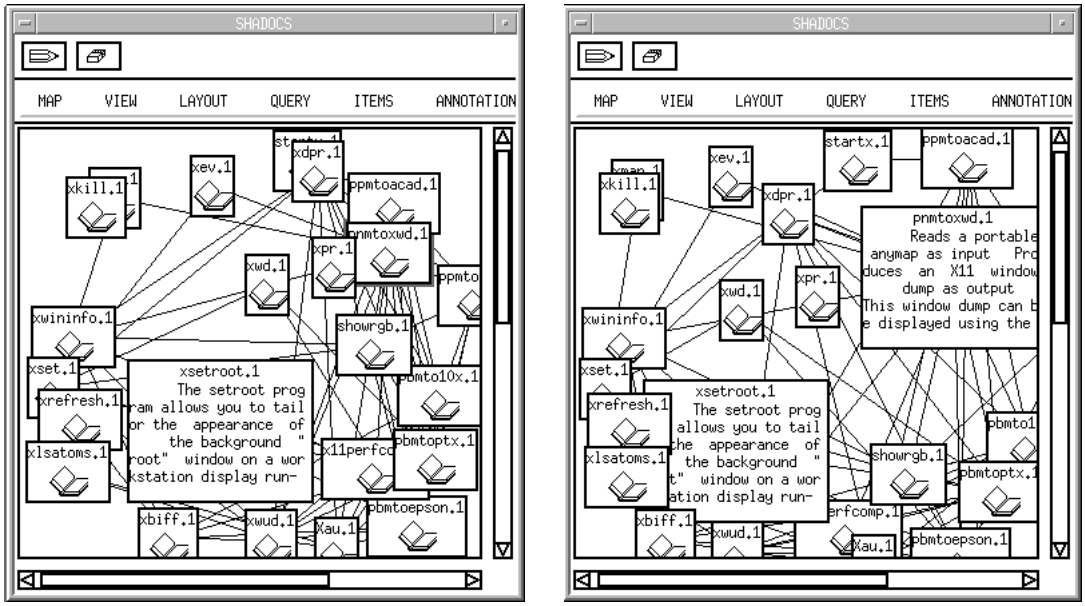

Figure 7: Fisheye views on the Document Map of figure 2

Fisheye view transformations make it possible to increase the size of parts of the map without losing the remaining items. With such a transformation, the position and size of each item is computed from the normal view, the distance between the focus and the given item and its *a priori* importance [Sarkar92]. The initial layout of the map serves as the normal view and the initial sizes of items serve as their *a priori* importance. The focus is set at a position defined by the user. Ideally, the focus should follow the mouse pointer but our current software and hardware platform does not support a sufficiently rapid response time.

We combine the fisheye view transformation with a semantic scaling transformation: the type of information that is visible for a given item varies with the size that was computed by the fisheye transformation for that item (hence the term *semantic fisheye view*). For example (figure 8), an item with a small display size shows a name and icon, while the same item with a larger display size shows the first few lines of the document. With a medium display size, the item shows a more detailed icon and additional information such as its relevance (number of stars).

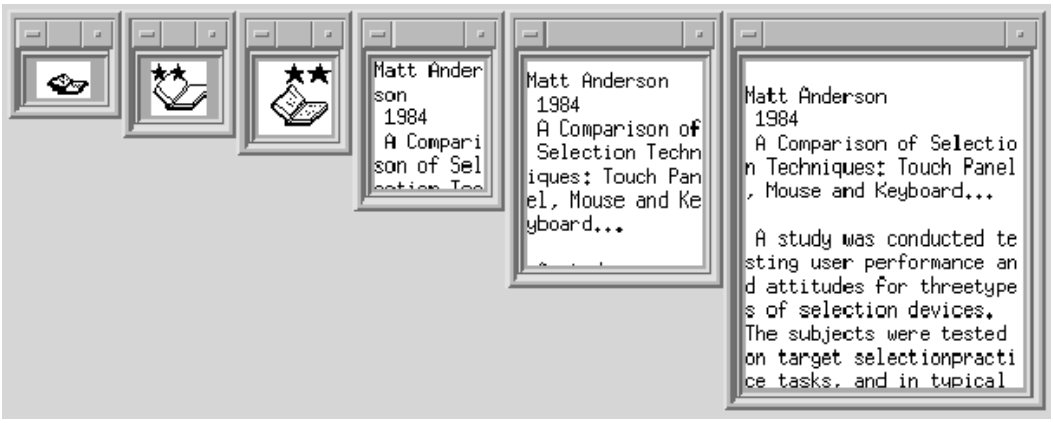

Figure 8: Displaying the same document at various levels of semantic zoom

Other types of fisheye views have been proposed [Noik93, Schaffer94]. These techniques have been designed for hierarchically structured documents. Since our documents are weakly structured, we found that the semantic fisheye view was more appropriate.

6.2. SEMANTIC ZOOM

The medium browsing level is intended for more detailed exploration. To achieve this, we allow users to zoom in on items of interest (figure 9). Unlike fisheye views, zooming reduces the level of context but enables the user to see a larger number of items in greater detail. Here too we use the semantic scaling mechanism to reveal various levels of detail according to the display size of each item (hence the term *semantic zoom*). The size depends upon each item's *a priori* size and the current zoom factor.

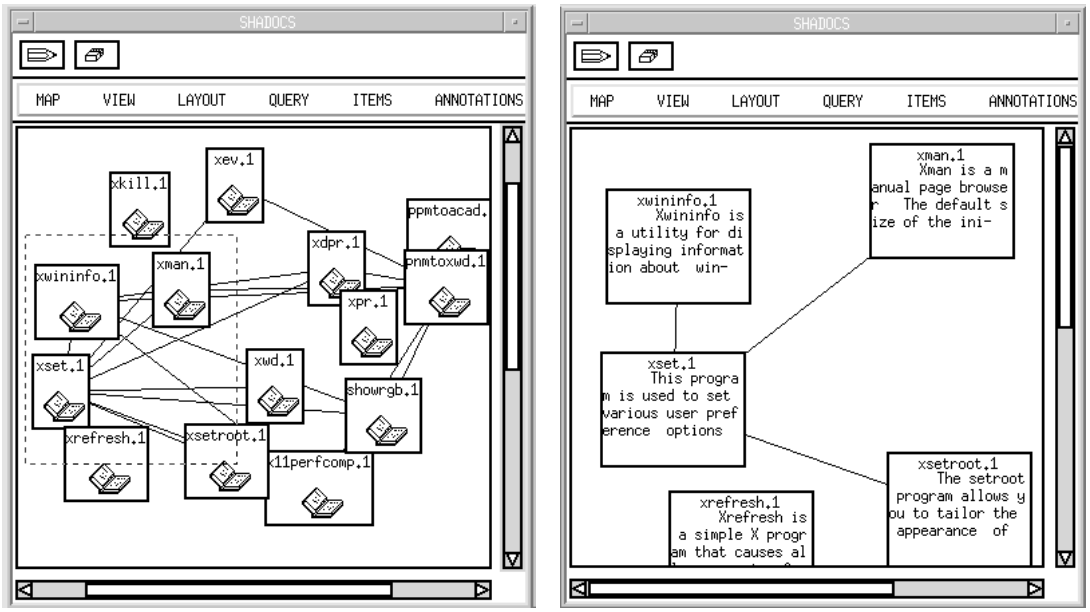

Figure 9: Zooming in on a part of the Document Map of figure 2

6.3. GRAPHICAL QUERY LANGUAGE

Like [Fox91], we consider querying as a form of navigation and we integrate querying and browsing in the same framework. Queries can be issued by selecting items and links directly on Topic Maps (figure 10) or on Document Maps. The selected items are translated into a boolean query according to simple rules: an OR operator is used between every two selected items unless a link between these items is selected, in which case an AND operator is used (figure 10). A query is then interpreted by using information retrieval techniques. The query is represented in the system by a vector $Q = (x_1, x_2,...,x_n)$ where *n* is the number of descriptors in the thesaurus. Each x_i is 1 if descriptor D_i is among the selected items of the map and is otherwise 0. For a Topic Map, non-null coordinates correspond to the topics selected on the map. For a Document Map, the non-null coordinates correspond to all the descriptors used in the representation of the documents selected on the map. This latter case is a type of query-by-example, which provides a useful extension to query languages for information retrieval [Stanfill91, Khale89].

.Empirical studies [Golovchinsky93] have also shown that graphical queries are easier to issue than textual queries and that users with relatively low computer experience are significantly better at expressing queries graphically rather than syntactically. Our query language is very similar to that of QRL [Golovchinsky93], except that in QRL query items are selected from the words of a document whereas in our system they are selected from maps. Our approach has several advantages. First, queries can be automatically expanded by using the thesaurus embodied in the Topic Map and the expansion can be easily controlled by the users. Second, the same graphical query mark-up technique can be used in Document Maps and in Topic Maps, providing consistency in the interface. Users can issue an initial query from a Topic Map and then refine it by querying the resulting Document Map. Finally, Topic Maps provide global views that help users better understand which topics can be found in the documents before

they actually issue a query. In contrast, QRL uses local views: the document from which the topics of a query are selected can be seen as a local view on the topics of the whole collection. We believe that these two approaches (global and local) correspond to different types of browsing tasks and therefore could be combined.

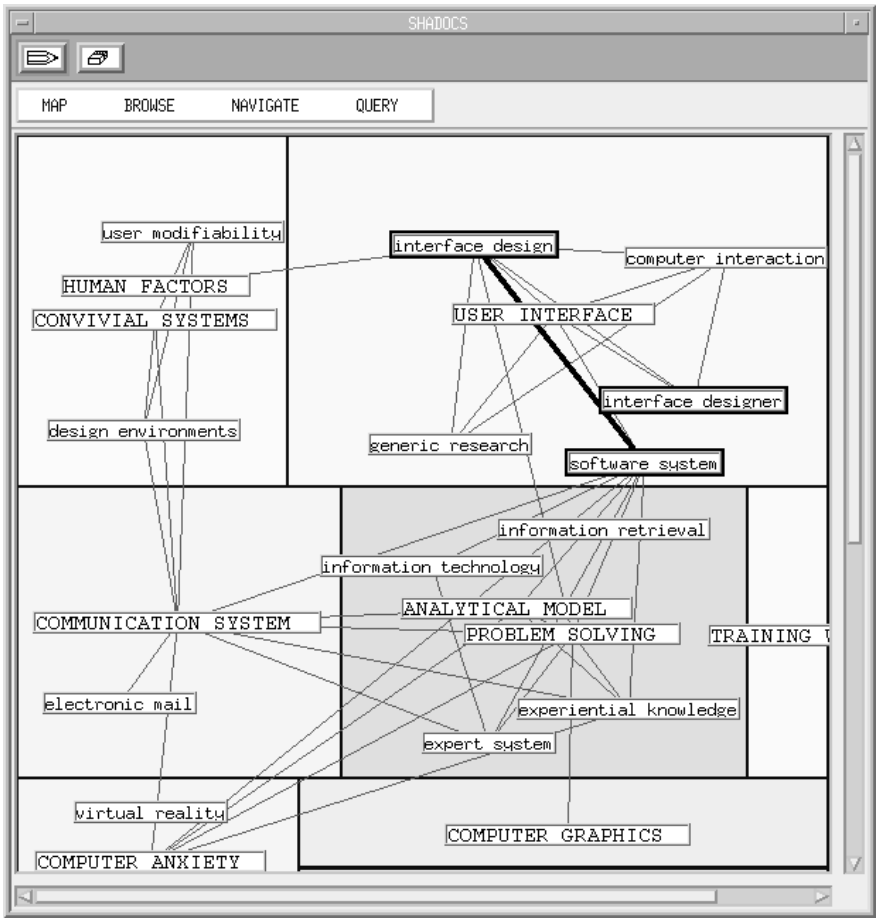

Figure 10: Expressing a query from a Topic Map. The query expressed above is *("interface design" AND "software system") OR "interface designer"* .

Other graphical query languages such as Graphlog [Consens94] or Gram [Amann94] have been developed on top of databases systems. These systems offer powerful query languages for querying structured data, such as logic programming. However graphical query languages based on information retrieval are more appropriate for handling topic searches within large amounts of weakly-structured text

Thesaurus driven query expansion

Information retrieval systems traditionally base query expansion on thesauruses or association lists. Topic Maps not only support thesauruses, but they also provide an immediate feed-back of queries as they are expanded. Users can control the expansion process by modifying selected items before the expanded query is processed. By permitting user-assisted query

expansions the system is less likely to expand queries in irrelevant directions as often happens with "blind" automatic query expansion systems.

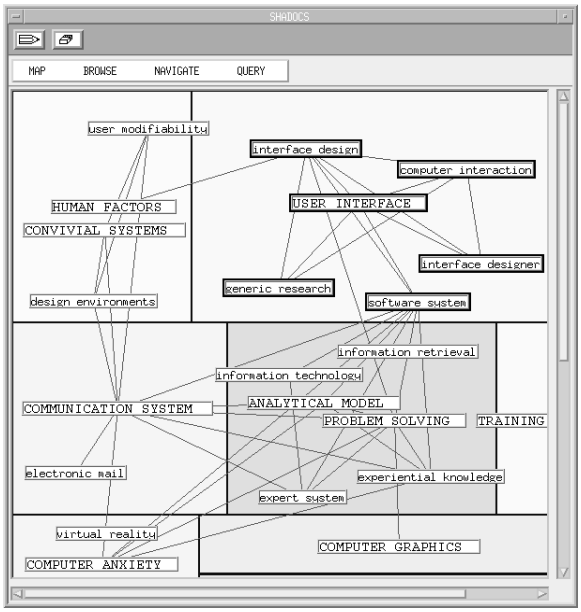

Figure 11: Extending a query to the enclosing regions

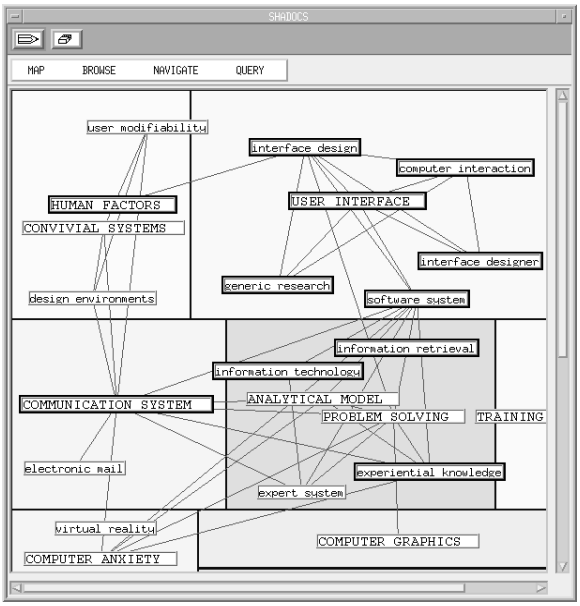

Figure 12: Extending a query to the closest neighbors

Interactive dynamic maps support two expansion strategies: a query can be expanded to include all the cities that lie in the thesaurus classes of the items of the original query (figure 11) or it can be expanded to include the cities directly connected to these items (figure 12). The first type of expansion is useful when the selected topics lie in a few classes of moderate importance while the second type is useful when the selected topics lie in a larger number of classes or in classes of larger importance.

7. Customization and re-usability of past search results

We consider customization to be an integral part of navigation, like querying. Customization allows users to suit maps to their needs and taste. The layout of a map can be changed with traditional editing commands and annotations of various kinds can be added. Customization helps users localize themselves: annotations act as landmarks that users remember and recognize easily. For example, readers can mark their own paths, annotate items, add post-it notes, etc. These annotations then serve as clues during further navigation by the owner of the map as well as by other users sharing the same IDM.

Maps are customizable documents that can be edited, stored in a library of maps, and exchanged among users. Maps can also be made active: an *active map* results from a query and is re-computed at regular intervals from the initial query, thereby tracking and filtering changes in the web.

8. Prototype system

In order to evaluate the concept of Interactive Dynamic Maps, we have implemented a prototype system called SHADOCS (SHAring DOCument System) [Zizi94]. All the figures of this article are hardcopies of displays generated by SHADOCS.

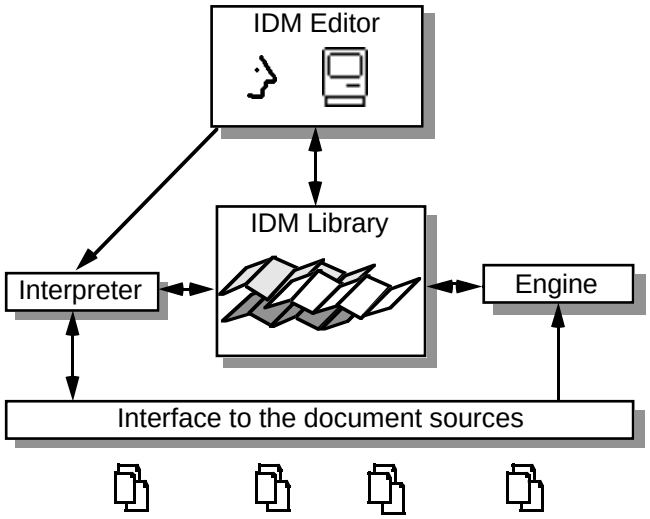

Figure 13: The architecture of SHADOCS

Figure 13 shows the architecture of SHADOCS. At the heart of the system is the *IDM library* which stores IDMs. The *engine* computes maps from the documents stored in the different document sources (as described in section 4). The *IDM editor* acts as the interface between the system and the users, who can view, customize and store IDMs in the library. Users can also issue queries that are sent to the *interpreter* to create new maps. The interpreter is also used to implement active maps by regularly querying the document sources. The *interface to the document sources* accesses the documents themselves. Since documents may be stored in various formats and may be accessible through various protocols (e.g. WAIS, Gopher, World-Wide Web, FTP), this component shields SHADOCS from dependencies on the document sources.

The prototype system is implemented with the O2 object-oriented database management system [O2, Delobel91]. We have used a local O2 database in this prototype to act as the interface to the different document sources. We are also developing interfaces to remote document sources.

9. Lessons learned

In order to evaluate the effectiveness of IDMs for exploring document sets, we have put several sets of documents in SHADOCS: a set of manual pages from the X Window System (62 documents), various subsets of the HCI Bibliography Project [Perlman94], including a set of 500 documents and a set of 4344 documents, and the whole bibliography (12041 documents). We have used these sets for a number of experiments [Zizi95] to evaluate the extraction of the descriptors, the classification, the layout algorithms and the query system. These experiments, together with informal evaluations with users, have led us to three major observations.

The number of descriptors. Without filtering, around 170,000 descriptors (two-word expressions) are extracted from the set of 12,041 documents from the whole HCI bibliography. With a simple filter based on the average number of occurrences of each descriptor, the number of descriptors drops to 5500, which is still far too many for a human user. This led us to develop an adaptive filtering strategy that is based on the number of descriptors we want to extract, using the extraction factor (section 4.3).

The quality of the dynamic clustering. The classes are quite significant with respect to the content of the documents, especially when using a "Miscellaneous" class. In other words, the items of a class exhibit high similarities and are very representative of the documents they were extracted from. An important parameter appears to be the number of classes in the partition, although this parameter seems to be less important when the number of descriptors grows. The notion of grain of a class (section 4.4) enabled us to dynamically adapt the number of classes to the descriptors extracted from a particular set of documents. The Miscellaneous class tends to decrease the initial number of classes in the partition, since several classes from the initial partition can often be emptied by applying the heuristic. This tends to counteract with the strategy that consists of increasing the number of classes to increase their grain size.

The level of interaction . The spatial metaphor appears effective and, as already noted in previous works (e.g., [Card91]), dynamic displays are helpful in understanding the content of an information space. In addition, the ability to edit and customize maps contributes to a better understanding of the set of documents. For example, moving items allows users to make the display more pleasing while giving a better sense of the dependencies among items. This encourages exploration by trial and error, which in turn is supported by the navigation features. At the same time, users are less likely to become disoriented, because of the stability of the display and their ability to create annotations and set landmarks.

10. Conclusion and Future Work

We have presented a new approach to hypermedia exploration that builds upon previous work in Information Retrieval, Information Visualization, and Human Computer Interaction. This approach is based on the concept of Interactive Dynamic Maps (IDMs). An IDM is a view of a web of documents, computed from the documents themselves, that users can interact with to navigate through and query the web of documents. The concept of an IDM has been implemented in a prototype system called SHADOCS, which we have used to conduct various experiments.

Next steps include improving and extending the system, including improving the computation of IDMS, improving the layout algorithms for visualization and perhaps adding 3D, and decreasing the interaction response times. We also plan to conduct additional user tests in real settings in order to evaluate the effectiveness of this approach. Finally, we would like to extend this to multimedia in addition to text documents. This will pose particular problems when automatically identifying descriptors. As a starting point, image analysis and signal processing techniques can be used to identify meaningful components in audio and video documents. Although this aspect is still an open problem, we believe that it will deserve more and more attention as electronic documents move from hypertext to hypermedia.

Acknowledgments

This work is partially supported by CNET-CNRS project Cesame. We also thank O2 Technology for their support and Wendy Mackay for her help with earlier versions of this article.

References

[Acksyn88] Robert M. Acksyn, Donald L. McCracken, and Elise A. Yoder. Kms: a distributed hypermedia system for managing knowledge in organizations. *Communications of the ACM*, 31(7):820-835, july 1988.

[Ahlberg94] Christopher Ahlberg and Ben Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with Starfield Display. *Proc. Human Factors in Computing Systems (CHI'94)*, ACM Press, April 1994, pages 365-371.

[Bernstein88] Mark Bernstein. The bookmark and the compass: orientation tools for hypertext users. *ACM SIGOIS Bulletin*, 9(4):34-45, october 1988.

[Bernstein91] Mark Bernstein, Franca Garzotto, Paolo Paolini, and Daniel Schwabe. Tools for designing hyperdocuments. In *Hypertext, Hypermedia Handbook*, pages 179-209. Mc Graw Hill, 1991.

[Bourigault92] Didier Bourigault, Lexter, un Logiciel d'Extraction De Terminologie, Colloque international de TermNet, Avignon, France, 1992.

[Bush45] Vanevar Bush. As we may think. *The Atlantic Monthly*, 176(1):101-108, July 1945.

[Card91] Stuart K. Card, George G. Robertson and Jock D. Mackinlay. The Information Vosualizer: an information workspace. *Proc. Human Factors in Computing Systems (CHI'91)*, ACM Press, April 1991, pages 181-188.

[Chignell93] Chignell, M. , Golovchinsky, G. , Poblete, S. and Zuberec, S., Information Visualization and Interactive Querying for Online Documentation and Electronic Books, ACM Hypertext '93, 1993.

[Consens94] M.P. Consens, F. Ch. Eigler, M.Z. Hasan, A.O. Mendelzon, E.G. Noik and A.G. Ryman and D. Vista, Architecture and Applications of the {H}y+ Visualization System, IBM Systems Journal, 33 (1), 1994.

[Crouch89] D.B. Crouch, C.J. Crouch, and G. Andreas. The use of cluster hierarchies in hypertext information retrieval. In *Proc. Hypertext'89*, pages 225-237, Pittsburgh, Pennsylvania, November 1989. ACM.

[Delobel91] Claude Delobel, Christophe Lecluse, and Philippe Richard. *Bases de données : des systèmes relationnels aux systemes à objets*. InterEditions, 1991.

[Diday82] E. Diday, J. Lemaire, J. Pouget, and F. Testu. *Eléments d'analyse de données*. Dunod, 1982.

[Foss89] Carolyn L. Foss. Detecting lost users: Empirical studies on browsing hypertext. Technical Report 972, INRIA, FRANCE, Février 1989.

[Fox91] E.A. Fox, Q.F.Chen, and R.K. France. Integrating search and retrieval with hypertext. In *Hypertext, Hypermedia Handbook*, pages 179-209. Mc Graw Hill, 1991.

[Furnas86] George W. Furnas. Generalized fisheye views. In *Proc. Human Factors in Computing Systems (CHI'86)*. ACM Press, 1986.

[Halasz87] F. G. Halasz, T. P. Moran, and R. H. Trigg. Notecards in a nutshell. In *Proc. Human Factors in Computing Systems (CHI'87)*, pages 45-52. ACM Press, 1987.

[Kamada89] T. Kamada and S. Kawai. An algorithm for drawing general indirect graphs. *Information Processing Letters*, 31(1):7-15, 1989.

[Khale89] B. Khale. Wide area information server. Technical Report 202, Thinking Machine Corporation, 245 First Street, Cambridge MA 02154, 1989.

[Larson92] R. R. Larson. Evaluation of retrieval techniques in an experimental on-line catalog. *JASIS*, 43(1):34-53, Jan 1992.

[Mackinlay91] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The Perspective Wall: Detail and context smoothly integrated. *Proc. Human Factors in Computing Systems (CHI'91)*, ACM Press, April 1991, pages 173-179.

[Meyrowitz86] B.J. Meyrowitz, K. E. Smith, and L. N. Garrett. Intermedia: Issues, strategies and tactics in the design of a hypermedia document system. In *Proc. ACM Conference on Computer-Supported Cooperative Work (CSCW'86)*, pages 163-174. ACM, 1986.

[Nielsen90] Jakob Nielsen. *Hypertext and Hypermedia*. Academic Press, 1990.

[Noik93] Emmanuel G.Noik, Exploring Large Hyperdocuments: Fisheye Views of Nested Networks, p 192-206, Proceedings Hypertext'93, ACM Press, 1993.

[O2] O2 technology, Versailles France. *O2 Reference Manual*.

[Padmini92] S. Padmini. Thesaurus construction, in *Information Retrieval: Data Structures and Algorithms*, W.B. Frakes and R. Baeza-Yates (Eds), Prentice Hall, 1992.

[Perlman94] Gary Perlman et al. The HCI bibliography project. Technical Report, Department of Computer and Information Science, The Ohio State University, 1994. Available on-line at ftp archive.cis.ohio-state.edu or http://hyperg.tu-graz.ac.at:80/230C57BD/Chcibib

[Rivlin94] Ehud Rivlin, Rodrigo Botafogo, and Ben Shneiderman. Navigation in hyperspace: Designing a structure-based toolbox. *Communications of the ACM*, 37(2):87-96, February 1994.

[Robertson91] George G. Robertson, Jock D. Mackinlay and Stuart K. Card. Cone Trees: Animated 3D visualizations of hierarchical information. *Proc. Human Factors in Computing Systems (CHI'91)*, ACM Press, April 1991, pages 189-194.

[Rodrigo92] Rodrigo A. Botafogo, Ehud Rivlin and Ben Shneiderman, Structural Analysis of Hypertext: Identifying Hierarchies Useful Metrics, *ACM Transactions on Informations Systems*, vol 10 (2), 1992

[Salton83] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw Hill, 1983.

[Sarkar92] Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. In *Proc. Human Factors in Computing Systems (CHI'92)*, pages 83-91. ACM Press, May 1992.

[Schaffer93] Doug Schaffer, Z. Zuo, L. Bartram, J. Dill, S. Dubs, S. Greenberg and M. Rosenberg, Comparing Fisheye and Full-Zoom Techniques for Navigation of Hierarchically Clustered Networks, Graphics Interface'93, p. 87-96, 1993.

[Shneiderman92] Ben Shneiderman. Tree visualization with tree-maps: a 2D space-filling approach. *ACM Transactions on Graphics*, 11(1):92-99, January 1992.

[Stanfill91] Graig Stanfill. Massively parallel information retrieval for wide area information servers. Technical report, Thinking Machines Corporation, 245 First Street, Cambridge MA 02154, 1991.

[Yankelovich88] N. Yankelovich, B.J. Meyrowitz, and S. M. Drucker. Intermedia: the concept and construction of a seamless information environment. *IEEE Computer*, 31(1):81-96, January 1988.

[Zizi94] Mountaz Zizi and Michel Beaudouin-Lafon. Accessing hyperdocuments through interactive dynamic maps. In *Proc. European Conference on Hypermedia Technology*, ACM, 1994.

[Zizi95] Mountaz Zizi. Cartes Dynamiques Interactives, une métaphore spatiale pour l'exploration des espaces informationnels, PhD thesis, LRI, Université de Paris-Sud, February 1995.

Improving drag-and-drop on wall-size displays

Maxime Collomb, Mountaz Hascoët
LIRMM, UMR 5506 CNRS
Univ. Montpellier II
Montpellier, France
{collomb, mountaz}@lirmm.fr

Patrick Baudisch
Microsoft Research
One Microsoft Way
Redmond, WA 98102
baudisch@microsoft.com

Brian Lee
Stanford Computer Graphics
Laboratory, Stanford University
California, United States
balee@graphics.stanford.edu

Abstract

On wall-size displays with pen or touch input, users can have difficulties reaching display contents located too high, too low, or too far away. Drag-and-drop interactions can be further complicated by bezels separating individual display units. Researchers have proposed a variety of interaction techniques to address this issue, such as extending the user's reach (e.g., *push-and-throw*) and bringing potential targets to the user (*drag-and-pop*). In this paper, we introduce a new technique called push-and-pop that combines the strengths of push-and-throw and drag-and-pop. We present two user studies comparing six different techniques designed for extending drag-and-drop to wall-size displays. In both studies, participants were able to file icons on a wall-size display fastest when using the push-and-pop interface.

Keywords: drag-and-pop, push-and-throw, wall-size display, drag-and-drop, pen input, touch-screen, interaction technique.

1 Introduction

With the emergence of wall-size displays (e.g., Liveboard [6] and SMART Board™), touch and pen input have regained popularity. Over the past years, more complex display systems have been created by combining multiple display units into wall-size display walls, such as the DynaWall [25], the iRoom SMART Board wall (iWall [14], shown in Figure 4), as well as display systems based on stitched projection, such as the Information Mural [11].

Touch/pen input requires users to physically display content in order to interact with it. This can become a problem when targets are out of reach, e.g., because they are located too high or too low or on a display unit that does not support touch/pen input [3]. Accessing icons located far away from the user may require users to physically walk over, requiring a target acquisition time roughly linear to distance [9]. Interactions that involve dragging objects tend to be particularly error-prone [22] and can be complicated further by the bezels separating screen units [3].

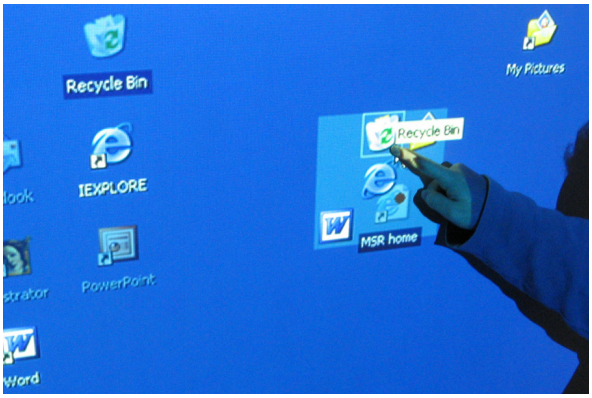

Figure 1: A user is dragging a web page icon into the recycle bin on a wall-size display. The proposed push-and-pop interaction technique has created a world-in-miniature around the user's finger that contains all valid target icons.

Researchers have proposed a variety of interaction techniques that simplify drag-and-drop from and to inaccessible screen locations, across long distances, and across display unit borders. Approaches include extending the user's reach (e.g., *push-and-throw* [12]) and bringing potential targets to the user (*drag-and-pop* [3], shown in Figure 4). Both techniques have their particular strengths, while facing their particular limitations, as we will discuss in detail in sections 3 and 4. We also present improvements addressing some of these limitations.

We then present a novel technique we call *push-and-pop* (Figure 1) that combines the world-in-miniature aspect from push-and-throw with the target-to-pointer approach of drag-and-pop. In two experimental comparisons with five competing drag-and-drop extension techniques, participants performed tasks fastest when using the push-and-pop interface.

2 Related work

Drag-and-drop is a well-known interaction technique for transferring or copying information using a pointing device, while avoiding the use of a hidden clipboard [26]. Several techniques have been proposed to adapt drag-and-drop, initially invented for use with a desktop-sized computer screen, to large screens.

The majority of work so far has focused on drag-and-drop with direct input devices, such as the mouse. Hyperdragging [21] allows extending drag-and-drop across physically separate displays. Manual And Gaze Input Cascaded (MAGIC) pointing [27] helps overcoming long distances by combining mouse input with gaze input. High-density cursors help users keep track of the mouse pointer during long-distance traversals [2]. Semantic pointing [4] and vector pointing [9] use a multi-scale navigation approach to allow users to cross very long distances with logarithmic access time [10].

Laser pointers [19] have been discussed for input on wall-size displays, but were found to be slower than touch input [18].

Techniques compatible with pen usage are based on a variety of different approaches. Pick-and-drop [22] and take-and-put [25] are based on point-and-click, but, unlike traditional drag-and-drop, they do not require users to maintain contact with the screen. The Frisbee technique by Kahn et al. [7] allows users to create a local view into a distant area of the screen, thereby allowing users to drag objects to arbitrary targets. These techniques also help users overcome obstacles, such as bezels separating display units. Marking menus [15] allow a user to perform a menu selection by either popping-up a radial (or pie) menu, or, for an experienced user, by making a straight mark in the direction of the desired menu item without popping-up the menu.

Other techniques offer additional functionality for interacting with targets that are out of reach. Throwing, for example, allows users to accelerate an object with a small gesture; the object then continues its trajectory based on its inertia [8]. The imprecision of human motor skills has prevented throwing from being used for reliable target acquisition.

Push-and-throw [12] and drag-and-pop [3] also fall into this category. In this paper, we focus on these two techniques and present extensions and design improvements. We describe these techniques in detail in the following section.

3 Push-and-throw and drag-and-pop

3.1 Drag-and-throw & push-and-throw

Drag-and-throw [12] was designed to address the limitations of throwing [8] by providing users with a real-time preview of where the dragged object will come down if thrown. This way, drag-and-throw allows users to tweak the throwing trajectory in order to assure that the right target is hit. Snap-to-target helps increase users' accuracy.

Push-and-throw [12] is a follow-up version of drag-and-throw. By letting users drag towards the target rather than away from it, push-and-throw was found to offer better affordance [12].

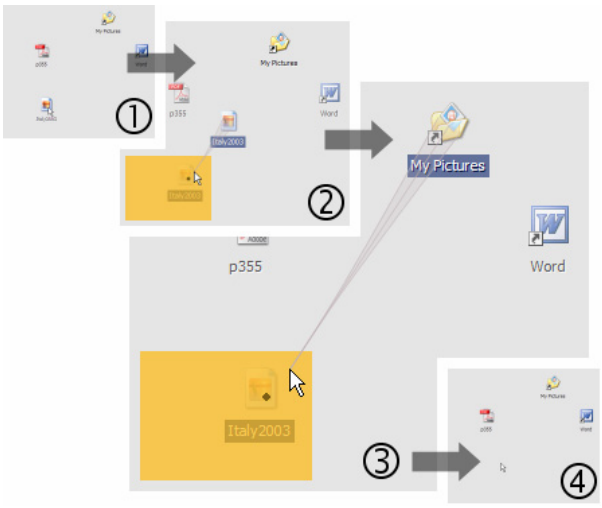

Figure 2: Push-and-throw walkthrough: the user is dropping the image icon located in the bottom left into the "My Pictures" folder located at the top right.

Figure 2 shows a walkthrough of push-and-throw. (1→2) Tapping the pen onto an icon causes the push-and-throw visuals to appear. The translucent rectangle, called the *take-off area*, represents a miniature of the desktop space. A translucent copy of the icon (the "tip icon") appears. It is connected to the cursor using a rubber band, which may be thought of as preview of the trajectory along which the icon will be "thrown" when released. Dragging the pen inside the take-off area causes the tip icon to move to the respective location on the real desktop. (3) As the user moves the tip icon over a target icon, such as a folder, the target icon provides the usual visual feedback, i.e., it changes its color. (4) When lifting the pen, the icon is filed.

Push-and-throw, originally inspired by the metaphor of the pantograph [5] is a combination of a go-go [20] and a world in miniature technique [24]. The main idea behind push-and-throw is to temporarily turn the pen/touch input, inherently a direct pointing device, into an indirect pointing device in order to traverse distances faster and to be able to reach locations further away or on a different screen unit. On the downside, it also leads to reduced resolution and accuracy.

Design improvements: rubber bands & acceleration

While the original version of push-and-throw [12] used simple lines to connect pointer and dragged icon, we created an improved version that uses a tapered rubber band inspired by drag-and-pop [3] as shown in Figure 2. As Baudisch et al. discuss, the shape of this type of rubber band provides users with an additional visual cue about distance. Unlike the rubber bands proposed by Baudisch et al., we used an asymmetric rubber band for push-and-throw, which looked more consistent

when linking the pointer (1 pixel wide) to an icon (32 pixels wide). This improved design was one of the interfaces compared in our user studies presented in Section 6 and 7.

One of the main limitations of the original push-and-throw is its lack of precision due to the size reduction that occurs when mapping the desktop to the take-off area. We address this issue by introducing non-linear acceleration, as it is common with indirect input devices, to push-and-throw. In *accelerated push-and-throw*, moving the pointer slowly results in a much slower motion of the dragged icon, helping users acquire small targets. In addition, the acceleration factor is reduced when the dragged icon is close to a target (similar to semantic pointing [4]). Accelerated push-and-throw also allows *clutching*, i.e., lifting and repositioning the pen/finger within a drag interaction. This allows users to reach very distant targets.

With acceleration, there is no more immediate correspondence between physical pointer location and the location of the dragged icon. As a consequence, the technique does not have a clearly defined take-off area anymore and we cannot provide a preview of it. Since accelerated push-and-throw therefore does not completely subsume tradition push-and-throw, we decided to include both designs in our user study.

3.2 Drag-and-pop

Drag-and-pop [3] uses the opposite approach to drag-and-throw. Rather than sending the dragged object to the periphery, it allows users to bring a selection of likely candidates to the user. This allows users to complete drag interactions in a convenient screen location. There is no scaling of pointer motion, so users can make use of the full resolution of their motor skills.

Figure 3 shows a walkthrough of *drag-and-pop*. (1) The user intends to delete a web page by dragging it into the recycle bin. (2) As the user starts dragging the web pages icon towards the recycle bin, icons that are of compatible type and located in the direction of the user's drag motion "pop up". This means that each of these icons produces a "tip icon" that appears in front of the user's pen. Tip icons are connected to the respective original icon using a rubber band. (3) The user drags the web page over the recycle bin and releases the mouse button. The recycle bin accepts the web page. Alternatively, the user could have dropped the web page over the word processor or the web browser icon, which would have launched the respective application with the memo. (4) When the user drops the icon, all tip icons disappear instantly.

In order to reduce clutter, drag-and-pop creates tip icons only for icons that are of matching file type, located far enough away from the dragged icon, and located within a certain angle from the user's initial drag direction. Drag-and-pop compacts the layout of all tip icons by placing tip icons on a denser grid and by eliminating empty rows and columns from that grid [3]. Users can abort drag-and-pop interactions at any time by moving the pen away from the tip icon cluster. This allows users to rearrange icons on the desktop. The rubber bands connecting tip icons with their original are designed to help users follow the transition when the tip icons appear and to *re*-identify the desired targets among the other tip icons. Drag-and-pop can be extended to allow users to access content in the periphery (drag-and-pick [3]).

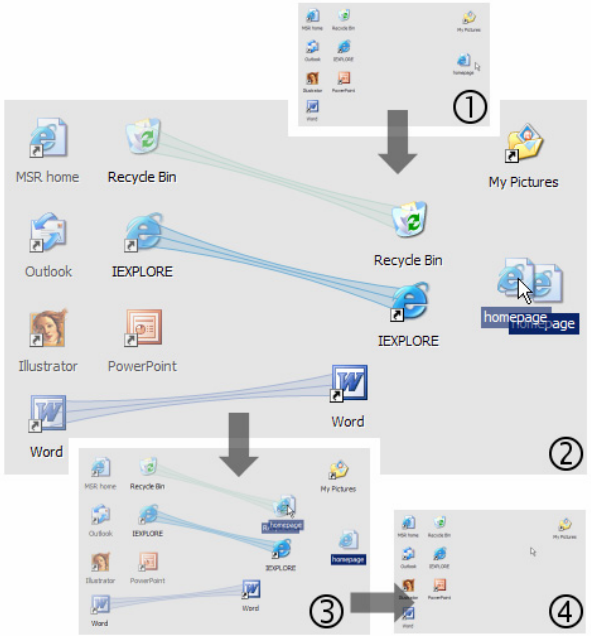

Figure 3: Drag-and-pop: Here the user drops the word file located at the right into the recycle bin.

The main limitation of drag-and-pop is that imprecise invocation gestures can cause the wrong tip icons to appear. In particular, Baudisch et al. found [3] that the arc-shaped full-arm drag motions users performed caused drag-and-pop to bring icons located in the extension of the first segment of that arc—this was typically *not* the direction to the target.

Design improvements: target sector and positioning

In order to address the limitations identified by Baudisch et al., we adjusted our version of drag-and-pop in two ways. First, we increased the size of the target sector and added extra tolerance for movements towards the top of the screen.

Second, in its original version, dragging towards another display unit sometimes makes the tip icon cluster appear fully or partially on that other screen unit.

The version of drag-and-pop used in our user study avoids this—it always places tip icon clusters in the display unit where the drag interaction was initiated.

Figure 4: Moving an object using the drag-and-pop technique on the iWall wall-size display.

4 Analysis and comparison of approaches

As listed in the previous section, push-and-throw and drag-and-pop have different strengths, but they also have different limitations. In order to allow creating a new technique that overcomes the limitations of both approaches, we take a closer look at these limitations and at the responsible design dimensions (Table 1).

4.1 Index of difficulty

All other factors kept constant, users can acquire closer targets faster than more distant targets (Fitts' law [17]). Push-and-throw and drag-and-pop both reduce the distance to the target in motor space. Push-and-throw amplifies pointer speed by a constant factor, but at the same time scales targets in motor space, so the only aspect that has an effect on the index of difficulty are snapping and acceleration. Drag-and-pop, in contrast, leaves target size unchanged. Packing targets more tightly therefore reduces the index of difficulty.

While index of difficulty does play a role in target acquisition in general, on wall-size displays the main factors are whether the target is in reach and how many bezels need to be crossed. In comparison to these factors, the impact of actual distance was found to be minor [3] and should therefore be expected to have only minor impact on the relative performance of push-and-throw with regard to drag-and-pop.

4.2 Need for reorientation

Push-and-throw-based techniques move the pointer all the way to the target, while drag-and-pop first moves potential targets to the pointer (Table 1). The difference, however, is merely a matter of the applied visuals. The underlying mechanism is similar: in motor space the user moves the pointer to a target that is within reach. In the case of push-and-throw the visuals appear in the target space, i.e., all over the wall-size display. In the case of drag-and-pop the visuals appear in the motor space, i.e., in the space reachable by the user.

The different visuals, however, have an impact on the interaction. Drag-and-pop involves a single fairly dramatic movement on the screen that requires users to reorient themselves. Drag-and-pop uses the rubber bands to minimize that impact, yet since different drag directions cause the tip cluster to be a little different every time, users need to pay attention when picking their target tip icon. Once users have identified the target tip icon, however, they can complete the interaction easily: the target is at a stable location and acquiring it requires only very little attention.

Push-and-throw, in contrast, requires users to constantly monitor the screen as it is virtually impossible for users to guess upfront where their finger has to be in order to acquire the target.

In our observation, the single motion caused by drag-and-pop impacts performance less than the continuous monitoring required by push-and-throw. We therefore paid close attention to avoid the need for continuous monitoring when designing *push-and-pop*.

technique	approach	need to reorient
drag-and-drop	—	never
pick-and-drop	—	never
push-and-throw	to target	constantly
accelerated…	to target	constantly
drag-and-pop	to pointer	once
New: push-and-pop	to pointer	once, later never

Table 1 : candidate techniques and design dimensions

5 Push-and-pop

Based on our analysis of push-and-throw and drag-and-pop, we created a new technique designed to combine the strengths of both techniques. We call this new technique *push-and-pop*—the name trying to convey that it builds on both *push-and*-throw and drag-and-*pop*.

Figure 5 shows a walkthrough. In the shown example, the user is dragging a word document into the recycle bin. The interaction proceeds as follows. (1) The user starts dragging the word document icon. (2) As a response, the system surrounds the pointer with a miniature representing the wall-size display—the *take-off area*—here containing the icons of the word application, a folder, and the recycle bin. (3) The user drags the word document icon over the recycle bin, which responds by highlighting itself with a rectangular frame. (4) The users lets go of the word document icon and the word document disappears in the recycle bin. The take

off area disappears. Figure 1 shows a photo of push-and-pop in use on a wall-size display.

In case users need to rearrange icons on the desktop, they can switch push-and-pop temporarily into a push-and-throw mode. Users invoke this functionality by moving the pointer back to the location of invocation, marked with a black circle in Figure 5.3.

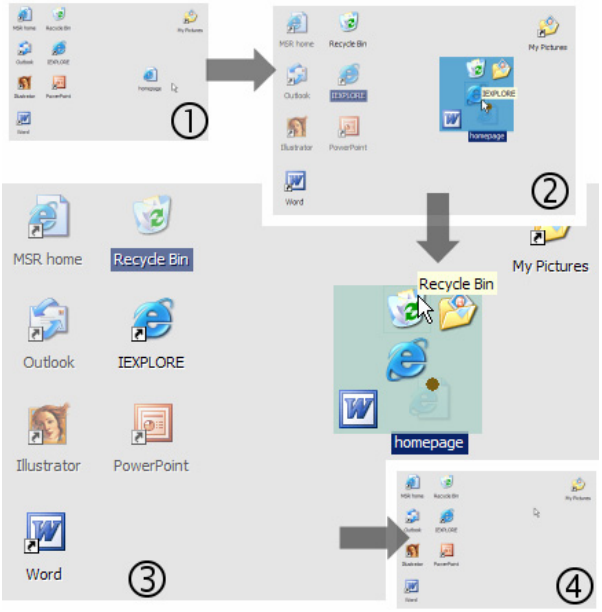

Figure 5: Push-and-pop walkthrough

The grid-like arrangement of the tip icons in the take-off area is taken directly from the drag-and-pop's layout algorithm [3]. It is created by placing icons on a small grid and by removing empty rows and columns. Unlike drag-and-pop which creates tip icons only for icons located in the drag direction, however, push-and-pop brings all target icons of matching type, independent of where they are located on the screen. This eliminates the risk of users invoking the wrong set of targets and also assures a stable, reproducible layout, thereby overcoming the two main limitations of drag-and-pop. Invocation of push-and-pop over the same icon type always results in the same take-off area, allowing users to perform the actual acquisition task based on muscle memory.

The resulting rectangular take-off area corresponds to the take-off area in push-and-throw. However, push-and-pop's take-off area offers two major benefits. First, the take-off area shows tip icons. This offers good readability even if the represented target is too far away to be readable. But most importantly it allows users to acquire the desired tip icon without the need for further reorientation. Second, rather than being a geometrically reduced version of the display, the miniature in the take-off is a *semantically* reduced version of the display in that only valid targets are contained. This allows push-and-pop to use full-size versions of the target icons allowing for an easier acquisition. However, to save space, we removed file names, instead revealing them as tool tips on hover.

6 First study: double wall-size screen

We conducted a user study comparing six drag-and-drop techniques for wall-size displays. The study served two main purposes. First, we wanted to learn more about the relative performance of the different techniques. We had several hypotheses. For sufficiently long distances, we expected all techniques to outperform traditional drag-and-drop. More specifically, we expected the techniques that required no or a one-time reorientation to outperform the techniques that required continuous tracking. Second, we wanted to validate the design of push-and-pop. Would it really outperform push-and-throw and drag-and-pop?

In order to extend our findings to longer distances, we later replicated the study on a three-unit display wall, as we report in Section 7. We will hold off with our discussion until after the second study.

6.1 Task

Participants' task was to perform drag-and-drop operations on a simulated Windows desktop. The task details corresponded largely to the original drag-and-pop user study reported in [3]. Figure 6 shows the icon layout used in the study. The icons to be filed appeared at the bottom right of the screen (the cluster of 10 icons at the bottom right of Figure 6). The target was successively displayed in one of 12 positions, which allowed us to obtain uniformly distributed indexes of difficulty. Unlike the original drag-and-pop study, we simplified the participants' task by always using the recycle bin icon as the target.

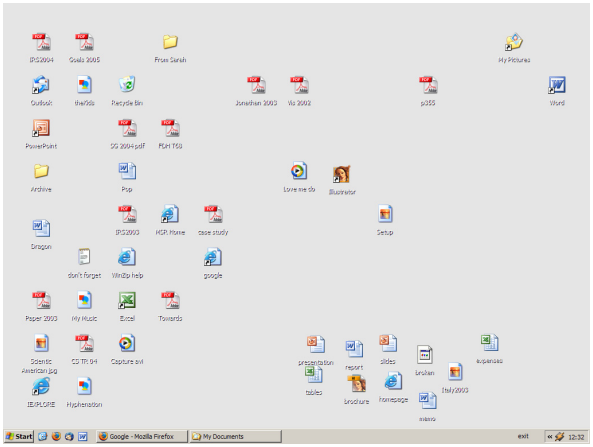

Figure 6: The icon layouts used in the study.

6.2 Apparatus & interfaces

The study was conducted on a combination of a wall-size back-projected display and a front-projected SMART Board™. Each of the two displays measured 5'/1.7m across and ran at 1024*768 pixel resolution. Pen input on the back-projected display was supported by a Mimio™ capture bar and a specialized pen. The Wall-screens were connected to a PC with a Pentium4 1.5GHz processor and 512MB of RAM.

Participants used six different interfaces: drag-and-drop, pick-and-drop, push-and-throw, accelerated push-and-throw, drag-and-pop, and push-and-pop. The pick-and-drop interface corresponded to [22] (A first click selects the object that has to be moved and a second one selects the target) with the difference that the interface in the study required users to briefly drag an icon in order to pick it up. This allowed differentiating drag operations from object selection.

6.3 Participants

Twelve participants (all male, one left-handed, students and researchers) were recruited internally. All participants but one had little or none experience with using wall-size displays.

6.4 Experimental design

We used a within-subjects design. Each participant performed 36 trials per interface, resulting in a total of 216 trials. The 36 trials were grouped in three blocks of 12 trials, with each trial corresponding to a different target positions. A Latin square was used to counterbalance order of interfaces and order of presentation. Target positions within each block were randomized. Participants received up to 5 minutes of training per interface before beginning the timed tasks.

We measured *Movement Time*, i.e., the time from the moment the participant tapped onto the icon to be filed to the moment the participant lifted the pen. We also measured *Error Rate*, i.e., the percentage of cases where the user released the dragged icon over the wrong target. In cases where participants accidentally released the dragged object over empty space, they had to pick it up again and complete the trial. We kept track of that as well.

6.5 Results

Task completion time: An ANOVA on median values for time showed significant differences with the type of technique ($p<0.001$). A Dunn's pair-to-pair comparison showed that all comparisons were significant except the comparison of accelerated push-and-throw and pick-and-drop.

Overall, participants performed the task fastest when using the push-and-pop interface, confirming our hypothesis. Drag-and-pop was only slightly slower. Next came accelerated push-and-throw and pick-and-drop. Participants performed worse when using push-and-throw. Confirming the findings by Baudisch et al. [3], drag-and-drop worked well as long as the target was located within the same display unit, but performed poorly when the task required participants to cross the bezel between screens. Averaged across distance, traditional drag-and-drop performed worst.

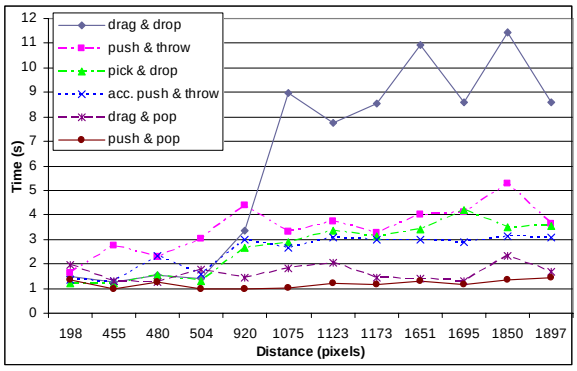

Figure 7: Mean task completion time for each technique depending of the ID of the task.

Errors rates: Figure 8a shows error rates. The number of cases where participants dropped an icon into the wrong target was generally low (less than two cases out of 36 trials per interface). The number of cases where participants temporarily dropped an icon, but then successfully filed it was much higher for push-and-throw and drag-and-pop (between 9% and 12%). Pick-and-drop, push-and-pop and accelerated push-and-throw offered much better results here (below 2%). Drag-and-drop does not count here, because participants had to temporarily drop the icon whenever crossing the bezel.

Subjective satisfaction: At the end of the experiment, participants ranked all six techniques by preference. Figure 8b shows a summary created by assigning 5 points for each first ranks, down to 0 points for last rank and averaging the results. In particular, 9 of the 12 participants ranked push-and-pop first.

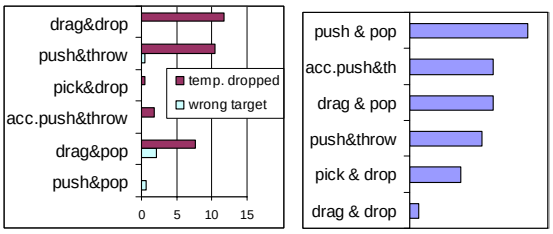

Figure 8: (a) Error rates for each technique. (b) Subjective preferences (higher is better)

7 Second study: triple wall-size screen

We replicated the study on a corresponding three-unit display wall in order to extend our findings to longer distances.

The second study was run on the iWall [14], a three back-projection SMART Board™ displays, each with resolution 1024x768 (Figure 4). Participants interacted with the board via touch, either using their fingers directly or holding a pen. The display was driven by a Pentium® II 500MHz with 256MB of RAM.

Six volunteers (4 females, 2 left-handed) participated in this study. All participants but one had very little experience with using wall-size displays.

The experimental design was identical to the first study, but instead of 3 blocks of 12 trials, participants performed 4 blocks or 12 trials per interface.

7.1 Results

Task completion time: An ANOVA on median task times showed significant differences with the type of technique ($p<0.001$). A Dunn's pair-to-pair comparison found all differences significant except the comparison of push-and-throw and accelerated push-and-throw.

Figure 9 shows the results. The resulting ranking corresponds largely with the first study. Participants were fastest when using push-and-pop, followed by drag-and-pop, pick-and-drop, and drag-and-drop. In this study participants performed worst when using push-and-throw/accelerated push-and-throw. Only for very long distances did these techniques perform better than traditional drag-and-drop. Note drag-and-drop and pick-and-drop performance depended on the target distance, while the performance all other techniques is largely distance independent.

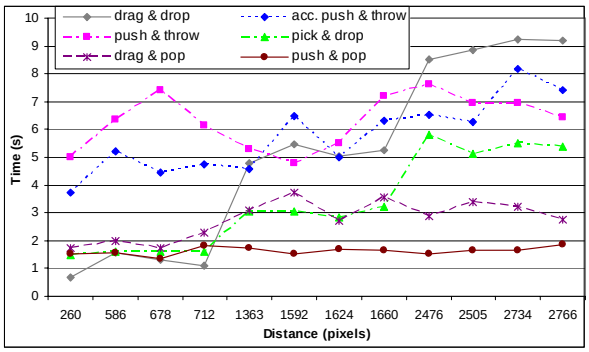

Figure 9: Mean task completion time for each technique depending of the ID of the task.

Error rates: Error rates corresponded to the first study with the exception that this time participants using accelerated push-and-throw performed a higher number of accidental drops (Figure 10a).

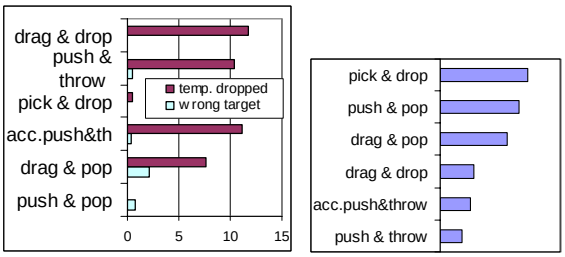

Figure 10: (a) Error rates for each technique. (b) User's subjective preferences (higher is better)

Subjective satisfaction: This time pick-and-drop scored slightly higher that push-and-pop (Figure 10b). The push-and-throw techniques scored last.

8 Discussion

The two user studies presented above provide some supporting evidence for the claims made earlier in this paper.

Confirming findings by Baudisch et al. [3] drag-and-drop performed well as long as source and target icons were situated in the same display unit, but failed quickly when long distances and bezels were involved. In addition, we found that pick-and-drop is affected by distance in a similar way, though to a lesser extent.

For all other evaluated techniques, target distance had comparably little impact on task performances. However, our studies seem to indicate a performance benefit of acquisition techniques that require a one-time reorientation over techniques that require continuous tracking. The two techniques that required only require a one-time reorientation (drag-and-pop and push-and-pop) achieved the best task times.

Overall, the study indicates that push-and-pop is indeed a useful technique. Push-and-pop outperformed all other techniques, including its ancestors, drag-and-pop and push-and-throw. Participants' subjective preference reflected this. Push-and-pop also offered a very low error rate, which is one possible explanation for the performance benefit in comparison with drag-and-pop. While participants using drag-and-pop needed to pay close attention to the directionality of their invocation gesture in order to avoid error, push-and-pop avoided this additional burden on users by always displaying all possible tip icons.

Among pointer-to-target techniques, accelerated push-and-throw performed significantly better than traditional push-and-throw. Despite the unexceptional performance of both techniques, this indicates that our design improvements were beneficial.

9 Conclusion

We presented an experimental comparison of six drag-and-drop techniques for wall-size displays and found

significant performance benefits for techniques that do not require users to continuously track their interaction, in particular the push-and-pop technique introduced in this paper.

We have focused on icon displacements. As future work, we plan to optimize the design of push-and-pop (e.g., by reintroducing rubber bands and solving scalability problems, such as those presented by desktops with numerous folders) and extending the presented techniques to other types of interactions, such as activation of menus and buttons.

Acknowledgements

We would like to thank the participants in our experiments, who spent significant amounts of time testing the interaction models.

References

[1] Accot, J and Zhai, S. Beyond Fitts' law: models for trajectory-based HCI tasks. In *Proc.CHI'97*, pp.295-302.

[2] Baudisch, P., Cutrell, E., and Robertson, G. High-Density Cursor: A Visualization Technique that Helps Users Keep Track of Fast-Moving Mouse Cursors. In *Proc. Interact'03*, pp. 236–243.

[3] Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P. Bederson, B., and Zierlinger, A. Drag-and-Pop and Drag-and-Pick: Techniques for Accessing Remote Screen Content on Touch and Pen-operated Systems. In *Proc Interact'03*, pp. 57–64.

[4] Blanch, R., Guiard,Y., and Beaudouin-Lafon, M. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proc. CHI'04*, pp. 519–526.

[5] Coxeter, H. S. M. *Introduction to Geometry*, 2nd ed. NewYork: Wiley, pp. 69–70, 1969.

[6] Elrod, S., et. al. Liveboard: a large interactive display supporting group meetings, presentations, and remote collaboration. In *Proc. CHI'92*, pp. 599–607.

[7] Khan, A., Fitzmaurice, G. Almeida, D., Burtnyk, N., and Kurtenbach, G. A remote control interface for large displays. In *Proc. UIST'04*, pp. 127–136.

[8] Geißler, J. Shuffle, Throw or Take It! Working Efficiently with an Interactive Wall. In *CHI '98 Late-Breaking Results*, pp. 265–266.

[9] Guiard, Y., Blanch, R., and Beaudouin-Lafon, M. Vector Pointing: Object vs. Pixel Selection in Graphical User Interfaces. In *Proc. GI'04*, pp 9–16.

[10] Guiard, Y., Beaudouin-Lafon, M., Bastin, J., Pasveer, D., Zhai, S. View size and pointing difficulty in multi-scale navigation. In *Proc. AVI'04*, pp. 117–124

[11] Guimbretiere, F., M. Stone, and T. Winograd. Fluid interaction with High Resolution Wall-Size Displays. In *Proc. UIST'01*, pp. 21–30.

[12] Hascoët, M. (2003). Throwing models for large displays. In *Proc. HCI'03*, pp. 73–77.

[13] Johanson B., Hutchins, G., Winograd, T., and Stone, M. PointRight: Experience with Flexible Input Redirection in Interactive Workspaces. In *Proc. UIST'02*, pp. 227–234.

[14] Johanson, B., Fox, A., and Winograd, T. The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms, *IEEE Pervasive Computing*, 1 (2), 67–74.

[15] Kurtenbach, G., Buxton, W. User learning and performance with marking menus. In *Proc. CHI'94*. pp. 258–264.

[16] MacKenzie, I.S., and Jusoh, S. An evaluation of two input devices for remote pointing. In *Proc. EHCI'01*, pp. 235–249.

[17] MacKenzie, I.S., Fitts' law as a research and design tool in human-computer interaction. *Human Computer Interaction, 1992*, 7, pp. 91–139.

[18] Myers, B., Bhatnagar, R., Nichols, J., Peck, C., Kong, D., Miller, R., and Long, C. Interacting At a Distance: Measuring the Performance of Laser Pointers and Other Devices. In *Proc CHI'02*, pp 33–40.

[19] Olsen, D. R. and Nielsen, T. Laser pointer interaction. In *Proc. CHI'01*, pp. 17–22.

[20] Poupyrev, I., Billinghurst, M., Weghorst, S., Ichikawa, T. The Go-Go Interaction Technique: Non-Linear Mapping for Direct Manipulation in VR. In *Proc. UIST'96*, pp. 79–80.

[21] Rekimoto, J. and Saitoh, M. Augmented Surfaces: A Spatially Continuous Work Space for Hybrid Computing Environments. In *CHI'99*, pp. 378–385.

[22] Rekimoto, J. Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments. In *Proc. UIST'97*, pp. 31–39.

[23] Smith, D.C., Irby, C., Kimball, R., Verplank, W., and Harslem, E. Designing the Star user interface, *Byte* 7(4):242–282, 1982.

[24] Stoakley, R., Conway, M., and Pausch, R. Virtual Reality on a WIM: Interactive Worlds in Miniature. In *Proc. CHI'95*, pp. 265–272.

[25] Streitz, N.A., Tandler, P. Müller-Tomfelde, C., Konomi, S. Roomware. In: Carroll, J.A. (Ed.), *Human-Computer Interaction in the New Millennium*, Addison Wesley, pp. 553–578, 2001.

[26] Wagner, A., Curran, P., O'Brien, R. Drag me, drop me, treat me like an object. In *Proc CHI '95*. pp. 525–530.

[27] Zhai, S., Morimoto, C. Ihde, S. Manual And Gaze Input Cascaded (MAGIC) Pointing. In *Proc. CHI '99*, pp. 246–253.

François Boutin and Mountaz Hascoët, Cluster Validity Indices for Graph Partitioning,
Proceedings of the Conference on Information Visualization IV'2004

Cluster Validity Indices for Graph Partitioning

François Boutin, Mountaz Hascoët
PhD Student, associate Professor – LIRMM – UMR 5506 – CNRS – University Montpellier II
francois.boutin@univ-montp1.fr, mountaz@lirmm.fr

Abstract

The aim of graph clustering is to define compact and well separated clusters using graph structure. Cluster's compactness depends on dataset and clustering method.

Different cluster validity indices were proposed especially in data analysis. Main indices are based on the comparison of intra and inter-cluster variability. Unfortunately they are often computed in Euclidian spaces when graphs are defined in metric spaces.

In this paper, we review cluster validity indices for graph clustering. We also propose adjusted measures.

We think that appropriate cluster validity indices can be added as visual tips in graph partition layouts. For instance the stronger a cluster is connected the darker its background is. Validity measures are helpful to understand but also to manage graph partitions.

Keywords : graph clustering, partitioning, cluster validity indices

1. Introduction

Various techniques have been developed to visualize and manage huge graphs [15]. Some of them are based on clustering techniques and provide graph partition. Their objective is to maximize intra-cluster connectivity (compactness) and minimize inter-cluster connectivity (separability). Clustering quality depends on the graph structure and the clustering technique. So we need appropriate validity indices to analyze clustered graph structure but also to compare clustering methods.

Note that cluster indices can be added to graph partition layouts using visual tips. For instance a compact cluster can be drawn with dark background. Visuals tips make graph structure easier to understand and to manage.

Various cluster validity indices were developed in data analysis [14]. They are usually based on variance analysis methods. The principle is to compare intra and inter-cluster variability using the famous theorem of Huygens: "Global variability is split up into intra and inter-cluster variability". Unfortunately such methods must be applied in an Euclidian space since they need to compute means and variances. Now, a graph is not defined in an Euclidian space but in a metric space. Therefore we have to use other indices to compute cluster compactness for a graph structure.

Little work has been done to present specific validity indices for graph clustering. Some indices are proposed in [13] to find the optimal number of clusters. In this work we review and compare various cluster validity indices. We also propose adjusted measures.

After introducing definitions (section 2), we propose different indices to define graph compactness (section 3). Next we describe cluster validity indices that depend on inter and intra-cluster connectivity (section 4). After that we propose indices based on connectivity between a node and its neighborhood (section 5). To end with we present indices to compare different clustering results (section 6).

2. Preliminary definitions

In this paper, we consider an undirected graph G, with N nodes v_i and E edges e_{ij} between nodes v_i and v_j. G is clustered into a set $\{C_1,\ldots,C_K\}$. N_i is the number of nodes and E_i the number of edges of cluster C_i. Moreover E_{ij} is the number of edges between C_i and C_j and E_i' as the number of edges between C_i and the other clusters.

Distance between v_i and v_j denoted $d(v_i,v_j)$ is defined as the length of the shortest path between v_i and v_j in G.

Different inter-cluster distances are defined. Most of them consist of single, complete and average linkage. They represent respectively the smallest distance, the largest distance, and the average distance between two nodes (one in each cluster) [2]. Distance between clusters C_i and C_j is denoted $d(C_i,C_j)$. In a same way we define $d(v_i,C_j)$ as the distance between node v_i and cluster C_j using either complete or average distance.

Two intra-cluster distances are mainly computed: complete and average diameter distance. They are respectively defined as the distance between the two most remote nodes, and the average distance between all couples of nodes that belong to a same cluster [2]. We denote $diam(C_k)$ the diameter of cluster C_k.

In practice, in this paper, we will use the average distance and the average diameter to compute indices.

3. Graph compactness

3.1. Compactness based on edge density

Simple indices are proposed for graph compactness:

For instance: $\frac{E}{N}$ or $\frac{E}{N^2}$

Such indices are easy to compute but they do not take into account the real structure of the graph. Indeed, two graphs can have a different structure while having the same number of nodes and links.

Figure 1, G_1 & G_2 have same compactness E/N=13/8.

3.2. Compactness index Cp

A compactness index Cp was introduced in [3] that takes into account graph connectivity. Consider a connected graph G. Cp is computed in quadratic time by:

$$Cp=\frac{Max-\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} d(v_i,v_j)}{Max-Min}$$

where Max and Min are, respectively, the maximum and minimum value of $\sum_i\sum_j d(v_i,v_j)$.

There are N(N-1)/2 couples (v_i,v_j) of distinct nodes. If Q is defined as the maximal distance between two nodes, $d(v_i,v_j)$ is between 1 and Q. So Min = N(N-1)/2 and Max = Q.N(N-1)/2

Figure 1, $Cp(G_1) = 0.48$ and $Cp(G_2) = 0.69$.

Now, to compute compactness for a disconnected graph, we can define a distance between two unconnected nodes as an arbitrary large value Q [3]. But the resulting index is depending on the choice of Q value.

3.3. New compactness index Cp*

We propose a new normalized compactness index denoted Cp* that use no constant Q. Indeed we consider a similarity measure instead of a distance.

Considering two nodes v_i and v_j, we define similarity: $sim(v_i,v_j) = 1 / d(v_i,v_j)$, if v_i and v_j are connected and $sim(v_i,v_j) = 0$, if v_i and v_j are disconnected. Note that sim value lies between 0 and 1. Now Cp* is computed by:

$$Cp^*=\frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} sim(v_i,v_j)}{N(N-1)/2}$$

Cp* is bounded by 0 and 1. If G is completely disconnected, Cp* = 0. If G is a complete graph, Cp* = 1.

Figure 1, $Cp^*(G_1) = 0.73$ and $Cp^*(G_2) = 0.69$.

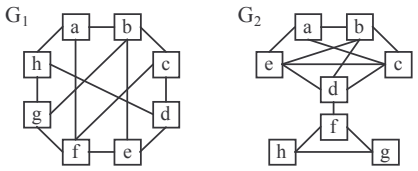

Figure 1. Same graph compactness for G_1 and G_2

4. Inter versus intra-cluster connectivity

To evaluate the quality of a graph partition we usually compare inter and intra-cluster connectivity. In this section, we present various indices using different definitions of inter and intra-cluster connectivity.

4.1. Indices based on diameter and distance

Dunn's index:

If C_i and C_j are the closest clusters according to average distance d and C_h is the cluster with largest diameter, Dunn' index [1][2][11][14] is computed by:

$$D(C)=\frac{d(C_i,C_j)}{diam(C_h)}$$

$d(C_i,C_j)$ and $diam(C_h)$ are respectively related to inter and intra-cluster connectivity. D is not robust since it depends only on few clusters and few links between them.

Figure 2, in both G_3 and G_4, closest clusters are C_2, C_3 and largest cluster is C_1. So, Dunn's indices are the same.

Davies Bouldin index:

Davies Bouldin index [2][9] is defined by:

$$DB=\frac{1}{K}\sum_{i=1}^{K}\max_{j\neq i}\left[\frac{diam(C_i)+diam(C_j)}{d(C_i,C_j)}\right]$$

Small values of DB correspond to compact clusters. Davies Bouldin index is more robust than Dunn's index.

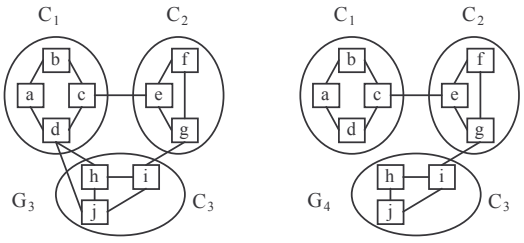

Figure 2. Same Dunn's index for G_3 and G_4

4.2. Indices based on inter & intra-cluster edges

***A cut index called MinMaxCut*:**

The cohesiveness of cluster C_i can be computed by:

$$\frac{E_i'}{E_i}$$

MinMaxCut is defined as [10] :

$$MinMaxCut=\sum_{i=1}^{K}\frac{E_i'}{E_i}$$

In [23], the objective is to minimize *MinMaxCut* function. Unfortunately, this index does not take into account the number of nodes N_i in cluster C_i.

Figure 3, *MinMaxCut* is the same for G_5 and G_6 (½) whereas G_6 contains components with smaller diameter.

Conductance of a cut:

Inter-cluster links are called *cut*. The conductance of a cut compares the size of the cut and the number of edges in the smallest induced subgraph [7]. If we consider the cut between C_i and C_j, we define the conductance by:

$$Conductance(C_i,C_j)=\frac{E_{ij}}{\min(E_i,E_j)}$$

Clusters with low conductance are well separated.

Coverage of a graph clustering:

Coverage of a graph clustering C is the fraction of intra-cluster edges within the complete set of edges [7].

$$Cov(C)=\frac{\sum_{i=1}^{K}E_i}{E}$$

So the larger the value of coverage the better the quality of clustering C. This index is easy to compute but it does not care about the number of nodes N_i in C_i.

Figure 3, coverage is the same for G_5 and G_6 whereas structure of graph G_6 is much more compact.

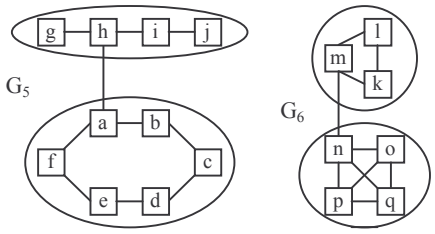

Figure 3. Same *MinMaxCut*, conductance and coverage

4.3. Indices using number of nodes and links

Performance of a clustering:

A clustering performance is defined in [7][22] by:

$$Perf(G)=1-\frac{\sum_{i<j}^{K}E_{ij}+\sum_{i=1}^{K}\left(\frac{N_i(N_i-1)}{2}-E_i\right)}{\frac{N(N-1)}{2}}$$

or

$$Perf(G)=1-\frac{\|False+\|+\|False-\|}{\frac{N(N-1)}{2}}$$

where $\|False-\|$ is th number of links between clusters (inter-cluster links) and $\|False+\|$ is the number of couples (v_i,v_j) in a same cluster that are not linked.

Note that performance can also be computed by:

$$Perf(C)=1-\frac{E\cdot(1-2\mathrm{cov}(C))+\sum_{i=1}^{K}\frac{N_i(N_i-1)}{2}}{\frac{N\cdot(N-1)}{2}}$$

Modularization quality MQ:

Modularization quality function is defined by the difference between intra & inter cluster connectivity[18]: Intra-cluster connectivity of C_i is computed by:

$$intra(C_i)=\frac{E_i}{N_i(N_i-1)/2}$$

where $N_i(N_i-1)/2$ is the maximum number of intra-cluster edges. Inter-cluster connectivity between clusters C_i and C_j is defined by:

$$inter(C_i,C_j)=\frac{E_{ij}}{N_iN_j}$$

$$Let\ define\ \overline{intra}=\frac{\sum_{i=1}^{K}\frac{E_i}{N_i(N_i-1)/2}}{K}\ and\ \overline{inter}=\frac{\sum_{i<j}^{K}\frac{E_{ij}}{N_iN_j}}{K(K-1)/2}$$

$$MQ=\overline{intra}-\overline{inter}=\frac{\sum_{i=1}^{K}\frac{E_i}{N_i(N_i-1)/2}}{K}-\frac{\sum_{i<j}^{K}\frac{E_{ij}}{N_iN_j}}{K(K-1)/2}$$

Unfortunately this index computes simple instead of weighted means when clusters can be of different size.

Figure 4, MQ ignore that C_1 is larger than C_2.

A new index denoted MQ*:

Previous index is easily modified to provide a weighted MQ. We denote this new index MQ*:

$$MQ^*=\frac{\sum_{i}E_i}{\sum_{i}\frac{N_i(N_i-1)}{2}}-\frac{\sum_{i<j}E_{ij}}{\sum_{i<j}N_iN_j}$$

Unlike MQ, MQ* takes into account the clusters' connectivity and their size.

Figure 4, C_1 is larger than C_2 with a smaller intra-cluster connectivity so MQ* is smaller than MQ (In fact, MQ* = 0.44 and MQ = 0.64).

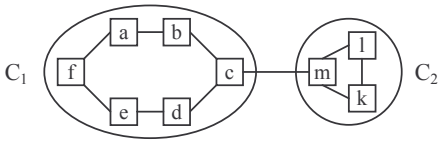

Figure 4. MQ* smaller than MQ

5. Indices based on node's neighborhood

We previously described global compactness indices. Now we present indices based on node's neighbors. The

clustering measure of a node is interesting in itself. Indeed, we can define if a node is well related to its cluster or not. Then outliers may be removed or reorganized.

5.1. Silhouette index

Let consider a node v_i that belongs to a cluster C_j. Let suppose that the closest cluster to node v_i (according to average distance) is denoted C_h. The silhouette index is defined in [2][21] by:

$$s(v_i)=\frac{d(v_i,C_h) - d(v_i,C_j)}{\max(d(v_i,C_j),d(v_i,C_h))}$$

Note that $-1 \le s(v_i) \le 1$. Moreover when $s(v_i)$ is close to 1, v_i is said to be "well clustered". When $s(v_i)$ is inferior to 0, v_i should be assigned to the nearest neighboring cluster.

Figure 4 s(m)=(2.5-1)/2.5=0.6, s(k)=(3.5-1)/3.5=0.71.

For a given cluster C_j we compute silhouette S_j:

$$S_j=\frac{\sum_{i=1}^{Nj} s(v_i)}{N_j}$$

A global silhouette value GS is also computed:

$$GS=\frac{\sum_{j=1}^{K} S_j}{K}$$

A new index denoted GS*:

Now, we propose to define GS* a new index that takes into account size of clusters:

$$GS^*=\frac{\sum_{j=1}^{K} N_j S_j}{\sum_{j=1}^{K} N_j} = \frac{\sum_{i=1}^{N} s(v_i)}{N}$$

Figure 4, C_1 is larger than C_2 so its contribution is more important in GS*.

5.2. Coverage measures

Naïve coverage measure

Let v_i be a node belonging to cluster C_j. Let $N(v_i)$ be the set of neighbors of v_i. *False positive set* (denoted $False_i+$) is defined as the set of nodes of C_i that does not belong to $N(v_i)$. *False negative set* (denoted $False_i-$) is defined as the set of nodes that belong to $N(v_i)$ but not to C_j. Naïve coverage measure is defined [20][22] as:

$$Cov(v_i) = 1 - \frac{\|False_i+\| + \|False_i-\|}{N-1}$$

Note that the performance index defined in 4.3 is also the average of $Cov(v_i)$.

Scaled coverage measure

Scaled coverage measure is computed by [20]:

$$ScalCov(v_i) = 1 - \frac{\|False_i+\| + \|False_i-\|}{\|C_j \cup N(v_i)\|}$$

If $\|True_i+\| = \|True_i-\| = 0$ then $\|C_j \cup N(v_i)\| = \|False_i+\| + \|False_i-\|$ and $ScalCov(v_i) = 0$ unlike $Cov(v_i)$.

5.3. Cluster index in Small World graphs

A specific cluster index for Small World graph is defined in [8]. The metric measures the edge density in the neighborhood of a node v_i. Let $N(v_i)$ be the neighborhood of node v_i with n nodes and e edges between them (without considering node v_i and edges from v_i). We compute $c(v_i)$ as:

$$c(v_i)=\frac{e}{\frac{n(n-1)}{2}}$$

Figure 4, c(m) = 1/3, c(k)=1, c(c)=0

The clustering measure of graph G is the average of clustering measures for all nodes:

$$c(G)=\frac{\sum_{i=1}^{N} c(v_i)}{N}$$

6. External indices to compare clustering results

We present various indices to compare two partitions $P=\{C_1,\ldots,C_K\}$ and $P' = \{C'_1,\ldots,C'_T\}$.

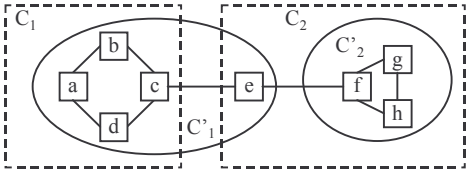

Figure 5. Comparison of partitions P and P'

6.1. Indices based on co-clusteredness

We present indices based on repartition of N(N-1)/2 couples of nodes (v_i,v_j) in partitions P and P' (see Table 1). They are presented in [14][17].

Partition of $\{(v_i,v_j)\}$	Same cluster in P'	Different clusters in P'
Same cluster in P	a	c
Different clusters in P	b	d

Table 1. Repartition of couples (v_i, v_j)

Jaccard Coefficient:

Jaccard coefficient defines similarity between P & P':

$$J = \frac{a}{a+b+c}$$

J computes the probability that two nodes belonging to a same cluster in a partition also belong to a same cluster in the other partition.

Folkes and Mallows index:

Folkes and Mallows introduced another index:

$$FM=\sqrt{\frac{a}{a+b}\cdot\frac{a}{a+c}}$$

a/(a+b) is the probability that two nodes belong to a same cluster in P if they belong to a same cluster in P'. Now, a/(a+c) is the probability that two nodes belong to a same cluster in P' if they belong to a same cluster in P.

Rand Statistic:

Rand statistic measures similarity between P and P':

$$R=\frac{a+d}{a+b+c+d}$$

Unlike J and FM, Rand statistic is calculated with d. It computes the probability that two nodes belong either to a same cluster or to different clusters in both P and P'.

Hubert and Arabie's statistic:

Hubert and Arabie [16] modified the Rand Statistic so that its maximum is one and its expected value is zero if classifications are selected randomly.

$$Hubert=\frac{a\cdot d-b\cdot c}{\sqrt{(a+b)(c+d)(a+c)(b+d)}}$$

Hubert's statistic is also called Phi statistic. It is in fact equivalent to the Pearson correlation coefficient [19]. Note that Phi can be computed using Chi square statistic:

$$\varphi^2=\frac{X^2}{n} \text{ with } n = a+b+c+d$$

Figure 5, a = 9, b = 4, c = 3, d = 12 and so J = 0.56, FM = 0.72, R = 0.75, Hubert = 0.50.

6.2. Indices based on probability measures

Now we propose methods based on probability to compare partitions $P=\{C_1,\ldots,C_K\}$ and $P' = \{C'_1,\ldots,C'_T\}$.

We denote f_{ij} the percentage of nodes of cluster C_i that belong to cluster C'_j. Note that:

$$\forall i\in\{1,\ldots,K\} \text{ and a cluster } C_i, \sum_{j=1}^{T} f_{ij}=1$$

Partition coefficient:

Bezdek [1] introduced the partition coefficient PC. Considering a cluster C_i:

$$PC(C_i)=\sum_{j=1}^{T} f_{ij}^2$$

$PC(C_i)$ is a value between 1/T and 1. If almost all nodes of C_i belong to a same cluster in C'_j, then $PC(C_i)$ is close to 1. Now, if nodes of C_i are randomly divided into all clusters of P' then $PC(C_i)$ is close to 1/T.

A global partition coefficient PC is computed:

$$PC(P/P') = \frac{1}{K}\sum_{i=1}^{K} PC(C_i) = \frac{1}{K}\sum_{i=1}^{K}\sum_{j=1}^{T} f_{ij}^2$$

PC(P/P') is also a value between 1/T and 1. Now, if PC(P/P') is close to 1/T, P and P' are almost independent. Moreover, if PC(P/P') is close to 1, then P is "close" to P'.

Figure 5, PC(P/P') = 0.84

Entropy of clustering:

In the same way, Bezdeck defined clustering entropy [6]. Considering a cluster C_i:

$$Entropy(C_i) = -\sum_{j=1}^{T} f_{ij}\cdot\log(f_{ij})$$

Note that, when x is close to 0, then x.log(x) is close to 0. So we consider that $0.\log(0) \approx 0$.

Entropy(C_i) is a value between 0 and log(T). If almost all nodes of C_i belong to a same cluster C'_k, then f_{ij} is close to 0 for $j\neq k$, and f_{ij} is close to 1 for j=k. So entropy(C_i) is close to 0 (since $0.\log(0)=1.\log(1)=0$). Now if nodes of C_i are randomly divided into all clusters of P', then f_{ij} is close to 1/T and entropy(C_i) is close to log(T).

A global clustering entropy is computed by:

$$Entropy\ (P/P') = \sum_{i=1}^{K} \frac{N_i}{N} Entropy\ (C_i)$$

Global entropy is also a value between 0 and log(T). If entropy(P/P') is close to log(T), P and P' are almost independent. Now if entropy is close to 1, P is close to P'.

Figure 5, entropy(C_1) = 0.72 and entropy(C_2) = 0. So entropy(P/P') = 0.45.

7. Conclusion

In this work we reviewed cluster validity indices for graph partitioning. We also proposed three new normalized indices: Cp*, MQ* and GS* from indices Cp, MQ and GS.

We first described global graph indices: Edge density is easy to compute but it does not care about connectivity. Cp is a compactness index that takes into account connectivity but it is depending on a constant Q. We defined a new normalized compactness index Cp* that use similarity measure instead of distance.

After that we presented indices to express cluster compactness using inter and intra-cluster connectivity: Dunn's index and Davies Bouldin index are computed using diameter and distance between clusters. Dunn's

index is easy to compute but it is not as robust as Davies Bouldin index. Cut indices and coverage index use the number of inter and intra-cluster edges. They are easy to compute but they do not take into account clusters' size. Performance and modularization quality MQ are computed using the number of nodes and the number of inter and intra-cluster edges. We proposed a normalized index MQ* that improves MQ.

Then we presented validity indices for nodes based on their neighborhood connectivity: Silhouette index GS and coverage measures can be used to decide if a node is well clustered or not. We defined GS* as a weighted measure of GS. Now clustering index C is well suited for Small World graphs.

To end with, we described external indices to compare clustering results: Rand Statistic, Jaccard coefficient and Folkes & Mallows index are easy to compute. Hubert & Arabie's statistic also called phi statistic defines Pearson correlation between partitions. Partition coefficient and Entropy are two similar indices based on probability measures.

In this paper, we reviewed cluster indices for graph partitioning. Now, a clustering technique can be applied recursively to organize a graph into a (hierarchically) clustered graph: a nested structure of clusters where each node belongs to an inclusion tree of clusters [12].

In a further work we plan to evaluate cluster indices for a clustered graph structure like *multi-level Outline Tree* [4][5]. We think that cluster validity indices can be used as visual tips to make a clustered graph layout easier to understand and to manage.

8. Bibliography

[1] Bezdek J.C., Pal N.R., "Some new indexes of cluster validity", IEEE Transactions on Systems, Man and Cybernetics, Vol. 28, Part B, 1998, pp. 301-315

[2] Bolshakova N., Azuaje F., Cluster Validation Techniques For Genome Expression Data, Technical Report TCD-CS-2002-33, September 2002

[3] Botafogo R.A., Rivlin E., Shneiderman B.: Structural Analysis of Hypertexts: Identifying Hierarchies and Useful Metrics. ACM Transactions on Information Systems, Vol. 10, No. 2. ACM, 1992. pp. 142-180

[4] Boutin F., Hascoet M., Focus Dependent Multi-level Graph Clustering. Proceedings of Advanced Visual Interface, AVI 2004

[5] Boutin F., Hascoët M., Focus-Based Clustering for Multi-Scale Visualization. Proceedings of IV 2003, IEEE, pp 53-59, 2003

[6] Bradley P.S. and Fayyad U.M., "Refining initial points for K-Means clustering," in Proc. 15th International Conf. on Machine Learning. 1998, pp. 91--99, Morgan Kaufmann, San Francisco, CA

[7] Brandes U., Gaertler M., and Wagner D., Experiments on Graph Clustering Algorithms, Proc. 11th Europ. Symp. Algorithms (ESA '03), Springer LNCS

[8] Chiricota Y. , Jourdan F., Software Component Capture using Graph Clustering. International Workshop on Program Comprehension (IWPC 2003)

[9] Davies D.L., Bouldin D.W., "A cluster separation measure", IEEE Transactions on Pattern Recognition and Machine Intelligence, Vol. 1, No. 2, 1979, pp. 224-227

[10] Ding Chris, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst Simon. Spectral min-max cut for graph partitioning and data clustering. Technical Report TR-2001-XX, Lawrence Berkeley National Laboratory, University of California, Berkeley, CA, 2001.

[11] Dunn J., "Well separated clusters and optimal fuzzy partitions", J.Cybernetics, Vol. 4, 1974, pp. 95-104

[12] Eades P., "Multilevel Visualization of Clustered Graphs," Proceedings of Graph Drawing'96, Berkeley, California, September,1996.

[13] Gunter S., Burke H., "Validation indices for graph clustering", in J.-M. Jolion, W. Kropatsch, M. Vento (eds.): Proc. 3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, 2001, pp. 229–238.

[14] Halkidi M., Batistakis Y., Vazirgiannis M., "On clustering validation techniques", JIIS, Vol. 17, 2001, pp.107-145

[15] Herman, I., Melançon, G., & Marshall, M. S. (2000). Graph Visualisation in Information Visualisation: a Survey. IEEE Transactions on Visualization and Computer Graphics, 6(1), 24-44.

[16] Hubert L.J., Arabie P., "Comparing partitions", Journal of Classification, Vol. 2, 1985, pp. 193-218

[17] Law M. H., Jain A. K. Cluster validity by bootstrapping partitions. Technical Report MSU-CSE-03-5, Department of Computer Science, Michigan State University.

[18] Mancoridis S., Mitchell B.S., Rorres C., Yih-Farn Chen, Emden R. Gansner: Using Automatic Clustering to Produce High-Level System Organizations of Source Code. IWPC 1998: 45-

[19] Pearson, K., 1900. Phil. Mag. Series 5, 50, 157.

[20] Ramaswamy L., Gedik B., Ling Liu: Connectivity Based Node Clustering in Decentralized Peer-to-Peer Networks. Peer-to-Peer Computing 2003: 66-73

[21] Rousseeuw P.J., "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis", J. Comp App. Math, Vol. 20, 1987, pp. 53-65

[22] van Dongen S., Performance criteria for graph clustering and Markov cluster experiments. Technical Report INS-R0012, National Research Institute for Mathematics and Computer Science in the Netherlands, Amsterdam, 2000

[23] Zhao, Y. and Karypis, G. Criterion functions for document clustering: experiments and analysis. Tech. Report #01-40, Departent of Comp. Sci. & Eng., U. Minnesota, 2001

François Boutin, Jérôme Thièvre, Mountaz Hascoët: *Multilevel Compound Tree – Construction Visualization and Interaction*, Proceedings of Interact 2005, Lecture Notes in Computer Science, LNCS 3585, Springer, pp. 847-860, 2005.

Multilevel Compound Tree - Construction Visualization and Interaction

François Boutin[1], Jérôme Thièvre[2], Mountaz Hascoët[3]

[1] LIRMM, CNRS, Montpellier, francois.boutin@univ-montp1.fr
[2] INA, Paris, jthievre@ina.fr
[3] LIRMM – CNRS, Montpellier, mountaz@lirmm.fr

Abstract. Several hierarchical clustering techniques have been proposed to visualize large graphs, but fewer solutions suggest a focus based approach. We propose a multilevel clustering technique that produces in linear time a contextual clustered view depending on a user-focus. We get a tree of clusters where each cluster - called *meta-silhouette* - is itself hierarchically clustered into an inclusion tree of *silhouettes*. Resulting *Multilevel Silhouette Tree* (*MuSi-Tree*) has a specific structure called *multilevel compound tree*. This work builds upon previous work on a *compound tree* structure called *MO-Tree*. The work presented in this paper is a major improvement over previous work by (1) defining *multilevel compound tree* as a more generic structure, (2) proposing original space-filling visualization techniques to display it, (3) defining relevant interaction model based on both focus changes and graph filtering techniques and (4) reporting from case studies in various fields: co-citation graphs, related-document graphs and social graphs.

1 Introduction

Graph clustering objective is to minimize inter-connectivity (edges between clusters) and maximize intra-connectivity (edges inside clusters). A multilevel clustering technique is used to organize large graphs 57. It provides a *hierarchical clustered graph* that consists of a graph of clusters where each cluster is itself hierarchically clustered. If we consider edges between clusters, we get a structure called *compound graph* 16 [section 2.2].

We are interested in multilevel clustering techniques that produce contextual graph views from any focus. The idea is to be able to organize information “around” a focus using graph connectivity.

Previous work on *MO-Tree* 3 introduced a focus based multilevel clustering technique that provides a *compound tree* structure [section 2.2]. It is a tree of clusters, where each cluster is itself hierarchically clustered. For instance, considering a connected graph in [Fig.1.], a *MO-Tree* view from focus 1 is displayed in [Fig.2.].

In this paper we define *multilevel compound tree* which improves the concept of *compound tree*. We also provide an algorithm for the construction of such structure.

We further propose a space-filling visualization technique and an interaction model for displaying and interacting with this structure. Moreover, we report results of three case studies using co-citation graphs, related-document graphs and social graphs.

In the next section, we present a synthesis of related work on graph clustering, and multilevel visualization/interaction techniques. Section 3 further describes the new *multilevel compound tree structure* called *MuSi-Tree* and its construction algorithm. In section 4, we propose space-filling visualisation techniques to display this structure and an associated interaction model. Finally, in section 5, we report the results of three case-studies that aim at:

– Producing real graphs visualization and interaction from various perspectives.

– Applying filtering techniques to produce understandable *MuSi-Tree* structures.

– Providing qualitative results from a set of 56 users [section 5.4].

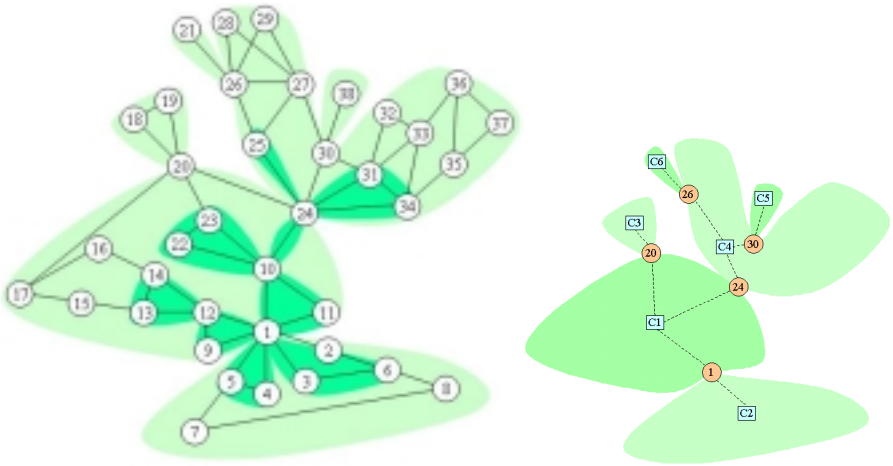

Fig.1. (a) Graph clustering into silhouettes (b) silhouette tree

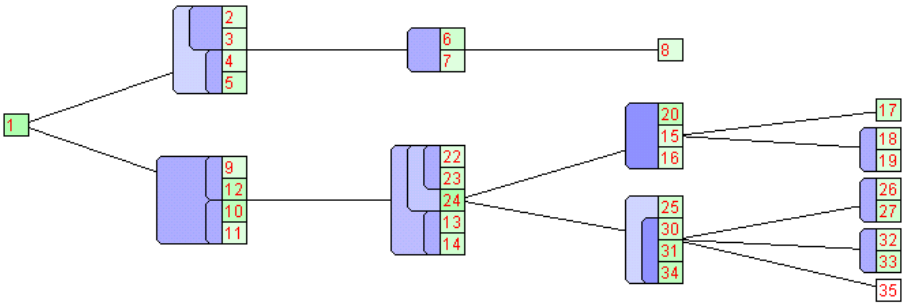

Fig.2. MO-Tree from focus 1

2 Related work

2.1 Graph partitioning techniques – a review

There are different ways to classify partitioning techniques 168. We propose main criteria and illustrate them with some methods (see table 1).

Firstly, geometrical techniques take into account nodes coordinates: for instance, *K-Means* 111 is a local clustering method that produces iteratively a k-partition using centroïds. Global geometric methods 6813 are based on energy models, inertial algorithms or graph bisection (with hyperplans or spheres).

Secondly, structural algorithms are based on graph connectivity. The principle of a local clustering technique is to minimize inter cluster connectivity using nodes exchange, agglomeration or separation. Global structural 1468 techniques use matrix operations: Markov clustering is a stochastic method based on flow computing while spectral techniques consist in mapping nodes into an eigenspace. Spectral techniques may be used to provide coordinates for geometric clustering methods.

Many algorithms 1113 use both global and local methods to refine clustering. Choice of clustering technique may depend on graph propriety (size, connectivity…), but also clustering constraints (balanced clustering, edge crossing minimization…).

Table 1: clustering criteria and methods

Partitioning criteria / Partitioning methods	Centroïd based – K-Means111	Geometric segmentation 68	Force directed based clustering 13	Inertial algorithm 68	Nodes exchanging – genetic 16	Agglomerative - greedy 1611	Min Cut 148	Spectral method 1468	Flow - Markov method148
geometric (x) vs. structural	x	x	x	x				→	
local (x) vs. global approach	x				x	x	x		
Iterative optimization (x) vs. exact	x	both	x		x	both	x		
hierarchical (x) vs. flat	→	both	both			x	both	both	
matrix computing.(x) vs. graph				x			x	x	x
stochastic (x) vs. determinist	x	x	x		x	both			x

Multilevel clustering techniques were developed to organize large graphs into hierarchical clustered graph [section 2.2]. For that purpose, some “flat” techniques are adapted. For instance *K-Means* can be applied recursively 11 (see → Table 1). Cutting techniques depending on a threshold may be computed with many thresholds.

Otherwise, hierarchical clustering techniques proceeds iteratively with merging or splitting the most appropriate clusters according some metric. Resulting dendrogram can be cut at different levels to provide a *hierarchical clustered graph* 111.

According to our criteria, our method is a structural-local-exact-hierarchical-graph-based-determinist graph clustering technique.

2.2 Multiscale structure – definitions and visualization

Now, we review and define new multilevel graph structures:

- A *hierarchical clustered graph* was defined 5 by a graph $G = (V, E)$ and a rooted tree T. Leaves of T are vertices of G. Other nodes in T are sets of nodes of G called *clusters*. T describes an inclusion relation between *clusters* so T is called inclusion tree. *2D* and 3*D* views [Fig.3] are proposed in 7: inclusion tree T is described either by inclusion areas [Fig.3.a] or dotted arrows [Fig.3.b].
- A *compound graph* consists of a *hierarchical clustered graph* with edges between clusters 16. *2D* and *3D* views are proposed in 57 [Fig.3.a,b]. Edges between clusters are drawn in black thin lines for small clusters and blue thick lines for large clusters: their removal induces a *hierarchical clustered graph*.
- We defined in 3 a *compound tree* as a *compound graph* with an adjacency tree of meta-clusters (largest clusters). For instance, [Fig.3.c] presents a *compound tree* since largest clusters belong to an adjacency tree. We proposed 3 flat *compound tree* visualization [Fig.2] of G [Fig.1.]*:* to avoid a cluttered view, edges between meta-clusters are displayed unlike edges between clusters in a meta-cluster.
- In this paper, we define a *multilevel compound tree* as a *compound tree* where each layer consists of an adjacency tree of clusters. For instance [Fig.3.d] meta-clusters but also small clusters belong to adjacency trees. We propose an algorithm that computes a specific *multilevel compound tree* structure called *MuSi-Tree*. This structure is general enough to be provided by other clustering techniques.

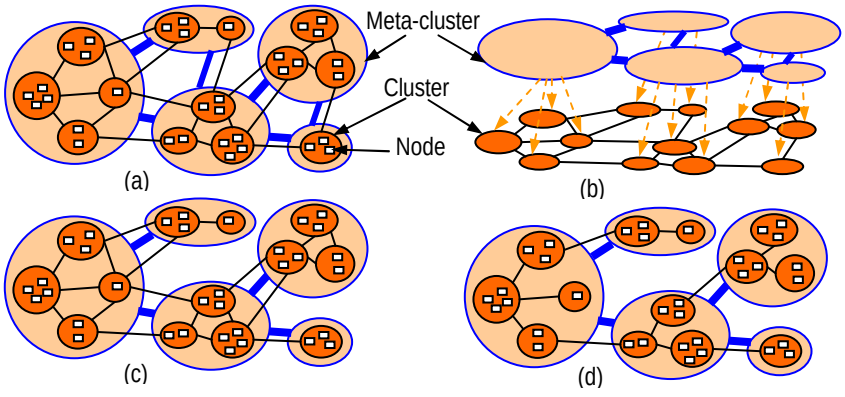

Fig.3. (a) & (b) compound graph, (c) compound tree, (d) multilevel compound tree

2.3 Visualization and interaction concepts – multilevel approach

2D and *3D* views of hierarchical clustered graphs based on tree layouts are described in 5 and 7 respectively. These visualizations support multilevel views by allowing user to choose level of visibility of cluster hierarchy, or manually collapse/expand clusters. Fisheye and full zoom methods are compared in 14.

On the other hand, the use of upgraded force-directed models to layout clustered graphs has been studied in 1017. The main idea here is to use different forces for intra and inter clusters links and add an invisible attractor vertex to force vertices of the same cluster to keep close to each other.

Yee and al. propose a focus-based radial tree layout of non-clustered graphs 18. The system builds a breadth-first spanning tree of the graph from a focus node selected by the user. The focus node is laid out at the centre of the display and the others nodes are located on concentric rings corresponding to their shortest distance to the focus. Angular position of a node depends on its parent position in the spanning tree. Angular width of each node is computed from the angle needed by its subtree. This system presents a particularly interesting work on animation of focus changes. When the user selects another focus, all nodes move to their new position, following an intuitive path with a smooth slow-in, slow-out timing. The *prefuse* toolkit 9 offers a smart implementation of this technique. Nevertheless, as mentioned earlier, neither particular visualization nor specific interactions have been implemented for multilevel data structures.

In this work we propose an add-on to *prefuse* 9 that supports the visualization of *multilevel compound trees* [Section 4]. This technique is used with *MuSi-trees* that are produced from our clustering technique. A *MuSi-Tree* consists of a particular case of *multilevel compound tree*. Nevertheless our visualization technique is not limited to *MuSi-tree*. It can be used for any type of *multilevel compound tree*. Since *multilevel compound tree* structure is general enough, we believe that this technique can be used to visualize other clustering techniques results.

3 Multilevel graph clustering technique

3.1 Main principles and definitions

Let us consider a connected graph $G = (V, E)$. A node is called *articulation node* if its removal disconnects G. Splitting (but not removing) articulation nodes disconnects G into maximal biconnected components or trivial components (two connected nodes). Indeed, any non trivial component is biconnected else it would be split up into components. Maximal biconnected or trivial components are called *silhouettes*.

Graph G is a *bipartite tree* that connects *articulation nodes* and *silhouettes*, see [Fig.1.b]. Indeed, if there was a cycle between *silhouettes*, then *silhouettes* should belong to a same component, what is impossible since they are maximal.

The *silhouette tree* decomposition can be applied to graph *G* but also to any connected sub graph *G'* of *G*. Considering a *silhouette* of *G'*, it naturally belongs to one *silhouette* of *G*. Consequently, we say that the *silhouette tree* of *G'* belongs to the *silhouette tree* of *G*. In [Fig.1.a], we present two *silhouette trees*: one for graph *G*, another one for a connected sub graph *G'* (see darkest included areas).

Now, the purpose is to be able to choose interesting sub graphs of *G*, in order to apply our algorithm recursively. We present our approach in [section 3.2].

3.2 Focus based clustering approach

We propose a clustering technique that takes into account a focus-node. So, we get different graph perspectives depending on the focus we select. Resulting computing is very efficient as explained in [section 3.3].

Nodes are displayed into K_{max} levels according to their distance to the focus. G_k is defined as the connected sub-graph of *G* that contains nodes at distance at most *k* from the focus. Sil_k is defined as the *silhouette tree* of G_k. The set of silhouettes trees $\{Sil_k\}$ is ordered with inclusion relation. It means that Sil_k includes Sil_i *if k is above i.*

$\{G_k\}$ is computed in linear time *using a breadth-first search.* Sil_{k+1} is also naturally computed from Sil_k [section 3.3] and resulting computing of $\{Sil_k\}$ is linear. The set of silhouettes $\{Sil_k, 0 \leq k \leq K_{max}\}$ is called *Multilevel Silhouette Tree* (*MuSi-Tree*).

Sil_{Kmax} is called *meta-silhouette tree*. It is a tree of *meta-silhouettes*. Each *meta-silhouette* is itself an inclusion tree of *silhouettes.*

3.3 Optimized Algorithm

Let consider graph G organized into K_{max} levels from a focus node. We describe our *MuSi-Tree* optimized algorithm [Fig.4]. $Sil_0 = \{focus\}$. Sil_{k+1} is computed using Sil_k:

- Level *k+1* is first organized into connected components. See [Fig.4], resulting nine connected components: *{ab}, {cd}, {e}, {f}, {g}, {hij}, {k}, {lm}, {no}.*
- If a connected component is related to only one node *v* on level *k*, *v* is an articulation node for Sil_{k+1} that connects a new *silhouette*. For instance, see [Fig.4], *{ab}, {cd}, {e}, {f}, {g}, {lm}* belong to new silhouettes.
- Else a large silhouette is created that includes silhouettes of Sil_k connected by this connected component. See [Fig.4]: components {*hij}, {k}, {no}*

So, the resulting *MuSi-Tree* structure is computed in linear time.

3.4 Invariant sets

We can get various MuSi-Trees depending on the focus we take. However, meta-silhouette tree structure remains the same. Indeed, global articulation nodes and meta-

silhouettes are graph invariants. Focus changes will be used in the interaction model and will benefit this property as explained in the next section.

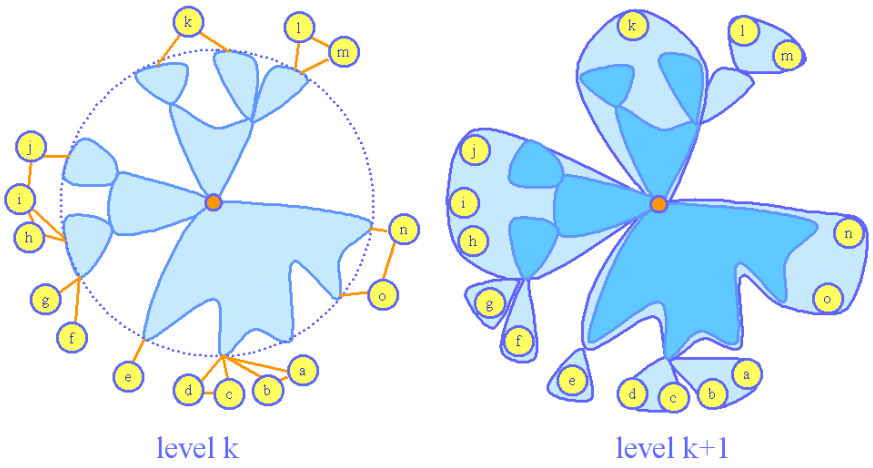

Fig.4. Graph clustering algorithm – step k+1

4 *MuSi-Tree* visualization and interaction

4.1 *MuSi-Tree* layout

The nature of our clustering, based on a focus node and distances between nodes and this focus, leads us to choose a hierarchical layout. Among existing hierarchical layout, radial layout is really well-suited. Nodes are located on concentric rings around focus, which makes distance between each node and the focus explicit 18. Moreover, radial layout makes a better use of screen space than classical top-down tree layout.

Our layout is quite similar to radial layout 18. Our main contribution is in the drawing of silhouettes. To perform this, we started by transforming *MuSi-Tree* into a well-suited spanning tree. This mainly implies particular ordering of spanning tree nodes to ensure (1) silhouettes graphical cohesiveness and (2) non-overlapping. Once nodes locations have been computed with respect to ordering constraint, we perform a depth-first traversal and draw each silhouette as the bounding shape of its nodes.

Initial graph (Fig. 5.a) is iteratively clustered by level into meta-nodes (light blue areas Fig. 5.b) using a former technique introduced in 3. Each silhouette is computed as a tree of meta-nodes (Fig 5.c). Geometric outline of this tree generates the drawing of silhouette (blue areas).

We attribute different colors to silhouettes trees to easily recognize non trivial bi-connected components in the graph. Each hierarchy of silhouettes is attached to an articulation node that is painted with the same color. Articulation nodes are not included in their corresponding silhouette to avoid multiple overlaps that would make the visualization too complex. The use of transparent colors to fill silhouettes helps to perceive their inclusion level: the deeper they are, the darker they look.

Visualization is built over the *prefuse* framework 9 that provides support for animated transformations on graph layout. It is used to animate our focus changes.

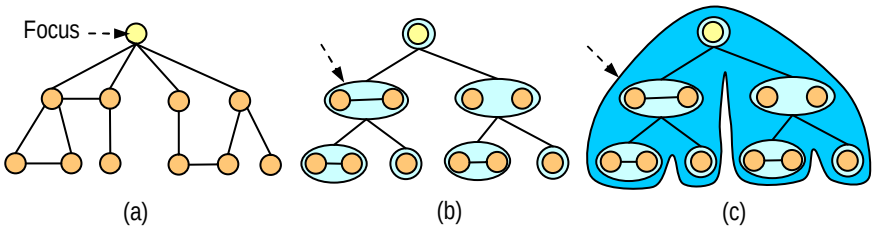

Fig.5. (a) Initial graph: hierarchical view (b) clustering by level (c) silhouette drawing

4.2 Changing focus

Transitions between focus changes are animated, to help the user in tracking objects of interest and keeping a coherent perception of graph structure 18. At this moment, only nodes and links are animated. The silhouettes are not displayed during transition because it is quite hard to get smooth animation of their transformation since their shape can change completely between two states, but we keep working on this interesting problem.

4.3 Dynamic Filtering

Graph visualizations provide great information on data structures and relations, and some properties can be displayed by using visual attributes like colours, size and shape. However, it is very hard to show the complete information attached to a node inside the graph view, and it is especially true for textual information.

To solve this problem, we use a traditional data table to display nodes and their attributes. This widget supports multiple sorts and filters. Combination of these two features allows users to perform quick searches and selections over the whole dataset. We propose different filtering features depending of nodes and edges attributes types:

- Regular expressions for textual attributes,
- Minimum and maximum threshold for numeric attributes,
- Temporal intervals for dates.

Filtering is easily reversible and its effects (dynamic insertions, nodes and links removals) are animated to offers users a good perception of changes.

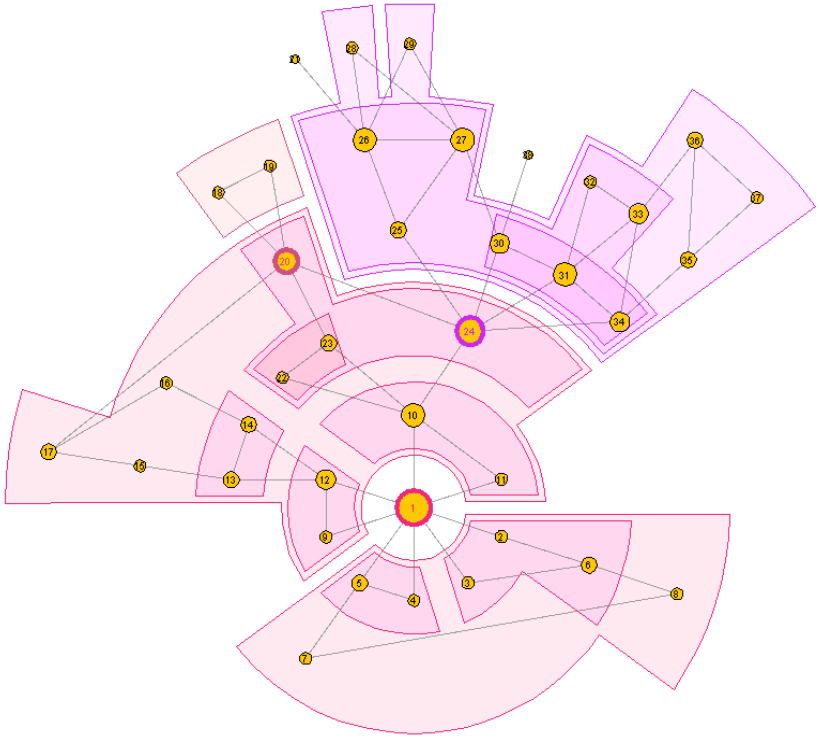

Fig.6. Multilevel Silhouette Tree (MuSi-Tree) from focus 1

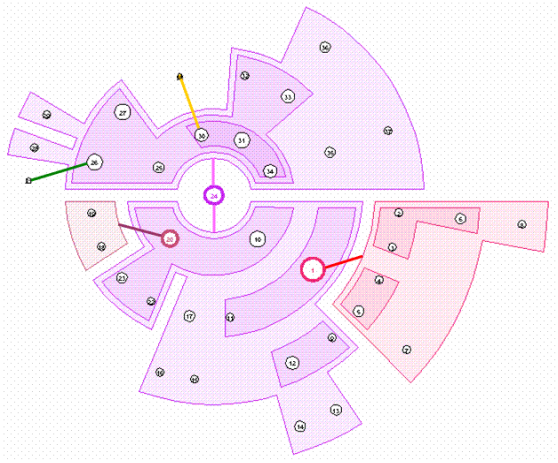

Fig.7. Multilevel Silhouette Tree (MuSi-Tree) – compact view from focus 24

5 Case studies

5.1 Citation graph

We have clustered a connected citation graph including 122 papers and 206 links. It was collected on Research Index (Citeseer) 12 from focus: “Navigation and interaction within graphical bookmarks” [Fig.8]. Three large silhouettes are related to: information visualization, database visualization and document classification [Fig.9]. All silhouettes and articulation nodes belong to a silhouette tree [Fig.9].

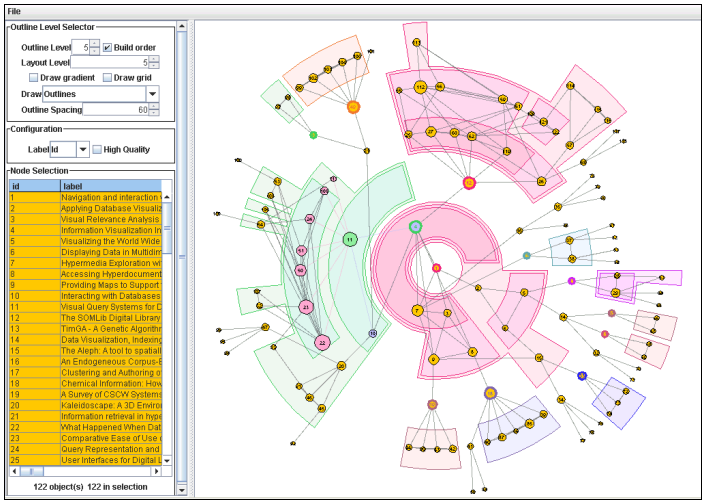

Fig.8. MuSi-Tree Viewer – citation graph clustering

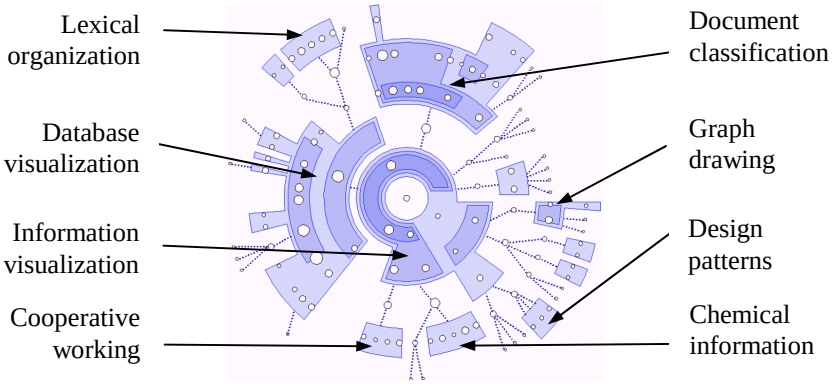

Fig.9. MuSi-Tree – compact view

5.2 Related-document graph

The aim is to propose a map of conferences "around" Interact 2005. For that purpose, we use *TouchGraph GoogleBrowser* 15 that iteratively produces a *spring* view of 101 conferences related to www.interact2005.org [Fig.10.b]. To manage a well connected graph, we consider nodes with 2 or more incoming edges. We get 200 edges.

We present *MuSi-Trees* from two foci: www.interact2005.org [Fig.10.a] and NYC 2004: www2004.org [Fig.11]. Resulting views share five non trivial silhouettes that represent five specific domains: visualization/interaction, ergonomics, computational linguistic, digital library and image processing. We get different articulation nodes between domains: NYC 2004 and SIGIR 2004 introduce digital library and computational linguistic. VIS 2005 connects visualization/interaction with image processing. HCII 2005 is a bridge between visualization/interaction and ergonomic.

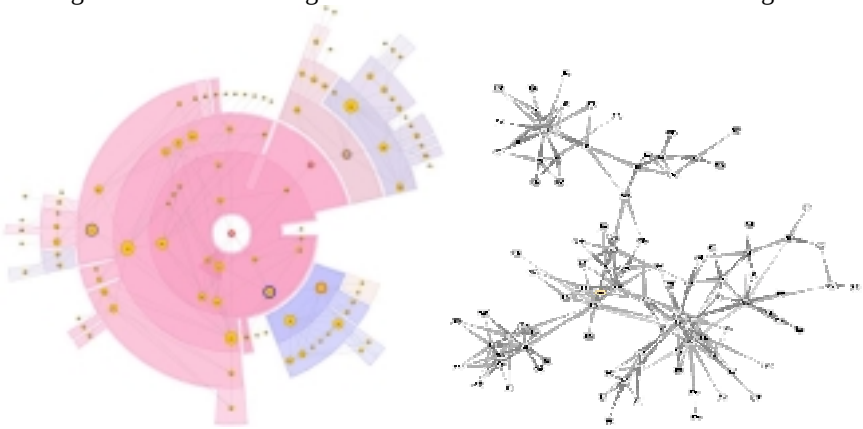

Fig.10. (a) MuSi-tree from focus "Interact 2005" , (b) TouchGraph view 15

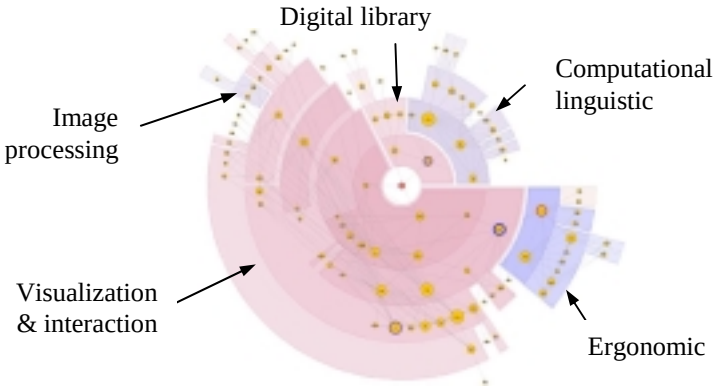

Fig.11. Organization from focus: "NYC 2004"

5.3 Social graph

In this study, 56 medical students divided into four work groups were asked about their friendship relations (strong or weak). To begin with, whole social graph including all relations (unilateral and bilateral, strong and weak) is displayed in [Fig.12]. We get a single well connected component.

Then, a filtering algorithm is applied to include bilateral strong relations. Social graph is displayed from the same focus [Fig.13]. We denote one large component in relation with four smaller components. In fact, each small group belongs more or less to a work group. Five students create relations between these groups.

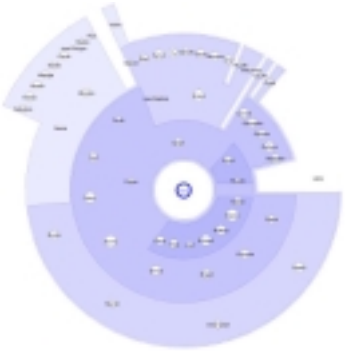

Fig.12. Whole relations between students

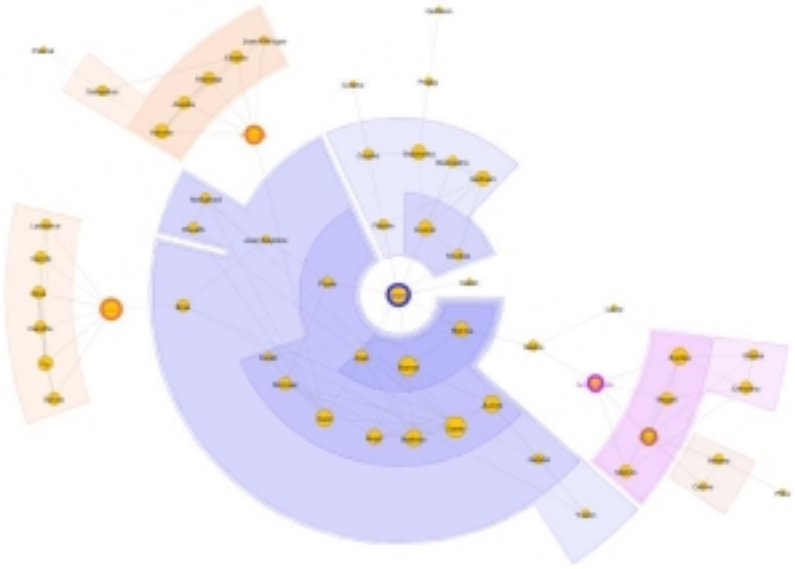

Fig.13. Strong bilateral relations between students

5.4 Qualitative evaluation

After collecting data, resulting social *MuSi-Trees* were presented to the four groups of medical students (without experience in information visualization). A five minutes show included visualization and interactive techniques: focus changing and group zooming. *MuSi-Tree* was presented as a "strong friendship map" around a student. No more explanation about the clustering algorithm was given. Students were invited to express themselves about their feeling on the clustering results, and the visualization/ interaction technique. The study reveals qualitative results:

- Majority of students considered the map as meaningful, even though they knew nothing about clustering technique.
- They recognized many natural groups of friends.
- They naturally understood dark areas as cohesive groups.
- Students were interested in changing focus to draw their own friendship map. They sometimes felt lost especially when focus moved from a silhouette to another.
- Students had sometimes difficulties to follow nodes when the graph was moving. Nevertheless, they often recognized silhouettes from their shape or their color.

This preliminary study invites us to practice a quantitative study to compare both our clustering technique, and our visualization/interaction technique with other ones.

6 Conclusion

In this paper we proposed a focus-based multilevel agglomerative technique. Resulting structure called *Multilevel Silhouette Tree* (*MuSi-Tree*) is easy to explore and easy to manage because of its features (*multilevel compound tree*).

We applied our algorithm to a citation graph, a related-document graph and a social graph. Citation graph was displayed in a compact view that reveals *silhouette tree* structure. We present a graph of related sites displayed from two foci. Resulting views share meta-silhouettes and articulation nodes. Applying filtering techniques help the user understanding clustering structure.

MuSi-Tree appeared to be particularly well suited to organize a *locally well articulated graph*. It consists of a graph with "some" articulation nodes in its various k-neighbourhood. It will be interesting to study *locally well articulated graph* indices. In order to improve visualisation of large graphs with high connectivity we propose to use filtering techniques in a pre-processing stage of our clustering technique. The objective will be to extract a "nice" *multilevel silhouette tree* structure.

More intensive evaluation is needed to better clarify conditions in which our clustering and visualization techniques can best benefit users interacting with large graphs. The question of what criteria should be used for the evaluation of clustering and visualization results is an open issue. A good evaluation of this work can only be a long and challenging task. We have already conducted many informal evaluations and studies on this work, but we are now deeply engaged into a more controlled evaluation approach. In this area, we plan to perform larger evaluation both in terms of analytical criteria 2 as in terms of controlled experiments involving real users.

References

1. Alpert C.J. and Kahng A.B. Recent Developments in Netlist Partitioning: A Survey, Integration: the VLSI Journal, vol. 19, pp. 1-81, 1995.
2. Boutin F. and Hascoët M., Cluster Validity Indices for Graph Partitioning, Proceedings of the Conference on Information Visualization IV'2004.
3. Boutin F. and Hascoët M., Focus Dependent Multi-level Graph Clustering. Proceedings of the Conference on Advanced Visual Interfaces, AVI 2004, ACM.
4. Brandes U., Gaertler M., and Wagner D.: Experiments on Graph Clustering Algorithms. Proc. ESA '03, LNCS 2832, pp. 568-579. © Springer-Verlag, 2003.
5. Brockenauer R. and Cornelsen S., Drawing Clusters and Hierarchies. In Michael Vauf-mann and Dorothea Wagner (Eds.): Drawing Graphs: Methods and Models, LNCS 2025, pp. 194 - 228. © Springer-Verlag, 2001.
6. Chamberlain B.L. Graph Partitioning Algorithms for Distributing Workloads of Parallel Computations. Technical Report UW-CSE-98-10-03, Univ. of Washington, 1998.
7. Eades P., Multilevel Visualization of Clustered Graphs, Proceedings of Graph Drawing'96, Berkeley, California, September, 1996.
8. Elsner U. 1997. Graph partitioning - A survey. Technical Report 393, Technische Universitat Chemnitz.
9. Heer J., Card S.K., and Landay J.A., prefuse: a toolkit for interactive information visualization. In CHI 2005, Human Factors in Computing Systems, 2005.
10. Huang M. L., Eades P. A Fully Animated Interactive System for Clustering and Navigating Huge Graphs, Proceedings of the 6th International Symposium on Graph Drawing (GD'98), Springer LNCS 1547, pages 374-383, 1998.
11. Karypis G., Han E-H and Kumar V. CHAMELEON: A hierarchical clustering algorithm using dynamic modeling. IEEE Computer, 32(8): 68-75, 1999.
12. Lee Giles C., Bollacker K.D., Lawrence S.: CiteSeer: An Automatic Citation Indexing System. ACM DL 1998: 89-98.
13. Noack A. An energy model for visual graph clustering. In G. Liotta, editor, Proceedings of GD 2003, LNCS 2912, pages 425--436, Berlin, 2004. Springer-Verlag.
14. Schaffer D., Zhenping Z., Grreenberg, S. et al. Navigating Hierarchically Clustered Networks through Fisheye and Full-Zoom Methods, ACM Transactions on Computer-Human Interaction volume 3-2, pages 162-188, 1998.
15. Shapiro A., TouchGraph, Dynamic Graph Layout Tool, www.touchgraph.com
16. Sugiyama K., Tagawa S. and Toda M. Methods for Visual Understanding of Hierarchical System Structures. IEEE Transactions on Systems Man and Cybernetics, 1981.
17. Wang X., Miyamoto I. Generating Customized Layouts, Proceeding of the 3rd International Symposium on Graph Drawing (GD'95), Springer LNCS 1027, 504-515, 1995.

18. Yee K.P., Fisher D., Dhamija R. & Hearst M.A. Animated Exploration of Dynamic Graphs with Radial Layout. Proceeding of IEEE Symposium on Information Visualization, 2001.