



HAL
open science

Increasing the expressive power of gesture-based interaction on mobile devices

Jessalyn Alvina

► **To cite this version:**

Jessalyn Alvina. Increasing the expressive power of gesture-based interaction on mobile devices. Human-Computer Interaction [cs.HC]. Université Paris Saclay (COmUE), 2017. English. NNT : 2017SACLS526 . tel-01744136

HAL Id: tel-01744136

<https://theses.hal.science/tel-01744136v1>

Submitted on 27 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Increasing The Expressive Power of Gesture-based Interaction on Mobile Devices

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'Université Paris-Sud

École doctorale n°580
Sciences et technologies de l'information
et de la communication (STIC)
Spécialité de doctorat : informatique

Thèse présentée et soutenue à Gif-sur-Yvette, le 13/12/2017, par

Jessalyn ALVINA

Composition du Jury :

Géry Casiez Professeur des universités, Université de Lille 1	Président
Shumin Zhai Ingénieur de Recherche, Google	Rapporteur
Andy Cockburn Professeur des universités, University of Canterbury	Rapporteur
Eric Lecolinet Maître de Conférences, Télécom Paris-Tech	Examineur
Wendy Mackay Directrice de Recherche, INRIA Saclay	Directeur de thèse

Increasing The Expressive Power of Gesture-based Interaction on Mobile Devices

A DISSERTATION PRESENTED

BY

JESSALYN ALVINA

TO

THE DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITÉ PARIS-SACLAY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
HUMAN-COMPUTER INTERACTION

UNIVERSITÉ PARIS-SACLAY
SACLAY, FRANCE
DECEMBER 2017

©2017 – JESSALYN ALVINA
ALL RIGHTS RESERVED.

Increasing The Expressive Power of Gesture-based Interaction on Mobile Devices

ABSTRACT

Current mobile interfaces let users directly manipulate the objects displayed on the screen with simple stroke gestures, e.g. tap on soft buttons or menus, pan to scroll a document, or pinch to zoom. These simple gestures on buttons, icons, or menus were a solid foundation to support the transition from WIMP to touch-based interfaces. However, today's smartphones offer a wide variety of functionalities, from communication, games, information consumption, office support, even art and creativity support [12]. To access a larger command space, the users are often forced to go through long steps, making the interaction cumbersome and inefficient. More complex gestures offer a powerful way to access information quickly as well as to perform a command more efficiently [5]. However, they are more difficult to learn and control. Gesture typing [78] is an interesting alternative to input text: it lets users to draw a gesture on soft keyboards to enter text, from the first until the final letter in a word. Since the gesture is drawn on top of a soft keyboard, each key acts as a “steering target” of how to draw the gesture.

I am interested in rethinking the access to a variety of functions, by building mobile interactions that are both easy to learn and control yet more powerful. I envision users just interact as they usually do with the existing systems, yet they can easily access more powerful functionalities. I am specifically interested in text entry, since most of mobile interaction involves command invocation and text entry.

I believe we can increase the expressive power of mobile interaction by leveraging the gesture's shape and dynamics and the screen space to produce rich output, to invoke commands, and to facilitate appropriation in different contexts of use. I argue that humans are capable to perform various movements with different level of precision, especially after a certain amount of practice. It is very difficult, if not impossible, to perform exactly the same movement twice. Each person also performs differently from one to another. Hence, if we can distinguish the variations in the gesture, I believe we can exploit the variability to express different concepts.

Once users gain more control over their variability, we can also introduce slightly more complex interactions that enable way more powerful functionalities.

In this thesis, I show that users' gesture performance vary substantially across users (due to e.g. biomechanics or personality) and context (e.g. activity or environment). I present three features that form a low-dimensional representation of gesture variation and users can significantly vary under different conditions. I design a new interaction technique, EXPRESSIVE KEYBOARD, that transforms the gesture variations into rich output, and demonstrate several applications in the context of text-based communication. Users can gesture type as they usually do, but get more power to express different concepts. I then introduce a series of controlled experiments that investigate the learning and appropriability aspects of the proposed technique. The result shows that users can deliberately control these additional features of their gestures as they gesture type, in addition to their natural gesture variations. Users quickly appropriated EXPRESSIVE KEYBOARD as soon as they used it and took the generated rich output as a feedback of their gesture-typing performance, which then improved their performance. I propose COMMANDBOARD, a gesture keyboard that lets users efficiently select commands from a large command space while supporting the transition from novices to experts. I demonstrate different applications of COMMANDBOARD, each offers users a choice, based on their cognitive and motor skills, as well as the size and organization of the current command set. Altogether, these techniques give users more expressive power by leveraging human's motor control and cognitive ability to learn, to control, and to appropriate.

Increasing The Expressive Power of Gesture-based Interaction on Mobile Devices

ABSTRACT

Les interfaces mobiles actuelles permettent aux utilisateurs de manipuler directement les objets affichés à l'écran avec de simples gestes, par exemple cliquer sur des boutons ou des menus, naviguer dans des documents ou pincer pour zoomer. Ces gestes simples sur des boutons, des icônes ou des menus étaient de solides bases pour passer des interfaces WIMP aux interfaces mobiles. Cependant, les smartphones offrent aujourd'hui des fonctionnalités variées, depuis la communication, les jeux, la consommation d'information, la bureautique ou bien même l'art et les outils supports de créativité [12]. Pour accéder à un espace de commande plus large, les utilisateurs sont souvent forcés de passer par de nombreuses étapes, rendant l'interaction inefficace et laborieuse. Des gestes plus complexes sont un moyen puissant d'accéder rapidement à l'information ainsi que d'exécuter des commandes plus efficacement [5]. Ils sont en revanche plus difficiles à apprendre et à contrôler. Le "Gesture Typing" (saisie de texte par geste) est une alternative intéressante au texte tapé: il permet aux utilisateurs de dessiner un geste sur leur clavier virtuel pour entrer du texte, depuis la première jusqu'à la dernière lettre d'un mot. Puisque le geste est dessiné sur un clavier virtuel, chaque touche devient une "cible directionnelle" qui dirige le tracé du geste.

Je m'intéresse à repenser l'accessibilité à de nombreuses fonctionnalités en créant des interactions mobiles qui soient à la fois faciles à apprendre et à contrôler tout en étant plus puissantes. J'envisage que les utilisateurs puissent continuer à interagir de la même manière que ce qu'ils font aujourd'hui tout en leur permettant d'accéder à des fonctionnalités plus avancées. Je m'intéresse en particulier à la saisie de texte car c'est, avec l'invocation de commande, le moyen le plus répandu d'interagir avec un mobile.

Je pense qu'il est possible d'augmenter le pouvoir d'expression de l'interaction mobile en tirant profit de la forme et de la dynamique du geste et de l'espace de l'écran, pour invoquer des commandes ainsi que pour faciliter l'appropriation dans différents contextes d'usage. Je soutiens que les humains sont capables d'exécuter divers mouvements avec différents niveaux

de précision, en particulier après un certain temps d'entraînement. Il est très difficile, sinon impossible, d'exécuter deux fois le même mouvement. Chaque personne exécute le même geste différemment des autres personnes. Ainsi, si l'on peut distinguer les variations des gestes, j'affirme que l'on peut exploiter la variabilité pour exprimer différents concepts. Lorsque les utilisateurs acquièrent plus de contrôle sur leur variabilité, on peut également introduire des interactions un peu plus complexes qui rendent possibles des fonctionnalités plus puissantes.

Dans cette thèse, je montre que l'exécution d'un geste par les utilisateurs varie grandement entre les utilisateurs (selon la personnalité ou pour des raisons biomécaniques par exemple) et entre les contextes (selon l'activité ou l'environnement, par exemple). Je présente trois caractéristiques qui quantifient la variabilité d'un geste. Je conçois une nouvelle technique d'interaction, EXPRESSIVE KEYBOARD, qui transforme la variation du geste en un résultat riche et je démontre plusieurs applications dans le contexte de la communication textuelle. Les utilisateurs peuvent écrire gestuellement (gesture type) comme ils en ont l'habitude, mais ils ont maintenant le pouvoir d'exprimer différents concepts. J'introduis ensuite une série d'expériences qui explorent l'apprentissage et l'appropriabilité de la technique proposée. Le résultat montre que les utilisateurs peuvent délibérément contrôler ces paramètres supplémentaires de leurs geste lorsqu'ils écrivent gestuellement, en plus de la variation naturelle de leurs gestes. Les utilisateurs se sont rapidement approprié EXPRESSIVE KEYBOARD dès qu'ils s'y sont habitués et ont utilisé le résultat comme un retour sur leur performance d'écriture gestuelle, ce qui leur a permis d'améliorer encore leur performance. Je propose COMMANDBOARD, un clavier gestuel qui permet aux utilisateurs de sélectionner efficacement des commandes parmi un vaste choix tout en supportant la transition entre les novices et les experts. Je démontre plusieurs applications de COMMANDBOARD, dont chacune offre aux utilisateurs un choix basé sur leurs compétences cognitives et moteur, ainsi que différentes tailles et organisations de l'ensemble des commandes. Ensemble, ces techniques donnent un plus grand pouvoir expressif aux utilisateurs en tirant profit de leur contrôle moteur et de leur capacité à apprendre, à contrôler et à s'approprier.

Contents

1	INTRODUCTION	I
1.1	Thesis Statement	4
1.2	Research Approach	4
1.3	Contributions	5
1.3.1	Collaborators	6
1.4	Thesis Overview	6
2	RELATED WORK	9
2.1	Gestures on Touchscreen Devices	9
2.2	Zero-order Gestures	10
2.2.1	Increasing Recognizability	11
2.2.2	Augmenting Zero-order Gestures	11
2.3	Unistroke Gestures	12
2.3.1	Challenges	13
2.3.2	Designing Gesture-based Interactions	14
2.4	Gesture Keyboards	15
2.4.1	Gesture Recognition Process	15
2.4.2	Learnability Aspects	17
2.5	Augmenting Soft Keyboards	18
2.5.1	Generating Expressive Output	19
2.5.2	Invoking Commands from Keyboards	19
2.6	Position of My Work	20
2.7	Generative Design Principals	21
2.7.1	Substrate	22
2.7.2	Co-Adaptation	22
2.8	To Leverage Gesture’s Shape and Dynamics	24
2.8.1	Spatial Features	25
2.8.2	Temporal Features	26
2.8.3	Technical Features	27
2.9	Summary	27
3	GESTURE VARIABILITY	28
3.1	Preliminary Observation on Gesture Variability	29
3.1.1	Procedure	30

	3.1.2	Observations	30
3.2		Experiment 1: Deliberate Gesture Variation	31
	3.2.1	Participants	32
	3.2.2	Apparatus	32
	3.2.3	Procedure	33
	3.2.4	Data Collection	34
	3.2.5	Results and Discussion	35
3.3		Quantifying Variation in Gesture-Typing	38
	3.3.1	Speed	40
	3.3.2	Inflation	41
	3.3.3	Curviness	42
3.4		Summary	44
4		TRANSFORMING GESTURE VARIATION INTO RICH OUTPUT	45
4.1		Expressive Keyboard	45
4.2		Applications	47
	4.2.1	Colorful Text	47
	4.2.2	Dynamic Font	49
	4.2.3	Parametric Emoticon	50
	4.2.4	Animated Emoticon	50
4.3		Technical Implementation	52
	4.3.1	Initial Implementation	52
	4.3.2	Revisited Implementation	53
4.4		Summary	54
5		LEARNABILITY AND APPROPRIABILITY	55
5.1		Experiment 2: Learning to Control Gesture Variations	55
	5.1.1	Description	56
	5.1.2	Participants	57
	5.1.3	Apparatus	57
	5.1.4	Procedure	59
	5.1.5	Data Collection	59
	5.1.6	Results	60
	5.1.7	Discussion	63
5.2		Learning Strategies	65
5.3		Experiment 3: Ecological Validity	68
	5.3.1	Description	68
	5.3.2	Participants	69
	5.3.3	Hardware and Software	69
	5.3.4	Procedure	69
	5.3.5	Data Collection	70

5.3.6	Results and Discussion	70
5.4	Implication For Designs	74
5.5	Summary	75
6	INVOKING COMPLEX COMMANDS FROM GESTURE KEYBOARDS	78
6.1	Revisiting Mobile Device’s Screen Real Estate	79
6.2	CommandBoard	80
6.2.1	Type-and-Execute Commands	82
6.2.2	Inline Gesture Shortcut	86
6.3	Gesture Chunking with COMMANDBOARD	88
6.4	Learnability Aspects	88
6.5	Cognitive and Motor Efficiency	89
6.5.1	Experiment Description	90
6.5.2	Method	91
6.5.3	Participants	91
6.5.4	Hardware and Software	91
6.5.5	Procedure	92
6.5.6	Measures	95
6.5.7	Data Collection	96
6.5.8	Result: Experiment	96
6.5.9	Result: Preferences Study	99
6.6	Discussion	102
6.7	Summary	102
7	CONCLUSION	105
7.1	Transforming Mobile Devices into Powerful, Personalized Tools	105
7.2	Contributions	107
7.3	Perspective	108
7.3.1	Building Human-Computer Partnership	109
7.3.2	Screen Spaces as Different Substrates	111
8	APPENDIX A: ACCESSING HIERARCHICAL COMMANDS FROM COMMAND-BOARD	112
	REFERENCES	121

Listing of figures

1.1	Gesture-typing [42, 82] is an efficient, easy-to-learn, and error tolerant technique for entering text on software keyboards. All four gestures in this figure are correctly recognized as “great”.	3
1.2	Overview of the thesis and the used design methods. It includes controlled experiments and informal observations. The understanding and theory from the studies served as the fundamentals to design and redesign new interaction techniques.	7
2.1	Different type of gestures based on the stroke length and the number of contact points.	10
2.2	The multi-channel recognition engine of gesture keyboards, taken from [42].	16
2.3	The difference between a novice and an expert when using gesture keyboards.	18
3.1	Gesture variations: a) <i>accurately</i> is straight, b) <i>quickly</i> is smooth, and c) <i>creatively</i> is inflated and highly varied.	34
3.2	Recognized creative gestures included: a) loop and cusp for <i>taxi</i> , b) visualization of crown for <i>queen</i> and c) scribbling on keys to create the stars or a constellation for <i>midnight</i>	36
3.3	Accuracy of all participants for KB-1 and KB-2 (range from 0 to 1) in the Experiment 1.	37
3.4	Speed ratio vs. angle by length. Participants slowed down when drawing short and zero-angle (straight-line) words, but maintained constant speed when drawing long words.	41
3.5	Recognized gestures from different participants in the Experiment 1.	43
4.1	The process of transforming gesture input into rich output with <i>Expressive Keyboard</i> . 1) The gesture keyboard recognizes the gesture into a word just as before, as if it is a “blackbox”. <i>Expressive Keyboard</i> adds additional steps (in grey): 2) quantifies the variations in the captured gesture input and 3) maps them to output properties. Finally, <i>Expressive Keyboard</i> renders the recognized word along with the output properties as a rich output.	46

4.2	The font engine lets users to draw their own typeface by adding control points that are connected to a cardinal spline. Users can include several strokes by detaching the splines. On the right are two examples of static typefaces defined using the font engine: the baseline (top-right) and the variation (bottom-right).	50
4.3	Manipulating the curviness and inflation in gestures with EXPRESSIVE KEYBOARD will generate emoticons with subtle different of expression. a) A gesture with straight stroke segments generates a little smile; b) a curvy gesture generates a big smile; and c) a curvy, inflated gesture with wobbles at the end generates a laughing face.	51
5.1	Participants are significantly more likely to control gestures based on output than on input.	60
5.2	Participants can intentionally control gesture size when asked (inflation ratio), but do not vary it otherwise.	62
5.3	Successful control of color through gesture: a) [<i>instruction: Different-Output</i>] bright-green indicates curviness, dark green indicates straight lines; b) [<i>instruction: Varied-Input</i>] inflating the gesture increased red values; c) [<i>instruction: recipients</i>] P8 changed the color deliberately for different recipients; d) [<i>instruction: express emotion</i>] P4 made curvier gestures with more detours for ‘happy’, and slow and less curvy gestures for ‘sad’.	63
5.4	Using Expressive Keyboard in Experiment 3: a) P2 naturally made curvy gesture (green) but made straight gestures (dark green) to emphasize some words, b) P6 deliberately made two words (“burger shop”) the same shape and color, c) P3 emphasized the first word, but then deliberately made his gesture more precise (dark color) instead of curvy.	72
5.5	Participants successfully varied or kept constant the inflation and speed consistency when instructed. They naturally made curvy gestures, but could make them more curvy when instructed.	73
6.1	The screen real estate in the current mobile interfaces.	80
6.2	COMMANDBOARD specifies multiple command entry spaces. New methods (in green) include typing a command name followed by an EXECUTE gesture; crossing a suggested command in the <i>command bar</i> ; or executing a unique gesture in the upper <i>command gesture input space</i> . An in-context dynamic guide shows gesture-command mappings.	82
6.3	A diagram describing TYPE-AND-EXECUTE technique of COMMANDBOARD. The text in green represents the new methods.	83

6.4	COMMANDBOARD creates a new <i>command gesture input space</i> above a soft keyboard. Users can: a) type ‘happy’ and use a dynamic guide to style it as bold; b) type ‘brightn’, draw an execute gesture and adjust the brightness slider; c) type ‘sans’, choose ‘sans mono’ and draw an execute gesture to change the font; d) type ‘color’, select yellow in the marking menu to change the brush color.	84
6.5	After typing an existing command name, a) a preview appears. Since “Line Spacing” is a sub-menu, its menu items appear in the command bar. After sliding across the command bar, b) performing a gesture will cancel writing, whereas c) performing a ^ gesture will execute the command. . . .	85
6.6	Gesture set: Grey circles indicate where to begin drawing.	92
6.7	Part A (Experiment): Each condition (Practice, Experiment, Pre-test, and Post-test) includes two blocks of six trials, one per technique, with three replications in the experimental condition. Part B (Study): Participants recompose 12 of their own messages, with free choice of technique.	93
6.8	Each trial presents instructions above the line, and the result below the line. Practice and Experimental conditions present a) INLINE GESTURE SHORTCUTS to draw, or b) MARKDOWN SYMBOLS to type, to issue the specified styling command. Pre- and Post-Test conditions present c) the styled text to reproduce with the specified technique, with no feedback.	93
6.9	Average time spent entering each word. <i>WO₂</i> is the styled word. Using INLINE GESTURE SHORTCUTS is significantly faster: almost double.	97
6.10	Average time spent gesture-typing (typing time) and issuing the command (command time). Participants drew quickly with INLINE GESTURE SHORTCUTS, but took significantly longer inserting MARKDOWN SYMBOLS.	98
7.1	Reciprocal co-adaptation focuses on the human-computer partnership in which both the user and the system learn from and adapt each other in order to give more power to the users.	109
8.1	COMMANDBOARD enables users to display and parameterize command from widgets: a) type ‘color’ draw an execute gesture and pick a color from the color palette; b) type ‘table’, draw an execute gesture and specify how many rows and columns in the inserted table.	113

Listing of tables

3.1	Gesture-typing accuracy in the Preliminary Observation with three participants.	30
3.2	Three word sets used in the Experiment 1. Each word set contains of twelve words of different length, letter repetition, and angle between stroke segments.	32
3.3	Subjective features the participants in Experiment 1 used to vary the gestures extracted from the questionnaire. A=accurately, Q=quickly, C=creatively.	39
4.1	Summary of mappings to generate colors and dynamic font with EXPRESSIVE KEYBOARD.	48
5.1	Experiment 2, Block 1: All instructions that vary according to FOCUS (<i>output</i> or <i>input</i>) and LEVEL OF RELIABILITY (<i>consistent</i> , <i>different</i> , or <i>varied</i>). Each condition has two goals, representing two categories of proficiency. The goal is measured through the value of the RGB component.	58
5.2	Experiment 2, Block 2: All instructions based on message recipient or sender emotion, replicated three times.	58
5.3	Accuracy in Experiment 3. Participants changed the way they gesture-type with different keyboards, and are more likely to explore with EXPRESSIVE KEYBOARD. Although using EXPRESSIVE KEYBOARD does not necessarily decrease word-accuracy, participants erased correct words to modify the output properties leading to low values for feature-accuracy.	71
6.1	Step comparison for COMMANDBOARD and pulldown on touch screen devices from when the users switch from typing to command selection until switching back to typing.	81
6.2	Participant ratings of how each technique helped them to style text (median values; * indicates a significant difference). Participants significantly preferred gestures in all categories except 'confidently'.	101

TO JESUS CHRIST AND MOTHER MARY,
TO WHOM I AM ALWAYS A BELOVED WEE LAMB.

Acknowledgments

A three-year period is a short Ph.D period yet a long journey of life. My life as a Ph.D student is full of adventure: a thorough learning process, an exciting experience, emotional rollercoasters, people come and go – all so generously contributed to the work I presented in this thesis, as well as to my personal development.

First and foremost, I offer my sincerest gratitude to my supervisor, Wendy Mackay, who has been supporting me throughout my research life. She was the one who convinced me that there is so much fun in research when I was still a master student, through her passions and teachings, and that I should pursue research. She is an amazing researcher who never runs out of exiting ideas and enthusiasm, which kept me positive during tough times in the Ph.D pursuit. It has been an honor to have the chance to work with her.

I would like to thank Shumin Zhai and Andy Cockburn for being my thesis reviewers, as well as Eric Lecolinet and Géry Casiez who served as parts of my defense committee. Shumin Zhai has given me valuable feedbacks on my work since the beginning of my Ph.D period, introduced me to Xiaojun Bi who was a great collaborator in one of my paper, and finally given me a very encouraging thesis review. I learned a lot from the detailed review and suggestions written by Andy Cockburn. All in all, thank you for taking your valuable time to read and examine my thesis in details, which inspired me to pursue my work in the future. I very much hope we can have the chance to work together in the future.

I could not be happier being a part of the ex)itu team in INRIA. All the members are exceptional researchers and students, from whom I learned a lot. I would like to thank Michel Beaudouin-Lafon who is always helpful especially during the paper submission process; and Joseph Malloch, a postdoctoral fellow in ex)itu, for being a great project collaborator.

Along my Ph.D journey, I have met valuable friends, with whom I shared so much joys, happiness, silly moments, frustrations, and hugs. Can Liu and Monireh Sanaei have been like sisters to me, with whom I can share anything – not only my research fiasco but also my life stories. Simon Perrault has been always been a great friend, always helpful in answering my questions from research to career. I delight in conversing with Daniel Strazzulla, for they always make me think about what I want in my life. Carla Griggio is not only an amazing supportive collaborator, but also a sweet friend who listened to my complains or noticed

how stress I was and gave me random hugs. I am really glad that Nolwenn Maudet, Ignacio Avellino, and I defended at the same period – thank you for the encouragement all along the thesis writing process. I had silly and fun moments sharing an office with German Leiva. As well, Michael Wessely, Arnaud Prouzeau, Stacy Yuan, Midas Nouwens, Abby Liu, Mai Ciolfi, Oleksandr Zinenko, Thibaut Jacob, Chengcheng Qu, and other students I met during my Ph.D – my life would be so grey without you guys. Our paths may separate now but I will always hope they intersect again someday, sometime.

I am grateful to have Indonesian friends in Paris, with whom I can feel like I have families even when I am so far away from home. The meals I had with Felicia Yusuf, Elita Hardjadinata, Stefani Angela, Tante Marietta, Tante Andriani, and Romo Bala tasted way better. Thank you for always praying for me. I really appreciate Johanes Narasetu and Tante Marietta for helping me with the *pot de thèse*, and for the amazing tiramisu. We did not see it ‘til the end, Martinus Putra Widjaja, but thank you for supporting me during my Ph.D period.

Finally, to my mother, father, and brother, who never give up on me, support me endlessly, send me all the love – you are the ones who kept me going through the hardest part of my Ph.D period.

This thesis was supported by ERC grant number: 321135, CREATIV: Creating Co-Adaptive Human-Computer Partnerships.

1

Introduction

The mass adoption of smartphones has transformed the everyday experience of users, as well as the design of mobile interactions. Thanks to the interactive high-resolution touchscreen, today's mobile interfaces are completely touch-based. The simplest form of touch-based interaction is tapping on menus or icons, that borrows the point-and-click metaphor in graphical user interface (GUI). The users can also swipe, pinch, or other complex gestures that involve several strokes or multi touch.

A gesture-based interaction is in fact a communication channel between the users and the system. It starts with the user transmitting a message by drawing a gesture on the touchscreen device. Hence, the gesture encodes the information that the user wants to communicate. The device then decodes the gesture into the intended message that can express different concepts, such as commands or text. For example, a user is looking at a collection of photos in her phone, and she wants to see the next photo. She performs a flick gesture that the gesture recognizer recognizes as a 'flick' to the left. The system interprets it as a command to display the next photo in the list.

One of the grand challenges of research in Human-Computer Interaction (HCI) is optimizing the bandwidth of the communication channel between users and the system. The capacity of the communication channel in gesture-based interactions, i.e. its *expressive power*, depends on *the number of different gestures* and how they can *express varying concepts* [4]. In the field of knowledge representation, expressive power (expressive adequacy) is related to

what a representation can say [77]: “*Two components of expressive adequacy are the distinctions a representation can make and the distinctions it can leave unspecified*”. In other words, expressive power can be measured through its power of distinction – by the situations it considers distinguishable. In programming languages, expressive power refers to a measure of the range of ideas expressible in a given programming language [23]. Increasing the capacity of the channel means more recognizable gestures to express more functionalities.

Today’s smartphones offer a wide variety of functions, from communication, games, information consumption, office support, even art and creativity support [12]. As a consequence, the command space becomes larger as well. To access a larger command space, the users are often forced to go through multiple steps e.g. searching the item in a menu, making the interaction cumbersome and inefficient. For example, users have to tap on a button to display a menu, then scroll through the menu to find the command they would like to invoke. Alternatively, users can perform a stroke gesture to invoke a command, i.e. gesture shortcuts. The challenge in using gesture shortcuts is *recognizability* and *learnability*. A gesture should be unique so that it is easy for the system to recognize the gesture. Nevertheless, the more gestures in the gesture set, the harder it is for each gesture to be both unique and simple. More complex gestures (e.g. different shapes, multi-touch, etc.) offer a powerful way to access information quickly as well as to invoke a command more efficiently [5]. However, they are more difficult to learn and control.

I am interested in rethinking the access to a variety of functions, by building mobile interactions that are both easy to learn and control yet more powerful. Given that gesture-based interactions have been supported and widely adopted in mobile devices, we can assume that there is a set of gestures that the users can already control. Hence, instead of adding more gestures in the gesture set, can we interpret the same gesture differently, for example depending on where (a part of) the gesture is drawn or how it is drawn? If so, users will potentially be able to interact as they usually do with existing systems, while obtaining easy access to powerful functionalities.

I am specifically interested in mobile text-input techniques, since text entry comprises about 40% of mobile activity [12]. Aside from communication, users also type search keywords, take notes, and compose documents with their mobile phones. While typing, users also often invoke commands, for example to change the text formatting. Switching from typing to command invocation back and forth is cumbersome and may interrupt the cognitive process [67, 61, 43]. Thus, in this thesis, I focus on augmenting the access to functionalities through soft keyboards.

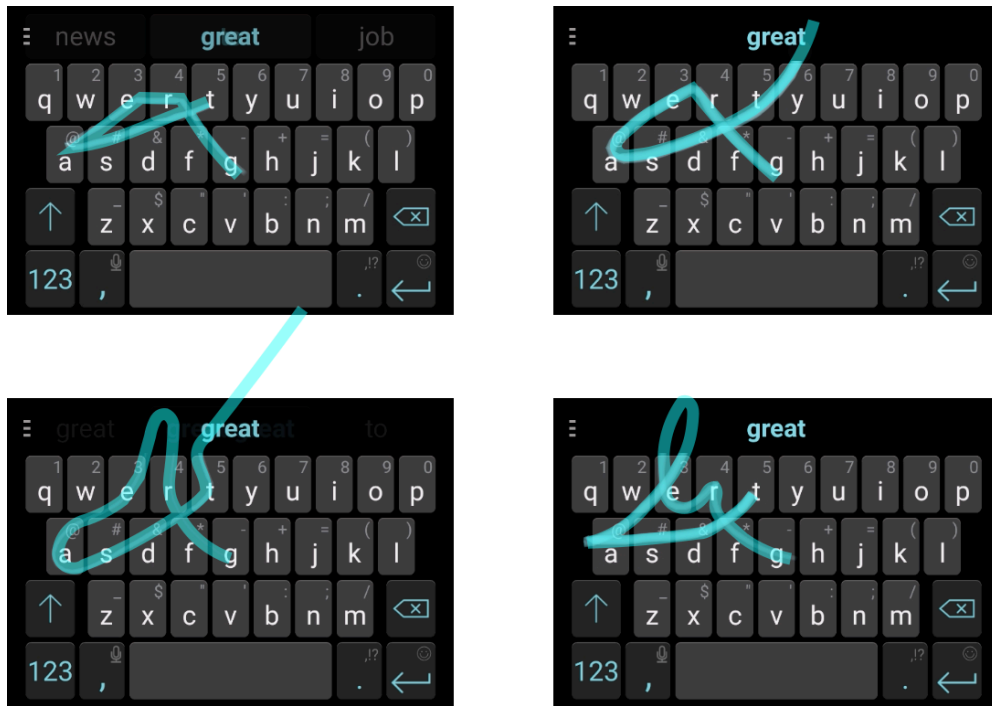


Figure 1.1: Gesture-typing [42, 82] is an efficient, easy-to-learn, and error tolerant technique for entering text on software keyboards. All four gestures in this figure are correctly recognized as “great”.

On most mobile devices, users tap on buttons on “soft” keyboards displayed on high-resolution 2D touchscreens to enter text. *Gesture typing* [42, 82] is a more interesting alternative, for it stands in between point-based and gesture-based paradigms: it lets users to draw a gesture on soft keyboards to enter text, from the first until the final letter in a word. Since the gesture is drawn on top of a soft keyboard, each key acts as a “steering target” users must go through. The recognition engine compares each word gesture to a pre-designed “template” representing the ideal word-gesture shape. Word-gestures are not unique for each word, but can be robustly matched using a combination of kinematic models, multidimensional distance metrics, and language models to resolve ambiguities. Gestures that vary significantly may thus still register as correct, as illustrated in Figure 1.1.

As with other soft keyboards, the goal of gesture-typing keyboards is to produce the single, “correct” typed word intended by the user; it is either correct or incorrect, and input variation is of interest only for the purpose of designing tolerant recognition systems. Gesture variation is treated essentially as a deformation of the correct shape and discarded as unwanted noise. To be sure, a small part of the variation is motor-system or digitizer noise, and can-

not be considered meaningful. However human experience with handwriting clearly shows the potential for personal and stylistically-communicative variation of output media through performed human gestures. How can we take advantage of such systems that already allow gesture variability while maintaining an extremely good recognition accuracy? Can users deliberately manipulate the shape and dynamics of their gesture *as* they write? Can we extend the keyboard's functionality to invoke commands *seamlessly as* they write?

1.1 THESIS STATEMENT

I believe we can increase the expressive power of gesture-based interaction by leveraging the screen space and gesture variation, to enrich communication, invoke commands, and facilitate appropriation in different context of use. Users naturally vary their gesture input due to biomechanics, personality, current activity, and environments. The existing gesture variations potentially carry additional information about the writers, the context, and the mood. Users can potentially control aspects of their gestures deliberately, for example to communicate intentions. Most advanced machine learning-based recognition algorithms often anticipate the variations, but remove them, since the goal is to produce a single “correct” text output. Capturing continuous features of the variation and mapping it to properties of the rendered text could re-enable some of the benefits of handwriting, such as recognizable personal styles, implicit communication of mood, activity, or context; and explicit communication of emphasis, sarcasm, humor, and excitement. Chunking the gesture into segments could enable a gesture be interpreted as commands and text at the same time.

1.2 RESEARCH APPROACH

After reviewing the related research literature, I decided to start with gesture typing as the means-to-an-end. Gesture keyboards have been widely adopted by the general public, with over 150 million copies installed. As shown in Figure 1.2, my work includes the following general approach:

- I used the *structured observation* technique [27] to explore how smartphone users gesture type, to gain insights about human variability that users make and the device can capture. A structured observation combines elements of controlled experiments to allow comparisons and a realistic task to enhance external validity. In contrast to a controlled experiment, the goal is not to assess a clearly defined hypothesis with the

data collection, but to create comparable conditions for observing common patterns and processes. This helped me to identify three gesture features that users significantly vary as they gesture type in different condition.

- I designed novel interaction techniques with the main goal of increasing the expressive power while assisting the learning process and facilitating appropriation. I used *co-adaptation* [51] and *substrate* [26, 41, 56] as the generative design principals. I will discuss both principals further in Chapter 2. I analyzed the structure of a soft keyboard and how the data is captured and interpreted based on the structure. My approach focused on two concepts: increasing the granularity of the data interpretation based on the same structure, and linking different structures. The proposed interaction techniques support discoverability, learnability, and appropriability.
- I conducted three controlled experiments to evaluate the efficacy of the proposed techniques as well as to gain deeper understanding on how users learn to control and appropriate the proposed techniques.

1.3 CONTRIBUTIONS

Through structured observations, I increase our understanding of the variability in users' gesture-typing input, that is significantly different across and within users. I demonstrate how we can quantify and then transform the *otherwise unused* gesture variations into rich output, to increase the expressive power of gesture-based interactions. I propose two interaction techniques:

1. EXPRESSIVE KEYBOARD, a gesture keyboard that lets users generate rich, expressive output by manipulating the variations in their gestures;
2. COMMANDBOARD, that lets users invoke a complex command e.g. hierarchical commands as they gesture type from soft keyboards.

I show that users can control their gesture and appropriate the system just by using it, while retaining the typing accuracy. I also show that users can draw gesture segments as a whole to

seamlessly switch between command invocation and gesture typing when using `COMMANDBOARD`. Selecting commands with `COMMANDBOARD` is almost twice as faster than mark-down language, and preferable by all of our participants. Finally, I discuss the concepts in terms of reciprocal co-adaptation and substrates to broaden the design space to other environments.

1.3.1 COLLABORATORS

I worked closely with my supervisor, Wendy Mackay, especially in all the design process (i.e. interaction techniques and evaluations), data analysis, as well as writing and publishing the papers. Additionally, I collaborated with Joseph Malloch, a post-doctoral fellow in ExSitu, who was actively involved in designing the `EXPRESSIVE KEYBOARD`'s concept and evaluations, as well as writing the published paper [2]. He also contributed to the early brainstorming of `COMMANDBOARD`'s concept.

I also collaborated with Carla Griggio, a Ph.D candidate under the supervision of Wendy Mackay in ExSitu, and Xiao Jun Bi, an Assistant Professor in Stony Brook University, USA. Carla Griggio was involved in the later design iteration process of `COMMANDBOARD` and reimplemented a dynamic guide technique, OctoPocus [8], that we used in `COMMANDBOARD`. She was also actively involved in designing and running the `COMMANDBOARD`'s evaluation, as well as analyzing and reporting the qualitative data. Xiao Jun Bi provided us with a gesture keyboard prototype that implements the gesture keyboard algorithm, and contributed in writing the Related Work section of `COMMANDBOARD`'s published paper [1].

1.4 THESIS OVERVIEW

The chapters are outlined as follows (see Figure 1.2).

Chapter 2 first summarizes related work on gesture-based interaction, emphasizing the advantages and th challenges in mobile environment. I then discuss existing research that focuses on increasing the expressive power of gesture-based interactions and the properties of gestures that we can take advantage of.

Chapter 3 investigates gesture characteristics that users vary when they gesture-type. I present a structured observation that focuses on intentional gesture variations, that a user deliberately produces to reach a certain goal, such as being fast. Based on the results, I define

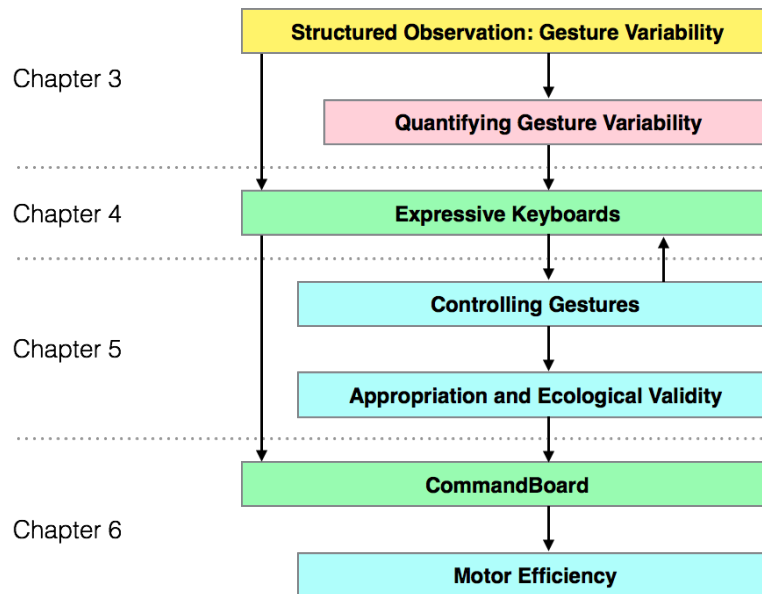


Figure 1.2: Overview of the thesis and the used design methods. It includes controlled experiments and informal observations. The understanding and theory from the studies served as the fundamentals to design and redesign new interaction techniques.

a set of gesture features to quantify the variations that the users manipulate differently across users or context.

Chapter 4 introduces EXPRESSIVE KEYBOARD, an approach that transforms the rich variation in gesture-typed input into expressive output. It enables users to express themselves through personal style and through intentional control. I present several output types and properties that users can manipulate with their gestures, such as text color, parametric font, dynamic emoticons, and animated content.

Chapter 5 focuses on the learnability and appropriability aspect of EXPRESSIVE KEYBOARD. I present two controlled experiments. The first one evaluates how users deliberately control their gesture variation to generate rich output with EXPRESSIVE KEYBOARD during the initial learning phase i.e. novice behavior. I also did an informal observational study to explore the effect of progressive feedback in user's strategy to learn how to use EXPRESSIVE KEYBOARD. The second one presents a controlled experiment that focuses more on the appropriation aspect of EXPRESSIVE KEYBOARD in a more ecologically-valid setting. The results show that when provided the tools, users quickly appropriate the system even when they are still learning how to use the system.

Chapter 6 introduces COMMANDBOARD, which lets users issue commands *as they type* through a gesture keyboard. COMMANDBOARD transforms the area above the keyboard into a command-gesture input space where users can draw gestures to select a command. I present two technique variations, TYPE-AND-EXECUTE and INLINE GESTURE SHORTCUTS along with demo applications.

Chapter 7 first presents an overall discussion focusing on how the work increases the expressive power of gesture-based interactions from different angles. I summarize the contributions in terms of scientific understanding and design, and gives directions for future work.

2

Related Work

This chapter begins with summarizing existing works for understanding the properties and the benefits of different types of gesture-based interaction. I review past research that focuses on increasing the expressive power of gesture-based interactions on mobile devices. This chapter also identifies missing points in the literature and positions my work in it.

2.1 GESTURES ON TOUCHSCREEN DEVICES

On touchscreen devices, a gesture is a 2-dimensional movement trajectory of a user's finger or stylus contact points with a touch sensitive surface [78]. Each sample point contains the information about the touch coordinate relative to the display dimension and a timestamp. Furthermore, a gesture can have pressure, making it a 3D gesture.

The expressive power of gestures is related to the gestures' design dimension – including more design dimension potentially enables us to generate more distinct gestures. The most common design dimensions are the gesture length and the number of contact points on the screen. A gesture without a movement trace, i.e. a tap, is often called zero-order gesture. Users can perform more than taps sequentially (e.g. double tap) or with multi fingers at the same time (e.g. multi touch). A gesture with single stroke, i.e. unistroke gesture, is called first-order gesture. Users can also draw two or more strokes sequentially or use different fingers at the same time to perform higher-order gestures. Another design dimension is visual-spatial


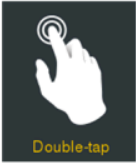
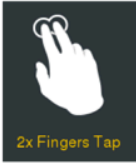
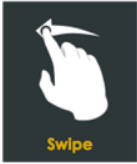
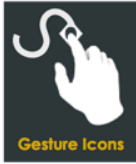

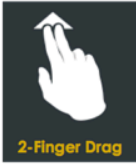
	Single Touch	Multi Touch
stroke length = 0	zero-order  Single-tap	zero-order multi-touch   Double-tap 2x Fingers Tap
stroke length > 0	first-order (unistroke)   Swipe Gesture Icons	higher-order   Multi strokes 2-Finger Drag

Figure 2.1: Different type of gestures based on the stroke length and the number of contact points.

dependency, in which the same gesture can be interpreted depending on where on the screen the gesture is drawn.

In the next two sections, I will describe both advantages and disadvantages of zero-order gestures and unistroke gestures, as well as previous research on how to increase the expressive power in different context.

2.2 ZERO-ORDER GESTURES

Like point-based interactions, a zero-order gesture (referred as a tap in the rest of the thesis) often highly depends on the graphical representation. In fact, a single tap is only powerful because it can be interpreted differently depending on which graphical object it points. Users point to different objects e.g. *menus*, *toolbars*, and *icons* to enter, retrieve, and select commands or text. Since the graphical objects are presented on the screen, non-expert users can quickly learn which commands are available. Small menus and toolbars allow users to quickly access common items, but do not help with large command sets, which may require extensive search and multiple physical operations to find the desired item [61]. Accessing a multi-level hierarchical menu forces the user to move through a multi-step process of selecting the appropriate category, and before finding the desired leaf.

The expressive power of tap gestures is related to how many objects can be included in the screen and how easy it is to point to them. However, no matter how good the optimization

strategy for screen real-estate use is, the display screen is limited [4]. The problem becomes worse in mobile environments where the screen size is very limited.

Having said that, in touchscreen devices, a tap input is highly oversampled in both space and time, from which we can derive various properties, such as duration (i.e. long tap), pressure, orientation, offset, etc. As such, we have the opportunity to explore more continuous forms of control to express different concepts with a tap input.

2.2.1 INCREASING RECOGNIZABILITY

In their study, Lee and Zhai [46] found that users can still tap on buttons even though their size is smaller than the average finger width. Nevertheless, as in any pointing task, tapping a small object requires more effort and precision. Bi et al. proposed FFitts' Law [11], that extends Fitts' Law [24] for predicting a finger touch input. They argued that pointing on touchscreen is less precise than in a desktop computer with a pointing device, and variability in a tap input exists even though the participants were asked not to worry about the speed [36]. Bi et al. took into account this variability in predicting the index of difficulty of finger touch, which can be affected by the speed-and-accuracy trade off and “the absolute precision” of human motor control. The latter may vary due to “fat finger” problems or motor impairment.

Most research in this topic mainly focuses on improving the touch accuracy, for example by taking into account the offset between the target and the touch input [35]. Azenkot & Zhai [7] investigated how users type on soft keyboards and found that touch offsets vary according to how they hold the device. Dynamic key-target resizing based on models of likely words or letter sequences increases the apparent accuracy of soft keyboards and partially resolves the fat-finger problem [34].

2.2.2 AUGMENTING ZERO-ORDER GESTURES

Another interesting direction on using the touch variability is to increase the expressivity aspect, for example to support subtle expression in text-based communication. Researchers also explored this topic in the context of physical keyboards. For example, Iwasaki et al. [38] added sensors to a physical keyboard to capture typing pressure and speed. KeyStrokes [59] uses shapes and colors to visualize typing style and text content. With soft keyboards, Buschek et al. [14] combined touch offset, key-hold time, and device orientation to dynamically personalize the font.

In September 2015, Apple launched iPhone 6s that includes built-in pressure sensor that enables capturing 3D touch^{*}, i.e. from light to normal to deep press. The pressure can be used to select different commands, e.g. a long normal-press on an app icon will allow users to delete or move the app, a deep-press to see all quick access shortcuts of the app; or to parameterize a command, e.g. vary line thickness [68].

Nevertheless, since the input is very simple, there is not much opportunity to increase its expressive power, relative to unistroke gestures.

2.3 UNISTROKE GESTURES

A unistroke gesture involves the spatial description of the trajectory (i.e. *shape*) and the time evolution of the motion (i.e. *dynamics*) [75], that lets users express varying things and concepts. A unistroke gesture can be drawn anywhere on the screen, thus it does not take the valuable screen space [5]. From unistroke gestures alone, we can already have a sufficiently large gesture set. In contrast to performing a sequence of discrete actions i.e. tapping on different objects, drawing a gesture is a kind of perceptual-motor skill involving continuous response, with little forgetting over long periods of time [76]. Users can also remember spatial patterns better than sequential patterns [21]. Appert & Zhai [5] suggested that using gestures to invoke command (gesture shortcuts) is more effective than using keyboard shortcuts. They found that gesture shortcuts are easier to learn and recall thanks to their spatial and iconic properties. Performing gesture-based interactions is related to human memory, which is categorized into *declarative memory* and *procedural memory* [73]. Declarative (cognitive) memory involves conscious, explicit thinking, such as perceiving and managing information and making sense. Procedural memory is related to behaviour or habits, thus it is done unconsciously and implicit. An expert in gesture-based interactions can potentially draw a gesture shortcut without much visual attention, in which case involves procedural memory.

Researchers have explored leveraging continuous unistroke gestures to select commands. A classic example is a Marking Menu [44], a contextual circular menu that supports executing commands via directional strokes. A Marking Menu can be extended in order to give access to a larger command space, for example by making it hierarchical [83] or combining it with drawn letters [66]. FlowMenu [33] extends a hierarchical marking menu to include parameter adjustment of an item. For example, a user can select a zoom command and specify

^{*}<https://developer.apple.com/ios/3d-touch/>

the zoom value in the sub-menu, or even type the value (when the desired number does not exist) without lifting the pen. Li [48] studied a world-deployment of a gesture-search system for smartphones, and demonstrated that gestures successfully supported users accessing data for their day-to-day mobile activities.

2.3.1 CHALLENGES

DISCOVERABILITY

With tap-based interaction, all the existing command items are usually included in the menu, thus novice users can easily discover all commands in the command space. In contrast, it is difficult to discover all available gesture shortcuts. Since the gesture shortcut can be drawn anywhere on the screen, distinguishing a gesture shortcut from a direct manipulation gesture is also challenging [48].

LEARNABILITY & RECOGNIZABILITY

Tapping on graphical objects e.g. menu only requires the system to *recognize* which object users point. Users can invoke a command by tapping on a menu item or an icon after finding where it is located in the menu. On the other hand, gesture-based interfaces require the users to *recall* the shape of the gesture shortcut and the system to correctly *recognize* the gesture and its corresponding command. Thus, the gesture should be unique so that it is easy for the system to recognize the gesture. However, the more gestures in the gesture set, the harder it is for each gesture to be both unique and simple. More complex gestures (e.g. different shapes, multi-touch, etc.) offer a powerful way to access information quickly as well as to invoke a command more efficiently [5]. The more gesture shortcuts are available, the more difficult it is for the system to recognize the gesture input and for users to remember [79].

SPEED-ACCURACY TRADE OFF

Performing a gesture shortcut is relatively more complex process that involves planning, memory, and motor control [79]. Users need to remember the shape and then execute the “precise” gesture, thus they need to practice in order to perform the gesture quickly and perhaps eyes free.

2.3.2 DESIGNING GESTURE-BASED INTERACTIONS

Ideally, gesture-based interactions should support discoverability, memorability, and learnability in order to offer more efficient alternative. Many attempts have been conducted to solve these problems from different perspective.

SUPPORTING TRANSITION FROM NOVICES TO EXPERTS

To support transition from novices to experts, a gesture-based interaction should be self-revealing. With Marking Menu [44], novices who are not familiar with the menu yet can pause, then the menu appears and they can discover all possible commands and just follow the direction to invoke the command. Experts can just draw to a certain direction without having to pause, in which case the menu is not displayed on the screen. OctoPocus [8] offers better support for learning gesture shortcuts, acting as a dynamic guide to help users follow the correct gesture template: If the user hesitates, OctoPocus appears, showing the remaining possible ways to finish the gesture. This highlights the need for progressive feedforward and feedback to support incremental learning, to help novices transition to expert users. Andersson and Zhai found that providing visual feedback affects how users perform the gesture: they draw a gesture bigger and faster yet take more time in order to complete drawing [3].

EASY-TO-REMEMBER GESTURES

Gestures are easier to remember if they are analogous to the physical effect or to well-known conventions. For example, a scrolling gesture mimics how we scroll a physical paper, or drawing a cross gesture to delete an item. However, analogous gestures can be complicated or ambiguous since they depend on users' prior experience. Alternatively, we can let users to define their own gestures. In their study, Nacenta et al. found that self-defined gestures are easier to remember [57].

To summarize, designing efficient and powerful gesture-based interaction on mobile devices has several issues. First, mobile devices have a limited screen space. Second, the larger the gesture set is, the more difficult it is to accurately recognise each gesture. If we make the shape of the gesture complex enough so that they are more likely to be recognised, then it is more difficult for users to learn and memorise the gesture. And last, there is also the issue of

efficiency, i.e. how we support the transition from novices to experts, and how the gestures can be used to express different concepts in different context and goals.

2.4 GESTURE KEYBOARDS

Gestures can also be used to enter text, for example by recognizing gestures as handwriting. Using handwriting as a text input means on mobile device is easy to learn, however it is considerably slow. Furthermore, handwriting recognition is not an easy task, because there are too many variability in the input. A letter may also consists of several strokes, making it more challenging to recognize. A single-stroke gesture design such as Graffiti or Unistroke [31] can simplify the recognition complexity, however they often require novice users to learn new symbols.

Gesture keyboards [80, 42, 82] offer an efficient, easy-to-learn, and error-tolerant alternative for entering text. Instead of tapping keys, users draw the shape of each word, i.e. a *word gesture*, beginning with the first letter and continuing through the remaining letters. The word-gesture recognition requires a multi-channel recognition engine [42], where the drawn shape is first compared to an “ideal” shape, i.e. from middle-point to middle-point of each key. The ideal shape of a word gesture is called “gesture template”.

2.4.1 GESTURE RECOGNITION PROCESS

Capturing the input The gesture keyboard captures all touch data from the first until the last letter. Users can see a visualization of the trajectory of their gesture as a feedback. The gesture trace feedback has a limited length, to avoid visual overload and occlusion. The touch data i.e. user’s word gesture is then sent to the recognition engine progressively.

Recognizing the word The input is compared to all gesture templates in the list of word candidates. At this point, we should assume that the user can write any word, and thus, all words in the language corpus are candidates. The recognition engine uses a multi-channel architecture to enable efficient, fast, and error-tolerant text input. The process of recognizing user’s input goes as follows (also see Figure 2.2).

1. Template pruning

The biggest challenge in recognizing user’s word-gesture input is that the gesture templates are not unique, i.e. there can be two or more words that have identical gesture

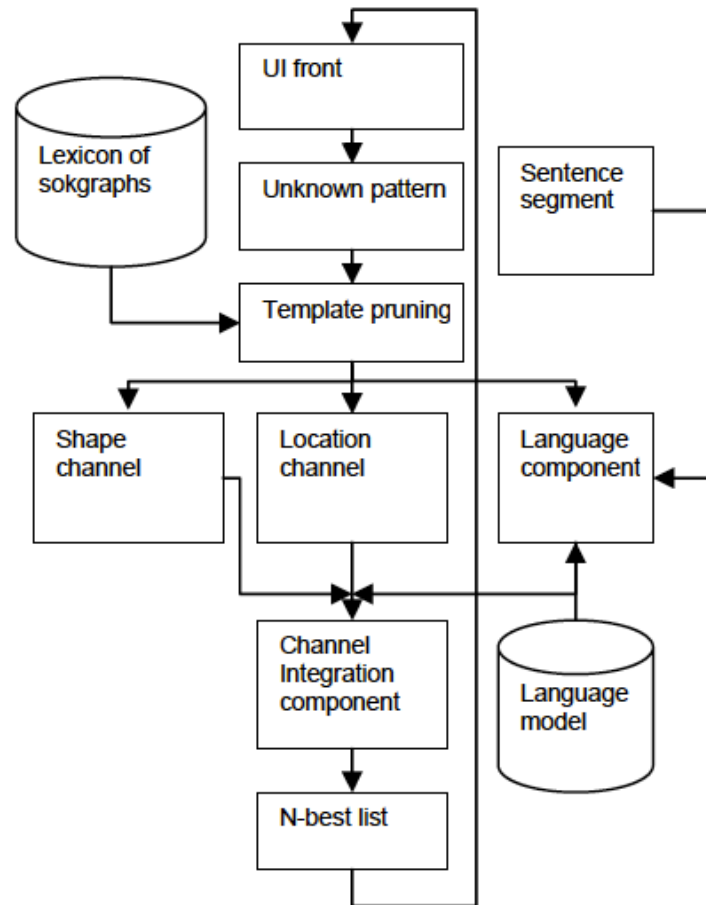


Figure 2.2: The multi-channel recognition engine of gesture keyboards, taken from [42].

templates. For example, “all” and “weep” have identical gesture templates. Thus, the gesture templates are first *filtered* based on the start (e.g. letter ‘a’) and the end (e.g. letter ‘l’) position of the gestures. If the start-to-start and/or end-to-end distances between a gesture template and the input is bigger than the predefined threshold, then the gesture template is removed from the list of word candidates.

2. Shape recognition channel

The input is then compared to the remaining gesture templates in the list one by one, by measuring the spatial similarity of the two. User’s input is first being normalized so that it has the same bounding box as the gesture template. The recognition engine then samples n -number of points in both the input and the gesture template – each

point in the input is proportional to a point on the gesture template. Next, the recognition engine measures the spatial similarity of the two, by calculating the point-to-point distances in sequence. As of now, each gesture template has a score of how similar it is to the user's input, which is the average point-to-point distance.

3. Location recognition channel

The recognition engine also considers the position of the gesture input relative to the keyboard layout (geometric center), in order to reduce the number of word candidates in the list. As of now, only completely identical gesture templates are left in the list, e.g. "lose" and "loose". A gesture template that starts and/or ends at almost the exact same location with the input has a higher score to be the actual final word output. The recognition engine also considers temporal features: If the user slows down at a letter, the recognizer weights the word candidate higher.

4. Language component

The recognition engine uses the context, language model, and a learning corpus to rearrange the word candidates in the list. Finally, the recognition engine produces an N-best list of word candidates, sort by relevancy to the contexts.

Displaying the final word output The recognition process is conducted progressively as the user moves her finger: at each touch, the gesture keyboard generates the N-best list of word candidates. The first is treated as the final result, the next three are displayed in the suggestion bar above the keyboard. The gesture keyboard may also auto-complete the current word-gesture, even before the user reaches the last letter of the intended word.

If a word-gesture is drawn outside the keyboard space, the gesture keyboard captures the touch event but stops recognizing the word.

2.4.2 LEARNABILITY ASPECTS

Since gesture keyboard is basically also a standard keyboard, users can alternate between typing and gesture typing without switching mode. Thus, it helps the new users to transition from tapping to gesturing.

Gesture keyboards also support a gradual transition from novices to experts in terms of drawing the word gesture. The transition takes advantage from the fact that the gesture of a particular word is always consistent, similar idea with Marking Menu [44], allowing users to

Beginner	Expert
During the initial learning phase	During the extended learning
Letter-to-letter tracing (a series of pointing/crossing actions)	Memory-driven gesturing
Visually guided (gestures tend to be more precise)	Recall driven (more sloppy gestures)
Closed-loop	Open-loop
Slow	Fast
Involve declarative memory (explicit)	Involve procedural memory (implicit, unconscious)
Involve voluntary actions (with a conscious goal)	Involve perceptual motor integration (outside of awareness)
	Long-lasting (due to procedural memory and perceptual motor integration)

Figure 2.3: The difference between a novice and an expert when using gesture keyboards.

build and memorize their own mental word shape representation. Expertise is reached when users transform from using declarative memory and voluntary actions, i.e. tracing from one key to another, to using procedural memory and perceptual motor integration, i.e. draw a shape (see Figure 2.3). The fundamental difference lies on the degree of visual guidance reliance. From the psychology perspective, the key to developing expertise, i.e. skilled, low attention, automatic behavior, lies in consistent mapping from stimuli to response.

2.5 AUGMENTING SOFT KEYBOARDS

In response to the high level of text entry on mobile phones, phone manufacturers are developing keyboard extensions that offer new capabilities, from suggesting emoticons to general search. The latest version of Google Keyboard (now called Gboard) [†] includes an in-context search engine. Users tap a button on the top-left of the keyboard to access the search engine, where they can directly type the search keyword, see the results, and share it.

[†]<https://play.google.com/store/apps/details?id=com.google.android.inputmethod.latin&hl=en>

2.5.1 GENERATING EXPRESSIVE OUTPUT

Researchers who study movement for music or dance often take a completely different perspective in interpreting gestures. They often emphasize the continuous qualities of human gestures over recognition: individual variation is valued rather than ignored or rejected. These researchers characterize gesture variation in terms of qualities of movement: spatial features [14, 17]); temporal features; continuity; power; pressure; activation; and repetitions [19]. In the artistic domain, the richness of gesture can be transformed into continuous output, e.g., [30], or to invoke a command [43] in a more integrated interaction.

Appert et al. designed several novel gesture vocabularies that rely on additional input channels such as tilt [71], pressure [70] and proximity [69]. These vocabularies of gestures remain simple to perform while offering at least as much expressivity as the whole set of existing graphical widgets, and without consuming any screen space. To increase the expressivity of text-based communication, Buschek et al [14] proposed TapScript that leverages the variation in typing input (i.e. tap) to increase the expressive power of mobile communication. Nevertheless, TapScript is based on typing discrete virtual buttons. Another alternative is to recognize handwriting, a continuous activity that not only communicates content but also provides additional information about the writer, e.g., rough letters from a writer in a hurry; and the context, e.g., shaky letters from writing on a moving bus. However, handwriting on a mobile touch screen device, particularly with a finger, is slow and tedious.

2.5.2 INVOKING COMMANDS FROM KEYBOARDS

Switching between typing and issuing commands comes with interruption costs, as in any multitasking environment [67]. As more and more mobile activities involve text entry, researchers have explored augmenting a keyboard with gestural commands, to reduce switching costs between the text entry and selecting commands. For example, Fuccella et al. [25] propose a technique in which using a two-finger touch gesture directly on top of a soft keyboard lets the user move the caret to select text. Command Strokes [43] employ additional buttons, e.g. COMMAND to enable keyboard shortcuts on gesture keyboards. Users can simulate using control keys on a physical keyboard, e.g. drawing a gesture that passes through COMMAND then C to perform COMMAND+C. However, users need to remember the shortcut in order to execute the command.

2.6 POSITION OF MY WORK

Users can also draw higher-order gestures, either sequentially or in parallel (multi-touch), to gain more physical and cognitive advantage [78]. For example, TapBoard 2 [40] enables pointing through a soft keyboard, adding support to bimanual interaction. Arpege [28] provides multi-finger chord interaction, with dynamic guides to show novice users where to place their fingers. In this thesis, I limit my investigation to unistroke gestures, since a unistroke gesture is often a component of multi-stroke or multi-touch gestures. In the rest of the thesis, the term “gesture” refers to a unistroke gesture.

In today’s mobile devices, gesture-based interaction is mainly used to invoke a command (i.e. gesture shortcut) and to enter text (i.e. gesture typing). I see the gesture keyboard as an interesting system to start my investigation with due to several reasons. First, gesture keyboards offer a faster and easy-to-learn alternative to text-input technique. Gestures that vary significantly may thus still register as correct, thanks to the algorithm’s kinematic models, multidimensional distance metrics, and language models. Second, more 40% of mobile activities involves typing, e.g. text messaging, text editor, browsing, etc [12]. Often the case, users must go back and forth between typing and command selection. Third, millions of people already use gesture keyboards in different mobile platforms. Augmenting gesture keyboards will potentially minimize users’ learning effort and the cost to switch view from keyboards to menu, *while preserving typing accuracy and speed*. Thus, in my thesis, I focus on increasing the expressive power of gesture-based interactions in text-based communications and command selections by augmenting gesture keyboards.

In this thesis, I propose different approaches to interpret gestures differently. The underlying idea of my approach is to interpret each gesture in the gesture set to express different concepts, by *leveraging gesture variations* and *leveraging the screen spaces* in order to increase the information transfer.

My first approach is based on the fact that people make movement variations. Due to their extremely fine motor control, people can perform various movements with different level of precision, especially after a certain amount of practice. For example, a professional violinist can control her fingers more precisely when playing a violin, generating a richer sound, than a beginner who is still learning. Nevertheless, it does not take a professional violinist to produce a rich sound. It is very difficult, if not impossible, to perform exactly the same movement twice. People also perform differently from one another. A violinist may play the same piece in a very different way depending on what she wants to express, or to which

audience she plays for. Another violinist, even with a similar level of skill, will also perform the same piece differently.

Movement variability also occurs in 2D environments, for example when writing on a piece of paper. It is almost impossible to draw identical letter ‘a’s twice [14]. We also vary our handwriting depending on the context, such as the intention, the condition, or the mood. We can easily identify our own handwriting, and our close friends or relatives are usually able to distinguish our handwriting from others’. In other words, the trajectory of a movement, such handwriting, potentially carries rich information about the author, the context, etc. Thus, we can leverage the additional information that naturally already exists to enrich communication, aside from the content itself. For example, in the past, the style of handwriting was used to determine one’s place in society [29]. Handwriting identification is also a universally engaged practice in a lot of different fields. Thus, for the first approach, I investigate whether movement variations also exist in gesture-typed input and whether we can quantify them, to see if we can interpret “the same word gesture” differently.

My second approach is based on visual-spatial dependency properties of the drawn gestures on the mobile screen space. In mobile interface, objects are represented virtually: an object is a drawn shape placed on a specific location in a specific view. For example, the “q” key is a labeled momentary box (i.e. a soft button) drawn on the top-left position in a QWERTY keyboard view. The boundaries between objects and/or views are also virtual. Hence, we can draw a gesture on top of the soft keyboard (i.e. gesture typing) and even to the upper space of the keyboard view. My idea is to leverage the spatial location in order to segment the gesture and to interpret each gesture segment differently. This is related to the notion of *gesture chunking*: grouping task elements into a larger, inseparable whole [78]. A gesture may involve several segments, that over time may be viewed as inseparable. In other words, a gesture can potentially be chunked into several segments, and each segment can be interpreted differently.

To systematically investigate both approaches, I used two generative design principals that I describe in the following section.

2.7 GENERATIVE DESIGN PRINCIPALS

Gesture keyboards are merely a means to an end towards the goal of building easy-to-learn but powerful interaction techniques, nevertheless we can take advantage of its sophisticated recognition algorithm and properties. In this thesis, I use two approaches, *substrate* and *co-*

adaptation, as generative design principals to help me re-frame design problems to reveal insights on how human beings interact with technology in the world.

2.7.1 SUBSTRATE

Substrates are software artefacts that handle specialized data, which are interpreted depending on the structure [26]. “*Substrates embody content, computation and interaction, effectively blurring the distinction between documents and applications*” [41]. Substrates can evolve over time and shift roles, depending on the perspective or the goal of defining the substrate. For example, a dot means completely different if put on top of a blank paper or musical bar lines. Here, the musical bar lines are the structure in the substrate that provides rules on how to interpret the data. Specialized substrates can be linked together to support more complex data operations and communicate each data and state to each other.

Klokrose et al. proposed Webstrate [41], that turns a webpage into interactive, shareable substrate. Each webstrate is a collaborative object: it has the same persistent data yet it may provide different representations for each user. For example, two users can edit the same document, one uses a WYSIWIG editor while the other uses a LaTeX-like editor. Each user can create their own tool, for example to insert a citation, and share it with other collaborators.

In this thesis, I analyze a soft keyboard from the perspective of substrates, to better understand the structure, the data, and how to interpret it based on the structure. A soft keyboard’s layout is the structure of the substrate. A tap on the same location can insert a different letter depending on the layout, e.g. QWERTY, ATOMIK [81], or an emoji keyboard. A gesture can be interpreted as a scroll if done on a web view, but it is interpreted as a word if done on a gesture keyboard. To augment soft keyboards with expressive and powerful interactions, my approach is to add more rules to the same structure in order to enrich the interpretation, and to link together different substrates in order to support more complex functionality.

2.7.2 CO-ADAPTATION

As described in Section 2.3, an ideal gesture-based interaction should support discoverability and learnability. Additionally, I want to also consider *appropriability* when designing interaction techniques. Wendy Mackay introduced the concept of *co-adaptation* [51], in which users *adapt to* the system to learn to control the system, as they *adapt* the system i.e. customize and appropriate it in ways unintended by the system designers, to meet their immediate needs in the current context of use. Co-adaptation also serves as a generative (re)design

tool, with which the designers can systematically analyze and react to the design in order to produce more ideas and new design directions [51, 50]. Co-adaptive systems should support both *learning* and *appropriation*.

LEARNABILITY

In their survey on learnability, Grossman et al. [32] organized the process of learning into two categories: initial learning and extended learning. Initial learning is related to novice users' experience, i.e. how a novice user learns how to use a system from the beginning until she "reaches a reasonable level of usage proficiency" [60] or "a predefined level of proficiency" [39]. Carroll and Rosson [20] highlighted two main paradoxes that may arise in the initial learning phase: production bias and assimilation bias. Both are related to the user's approach when using and learning a new system. Production bias is related to how novice users tend to jump right in when introduced to a new system, in order to get things done. Production bias reduces user's motivation to invest time in learning, and learning-by-exploring can be costly. To increase users' motivation to learn, we can add *intrinsic* rewards in the system such as achievement, satisfaction of curiosity, etc.; or reduce the learning cost, such as adding 'undo' function.

Assimilation bias is related to the tendency to interpret new systems based on what they perceive and their past knowledge. While often useful, as with transfer learning, it may also mislead and hinder users from actually learning about the system. System designers need to be careful when designing the system, to reduce the potential misleading connections to existing systems, and only make similar functionalities when they are truly similar.

Extended learning is related to a quality of use overtime [9], in which experienced users are able to select a more efficient alternative, i.e. involves fewer steps [15]. Nielsen [60] argued that extended learning is related to expert users' experience, in which they have become efficient and their level of performance is already steady. To measure efficiency, we must define what it means to be an "expert", for example how much time it takes for the expert users to perform a predefined task. Nielsen also includes memorability in extended learning, in which an expert is able to recall from their previous learning experience. Thus, a good system should be easy to learn and remember, and supports transition from novices to experts.

APPROPRIABILITY

Appropriation is a phenomenon in which users reinterpret the systems based on what they know they can do to get what they need [22]. Appropriation emerges through different environment, needs, and even ownership, to the point that a piece of technology becomes a highly-personalized tool that others may not be able to use [22]. For instance, a user may use a digital camera as a mirror aside from taking pictures. People use upper-cased letters to express emphasis when texting. The use of spreadsheets has evolved from creating tables to programming, interactive forms, even collaborative tools in business setting [58]. In extreme cases of appropriation, users may reinterpret the artifact by changing its structure and functionality, such as when people “hack” the purpose of IKEA products’ pieces to create new items [65].

The attempt to appropriate can happen anytime along the use of the system, whether or not users have gained full control of the system [51]. For example, novice users may appropriate the system when they just begin learning, or something breaks, or even just when they are bored or on a whim. The more users gain control over the system, the more likely they are to appropriate the system. For example, they may notice their repeated usage pattern or when they think the system is already “too slow” to accommodate what they want to do, and thus are encouraged to customize it – if the system allows. Thus, anticipating appropriation when designing a system can increase the possibility of more powerful use.

2.8 TO LEVERAGE GESTURE'S SHAPE AND DYNAMICS

Lesaffre et al. [47] argued that musical embodiment, i.e. controlling body movement to produce rich sounds, is related to the coupling of *action* and *perception*. For example, when producing a specific sound on a drum, a drummer may first hit the drum with a stick (action) to hear the sound it produces (perception). He perceives the produced sound, which then may change how he hits the drum to produce a specific sound, such as hitting harder or to make it louder or hitting on the side (snare) to make it softer. On the other hand, music playing can also benefit from experience, i.e. a professional drummer may be able to produce the desired sound without relying too much on perception. Thus, when instructing someone to produce a specific sound, we can provide the final sound we want him to produce, or tell them how to hit the drum to produce the final sound.

In gesture-based interface, the gesture variation can be affected by movement cost [62],

interaction metaphors, and system behaviour. Tu et al. [72] investigate how gestures vary when drawn with a stylus versus a finger; and when sitting versus walking. They found that drawing time and accuracy dropped when walking, as compared to sitting. Another important design consideration is the mapping from gesture variations to rich output. For example, if the goal is to make the system ‘fun’ and challenging, the system should encourage curiosity [59]. Hunt et al. [37] found that continuous, multi-parametric mappings encourage people to interpret and explore gestures, although learning these mappings takes time.

I summarize and identify gesture properties that we can derive from a gesture’s shape and dynamics and how the input is captured in mobile devices, based on existing research.

2.8.1 SPATIAL FEATURES

Spatial features are related to a gesture’s shape, of how much space occupied [19].

Offset The offset measures the spatial similarity of two gestures, which reveals how accurate the users in pointing (when typing) or steering (when gesture-typing). In typing [14], the offset is how close the touch from the middle point of the key. In gesture-typing [42], the users’ gesture is compared to a gesture template, which is a straight line from middle-point to middle-point of the letter in a word. Both of the model and users’ gestures are divided into a certain amount of sampling points. The offset is calculated for each sampling point.

Curviness While the gesture template consists of straight-lines and corners, the actual users’ gesture may also consist of curves [17], for example when they cut corners. This is also related to the speed [43], users tend to make curvier gestures (i.e. smooth curve instead of sharp turning) when they do it fast.

Power / Pressure We can measure how much pressure the users put when gesturing [19]. Unfortunately, only some of the latest mobile devices include a real pressure sensor in their screens. We can measure the “fake pressure”, from the assumption that the touch area of the finger on the screen is bigger when the users press harder.

Gesture length In gesture-typing, it is interesting to compare the gesture length of the users’ gesture and the gesture template. A longer gesture may increase the accuracy of the

gesture recognition [43]. The gesture length can also be affected by overshooting or cutting corners [62].

2.8.2 TEMPORAL FEATURES

Temporal features are related to a gesture's dynamics, that describes a gesture performance over time [19].

Duration When typing, the time to enter an input i.e. a letter is the key hold time [14, 59]: the time spent from when the finger touches down until it is lifted. When gesture-typing, the input is a word-gesture and the duration is calculated from when users put their finger on the screen until they lift it.

Speed We can measure the typing speed of a text input by dividing the number of words written in a certain amount of time (duration). One of the most common way to measure typing speed is word per minute, which tells us how many 5-letter input chunks users can type in a minute. In gesture typing, aside from the typing speed, we can also measure the drawing speed, by dividing the total length of traced distances (in pixels) by the duration (in seconds). We can also measure its derivative, such as acceleration and jerk.

Gap time between outputs We can measure the time users spend to start writing a new word, after finishing the previous one [43]. It is calculated from the moment they lift their finger to write the previous word until they put their finger on the screen again to write the new one. Novice users may spend more time switching from one word to another, since they are still highly visually-guided [43].

Wobble Users may make a wobble within their gestures, for example when they want to emphasise on a key i.e. writing a double-letter word like 'loose'. We can recognise a wobble when there are a lot of movements within the same small area, sometimes with higher speed. The issue is to differentiate it from the noise, due to how the mobile device captures the touch events.

Fluidity / Dwelling / Chunking When gesturing, users may pause or slow down (i.e. dwell) in the middle of gesturing. Technically, dwelling happens when the momentary velocity is

zero or relatively lower [43], which makes the movement jerky [19]. Users may do it accidentally because they hesitate, interrupted, or looking for the next key [43]. Dynamically, “dwelling” can also happen when changing the movement direction, i.e. on corners [62]. This suggests that users may separate the activity into separate smaller activities, as in chunking the gesture. We can analyse the features of each chunk separately, to understand the process of gesturing further. The more fluidly a user gestures, the less chunking happens.

2.8.3 TECHNICAL FEATURES

Aside from finger touch input, we can also take advantage of how the gesture is captured by the devices, i.e. using sensors.

Device Orientation Users may hold the device differently, depending on the context, which changes the device orientation. For example, when the user is lying down on their side, the orientation tilts. The device orientation is often measured using the accelerometer sensor in mobile devices [14].

Device movement Users may be on the move when writing, causing the device to shake. The device movement is often measured using the accelerometer sensor in mobile devices.

2.9 SUMMARY

This chapter began with a review on gesture-based interactions on mobile devices, emphasizing different approaches to increase its accuracy and expressive power. I discussed about gesture keyboards, an interesting gesture-based system that lets users enter text by drawing gestures. I highlighted the opportunities of leveraging the gesture variation and spatial location to increase the expressive power of gesture keyboards. The last section of this chapter summarized gesture features that we can derive from a gesture’s shape and dynamics.

To successfully allow users manipulate their gesture variations to generate rich output, we must first determine the extent to which user’s gestures vary when gesture-typing; whether they vary significantly; and if we can reliably detect specific features within those variations. The next chapter begins the investigation on real users’ gesture variability, i.e. what gesture features users vary when they gesture type.

3

Gesture Variability

Handwriting is a form of *rich output* generated by humans, i.e. the output may contain additional information aside from the content (i.e. the words) itself, e.g. information about the writer and the process of writing. I am interested in combining these benefits of handwriting with the benefits of digital writing: users can explicitly or implicitly communicate more information in their text continuously *as* they write, while giving them full control of how and when the additional information is presented in the final output. Current mobile devices already include high-resolution sensors capable of measuring the variation, and commercialized gesture typing keyboards are widely installed and are already designed to tolerate deformations of the “ideal” gesture template. Rather than throwing away these existing gesture variations, I am interested in “recycling” them. Capturing continuous features of the gesture variations and mapping it to properties of the rendered text could re-enable some of the benefits of handwriting, such as recognizable personal styles, implicit communication of mood, activity, or context; and explicit communication of emphasis, sarcasm, humor, and excitement. Before we can do so, we need to determine if the gesture variation can be quantified as detectible features which can then be mapped to rich output variation.

This chapter describes a preliminary observation and an experiment investigating gesture variability when gesture typing. It also describes several definitions of gesture features inspired from the observations that significantly varied depending on the condition and participants.

3.1 PRELIMINARY OBSERVATION ON GESTURE VARIABILITY

The initial step towards leveraging the variation in gesture-typed input is to investigate how gesture-typing performance varies substantially within and across users (due to biomechanics or personality), or context (activity or environment). I started with a preliminary observation of how users gesture-type with gesture keyboards, to see if there are visually noticeable gesture variations a user makes, and how it is different from one person to another. The goal is to vary the conditions as systematically as we would in a formal experiment. I also want to gather insights about factors that may affect gesture variations, before designing a formal experiment.

We suspected that users would gesture type differently depending on the context. For example, we expected novices to gesture type more carefully while experts are already used to the error-tolerance feature of gesture keyboards. We also suspected that placing users under time constraints would encourage them to gesture type faster. These led us to choose three different INSTRUCTION: *accurate*, *quick*, and *creative*. We asked the participants to gesture type twelve words “as *accurately* as possible”; “as *quickly* as possible while still being accurate”; and “as *creatively* as possible”.

The *accurate* condition should provide the minimum level of variability for novice gesture-typists as they try to match the word shape as closely to the template as possible.

The *quick* condition might realistically be found in real-life gesture-typing under time constraints, and presumably results in greater variability and divergence from the template.

The *creative* condition was designed to provoke more extreme variation, in which we ask the users to “imagine” different ways to enter the word with gestures. It is not intended to match a real-world gesture-typing scenario.

We chose two participants who had each tried gesture keyboards once but have never used it since, and one participant who used it daily. We provided two devices: a tablet (Samsung Galaxy Tab Pro) running Android 4.4.2 and a mobile phone (Nexus 5) running Android 5.1. P₁ used the tablet; P₂ and P₃ (the frequent user) used the phone. We recorded the screen and audio recorded the participant’s verbal comments.

Table 3.1: Gesture-typing accuracy in the Preliminary Observation with three participants.

Participant	ACCURACY			
	<i>accurate</i>	<i>quick</i>	<i>creative</i>	TOTAL
P ₁ (<i>non user</i>)	68%	77%	45%	63%
P ₂ (<i>non user</i>)	83%	84%	35%	68%
P ₃ (<i>daily user</i>)	84%	88%	10%	61%

3.1.1 PROCEDURE

Participants sit on a chair while holding the tablet or the phone comfortably with their left hand. They first try to use the gesture keyboard to make sure they can do it. Next, the observation begins: a word is presented to the participants and they must gesture type it while following an INSTRUCTION, ten repetitions for each. They are encouraged to think aloud while completing the task. The researcher observes and takes notes of the success of each attempt. Upon finishing, the participants are interviewed.

3.1.2 OBSERVATIONS

Even with only three participants, we see three clearly different gesture styles. P₁, a novice user who used the tablet made the sharpest, most angular gestures. P₂ held the mobile phone with his right hand and gesture-typed with the thumb of one hand. His gestures tended to be faster and more sloppy, but also performed an idiosyncratic “under-loop”. This is particularly interesting: while we hypothesized that non-users would be more likely to slow down and carefully follow the gesture template, P₂ quickly picked up the error-tolerant feature of the gesture keyboards and started gesture typing quickly. P₃, who used gesture keyboards daily, held the phone with her left hand and gesture-typed with the right index finger, making more “loopy” gestures when turning or reaching a key target.

We also see how the participants deliberately changed the way they gesture-type according to the instruction. When participants wanted to gesture type more *accurately*, they slowed down to draw the gesture from middle point to middle point of each key more precisely. In the *creative* condition, they explored how to draw the gesture differently while still ensuring the gesture correctly recognized. If they made a mistake because they included too many variations in their gesture, they tried to be more careful in the next trial. The non-user participants who did not use gesture keyboards daily were also more careful when gesture-typing in the *creative* condition (see Table 3.1), and their accuracy was not as low as the daily user (i.e.

P₃).

This preliminary observation is not a controlled experiment with formal data collection and statistical analysis, thus we cannot draw significant conclusions out of the data. However, it gave me two important insights: there is a high chance that gesture-typing performance vary within and across users due to e.g. personal style and intention; and user gesture-typing behaviour is affected by the recognition algorithm. Based on these insight, we designed an experiment investigating deliberate variations users make when gesture-typing.

3.2 EXPERIMENT 1: DELIBERATE GESTURE VARIATION

We designed an experiment which goal is to systematically observe the amount of gesture variations that users employ when asked to gesture type in different manners, as well as how the variations differ across users. If users can deliberately vary their gestures, then we can quantify and map them into various output properties.

From the preliminary observation, we learned how feedbacks from the recognition engine may affect gesture-typing behaviour, such as typing more quickly and sloppy because they realized they did not have to be precise. This may induce unwanted noise, since our main goal is to see to what extent users can deliberately vary their gestures. Thus, in this experiment, we decided to remove this potential source of error, by choosing participants who are non users and asking them to gesture-type on a non-interactive Wizard-of-Oz (WOZ) keyboard. This is to ensure they are not affected by the error-tolerant feature of the recognition algorithm.

I conducted a within-participants experiment with three types of INSTRUCTION as the primary factor: Participants gesture type specified words “as *accurately* as possible”; “as *quickly* as possible while still being accurate”; and “as *creatively* as possible, have fun!” (as described in Section 3.1).

The shape of gesture typing’s input vary depending on the letters forming word and their positions in the soft keyboards. Most word gestures include several stroke segments, with different angle between the segments, but some are straight lines, for example “pure” in QWERTY keyboards. Two or more words may also have exactly the same word gesture at the exact same location, for example “lose” and “loose”. In Chapter 2, we learn that these factors affect the recognition accuracy – identical gesture templates are harder to recognize. Furthermore, angle between segments may affect the motor control complexity [62]. Thus, we systematically distributed these potential source of noise over the testing word set. We chose three sets of 12 words (see Table 3.2) that vary systematically according to three dimensions:

Table 3.2: Three word sets used in the Experiment 1. Each word set contains of twelve words of different length, letter repetition, and angle between stroke segments.

WORD-GESTURE CHARACTERISTICS			WORD SET		
LENGTH	LETTER REPETITION	ANGLE	WORD SET 1	WORD SET 2	WORD SET 3
short	single	zero	<i>your</i>	<i>pure</i>	<i>per</i>
short	single	acute	<i>wax</i>	<i>vein</i>	<i>sigh</i>
short	single	obtuse	<i>lose</i>	<i>taxi</i>	<i>back</i>
short	double	zero	<i>all</i>	<i>zoo</i>	<i>peer</i>
short	double	acute	<i>jazz</i>	<i>feel</i>	<i>fell</i>
short	double	obtuse	<i>fill</i>	<i>loose</i>	<i>knee</i>
long	single	zero	<i>queue</i>	<i>power</i>	<i>query</i>
long	single	acute	<i>midnight</i>	<i>joking</i>	<i>exorcize</i>
long	single	obtuse	<i>bracket</i>	<i>headache</i>	<i>jewel</i>
long	double	zero	<i>pepper</i>	<i>puree</i>	<i>tweet</i>
long	double	acute	<i>vaccine</i>	<i>queen</i>	<i>middle</i>
long	double	obtuse	<i>loose</i>	<i>syllable</i>	<i>arrive</i>

- i word LENGTH: short (no more than 4 characters), long (more than 4 characters)
- ii ANGLE between stroke segments: zero, acute, obtuse
- iii LETTER REPETITION: single (e.g., *lose*), double (e.g., *loose*)

For example, the word *puree* is long, with a double letter ‘e’, and a zero drawing angle, i.e. a straight line on the keyboard; *taxi* is short, with a single letter and at least one obtuse angle: the chunk *axi*. Each letter appears at least once in each set.

3.2.1 PARTICIPANTS

We recruited seven men and five women, all right-handed, mean age 26. All use mobile phones daily, but none had used gesture-typing prior to this experiment.

3.2.2 APPARATUS

I developed a custom Android application running on an LG Nexus 5 (Android 5.1) smartphone. It displays a non-interactive Wizard-of-Oz (WOZ) keyboard that matches the po-

sition and dimensions of a standard QWERTY English keyboard. The application takes a screenshot after every word gesture drawn by the participants.

We use the *remulation* keyboard evaluation technique described in [10]. The WOZ keyboard collects gesture coordinates from the participants and generates log files. I wrote a monkeyrunner jython script that opens a connection to an Android device and continually sends a stream of gesture coordinates. The script feeds the recorded gesture coordinates to the Android MonkeyRunner event simulation tool on a desktop computer that communicates with the tethered mobile phone using the Android Debug Bridge. This process simulates the gesture typing activity on a working gesture keyboard, which then generates the word output. We use two commercial gesture keyboards (referred to anonymously as KB-1 and KB-2) representing the state-of-the-art for Android, with over 150 million copies installed collectively. Both keyboards have identical dimensions but different recognition algorithms^{*}. The script also sends a random word between each prompt word and erase personalization data after each participant. This ensures that the recognition results cannot be contaminated by the adaptation and personalization features in the keyboards.

3.2.3 PROCEDURE

An experiment session lasts approximately 50 minutes. Participants sit in a chair and hold the phone comfortably in their left hands, so they can perform all gestures with their right index finger. Participants are encouraged to talk aloud as they draw each word. During initial training, participants may practice until they feel comfortable using the gesture-typing technique.

Each trial in the experiment begins with an instruction displayed at the top of the screen, e.g., “Draw as accurately as possible”, with a word centered below, e.g., *queue*, and a soft keyboard at the bottom of the screen (see Figure 3.1a). The trial ends when participants lift their finger, after which they answer a multiple-choice question as to their level of confidence: “Do you think you wrote *vein*?” (*Yes*, *No*, or *Not sure*). Each word is presented as a sub-block with 10 replications.

The experiment consists of 360 trials (12 words x 3 instructions x 10 replications). Sessions are organized into three blocks of 12 words, according to the instruction. All participants begin with the *accurately* instruction; *quickly* and *creatively* are counterbalanced for order across

^{*}We anonymize the keyboards here since the goal of our investigation probably differs from that of the keyboard designers and the samples we collected may not match typical uses of the keyboards.



Figure 3.1: Gesture variations: a) *accurately* is straight, b) *quickly* is smooth, and c) *creatively* is inflated and highly varied.

participants. The 12 words are chosen from the three word sets; counter-balanced within and across participants. I chose four words from each word set while ensuring that every word appears four times at the end, each in a different session.

3.2.4 DATA COLLECTION

We record the touch coordinates in order to extract spatial and temporal characteristics of each gesture. A gesture is a series of touch data in tuples: $(x, y, \text{timestamp})$ where x and y are the touch spatial location relative to the screen (i.e. x -axis and y -axis) in pixels, and the `timestamp` is the time of the touch occurrence in milliseconds. We later simulate the gesture data on gesture-typing recognizers, KB-1 and KB-2, to derive Accuracy, i.e. the recognizer score for the intended word ($True=1$, $False=0$). We also record the participant's ConfidenceRate – an ordinal measure of the post-trial answers ($Yes=1$, $Not\ Sure=0.5$, $No=0$). The post-questionnaire asks participants to describe how they varied their gestures according to each instruction. We also take a screenshot of each gesture; record a kinematic log of each gesture, using screen capture; and audio record the participant's verbal comments.

3.2.5 RESULTS AND DISCUSSION

We are interested in investigating the extent to which the participant's gestures vary as they gesture-type. We first examined the subjective measures obtained through the post-questionnaire and looked at the existing variability in gesture data to identify candidates for gesture features. I briefly report the analysis regarding recognition rate and confidence rate before moving to the objective measure of the feature candidates.

We collected 4320 unique gestures. We removed 22 outliers (0.5%), defined as when:

1. a participant said they made a mistake, e.g. accidentally lifting the finger before finishing the gesture
2. they answered *no* to the post-trial question
3. gesture length was <100 pixels

Significance rates for confidence or recognition rate were not affected.

GESTURE VARIABILITY

Like [82], we found that participants viewed the word-gesture as crossing through “targets” i.e. each letter in a word. Participants changed the way they drew depending upon their perception of the instructions. Seventy-five percent (9/12) of the participants said they “pass through all the letters” and “stay in the [letter] box” in the *accurately* condition. This results in straight-line gestures (Figure 3.1a) that closely resemble the gesture template. Not surprisingly, participants drew faster when asked to draw *quickly*, which resulted in smoother, more curvy gestures (Figure 3.1b), although two participants mentioned that they tried to draw straight lines to make them shorter and take less time. Almost half (5/12) said they explicitly ignored precision when they drew *quickly*. Participants interpreted the *creatively* instruction as either to “draw shapes along the way from one key to another” (7/12), or to “be comfortable, likeable, or suitable” (5/12). The *creatively* instruction resulted in the highest variation across gestures, as shown in Figure 3.2.

Some participants changed their behaviour in response to the order of the instructions. For example, P₃ took advantage of the creative instruction to discover a faster and more comfortable technique, which changed his behaviour on the quickly instruction: “*If I had an ‘accurate’ before ‘quick’, I would just repeat the same move. If I had a ‘creative’ trial before, I would pick the stroke that I found the most comfortable to use.*”



Figure 3.2: Recognized creative gestures included: a) loop and cusp for *taxi*, b) visualization of crown for *queen* and c) scribbling on keys to create the stars or a constellation for *midnight*.

Table 3.3 summarizes the keywords used by the participants to describe how they varied their gestures for different conditions. Analysis shows that the participants varied their gestures by varying three main feature candidates: speed, inflation, and curviness. Since a word-gesture has to cross certain targets, scaling the gesture is not applicable; hence the participants *inflated* the shapes (as in Figures 3.1b and 3.1c) while still trying to cross the letters as a means to create a larger gesture.

ACCURACY

Although recognition is not our primary goal, we are interested in how the different recognizers reacted to the variation in the users' gestures. We expect low accuracy since both recognizers (KB-1 and KB-2) are known to use language context to resolve ambiguities between word shapes in normal use, and participants did not receive accuracy feedback during the experimental trials. The instruction *creatively* was specifically intended to provoke wide exploration and was not expected to give recognizable results – in fact, high recognition rates would indicate a failure to provoke adequate exploration.

Accuracy for KB-2 (73.6% of the gestures recognized correctly) was significantly higher than for KB-1 (46.7%). One contributing factor is that KB-1 treats leaving the keyboard space as a cancellation. However, even when these trials are removed, KB-2 achieved a significantly higher recognition rate (75% vs. 53%).

INSTRUCTION also has a significant effect on Accuracy for both keyboards ($F_{2,22}=140.6$ and $F_{2,22}=106.3$ for KB-1 and KB-2 respectively, all $p<.0001$). A post-hoc analysis with Tukey HSD showed that accuracy for KB-2 is significantly lower for *creatively* (mean=62%), but no significant differences between *accurately* (82%) and *quickly* (79%) obtained. Surprisingly, for

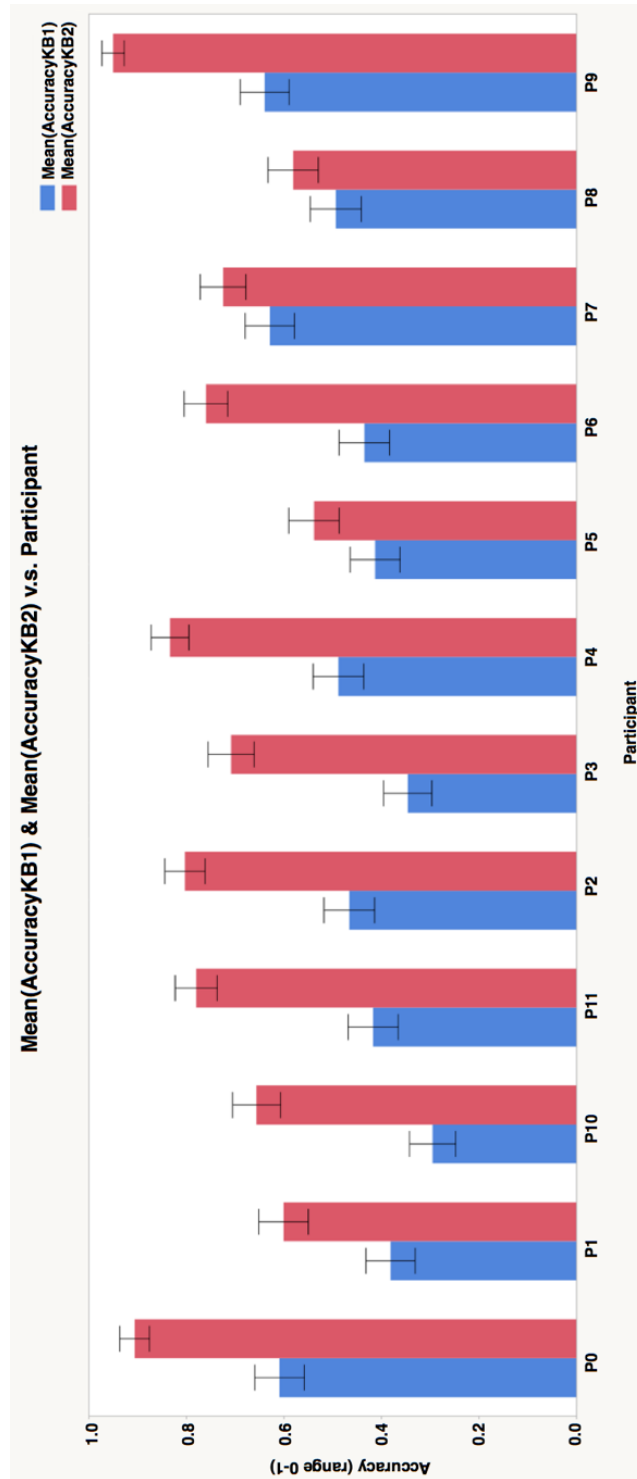


Figure 3.3: Accuracy of all participants for KB-1 and KB-2 (range from 0 to 1) in the Experiment 1.

KB-1, a post-hoc analysis with Tukey HSD showed that the accuracy of each instruction is significantly different from the others. The instruction *creatively* has the lowest accuracy (mean KB-1=34%). However, the *quickly* instruction resulted in fewer errors (mean accuracy=57%) than instructions executed *accurately* (mean accuracy=53%). This may be due to the different algorithm each keyboard uses.

A post-hoc analysis with Student's *t*-test shows that some participants can draw recognizable gestures significantly better than the other participants (see Figure 3.3). For KB-1, P0, P7, and P9 have significantly higher accuracy than other participants. For KB-2, P0 and P9 have significantly higher accuracy than other participants. While the accuracy may change when the participants receive the accuracy feedback, it may suggest that every participant has their own perception of how to draw a "correct" gesture, which thus affects accuracy.

CONFIDENCE RATE

Participants expressed high overall confidence in their performance. Participants were most confident when they tried to write *accurately*, and least confident when they wrote *quickly*. For *creatively*, participants are less sure whether or not they wrote the intended word: 12.4% *Not Sure* and 5.5% *No*. Their comments indicate that they drew more carefully in the *accurately* condition, which appears to have increased their confidence.

In summary, users vary their gestures in response to different instructions while maintaining high confidence in performance, despite lack of formal training. Eight participants were eager to pursue more expressive gestures when writing, especially in place of emoticons or when communicating with friends and family. This suggests that generating rich output from soft keyboards is possible under user control, as long as the recognizer can reliably identify key features of the user's gestures.

3.3 QUANTIFYING VARIATION IN GESTURE-TYPING

Arguably, there are several causes of the variations in gesture-typed input, and they can be *intentional* and *unintentional*. In Experiment 1, the participants intentionally varied their gestures depending on the instruction, e.g. gesture type faster when asked to be as quickly as possible. To some extent, there are also unintentional variations, either *explicit* i.e. personal style or *implicit* e.g. writing in a bumpy bus or when being angry. I would like to argue that

Feature	# Participant	Example Quotes
Speed	10	“To be sure to stand on every letters and do it slowly.” (P0:A) “[...] touch every letter that formed the word and finish it faster.” (P10:Q)
Inflation	8	“I would start from inside the first letter and then just aim for the next letter in the most direct way.” (P3:A) “I did exaggerated gestures, while trying to introduce variations.” (P7:C)
Curve	6	“to use ‘straight’ lines instead of curvy ones.” (P4:Q) “I did circular swipes.” (P11:C)
Pressure	1	“[...] to put more pressure in the correct boxes.” (P5:A)

Table 3.3: Subjective features the participants in Experiment 1 used to vary the gestures extracted from the questionnaire. A=accurately, Q=quickly, C=creatively.

these source of variations are not necessarily “noise” if the goal is to communicate a message to other people – they can potentially enrich a text message written by a user to another person. For example, writing in a bumpy bus may result in a very sloppy gesture. When the information is expressed in the rendered text, it can be an interesting contextual information e.g. if the user is writing to her closed ones. Our goal is to extract a few measurable features from the “noise” that is currently discarded by gesture-typing systems, and to use these features to determine whether our concept of “expressive” gesture-typing is viable. We are not trying to create a complete taxonomy of gesture variations, nor to determine users’ emotion based on the gesture variability – we simply aim to increase the information transfer in text-based inter-personal communication by enabling users to generate rich output based on how they gesture type. This suite of features can be improved and extended in the future.

To determine if the variation identified in Experiment 1 can be quantified as detectable features, which can then be mapped to text output variation, we considered candidates from the Experiment 1 (as summarized in Table 3.3) as well as the previously explored gesture characteristics described in Section 2.8. We evaluated them with respect to their applicability to gesture-typing. Word-gestures are restricted by keyboard layout properties, such as letter size and position, making features such as orientation, direction, or scale difficult to control while gesture-typing. We performed an ANOVA[†] to determine the effect of INSTRUCTION (user intention) on each feature.

[†]All analyses are performed with SAS JMP, using the REML procedure to account for repeated measures.

3.3.1 SPEED

Speed is a common measure of gesture variation, e.g., for typing activity [59, 18] and modeling the production time of a gesture [13, 17]. For gestures, speed is calculated by dividing the total length of traced distance (in pixels) by the total time (in milliseconds). We found that average drawing speed is significantly affected by INSTRUCTION ($F_{2,22}=216.8$, $p<.0001$), and they are significantly different from each other. Drawing *quickly* is the fastest (mean=1.8 px/ms), followed by *creatively* (mean=1.3) and *accurately* (mean=1.07). A post-hoc analysis with Student's t -test shows that some participants can draw gestures significantly faster than other participants. The data suggests that participants vary the drawing speed depending on the condition and drawing speed is different across participants.

Since our goal is to enable more fine-grained gesture control, we would like to increase granularity by examining each gesture over chunks of movement instead of as a whole. Given a sequence of points $P = \langle (x, y) \rangle$, we can divide P into n (equal) chunks, where P_i is the i^{th} sub-sequence. Thus, we calculate the average speed (v_i) of each chunk as follows:

$$v_i = \frac{\sum_{j=0}^n \sqrt{(x_j - x_{j+1})^2 + (y_j - y_{j+1})^2}}{T_i} \quad (3.1)$$

where T_i is the total time for P_i . Parameterizing n increases the possibility of exploration in the feature space, which may give similar information to acceleration but with less noise. We started with $n = 2$ and compared the speed of the first and last half of each gesture to measure *speed consistency*:

$$R_{i,j} = \frac{v_j}{v_i} \quad (3.2)$$

We found that participants were more likely to start quickly and then slow down (mean $R=0.83$; ratios less than 1 indicate drawing more slowly). This is in line with Fitts' Law [24], that users tend to slow down when they are getting closer to the pointing target. An ANOVA showed that INSTRUCTION significantly affects the speed ratio ($F_{2,22}=35.4$, $p<.0001$). Drawing *creatively* results in the most constant rhythm (0.91), followed by *accurately* (0.82) and then *quickly* (0.77). Additionally, different patterns obtain when writing a long word as opposed to a short one. Figure 3.4 shows that participants performed faster at the end of long words with obtuse angles (1.1), such as *jewel*, performed at constant speed with acute angles (1.0), such as *joking*, and slowed down when angle=zero (0.76), such as *pure*. This suggests that participants may separate long words into separate chunks and then draw each chunk more

consistently.

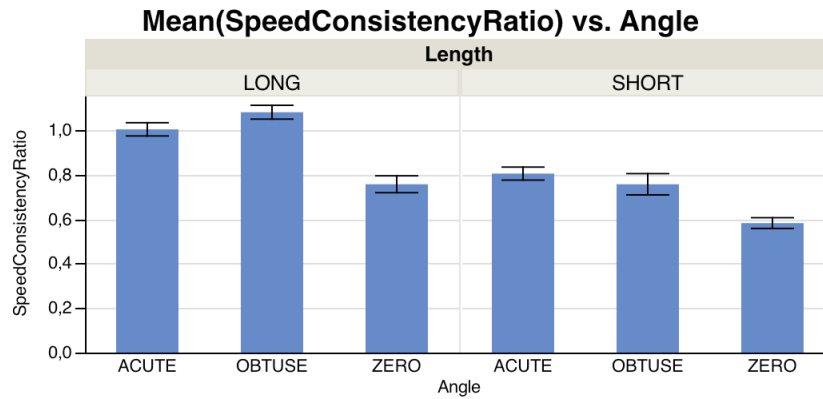


Figure 3.4: Speed ratio vs. angle by length. Participants slowed down when drawing short and zero-angle (straight-line) words, but maintained constant speed when drawing long words.

In summary, using velocity profiles as a feature reveals richer information about the gesture performance, i.e. although the overall drawing speed for *quickly* is faster, participants actually start quickly and slow down the most at the end, and each participant may draw the gesture at different speed.

3.3.2 INFLATION

Some properties of a gesture, e.g., direction and residual momentum, may also result in unintentional inflation or overshooting that goes beyond the limits of the keyboard itself. More interesting are deliberate gestures drawn outside the keyboard, as in Figure 3.1c. Since the gestures are constrained to pass approximately through each letter key in a word, we refer to this variation as *inflation* rather than *size*, but it can nevertheless be quantified using the ratio between the minimum bounding boxes of the performed and template gestures. A bounding box (in *pixels*²) is a two-dimensional rectangle that defines the extent of the gesture; the area is a multiplication of its length and width. We calculated the ratio between the participants' gestures and the gesture template's bounding box (Rb):

$$Rb_{word} = \frac{B_{gesture}}{B_{template}} \quad (3.3)$$

We found a significant effect of INSTRUCTION on inflation ($F_{2,22}=185.9$, $p<.0001$). A post-hoc analysis with Tukey HSD showed that words drawn *creatively* are significantly more likely to

be inflated; with no significant difference between *accurately* and *quickly*, which corresponds to qualitative observations. Some participants intentionally drew outside the keyboard as they moved from one key to another, or drew very large gestures, beyond the normal bounding box.

A post-hoc analysis with Student's *t*-test shows that some participants draw gestures significantly more inflated than the other participants. Three participants who in general drew more inflated gestures: P₅, P₈, and P₁₀. This is mainly because they tended to make more extreme jumps and inflated gestures when the gesture template is a straight line, e.g. 'pure'.

Although Long et al. found that visually the size of the bounding box is not a strong differentiating factor in gesture similarity [49], our data show that word-gesture inflation varied according to condition, with a corresponding effect on the minimum bounding box ratio.

3.3.3 CURVINESS

While a word-gesture template consists of lines and corners, in practice a gesture may also include curves [17]. In our data, calculation of a normalized curviness metric was complicated by the need to compare word templates with varying numbers of corners. We consider the absolute instantaneous angle among three points using tangents. Given a sequence of points $P = \langle p_i \rangle_{i \dots N}$, where $N = \text{size}P$, ϑ is the angle between vector $\vec{u} = \overrightarrow{p_i p_{i+1}}$ and vector $\vec{v} = \overrightarrow{p_{i+1} p_{i+2}}$ where $p_i, p_{i+1}, p_{i+2} \in P$, calculated as follows:

$$\vartheta = |\text{atan2}(|\vec{u} \times \vec{v}|, \vec{u} \cdot \vec{v})| \quad (3.4)$$

where *atan2* is a function of arctangent in Android that returns a value in the range $(-\pi, \pi)$. In our analysis, ϑ is in degrees $(0 - 180^\circ)$.

To emphasize the relative variations over the gesture, we measure the curviness (in degrees) by the standard deviation of all angles, calculated as follows:

$$\text{curviness} = \sqrt{\frac{1}{N-2} \sum_1^{N-2} \left(\vartheta - \left(\frac{\sum_1^{N-2} \vartheta_i}{N-2} \right) \right)^2} \quad (3.5)$$

The standard deviation is close to zero for straight lines but higher for curvy lines. For gestures consisting of several segments, e.g., *taxi*, even if the lines are drawn straight, the standard deviation is still affected by the corners. While different words may have different numbers

of corners with different angles, we can still reliably distinguish a curvy gesture by setting a threshold value. If all lines are drawn relatively straight, the standard deviation is around 12; cusps (acute corners), scribbles, and mixed straight-line/curves are >12 ; while arcs (i.e. curves) and loops (obtuse corners) are < 12 .

An ANOVA showed that participants increase curviness when writing *quickly*, which is significantly different ($F_{2,22}=60.5, p<.0001$) from *accurately* or *creatively*. These results correspond to the qualitative observations.

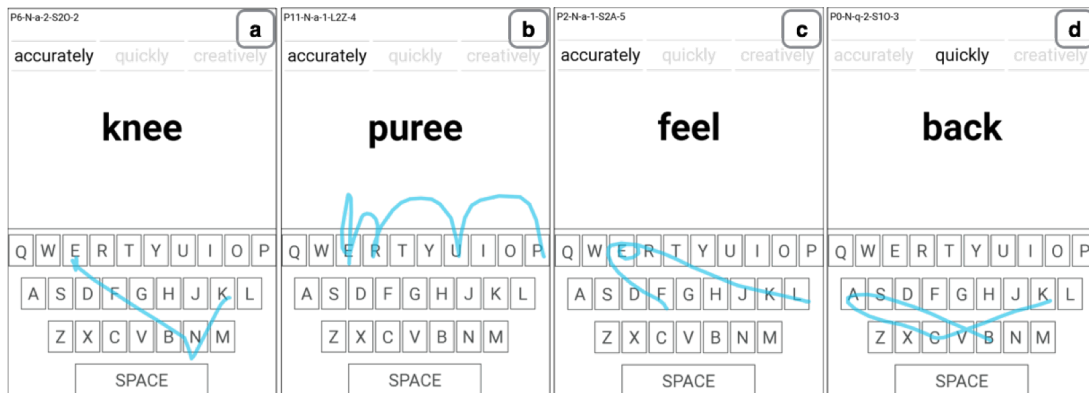


Figure 3.5: Recognized gestures from different participants, which highlights personal difference in terms of the curviness in the drawn gestures. a) P6 drew a cusp and a wobble. b) P11 drew a jumping gesture to avoid passing through other letters when writing 'puree'. c) P2 used a combination of straight lines and loops to write 'feel'. d) P0 drew curvy gestures with obtuse turning between stroke segments.

A post-hoc analysis with Student's t -test shows that some participants can draw gestures significantly more curvy than other participants (see Figure 3.5). The test presented five significantly different categories, e.g. P6 and P11 made significantly more cusps, wobbles, curves, and jumps in their gestures than other participants (mean curviness 17.9 and 17.1 respectively) while P3, P8, P7, and P1 made the most straight gestures (mean curviness 12.5, 12.1, 11.9, 11.8 respectively).

In summary, participants tend to draw more curvy gestures when writing *quickly* and straighter gestures when writing *accurately*. This is not surprising when speed is the goal, given that the human motor system maximizes smoothness to reduce movement cost [62]. The participants may also have different styles when it comes to curviness when they gesture-type.

3.4 SUMMARY

The Experiment 1 in this chapter confirms that gesture-typing performance varies across participants and experimental conditions, i.e. the intention of writing to be accurate, quick, or creative. We can reliably extract three key gesture characteristics – curviness, inflation, and speed – that varied significantly according to the instruction given to the participants. We observed large individual differences across participants with respect to the three features. Perhaps not surprisingly, this suggests that each individual is likely to generate distinctive, personal gesture styles, just as they do with their handwriting.

The next step is to create a technique for mapping user-generated characteristics to continuously variable output, enabling user- and context-dependent variation to appear in the output, and to test whether users can *intentionally* transform the variation of their gestures into rich output.

4

Transforming Gesture Variation into Rich Output

4.1 EXPRESSIVE KEYBOARD

I introduce **EXPRESSIVE KEYBOARD**, a gesture keyboard that leverages the variations in gesture-typed input to produce rich output. The rich output includes additional information about the gesture variability aside from the content (i.e. the words), which may reflect the process of writing. However, we are not interested in ‘identifying’ expression or emotion based on gestures, since we believe richer interpretations will result from a system in which users can develop their own communication contexts and related meanings. More importantly, **EXPRESSIVE KEYBOARD** opens up the possibility for increasing information transfer in textual communication with an instrument that enables users to express themselves through personal style and through intentional control.

EXPRESSIVE KEYBOARD takes advantage of gesture keyboards, which recognition algorithm tolerates gesture variability. **EXPRESSIVE KEYBOARD** adds a layer of gesture analysis, separate from the recognition process, that quantifies the differences between the gesture template and the gesture actually drawn on the keyboard. These features can then be mapped to output properties and rendered as rich output. I describe the process step by step, as fol-

lows.

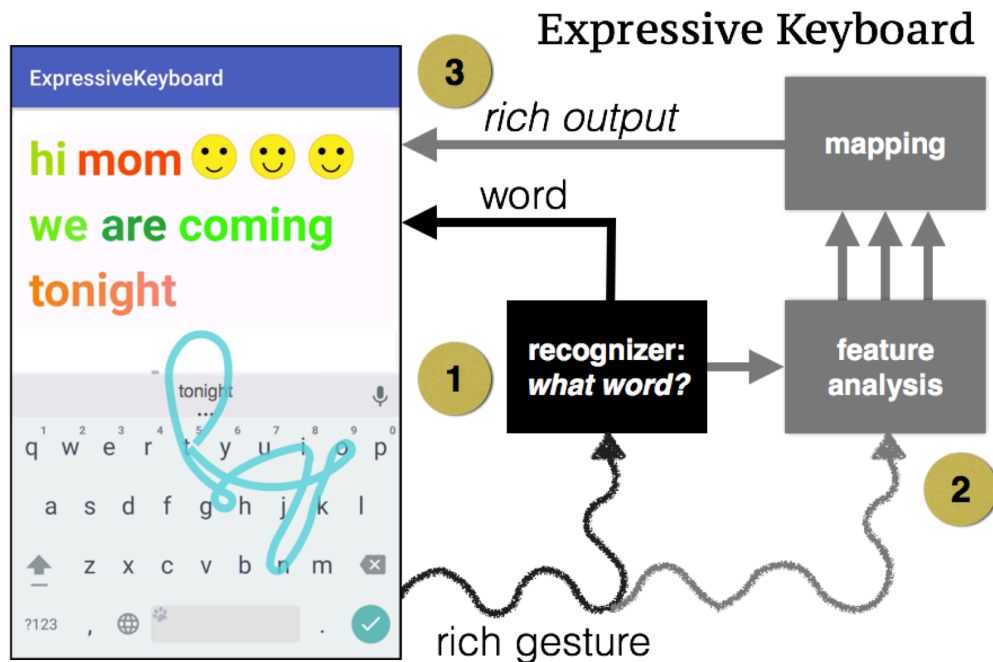


Figure 4.1: The process of transforming gesture input into rich output with *Expressive Keyboard*. 1) The gesture keyboard recognizes the gesture into a word just as before, as if it is a “blackbox”. *Expressive Keyboard* adds additional steps (in grey): 2) quantifies the variations in the captured gesture input and 3) maps them to output properties. Finally, *Expressive Keyboard* renders the recognized word along with the output properties as a rich output.

WORD RECOGNITION The process starts when the gesture keyboard captures users’ gesture input. A gesture input is a series of touch data in quadruplets: (action, x, y, timestamp) where action is either TOUCH_DOWN, TOUCH_MOVE, or TOUCH_UP; x and y are the touch spatial location relative to the keyboard view (i.e. x-axis and y-axis) in pixels; and the timestamp is the time of the touch occurrence in milliseconds. A gesture always starts with TOUCH_DOWN i.e. when the user puts her finger on top of the keyboard, followed by a series of TOUCH_MOVE as she drags her finger, and ends with TOUCH_UP after she lifts her finger from the keyboard.

As soon as the user starts drawing a line connecting the letters in a word, the recognition engine recognizes the word-gesture input to come up with most-likely word output. This process of capturing and recognizing the gesture is done progressively. As it is, the traditional

gesture keyboard throws away the additional information i.e. how the gesture is performed – the only output is the recognized word. This process is shown in Figure 4.1 on Step 1.

GESTURE FEATURE ANALYSIS EXPRESSIVE KEYBOARD adds a layer of gesture feature analysis that quantifies the gesture variability in users' input (see Figure 4.1 Step 2). The gesture keyboard provides the gesture data, the final recognized word, and the coordinate of each key in the keyboard – from which the gesture features are calculated. This process does not interfere with the word recognition process, thus we can use any word-gesture recognition algorithm. We can use any gesture feature, as described in Section 2.8. To start, I used curviness, inflation ratio, and speed consistency ratio, as described in Section 3.3.

MAPPING AND RENDERING EXPRESSIVE KEYBOARD maps the gesture features e.g. curviness, inflation ratio, and speed consistency ratio to output properties. The gesture features can be mapped to different output properties aside from text properties, such as emoticons or animation (see more details in Section 4.2). The final result is the recognized word, displayed according to the variation inherent in the user's gesture input. Users can thus vary their gestures and see the corresponding change in the final output.

4.2 APPLICATIONS

EXPRESSIVE KEYBOARD concept can be implemented on any device in different environment, such as mobile phones, tablets, smartwatches, and even mid-air operation e.g. Vulture [55]. In this thesis, I focus on the interaction on mobile phones. Below I provide several applications of EXPRESSIVE KEYBOARD that focus on inter-personal communication systems.

4.2.1 COLORFUL TEXT

EXPRESSIVE KEYBOARD can transform gesture variations into the full range of RGB colors. I mapped inflation ratio to *red*, curviness to *green*, and speed consistency ratio to a *gradient of blue*. Gestures that use a constant speed when following the gesture template – a straight line from middle point to middle point of each letter in the word – map to the color black. As the user slows down, the text output turns from blue at the beginning of the phrase to another color at the end of the phrase. This mapping makes it technically feasible for users to

Table 4.1: Summary of mappings to generate colors and dynamic font with Expressive Keyboard.

Gesture Variability	Properties of Color	Properties of Dynamic Font
<i>Inflation Ratio</i>		
User's gesture has bounding box of the same size with the gesture template [inflation ratio = 1.0]	RED = 0	font thickness = 10
User's gesture has bounding box bigger [inflation ratio ≥ 1.5] smaller [inflation ratio ≤ 0.5] than the gesture template	RED = 255	font thickness = 20 (i.e. bold)
<i>Curviness</i>		
User's gesture consists of straight stroke segments	GREEN = 0	offset to control points = 0 (i.e. all 'a's are identical)
User's gesture consists of curves, cusps, wobbles, or loops	GREEN = 255	offset to control points = x where $0 \leq x \leq 50$ (i.e. all 'a's are unique)
<i>Speed Consistency Rate</i>		
Consistent speed over the gesture [speed ratio = 1]	BLUE = 0	the baseline font
Speed up [speed ratio ≥ 1.5] or slow down [speed ratio ≤ 0.5]	BLUE = 255 (speed up: any color to blue; slow down: blue to any color)	the font variation

generate all possible RGB color combinations. Table 4.1 summarizes the mapping of gesture features to the RGB components.

I chose a quadratic mapping between the gesture features (e.g. speed consistency rate) and the output properties (e.g. blue in RGB) so the rate is change (i.e. the change in the value of output property by the elapsed speed consistency rate) is small when the gesture is more or less not inflated (speed consistency rate ≈ 1.0) but bigger when the gesture is inflated. This makes the “black” color easier to produce although the users do not gesture type precisely like the gesture template (straight line segments from middle point to middle point) with perfect consistent speed. Likewise, to produce text with other color, they need to distinctly manipulate their gesture variation.

Figure 4.1 shows an example of how one can use EXPRESSIVE KEYBOARD to generate col-

orful text. The user naturally makes curvy gestures, thus her “default” output is green text. When she wants to emphasize something, e.g. “Mom” or “tonight”, she changes the way she usually gesture-type to produce different color. For example, she wants to express her excitement that she is coming home that night, so she makes curvy, inflated gesture while speeding up when writing “tonight” (as shown in the figure), thus generating a text which color gradually changes from orange to pink.

4.2.2 DYNAMIC FONT

I implemented an EXPRESSIVE KEYBOARD that maps gesture features to a user-defineable dynamic font. The dynamic font is created through a simple font engine that lets users define static typefaces by connecting control points to form each letter (Figure 4.2). Users can create several typefaces and dynamically interpolate between them to generate new intermediate fonts continuously. The interpolation between n typefaces changes the position of component control points based on a weighting function:

$$typeface_{interpolated} = \frac{\sum_{j=1}^n typeface_j weight_j}{\sum_{j=1}^n weight_j} \quad (4.1)$$

where each typeface is a vector of control points for each letter.

Every font is tagged by an ID number. In the beginning, the users can choose an ID number and start defining the font from ‘a’ to ‘z’ (or continue to define from where it’s left off, for example if previously they quit after defining ‘g’ then they can continue defining ‘h’, etc). This font serves as the *baseline font*.

Once the baseline font has been defined, the users can enter the same ID to:

Create a font variation by editing the baseline font to enable interpolation. Here users can edit the position of the control points to create the variation. The number of the control points must match its baseline.

Edit the baseline font Users can edit the position or add/delete control points.

The speed consistency ratio is mapped to the weight ratio for font interpolation. The inflation ratio is mapped to stroke thickness (bold). The curviness is mapped to the magnitude of random offsets applied to each control point. Table 4.1 summarizes the mapping of gesture features to the properties of the dynamic font.



Figure 4.2: The font engine lets users to draw their own typeface by adding control points that are connected to a cardinal spline. Users can include several strokes by detaching the splines. On the right are two examples of static typefaces defined using the font engine: the baseline (top-right) and the variation (bottom-right).

4.2.3 PARAMETRIC EMOTICON

We can use EXPRESSIVE KEYBOARD to control subtle change of expression in emoticon, to increase the granularity of expression that can be conveyed through emoticon. I developed a parametric emoticon engine where users can choose a keyword to insert a parametric emoticon which expression changes depending on how the keyword is gesture-typed. As a demonstration, I chose a keyword ‘emoji’ to insert a parametric smiley face. I mapped curviness to the size of the smile, and the inflation ratio to the type of smile. More curviness will cause a big smile (Figure 4.3b), while less curviness will only generate a little smile (Figure 4.3a). When combined with high inflation ratio (> 1.2), the big smile will turn into a laugh (Figure 4.3c).

4.2.4 ANIMATED EMOTICON

I worked with a Master student, Chengcheng Qu, to develop an application of EXPRESSIVE KEYBOARD that transforms users’ drawn gesture when gesture-typing into dynamic



Figure 4.3: Manipulating the curviness and inflation in gestures with Expressive Keyboard will generate emoticons with subtle different of expression. a) A gesture with straight stroke segments generates a little smile; b) a curvy gesture generates a big smile; and c) a curvy, inflated gesture with wobbles at the end generates a laughing face.

animated emoticons. As a demonstration, Chengcheng Qu implemented three animation emoticons: “smile”, “cry”, and “sad”. Users manipulate the curviness of their gesture input to control the final value of the animation properties. For example, for the emoticon “cry”, the property is the shape of the tears. When the users gesture type “cry”, the more curvy their gesture is, the more overflowing the tears will end up.

After the users finish gesture-typing and lift their fingers, the emoticon is rendered on the screen and the animation starts. When animated, a very curvy gesture generates an emoticon with the tears gradually changing from welling up to overflowing (the tear-flow expands from short to long). A straight gesture generates an emoticon with welling up tears, with almost not noticeable animation, as if it is a static emoticon. In the case of “smile” emoticon, more curviness will cause a grin, while less curviness will only generate a slight smile, likewise with the “sad” face. To avoid visual saturation, i.e. endlessly moving animation can be disturbing, the animation stops after repeated 5 times. If the users want to invoke the animation again, they could double tap the emoticon, and the animation will restart.

We also implemented some functions for users to modify their generated emoticon: If users wants to modify it after the gesture-typing, they can first put their finger on this emoticon, and then drag up or drag down. In the case of “cry” emoticon, the tears can be shorten by tapping on the emoticon and drag up. The same operation can be applied to the “smile” and “sad” emoticons.

4.3 TECHNICAL IMPLEMENTATION

4.3.1 INITIAL IMPLEMENTATION

The first implementation of EXPRESSIVE KEYBOARD used a commercial gesture keyboard, which did not give us access to the recognition results. We treated the gesture keyboard as a “black box” that recognizes the word output, and captured the gesture data using Android Debug Bridge (ADB), while the gesture keyboard we used was running live as users gesture type with EXPRESSIVE KEYBOARD.

The touch events are captured by Android Debug Bridge using the `adb shell getevent` command, running on a desktop computer connected to the device via a USB cable. ADB captures all the touch events and display all the information on the terminal. I wrote a simple `shell script` that forwards the ADB’s output to a `python` program (`process.py`). The `python` program extracts the gesture data, which is a series of touch event in tuples: (`x`, `y`, `timestamp`) where `x` and `y` are the touch spatial location relative to the screen (i.e. `x`-axis and `y`-axis) in pixels, and the `timestamp` is the time of the touch occurrence in milliseconds. When a gesture is complete, i.e. from when the user starts gesture-typing until she lifts her finger from the keyboard, the `python` program generates a text file (`gesture.txt`) containing the gesture data. The gesture data is then sent back to the device and post-processed by a foreground Android application.

The foreground application observes user’s typing activities through a text field. As soon as the user finishes gesture-typing a word, it reads the file `gesture.txt` in the device and stores the gesture data within the application. The foreground application then calculates the gesture features, maps them to output properties, and renders the output on the screen canvas.

This implementation has three issues. First, as the foreground application has to rely on the ADB terminal to capture the gesture data, there is a delay from after the user finishes gesture-typing until the gesture data is pushed into the device. From the pilot study, the foreground application has to wait at least 0.5 second to ensure the gesture data has been complete and correctly pushed into the device. While this was not a big problem for our experiments, this situation is not ideal for real use. Second, since the process requires the ADB to push a text file (`gesture.txt`) into the device, the Android device must be rooted. These two issues make it impossible for us to evaluate EXPRESSIVE KEYBOARD in the wild. Last, the whole process of capturing-and-pushing the gesture data cannot be done progressively, since it would be a too-heavy process for the device. The users can only see the final output of

EXPRESSIVE KEYBOARD after they release their finger. The last issue appears to be quite a problem, as we found out later in Section 5.1.

4.3.2 REVISITED IMPLEMENTATION

Later, I collaborated with Xiao Jun Bi, an Assistant Professor in Stony Brook University, New York, USA. He provided us with a gesture keyboard prototype that implements an improved version of the original gesture keyboard algorithm, SHARK2 [42]. The recognition algorithm is similar to an early version of KB-1 we used in the experiment on deliberate gesture variation (Section 3.2). The keyboard prototype uses Google Protocol Buffer^{*} to communicate data related to the gesture keyboard: the gesture data, the list of word candidates from the recognition engine, and the position of each key in the keyboard. A foreground Android application is needed to receive and process the data.

The new keyboard prototype allows us to run EXPRESSIVE KEYBOARD live, without having to rely on ADB. This enables us to do the gesture feature calculation and the word recognition *at the same time*. The users can also use it anytime, anywhere. They only need to install both the keyboard prototype and the foreground application to use EXPRESSIVE KEYBOARD. This solves all three issues with the early implementation of EXPRESSIVE KEYBOARD.

Furthermore, the new EXPRESSIVE KEYBOARD prototype also includes progressive feedback. As soon as a user puts her finger on the gesture keyboard and start drawing, the recognition engine suggests a most likely word as the output. Thus, the recognition is done progressively, and generates a set of intermediate word output. For example, when writing ‘hello’, the intermediate word output includes ‘he’, ‘hell’, and finally ‘hello’. The gesture feature extraction is also done progressively, and the user also sees the intermediate output properties (e.g. color) rendered on the intermediate word output. With the progressive feedback, the users can see the generated output properties on-the-go, thus it may ease the gesture variation manipulation. For example, a user wants to write a yellow ‘hello’ i.e. she must make the gesture inflated (red) and curvy (green). During the intermediate result, she only gets e.g. a green ‘he’. Thus, she understands that she must inflate the gesture more to have an orange ‘hello’ at the end.

^{*}<https://developers.google.com/protocol-buffers/>

4.4 SUMMARY

EXPRESSIVE KEYBOARD transforms the rich variation that already exists in users' gesture-typing input into rich output, including dynamic font, emoticon with subtle expression change, and animation. Users can gesture type just exactly as they do with a standard gesture keyboard, yet with EXPRESSIVE KEYBOARD more information are included in the output. Simply mapping unintentional variation to small differences in the rendered output would generate a degree of expression that could be recognizable as personal styles or contextual information. A user may not specifically control their gesture when writing on a bumpy bus, yet the output can reflect the writing process – something that handwriting can capture easily. Conscious control of this variation would not be required to implicitly communicate style, personality, context, or mood. EXPRESSIVE KEYBOARD demonstrates how we can increase the expressive power of mobile communication system by leveraging the otherwise-unused gesture variability when gesture typing.

In real usage, users should be able to vary the sensitivity of the output variation or turn it off completely [6], especially in cases where the user prefers to generate more formal output. Users should also be able to (re)design their own feature detectors, text-rendering properties, and mappings linking the two. Users may also want to store a particular style that they generate earlier for future use.

I believe that deliberate variation is more interesting, as users can use and appropriate it to intentionally convey more information when communicating with other people. Just as they do with pen and paper, people can intentionally change the color or the size of their handwriting to attach different meaning when taking notes, e.g. black is for normal content while red is for comments. In the next chapter, I will investigate the intentional-control aspect of EXPRESSIVE KEYBOARD, i.e. how users learn to intentionally vary aspects of their gesture in order to generate a specific output, and in what context such case may be useful.

5

Learnability and Appropriability

Ideally, gesture-based interaction should be easy to learn and control, in order to offer an efficient alternative input channel [4]. EXPRESSIVE KEYBOARD leverages the existing gesture variations in gesture-typing input to generate rich output. Users can gesture type just like they usually do, but they generate rich output that potentially carries additional information about e.g. personal style or the current context. That said, EXPRESSIVE KEYBOARD can be more powerful when users can intentionally control the granularity of their gesture variability, to deliberately express intent. In this chapter, I describe an experiment that investigates whether novice users could selectively control aspects of their gestures to generate desired output with EXPRESSIVE KEYBOARD during the initial learning phase. The data analysis deepens our understanding of how novice users learn to control their gesture characteristics, which led us to design an informal observational study to see what factors may affect the learning strategies. Afterwards, I introduce Experiment 3, that investigates the appropriability aspects of EXPRESSIVE KEYBOARD in an ecologically-valid context.

5.1 EXPERIMENT 2: LEARNING TO CONTROL GESTURE VARIATIONS

The goal of EXPRESSIVE KEYBOARD is to build an expressive channel between users and the system, to increase the information transfer with an instrument that enables users to express themselves through unintentional variations (e.g. situation, personal style) and through in-

tentional control. In [Chapter 3](#), I show that users can vary their gestures deliberately under different conditions. I want to understand to what extent users can control the variability when they use EXPRESSIVE KEYBOARD for the first time, and how deliberately modifying the gesture's shape and dynamics affects accuracy. Additionally, I am also interested to investigate what factors may affect the process of learning to control the variations.

5.1.1 DESCRIPTION

We designed an experiment to explore the “initial learning” phase of controlling gesture variations. The goal of the experiment is to investigate whether novice users can explicitly control both their movement (gesture input) and the final result (gesture output), while maintaining the recognition accuracy. When a novice user immediately gesture type with EXPRESSIVE KEYBOARD, what is the more suitable instruction, to tell her what to do (i.e. the action) or what output properties to generate (i.e. the perceived goal)? How proficiently can users control the aspect of their gestures? Can users produce desired output properties and produce the same (or different) output properties if they are asked to repeat the task?

For this experiment, we chose RGB color as the output parameter space, since it is continuous, easy to quantify, and has relatively unambiguous semantics – most people agree on the meaning of the descriptor “red” as opposed to e.g. “messy”.

The experiment consists of two blocks. Block 1 is a [2x3] within-participants design with two primary factors:

F1 FOCUS

I *input*

O *output*

F2 LEVEL OF RELIABILITY

C *consistent*

D *different*

V *varied*

The factor FOCUS represents the type of instruction given to the participants, whether we ask them to focus on how they should control their *input* or on the resulting *output* properties.

The factor LEVEL OF RELIABILITY represents the level of performance reliability when we ask them to replicate the same instruction.

Participants are asked to gesture type phrases either by controlling a particular characteristic of their *input*, e.g. “Go outside of the keyboard”; or by controlling a characteristic of their *output*, e.g. “Try to make each phrase the same color of green”. For each type of FOCUS, participants are asked to draw phrases that are *consistent*, *different*, or *varied*, as shown in Table 5.1. As such, each trial condition has *two goals*: a specific *color* and different *levels of reliability*. For example, the two goals of *varied-output* condition “Make each phase include at least two colors; use as many colors as you can” are 1) to make the *blue* component high, which can be accomplished by speeding up or slowing down; and 2) to vary other color component (green and/or red) to create varied colors within each successive phrase. We set two goals for each condition, one focuses on *controllability* and the other on *reliability* so that we can get two different categories of “proficiency”. Regarding *reliability*, we hypothesize that repeatedly producing the same color *consistently* is arguably more difficult than *differently*, and *varied* will be relatively the easiest.

Additionally, we are also interested if users can deliberately change the way they gesture type when they write for different recipients or to express different emotional states. Block 2 is a one factor within-participants design with two levels: *message recipient* and *sender emotion*. The task is open-ended: participants can choose how to interpret these instructions and make their own mappings between the instruction and the resulting variation in the output. Participants gesture-type three phrases, to three different recipients or to express three different emotions, as illustrated in Table 5.2. For example: “Draw the phrase for your partner” or “Express how you feel: happy”.

5.1.2 PARTICIPANTS

We recruited five right-handed men and seven women (mean age 27). All use mobile phones daily. Four use gesture-typing daily, the others are non-users. No participants had participated in Experiment 1.

5.1.3 APPARATUS

We used the same LG Nexus 5 (Android 5.1) smartphone as in Experiment 1, running EXPRESSIVE KEYBOARD that maps the gesture features to RGB colors (as described in Section 4.2.1).

Experiment 2: Block 1			
FOCUS	LEVEL OF RELIABILITY		
	<i>consistent</i>	<i>different</i>	<i>varied</i>
<i>output</i>	Make each phrase the same black color.	Make each phrase a different shade of green.	Make each phrase include at least two colors.
<i>goals:</i>	1) red < 70 and green < 70; 2) difference of all components < 20	1) green > 60 and green > red; 2) difference(green) > 20	1) blue > 100; 2) difference of any component > 20
<i>input</i>	Scribble on each letter.	Draw at different speeds.	Draw outside the keyboard.
<i>goals:</i>	1) green > 100 and blue > 100; 2) difference(green) < 20 and difference(blue) < 20	1) blue > 150; 2) difference(blue) > 20	1) red > 100; 2) difference of any component > 20

Table 5.1: Experiment 2, Block 1: All instructions that vary according to FOCUS (*output* or *input*) and LEVEL OF RELIABILITY (*consistent*, *different*, or *varied*). Each condition has two goals, representing two categories of proficiency. The goal is measured through the value of the RGB component.

Experiment 2: Block 2	
Recipient	Draw the phrase for your partner Draw the phrase for your boss Draw the phrase for a stranger Draw the phrase for your niece Draw the phrase for your best friend Draw the phrase for your parents
Emotion	Express how you feel: happy Express how you feel: angry Express how you feel: frustrated Express how you feel: sad Express how you feel: busy Express how you feel: bored

Table 5.2: Experiment 2, Block 2: All instructions based on message recipient or sender emotion, replicated three times.

We chose KB-2 for the gesture-typing recognizer since it allows drawing outside the keyboard area.

5.1.4 PROCEDURE

Sessions last from 30-60 minutes. Participants sit in a chair and hold the phone comfortably in their left hand, so they can perform all gestures with their right index finger. Participants are encouraged to talk aloud as they draw each word. They are asked to practice until they feel comfortable with the recognizer and can reliably produce different colors.

Each trial displays an instruction at the top of the screen, e.g. “Try to make each phrase a different shade of green”; with a three- or four-word phrase centered below. Participants gesture-type three phrases in succession, on three separate lines, according to the condition (see Figure 5.3). For example, the *varied-output* condition “Make each phrase include at least two colors; use as many colors as you can” is accomplished by speeding up or slowing down to create varied colors within each successive phrase.

Phrases were chosen randomly from MacKenzie and Soukoreff’s three- or four-word phrase sets [53]. In both blocks, participants may write the phrases as often as they like, before pressing ‘next’ to submit the current result and move to the next trial. After each condition, participants used a five-point Likert-style scale to rate how their output compared to their expectations.

The complete experiment consists of 30 trials: Block 1 includes 18 trials (2 FOCUS x 3 LEVEL OF RELIABILITY x 3 replications); and Block 2 includes 12 trials (2 INSTRUCTIONS x 6 replications). Trials are counter-balanced within each block and across participants using a Latin Square.

At the end of the experiment, participants are asked to explain how the system generated colors in block 1; how they generated variations when asked to write to a particular recipient; and how they expressed specific emotions in Block 2. We intentionally did not inform participants how the mappings work to see if users are able to implicitly control their gesture variations in order to generate desirable output just by using the system.

5.1.5 DATA COLLECTION

In addition to touch events, we record values for `CorrectRate`, `WordAccuracy` and `FeatureAccuracy`. `WordAccuracy` is when the gesture produced the correct word; and `FeatureAccuracy` is when the gesture produced both the correct word and correct output properties. We also measure inflation, curviness, and speed consistency ratio. To reduce noise

caused by dependencies in word characteristics, and to increase variation in general, we average each measure progressively throughout each phrase.

CorrectRate (0–2) measures the participant’s success of fulfilling both goals in each condition. We defined a threshold value for each condition based on a pilot test and results from Experiment 1. For example, in the *varied–output* condition, a successful trial has 1) a high blue value ($> 100/255$); and 2) at least one other RGB color component in RGB that differs from the other phrases (difference $\geq 20/255$). Fulfilling both goals results in **CorrectRate**=2, whereas fulfilling only one results in **CorrectRate**=1.

We count number of errors based on how many times the participant erased a word before submitting a results. We record the screen and audio throughout to capture verbal comments.

5.1.6 RESULTS

Participants were able to control the variation in their gestures (overall **CorrectRate** is 1.3 out of 2.0): 75% of the trials met at least one goal of each condition. Of those successful trials, about two third of them (51%) met both goals of each condition. Of the 24% trials that only met one goal, there is no significant difference between succeeding controlling the *color* and *reliability* (48% v.s. 52% respectively).

An ANOVA showed that both **FOCUS** and **LEVEL OF RELIABILITY** significantly affect **CorrectRate** ($F_{1,11}=18.5$ and $F_{2,22}=28.7$ respectively, all $p<.0001$). Participants achieve significantly higher success rates when they focused on the output (1.4) than when focusing on the in-

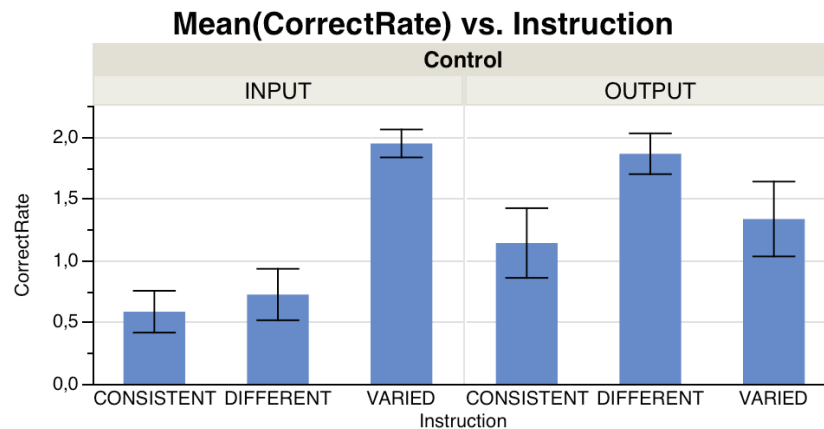


Figure 5.1: Participants are significantly more likely to control gestures based on output than on input.

put (1.1) (Figure 5.1). A post-hoc test with Tukey HSD also showed a significant interaction between FOCUS and LEVEL OF RELIABILITY ($F_{1,11}=37.5$, $p < .0001$). Participants were most successful in the *varied-input* and *different-output* conditions (1.9 and 1.86 respectively) which are both significantly different from the others. The least successful conditions were the *consistent-input* and *different-input* (0.6 and 0.7 respectively).

VARIATION IN GESTURE FEATURES

A post-hoc analysis with Tukey HSD revealed a significant interaction between FOCUS and LEVEL OF RELIABILITY for all three gesture features: curviness, inflation ratio, and speed consistency. Participants were clearly able to control the inflation ratio (the size of the bounding box), as in the *varied-input* and *varied-output* conditions, but otherwise chose not to. Participants drew curvier gestures when in the *consistent-input* and *different-output* conditions and significantly more straight gestures in *consistent-output*. However, their natural inclination is to draw curvy gestures with non-constant speed. Explicitly controlling speed consistency appears to be more difficult.

The post-questionnaire on expectation-match level reveals that overall, 70% of the tasks matched their expectations (41% strongly satisfied, 18% neutral, and 12% dissatisfied). Based on the measurement criteria, 76.4% of the tasks successfully fulfilled at least one condition (50% fulfilled both).

ACCURACY

The overall **WordAccuracy** is 81.8%, which suggests that the participants are able to control both their input and output while retaining reasonable accuracy as compared to baselines from Experiment 1. Overall **FeatureAccuracy** is lower at 46.3% (56.6% of correct words), which suggests that the participants sometimes re-wrote correctly recognized words to modify their output properties. They erased an average 4.4 words before completing a trial: an average 1.5 words due to incorrect words (i.e. typing mistakes) and 2.9 due to incorrect features. However, we found that the participants erased more incorrect words during the first trial replication (mean 2.3 vs. 1 word for the last trial replication). They also erased more correct words to modify the output properties in the first trial replication (mean 4 vs. 2). If we assume that the first and second replications are “practice”, since the users never used EXPRESSIVE KEYBOARD prior to the experiment, the **WordAccuracy** increases to 85.4% and

the FeatureAccuracy to 54.8%. Clearly, the learning curve (from first to third replication) had not reached a stabilized performance.

Both FOCUS and LEVEL OF RELIABILITY significantly affect the number of errors. Participants in conditions that focused on output made significantly more errors than in conditions focused on input ($7.2 > 2.0$, $F_{1,11}=27$, $p<.0001$). Participants also erased significantly more often when trying to be consistent than when trying to be different ($6.5 > 2.9$, $F_{2,22}=4.3$, $p<.0001$).

CONVEYING EMOTION & WRITING FOR DIFFERENT RECIPIENTS

Participants varied their gestures when expressing certain emotions or when writing to different recipients. Participants used different strategies: six deliberately varied their gesture input; five varied their gesture output; and only one participant varied both.

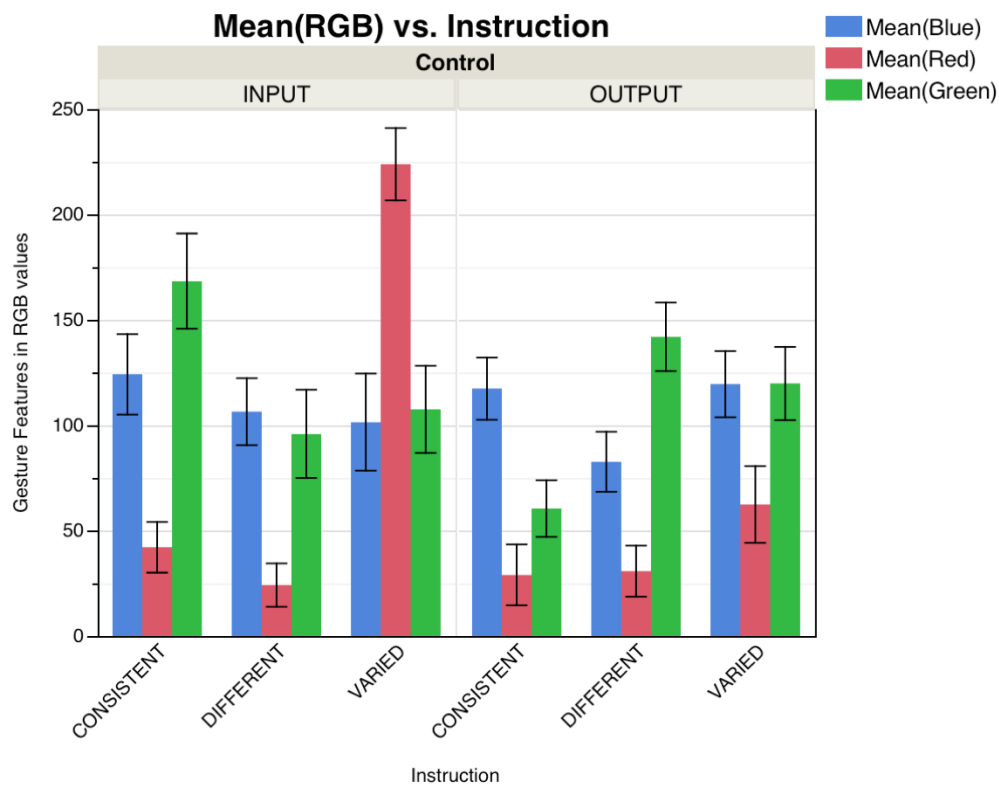


Figure 5.2: Participants can intentionally control gesture size when asked (inflation ratio), but do not vary it otherwise.

P2-F-00 a Try to make each phrase a different shade of green breathing is difficult	P9-N-IV b Go outside of the keyboard time to go shopping	P8-N-HP2 c Draw for 1) your niece 2) your best friend, 3) your parent earth quakes are predictable	P4-N-HE1 d Express how you feel 1) happy, 2) frustrated, 3) sad we went grocery shopping
breathing is difficult	time to go shopping	earth quakes are predictable	we went grocery shopping
breathing is difficult	time to go shopping	earth quakes are predictable	we went grocery shopping
breathing is difficult	time to go shopping	earth quakes are predictable	we went grocery shopping

Figure 5.3: Successful control of color through gesture: a) [*instruction: Different-Output*] bright-green indicates curviness, dark green indicates straight lines; b) [*instruction: Varied-Input*] inflating the gesture increased red values; c) [*instruction: recipients*] P8 changed the color deliberately for different recipients; d) [*instruction: express emotion*] P4 made curvier gestures with more detours for ‘happy’, and slow and less curvy gestures for ‘sad’.

When the hypothetical recipient was their boss or their parent, five participants reported they wanted to make the text color darker, and thus wrote more slowly and accurately. In contrast, when writing to a close friend or child relative, seven participants said they drew more slowly, with curvier gestures and detours, resulting in brighter colors (Figure 5.3).

Three participants chose pink or red to write to their partners; P9 drew a heart shape that left the keyboard area. Participants expressed negative emotions using slower, straighter gestures, resulting in darker colors. Four participants associated ‘angry’ with greater speed and most expressed being busy by drawing faster, curvier gestures. Only one participant stated that they did not change their style of gesture-typing according to her emotion or the recipient of her message.

5.1.7 DISCUSSION

Controlling Aspects of Gestures In this experiment, we introduced two levels of proficiency (i.e. goals) to see how far users could control their gesture variation just by using it. 75% of the trials met at least one goal of each condition, and of those successful trials, about two third of them (51%) met both goals of each condition. Given that all users tried EXPRESSIVE KEYBOARD for the first time and we did not inform them in advance how the mapping worked, we were surprised at how well they were able to control the aspects of their gestures. Perhaps, focusing on one goal (instead of two such as in this experiment) would be easier for them during the initial-learning phase.

Focusing on Gesture Characteristics v.s. Output Properties Participants performed better when instructed to focus on the output of their gestures i.e. to explicitly control the color of a phrase, than to focus on the characteristics of their input, i.e. to control the curviness, size and speed of their gestures. Some participants took advantage of the variations in the

color feedback to reflect upon and modify their performance, which was not possible for the *input* conditions. We observed that participants faced a trade-off during the first time using EXPRESSIVE KEYBOARD: they expended more effort rewriting the phrase until they got the desirable colors, but in the end were more successful in fulfilling the instructions. This suggests that perhaps it is more suitable to tell people the goal (of what they must produce) and let them explore during the initial learning phase.

Ongoing Learning Process Participants were able to deliberately modify their gesture characteristics while maintaining accuracy. However, they sometimes erase correct words to modify the output properties. Clearly, the participants need more time to practice, since most participants were using gesture-typing for the first time. While both types of accuracy must be improved, we believe this is a promising start: novice users are unlikely to have a clear understanding of how gesture recognition algorithms work, but this should not prevent them from generating the desired output properties by varying their gestures.

We also suspect that the recognizer affected participants' behavior when gesture-typing. For example, although participants had no trouble understanding the CI (consistent-input) 'scribble' instruction, it had the lowest correct rate and half of the participants stated that it was not easy to perform. In the course of the experiment, we observed that scribbling sometimes caused a recognition error that forced participants to repeat or change their gesture input strategy. However, considering that we were using a gesture recognizer designed for a different purpose, with no special modifications, the error rates were surprisingly low.

Manipulating spatial features is easier than temporal features Participants demonstrated that they are capable of drawing certain types of gestures, e.g. extremely curvy or large gestures, even if they do not choose to do so when typing without rich output. By contrast, participants have difficulty maintaining a constant gesture-typing speed. Gestures typically start quickly and then slow down, which with our mapping produces a color gradient in the *varied-output* condition. This should be a particularly simple gesture to control, but interestingly, only three found it easy and none could articulate how it works.

To summarize, despite being able to control the gesture features to generate desirable output properties, the participants had to try several times to achieve the goals.

5.2 LEARNING STRATEGIES

In Experiment 2, the participants could only see the resulting color after they finished gesture-typing. I observed two strategies that users used to produce the desired output:

1. *Trial-and-error with brute force*

Users just draw a gesture without a pre-planning and see what the colour it produces. If they do not like it, they erase the (correctly-recognized) output and re-draw. This is perhaps more likely to happen when they do not know the mapping in advance, but it may also happen even when they do.

2. *Pre-planning and post-planning*

If they already know the mapping, they normally do a pre-planning: they first plan how to generate the desired output properties and then how to perform the gesture before actually drawing the gesture.

Adding a progressive feedback to EXPRESSIVE KEYBOARD may help users to build their mental model of how the mapping works, and thus will improve their performance. I worked together with Chengcheng Qu, a Master student working with Wendy Mackay, to run an informal observational study focusing on how to see how advance knowledge of the mapping and different feedback mechanism affect the learning strategies. I implemented a progressive feedback on EXPRESSIVE KEYBOARD, as described in Section 4.3.2. With the new EXPRESSIVE KEYBOARD prototype, the users can see the intermediate word output along with the intermediate output properties such as colors.

We recruited two right-handed men. All use mobile phones daily. None of them use gesture typing daily, but they are familiar with gesture keyboards. We informed participant P2 about how the mapping worked, but not P1. We used the same LG Nexus 5 (Android 5.1) smartphone as in Experiment 1, running EXPRESSIVE KEYBOARD prototypes with and without progressive feedback that map the gesture features to RGB colors (as described in Section 4.2.1).

PROCEDURE

The study lasts approximately 45 minutes. Participants sit in a chair and hold the phone in their left hands, so they can gesture type with their right index finger. We encourage the participants to talk aloud as they gesture type with EXPRESSIVE KEYBOARD, emphasizing

the strategies they use to generate the desirable colors. Before the observation starts, we let them practice with EXPRESSIVE KEYBOARD (with or without progressive feedback) until they are comfortable.

We ask the participants to compose two text messages for each feedback type:

T1 [User-Generated] Write a sentence to your crush asking him/her out for a dinner.

T2 [User-Generated] Write a sentence to your supervisor asking to schedule a meeting.

Participants always start with task T1 with first EXPRESSIVE KEYBOARD with feedback at the end and then EXPRESSIVE KEYBOARD with progressive feedback, before moving on to the next task. In total, each participant finishes six trials (2 task x 2 feedback mechanism).

We count how many times they delete a word output and rewrite it, despite of whether it is a recognition error (i.e. incorrect word) or feature error (i.e. incorrect color). We use the screen capture to record all of the typing activities and audio record the participant's verbal comments.

PHENOMENON 1: PRE-PLANNING AND POST-PLANNING STRATEGY

In general, we observed that both participants used the *pre- and post-planning* strategy for any feedback mechanism. Participant P1, who did not know how the mapping worked, started with the *trial-and-error* strategy in the practice session. After trying several times, he started to build an ad hoc interpretation of how the mapping worked, and took it as a consideration when writing the next word. In the end, P1 switched to the *pre- and post-planning* strategy. Meanwhile, participant P2, who knew in advance how the mapping worked, went directly to the *pre- and post-planning* strategy.

Both participants claimed that they did not change their strategies when switching from EXPRESSIVE KEYBOARD with feedback at the end to the one with progressive feedback. However, the experimenter observed that they actually changed their strategies during the gesture execution. If, in the middle of execution, they already see that the color is not going to turn into what they want in the end, they halt the gesture execution by releasing their fingers, erase the wrong output, and then re-execute plan more carefully. For example, a participant wants yellow-colored "hello", it means the red and green components should be high. A high red component maps to an inflated gesture, and a high green component maps to a very curvy gesture. He now has established a plan: gesture type "hello" with consistent speed, while making the gesture inflated and curvy. As the EXPRESSIVE KEYBOARD (with progressive

feedback) is showing the intermediate output, he sees a green “hell”. He realizes that it is too late to inflate the gesture at that point, so he just lifts his finger. He erases the green “hell”, and starts over. If the final result is an orange “hello”, he realizes that he did not make the gesture curvy enough. Finally, he starts over, carefully keeping the speed consistent while inflating a very-curvy gesture.

PHENOMENON 2: ADVANCE KNOWLEDGE AND PROGRESSIVE FEEDBACK IMPROVE CONFIDENCE

We observed that P₂ who knew the mapping in advance was more encouraged to form expectations on what color he wanted to generate. In contrast, P₁ did not set any specific color expectation when using EXPRESSIVE KEYBOARD with feedback at the end – he only mentioned that he wanted to make all words the same color when we asked him to compose a sentence to his supervisor. This is perhaps because P₂ felt like he had controls over the output more than P₁, which then encouraged him to be more creative.

However, when P₁ switched to using EXPRESSIVE KEYBOARD with progressive feedback, he started to set a color that he wanted to generate. P₁ erased more times when using EXPRESSIVE KEYBOARD with the progressive feedback: 12 v.s. 23 out of 14 words for without and with progressive feedback respectively. When asked, he mentioned it was because he understood better how to generate the color he liked and thus more motivated to try. Both participants reported that they were more confident in controlling the color with the progressive feedback. This phenomenon may suggest that the more participants feel they “understand” how to control the output generation, the more they are encouraged to put more effort into learning the system. Although the participants in Experiment 2 could still generate the desired output properties, informing them about the mapping may affect their learning process.

In summary, our observation discovers two interesting phenomena. First, advance knowledge of the mapping may not affect user strategy when using EXPRESSIVE KEYBOARD, since the participants tried to follow a mental model of how the mapping worked. Second, knowing the mapping in advance may cause users to feel more confident to build expectation. This may also inspire the users to use the colors to express different concepts more.

5.3 EXPERIMENT 3: ECOLOGICAL VALIDITY

Writing is a complex form of interaction that involves motor abilities, context, and other cognitive aspects of human behaviour. Writing performance or experience may change depending on the context of use [64, 74]. The degree of expression may also change depending on the text the users are composing, thus composition task in a more realistic setting can capture more insights than standard transcription task [74]. Likewise, EXPRESSIVE KEYBOARD is designed to encourage in-context appropriation, i.e. users will invent new ways of using EXPRESSIVE KEYBOARD as they use it in real-life context. Hence, I am interested to investigate how users choose to use EXPRESSIVE KEYBOARD to fit their preference when they are composing a text message while still learning how to use the system in a more realistic context, i.e. when a user is chatting with his/her friend. I also want to compare how using EXPRESSIVE KEYBOARD affects gesture typing behaviour.

5.3.1 DESCRIPTION

We designed a third experiment to explore how EXPRESSIVE KEYBOARD is used in a more ecologically-valid setting, e.g. chatting with friends. The experiment is divided into three sessions:

Chat We set up a live conversation between a pair of participants. We specifically chose participants who knew each other, and asked each pair to text message each other using a chat application, so that the participants understand better about the content as well as the context. We want to observe if it encourages them to produce rich output and systematically compare it with when they write a prescribed text (later in the *simulation* session).

Simulation We asked each participant (individually) to re-type five of their sentences using EXPRESSIVE KEYBOARD, to simulate the use of EXPRESSIVE KEYBOARD when the gesture variations are mapped to more complex features than color. The goals are to see 1) how first-time users appropriate the system as they learn to control it; and 2) if users are more likely to put more effort in learning to control their gesture variations if they write their own text. We also compare EXPRESSIVE KEYBOARD to a standard gesture keyboard, to see if participants change the way they gesture type if they get rich output.

Quiz We asked the participants to produce text with specific goals e.g. “bold text” using

EXPRESSIVE KEYBOARD, to see how much users learn to deliberately control their gesture variations to generate the desired output properties.

For the *simulation* session, we use a [2x2] repeated measures design with two factors:

F1 KEYBOARD TYPE: {*standard keyboard* v.s. EXPRESSIVE KEYBOARD}

F2 TEXT TYPE: {*user-generate* v.s. *prescribed*}

The *standard keyboard* is a standard gesture-typing keyboard. The EXPRESSIVE KEYBOARD generates a *dynamic font* whose shape and colors changes depending on the gesture features. Considering the previous informal observation, in this experiment we inform the participants about the mapping and provide a cheatsheet that they can take a look anytime during the experiment.

5.3.2 PARTICIPANTS

We recruited six pairs of friends, seven men and five women (age range 19-40, mean 25.4); all use mobile phones daily. Half of them use gesture-typing daily, the others are non-users. No participants had participated in the two previous experiments.

5.3.3 HARDWARE AND SOFTWARE

For the *chat* session, we customized an Android chat application^{*} to capture the gesture data as well as the typed words. For the *simulation* and *quiz* sessions, we developed a custom Android application that presents either the *standard keyboard* or an EXPRESSIVE KEYBOARD that renders the dynamic font as described in Section 4.2.2. We predefined a font with $n=2$; one typeface is more skewed (italic) than the other. Along with the dynamic font, we also used the color mapping as described in Section 4.2.1. For recognition, we used KB-2 on the same LG Nexus 5 (Android 5.1) smartphone as in the previous experiments.

5.3.4 PROCEDURE

Participants sit comfortably in a chair while gesture-typing. The experiment begins with the *chat* session, in which each pair of participants chat for 15 minutes without any restriction on

^{*}AndroidHive: <http://www.androidhive.info/>

how to gesture-type. Afterwards, we select five sentences from the chat as the *user-generate* text.

The *simulation* session consists of three blocks of five trials each. This is an individual task where the participant has to write five *user-generate* and five *prescribed* sentences (from news, blogs, etc). Participants are instructed to gesture-type as if writing to their peer and to assume the previous peer will see the same output. The session always starts with the *prescribed* text with *standard keyboard*, followed by an introduction to the EXPRESSIVE KEYBOARD and the mapping used. Participants are encouraged to practice to understand how the system works; no participant practiced longer than five minutes. For the next two blocks participants are asked to write both the *prescribed* and *user-generate* texts (counter-balanced across participants). We do not specifically tell them how to use EXPRESSIVE KEYBOARD and let them use it as they like. Throughout the session, we ask the participants to describe aloud what they want to do and what they are thinking.

The third session is a *quiz* (three blocks of three trials). We ask them to gesture-type “hello” three times with specific output goals: 1) bold and red, 2) italic and containing blue, 3) green. This is to confirm their success rate in controlling aspects of their gestures after using EXPRESSIVE KEYBOARD in a more realistic context. Finally, we interview them regarding their preference and how they might define their own features and mapping if they could. An experiment session last for 40 minutes.

5.3.5 DATA COLLECTION

We calculate the three gesture features (curviness, inflation ratio, and speed consistency ratio) as well as WordAccuracy and FeatureAccuracy as in Experiment 2. We log the timestamp and 2D coordinate of each touch event, and record the screen and audio to capture verbal comments.

5.3.6 RESULTS AND DISCUSSION

Out of 2312 performed gestures from the *simulation* session, we removed 183 that were not gesture-typed, most of which were single-letter words. Three of them were picked from the suggestion bar, five were words that could not be successfully recognized e.g. “folate” while the rest were single-letter words (e.g. “I”). For the *quiz* session, we collected 108 gestures and removed 2 outliers where the participants had lifted their finger at the start of the gesture.

KEYBOARD TYPE	TEXT TYPE	
	<i>user-generate</i>	<i>prescribed</i>
WordAccuracy		
<i>standard keyboard</i>	95%	75.4%
Expressive Keyboard	68.5%	79.5%
FeatureAccuracy		
Expressive Keyboard	55.9%	61.7%

Table 5.3: Accuracy in Experiment 3. Participants changed the way they gesture-type with different keyboards, and are more likely to explore with Expressive Keyboard. Although using Expressive Keyboard does not necessarily decrease word-accuracy, participants erased correct words to modify the output properties leading to low values for feature-accuracy.

STANDARD GESTURE-KEYBOARD VS. EXPRESSIVE KEYBOARD

Accuracy We ran an ANOVA test to compare performance of KEYBOARD TYPE and TEXT TYPE. WordAccuracy is significantly affected by KEYBOARD TYPE ($F_{1,11}=30.7$, $p<0.0001$) (see Table 5.3). There is a significant interaction between KEYBOARD TYPE and TEXT TYPE ($F_{1,11}=15$, $p=.0001$). However, further analysis with Tukey HSD showed that the only significant difference is found when the participants wrote *user-generate* text using *standard keyboard* compared to EXPRESSIVE KEYBOARD. There was no significant difference between *standard keyboard* and EXPRESSIVE KEYBOARD when writing a *prescribed* text, although EXPRESSIVE KEYBOARD's WordAccuracy is a bit higher (Table 5.3). This suggests that the use of the EXPRESSIVE KEYBOARD does not necessarily decrease WordAccuracy; instead the context when chatting may help the participants better match the design parameters of the recognizer.

However, with EXPRESSIVE KEYBOARD, there were cases in which the participants erased a correctly-recognized word to modify the output properties: 18.9% for *user-generate* and 23.3% for *prescribed*. The overall FeatureAccuracy, which represents accuracy of EXPRESSIVE KEYBOARD is shown in Table 5.3. The time spent to draw a gesture also significantly increased when using EXPRESSIVE KEYBOARD ($F_{1,11}=64.9$, $p<.0001$), mean 2 seconds per word, while they spent the least time when chatting (0.7spw).

Gesture Variability There is a significant effect of KEYBOARD TYPE on inflation and curviness ($F_{1,11}=30.6$, $p<.0001$ and $F_{1,11}=5.6$, $p<.0177$ respectively). The participants significantly inflated their gestures when using EXPRESSIVE KEYBOARD (mean 1.6) as compared to stan-

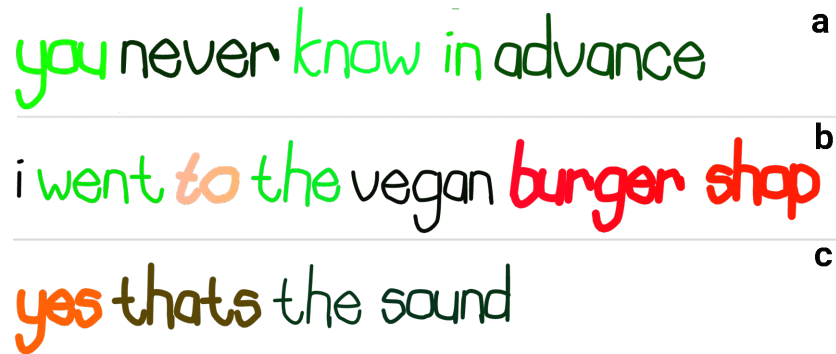


Figure 5.4: Using Expressive Keyboard in Experiment 3: a) P2 naturally made curvy gesture (green) but made straight gestures (dark green) to emphasize some words, b) P6 deliberately made two words (“burger shop”) the same shape and color, c) P3 emphasized the first word, but then deliberately made his gesture more precise (dark color) instead of curvy.

ard keyboard (mean 1.2). There is no significant difference with regards of the speed consistency.

Generating Rich Output From the post-questionnaire, we learned that the participants took advantage of the fact that they could change the output properties. All participants stated that they changed the way they gesture-typed with EXPRESSIVE KEYBOARD. Most used it to highlight a specific word or phrase in the sentence, rather than trying to control the appearance of every word. Two of them mentioned they changed the properties to match their intonation when reading the sentence. Three of them stated that they expressed their feelings or mood when writing, e.g. “when it’s something happy I tried to write faster [so] that text becomes green and blue” (P6).

An interesting use of EXPRESSIVE KEYBOARD is to reflect on their own gesture-typing habits. P1, P10, and P11 realized that they tend to make curvy gestures, and they deliberately let the output change according to their natural input style. On the contrary, P3 and P8 made a special effort to make the output text as similar as possible (e.g. all black). This suggests that continuous changes to output properties can provide important feedback for the participants and may change their behaviour when gesture-typing – not only to customize the output, but also to try to gesture-type more precisely.

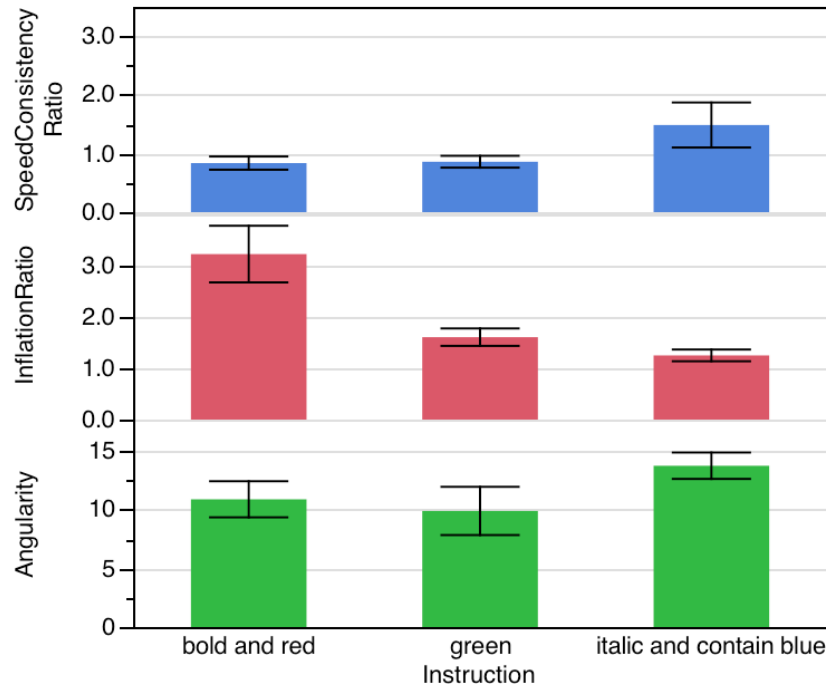


Figure 5.5: Participants successfully varied or kept constant the inflation and speed consistency when instructed. They naturally made curvy gestures, but could make them more curvy when instructed.

CONTROL OF GESTURE FEATURES

In the *quiz* section, the average *WordAccuracy* was 86%. Participants were most accurate when controlling speed consistency (33 out of 36 trials, or 92%) and curviness (32/36 or 89%), but less accurate when controlling inflation (26/34 or 76%). Further analysis revealed that the recognition error was caused by 1) too much deformation (6 out of 15 errors), 2) faulty start or end position (6/15), and 3) removing the finger too early (3/15). This suggests that while many factors affect recognition rate, certain types of intentional variation may increase error to a small extent.

The instruction had a significant effect on all the gesture features (all $p < .0001$). A post-hoc test with Tukey HSD showed a significant difference between the instruction “italic” and the others. In Figure 5.5, we can see that in general the participants speed-up (mean rate=1.5) with higher variability when asked to make the output italic; however both the value and the variability dropped for instructions which required them to keep the speed constant (0.86 and 0.84 for “bold” and “green” respectively).

Similar significant differences also appear for inflation rate. Pairwise t-tests with Tukey HSD showed that there is a significant difference between the instruction “bold” and the others. However, for “green”, the inflation rate is a bit higher (mean=1.6). Based on our observation, this is because the participants overshot when trying to make a curvy gesture.

Meanwhile, the gestures are quite curvy for all instructions: with means 10.9, 9.9, and 13.8 for “bold”, “green”, and “italic”, respectively (a value of 12 indicates minimum curviness). Gestures are most curvy when instructed implicitly, however, it is not significantly different from “bold”. This suggests that inflating the gesture also increased curviness. Overall, participants naturally made curvy gestures, but were able to increase curviness when necessary.

In summary, when using EXPRESSIVE KEYBOARD to write their own text, the participants were more likely to put effort in controlling how the output should look like. This significantly affects the word recognition accuracy, since the participants explored more to produce the output they liked. When they wrote prescribed texts, the word recognition accuracy was not affected. The context when chatting may help the participants type more fluidly and thus better match the design parameters of the recognizer. However, as we found in Experiment 2, participants also erased correct words to modify the output properties more with EXPRESSIVE KEYBOARD. They also spent more time to finish a trial when using EXPRESSIVE KEYBOARD. Since all participants were necessarily novice users, a more longitudinal study of EXPRESSIVE KEYBOARD is needed to determine whether accuracy improves with experience. Additional factors may also affect user performance when using EXPRESSIVE KEYBOARD, for example, in Experiment 3 six participants mentioned that they sometimes changed their hand position to increase the precision of their finger in gesturing, e.g. from using thumb to index finger.

5.4 IMPLICATION FOR DESIGNS

Revealing Past Writing Process We can use EXPRESSIVE KEYBOARD to reveal the past process of their own writing, either to enrich interpersonal communication or to improve typing accuracy. Just like in verbal communication, one may struggle in choosing the right diction to express what they really mean, for example by pausing or correcting herself when she is not sure. In text-based communication applications, this may be inferred by looking at the feedback of whether the sender is typing or not, e.g. a text feedback “*Alice is typing*” or a blinking (...) ellipsis. However, it is rather ambiguous since there can be many reasons why

the sender takes a while to type in the whole sentence. For example, we can change the color based on whether a word has been changed a lot or the sender hesitates during the writing process. From the receiver's point of view, it may communicate a subtle information. From the sender's point of view, it may also act as a "pre-send mechanism" that lets her check whether the final output is desirable or not. To push the idea even further, we can let users to save and appropriate their pattern for future use, such as suggesting a better diction alternatives or opting not to show a particular typing habit to the senders.

Inserting and Generating Emoticon Emoticons are often used to substitute or accompany text as a way to convey emotion expression [45]. In today's commercial chat applications, users usually choose one (or more) from a list of predefined emoticons. Alternatively users can also type a particular keyword, e.g., 'sad' to produce :(, draw a gesture [63], or search through an emoticon recommendation system [72]. EXPRESSIVE KEYBOARD can improve an emoticon recommendation system, for example, by sorting the suggestions based on the gesture characteristics. Additionally, the participants mentioned that they would like to generate a parametric emoticon. For example, they would like to control the color, the size, or the number of the emoticon by manipulating the gesture characteristics. We can also use EXPRESSIVE KEYBOARD in the context of text-to-speech generator, in which the spoken intonations change depending on how the words are gesture typed.

Invoking Commands We can use EXPRESSIVE KEYBOARD to select simple, discreet commands, such as mode switching during text editing. For example, normal gestures to write English words while inflated ones to write German words; slow gestures generate thick strokes while fast ones generate thin strokes; or inflating the gesture as a way to "tell" the keyboard not to auto-correct the word. We can broaden the idea to other context such as web searching, for example, normal gestures invoke a web search while inflated ones an image search. We can also interpret the gesture variations into urgency levels. For example, when writing an item in a to-do list, a fast gesture will automatically set the priority high. We can potentially include the urgency level into the notification sent to the receiver.

5.5 SUMMARY

Experiment 2 explored initial learning aspect of EXPRESSIVE KEYBOARD, in which the participants just used it without proper training. All of the participants were not daily users

of gesture keyboard – not to mention EXPRESSIVE KEYBOARD – yet, they were able to keep the gesture-typing accuracy (81.8%) at a reasonable level while using EXPRESSIVE KEYBOARD. However, the participants sometimes erased correctly-recognized words (overall 56.6% of correct words) to change its output properties. We believe that this is a promising start, since this experiment focuses on “immediate use” in which the participants tried using EXPRESSIVE KEYBOARD in about an hour and we did not inform users about how the mapping worked.

Besides, I would like to emphasize that we should not see EXPRESSIVE KEYBOARD as a keyboard that as a “color picker”. Obviously, compared to a standard color picker, users will be able to pick the color more accurately and efficiently than by manipulating their gesture characteristics. EXPRESSIVE KEYBOARD emphasizes the value of human variability that exists in gesture-typing input, instead of treating it as a deformation to a “correct” gesture input. The participants were able control aspects of their gestures to generate the desired color (success rate 75%), however the observed learning curve had not achieved a stabilized performance. A longitudinal study will be needed to see how users learn to control the aspects of their gestures over time (i.e. expert performance) and in which state the learning curve will reach a stabilized performance.

Experiment 2 highlights the importance of *feedback*, in this case, the color output, both to better reveal how the mapping between gesture characteristics and color components to the users, and to apprise users of their performance. The color may also act as an intrinsic reward that motivates users to better learn controlling their gestures. Hence, focusing on the color (i.e. output) led to be more successful in fulfilling the instructions. We also learned that knowing the mapping in advance may help the participants to gain more confidence when using EXPRESSIVE KEYBOARD for the first time, which then encouraged them to explore and do more things.

Experiment 3 explores the use of EXPRESSIVE KEYBOARD that emerged as the users gesture type in a more realistic context. We found that the users found it enjoyable to be able to generate rich output when they were communicating with their friends. They were willing to put more effort in learning to carefully control aspects of their gestures as they wrote. Experiment 3 also confirms that participants were able to selectively control curviness, speed consistency, and inflation when they want to, although they naturally draw curvy gestures and slow down at the end of their gesture.

EXPRESSIVE KEYBOARD successfully demonstrates how taking advantage of both machine learning and the richness of human’s motor control can increase the expressive power of gesture-based interfaces such as gesture keyboards. Users can gesture type naturally, yet

the output contains richer information. Users do not need to deliberately control their gestures to be able to implicitly communicate different things in their output. Nevertheless, users can also control their gesture deliberately, and reach a reasonable proficiency level after using EXPRESSIVE KEYBOARD, e.g. for less than an hour during the experiments. Once they used EXPRESSIVE KEYBOARD, they quickly thought of other possibilities of how they wanted to use it in their daily lives. This demonstrates how EXPRESSIVE KEYBOARD not only encourages but also inspires *users* to new ways of using the system to fit their needs, and possibly *designers* to “recycle” existing data in a “presumably” closed system such as gesture keyboards.

6

Invoking Complex Commands from Gesture Keyboards

In [Chapter 4](#), I demonstrate how we can increase the expressive power of a gesture keyboard by mapping the gesture variation into rich output. Users can gesture type as they usually do, but the output contains additional information that can be used to express context, mood, or personal style. They also can deliberately manipulate their gestures to communicate intent, e.g. sarcasm, or subtle emotion change. On the other hand, users often issue a command in between typing activities on their mobile devices. To issue a command, mobile devices rely on buttons, menus and dialog boxes, which restricts the available command set to what fits on a tiny screen. When the command set is big, the system often needs to hide the keyboard and reuses the space to display the menu. This causes inefficiency when switching back and forth from typing to issuing a command (see [Table 6.1](#)).

I am interested in designing a simple, yet powerful method of issuing commands from a mobile device, that enables a variety of alternatives depending upon the task, the user's cognitive and motor skills, and the size and structure of the current command space. Appert and Zhai argued that a soft keyboard takes the valuable screen space, and thus, a free form gesture-based command selections may be preferable [5]. However, most of mobile activities include typing: texting, searching, naming contacts or calendar items, taking notes, etc. Why not let users access a large command space from the keyboards?

This chapter describes my work that focuses on broadening the access to a large command space from a gesture keyboard. I start with analyzing the structure of the device screen space, to maximize the screen real-estate use. Then, I describe a novel interaction technique called `COMMANDBOARD`, which turns the space above the keyboard into a general-purpose command gesture space, to enable more sophisticated command generation. I then show how `COMMANDBOARD` leverages these to provide users with a variety of simple, yet powerful command invocation techniques.

6.1 REVISITING MOBILE DEVICE'S SCREEN REAL ESTATE

In traditional interactive displays, a soft keyboard usually appears at the bottom of the screen, taking almost half of the valuable screen space. The soft keyboard's output is linked to another display area on the screen – the output space. The two are linked, such that user actions on the soft keyboard produce output on the related display space. For example in a chat application, the soft keyboard appears at the bottom of the screen and the text output that is typed (or drawn) by the user appears within the display area above, to create a conversation (Figure 6.1). Mobile applications often include menus, e.g. tapping the top-right icon in the Message application shown in Figure 6.1 opens up the “Setting” menu. The menu view often appears on top of the current view, occluding the content and hiding the keyboard. Once they finish selecting the command, the users often have to explicitly tap on the text field to re-display the keyboard before continuing typing. Switching back and forth from typing view to menu view is cumbersome and may interrupt the cognitive process [67, 61, 43].

My goal is to provide a more powerful alternative command invocation technique that avoids this back-and-forth view switch. One possible solution is to let users invoke commands from the keyboard, like invoking keyboard shortcuts from a physical keyboard. Each screen space described in Figure 6.1 is a substrate that has a structure and a set of rules on how to interpret the data. The text input space, i.e. the keyboard view, uses the `QWERTY` layout as its structure, and interprets the gesture drawn on top of it as a word gesture. The suggestion bar is another substrate that displays word candidates produced by the keyboards and receives touch input such as a tap. The upper space displaying the content is also another substrate. A slide gesture done in the text input space is interpreted as a word gesture, while in output space it is interpreted as a scroll. We can redefine the screen space's substrates, so that all areas of the screen can be used to invoke commands without having to switch the view from keyboard to menu. I propose `COMMANDBOARD`, which adds contextual substrates

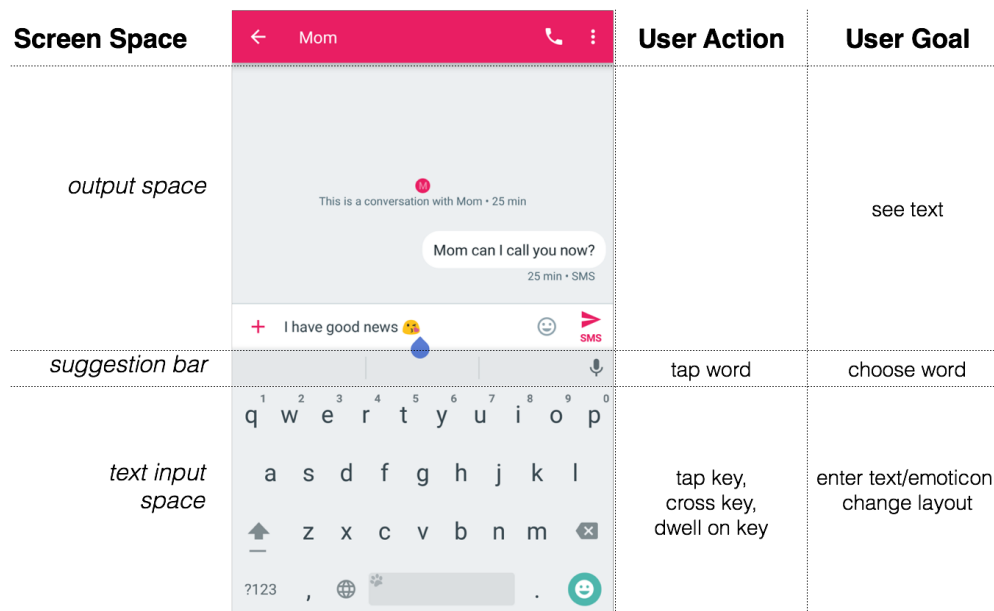


Figure 6.1: The screen real estate in the current mobile interfaces. Users type on the bottom (text input) space, choose word suggestions from the suggestion bar, and see the content in the upper (output) space. Each space is a *substrate* that has a specific structure and different way of interpreting touch input.

that interpret a wide variety of gestures drawn above the keyboard.

6.2 COMMANDBOARD

COMMANDBOARD extends the functionality of a soft keyboard to an efficient command-entry tool that provide access to a large command space. We build on a key insight from the gesture keyboard, i.e. that the system can recognize users' gestures as they cross over the keys, and interpret them as text. COMMANDBOARD generalizes this idea by creating an additional interaction layer, above the keyboard, for interpreting free-form gestures. We can think of this as extending a transparent interactive a substrate above the keyboard, where users can still see the usual display, but also issue gesture commands. This creates a general-purpose gesture command input space that supports a variety of command entry techniques.

COMMANDBOARD takes full advantage of the limited screen real estate on a smartphone. Figure 6.2 shows four discrete interactive substrates, extending the traditional interaction spaces described in Figure 6.1. As with gesture keyboard, the lower space is dedicated to generating text input or emoticons via tapping, crossing or dwelling on keys. Users can also

Table 6.1: Step comparison for CommandBoard and pulldown on touch screen devices from when the users switch from typing to command selection until switching back to typing.

Pulldown Menu	COMMANDBOARD'S INLINE GESTURE SHORTCUTS	COMMANDBOARD'S TYPE-AND-EXECUTE
1. tap the menu bar to display it 2. decide which menu has the command item 3. look for the menu name 4. tap on the menu name to display the command items 5. look for the command item 6. tap the command item — command is executed 7. tap the textfield to hide the menu bar and redisplay the keyboard — continue typing	1. gesture-type a word 2. go up to the upper space 3. perform the gesture to select the command item — command is executed — continue typing	1. gesture-type the command keyword 2. cross the command item in the command bar while going up to the upper space 3. perform the execute gesture — command is executed — continue typing

swap keyboards, e.g. numeric or emoticon. COMMANDBOARD includes additional features (marked in green), which let users specify command names for later execution via a gesture.

The gesture keyboard provides an optional *suggestion bar* in the middle where users tap to choose among suggested words. COMMANDBOARD offers a similar *command bar* that users cross to choose among suggested commands. Finally, COMMANDBOARD transforms the upper display area at the top of the screen into a *command-gesture input space* where users can draw an execute gesture to issue the current command name, or draw a unique command gesture. Since this overlay is transparent, users can see the underlying display, such as the current chat conversation.

COMMANDBOARD exists in harmony with existing command-generation techniques, such as menus and buttons, but also offer novices the opportunity to transition into power users, able to execute commands fluidly, at the tip of their fingers. I introduce two most basic tech-

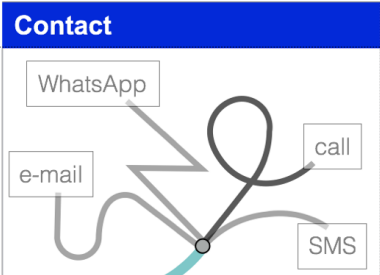
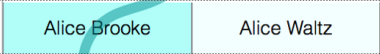


Screen Space	Contact	User Action	User Goal
<i>command-gesture input space</i>		draw gesture	execute command
<i>command bar</i>		cross command option	choose command option
<i>suggestion bar</i>		tap word	choose word
<i>text input space</i>		tap key, cross key, dwell on key	enter text/emoticon, change layout, specify command

Figure 6.2: CommandBoard specifies multiple command entry spaces. New methods (in green) include typing a command name followed by an EXECUTE gesture; crossing a suggested command in the *command bar*; or executing a unique gesture in the upper *command gesture input space*. An in-context dynamic guide shows gesture-command mappings.

niques: TYPE-AND-EXECUTE and INLINE GESTURE SHORTCUTS.

6.2.1 TYPE-AND-EXECUTE COMMANDS

Although novices may need to search through menus to discover the available commands, frequent users are usually familiar with both the commands and their names. Navigating through menus can be time-consuming, especially if the user forgets where the desired command is classified within a hierarchical menu. Some graphical user interfaces offer a *search bar* where typing the keyword or command name displays its location in a pull-down menu, if the command exists. Clicking on the search result issues the command, as if it had been selected from the menu. COMMANDBOARD offers a similar function by letting the user type any command name from the keyboard, and then execute it directly by drawing the execute gesture in the display area above the keyboard.

KEYWORD SEARCH Since COMMANDBOARD co-exists with traditional menus, we have prior knowledge about the current set of command names or *command-gesture input space*.

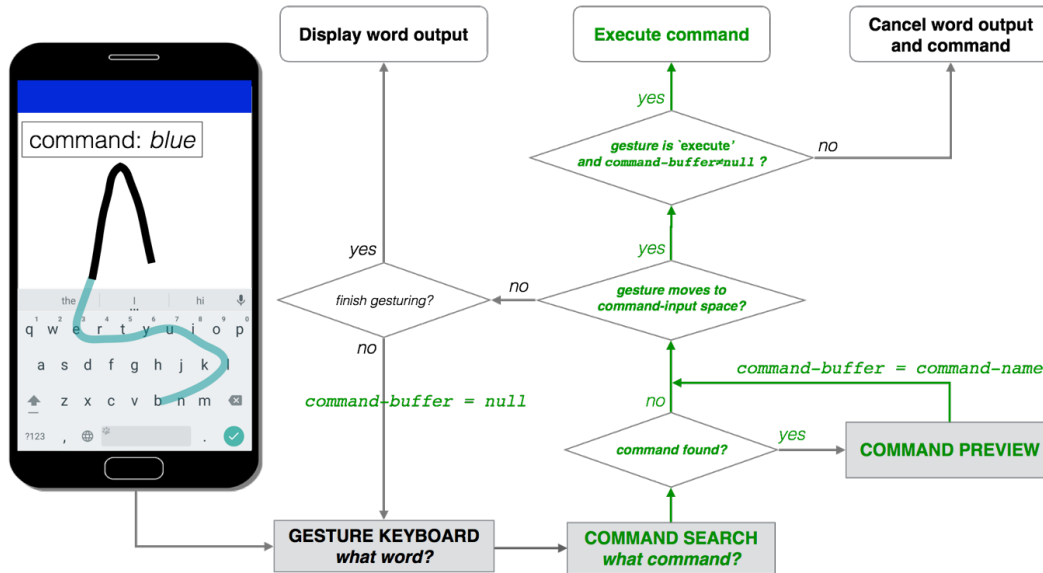


Figure 6.3: A diagram describing TYPE-AND-EXECUTE technique of CommandBoard. The text in green represents the new methods.

When a user gesture-types a word, COMMANDBOARD’s TYPE-AND-EXECUTE technique examines the first four words suggested by the keyboard recognition engine to see if any is a keyword in the command space. The TYPE-AND-EXECUTE technique treats each element of a compound command name as a search keyword. For example, both “line” and “spacing” can be used to find LINE SPACING. Users need only type the first unique letters of a long command name and the system will suggest the full command. For example, typing ’brightn’ produces the BRIGHTNESS command in the command bar, which can then be invoked by performing the execute (/) gesture (Figure 6.4b).

COMMAND PREVIEW If the keyword search is successful, the TYPE-AND-EXECUTE technique displays a preview: the full command name appears at the top of the screen. The keyboard continues to recognize the word-gesture as the user types. Thus, when the preview appears, the TYPE-AND-EXECUTE technique stores the keyword so that even if the recognized word changes as the user slides her finger upward, the command keyword remains the same. If the user continues gesture-typing, the preview disappears. If the user releases her finger within the keyboard space, the word appears as normal text.

COMMAND EXECUTION If, after typing a recognized command name, the user continues to slide her finger upward, she enables the command-gesture input space. If she now performs a \wedge gesture, the TYPE-AND-EXECUTE technique will execute the corresponding command. This allows the user to perform any command directly from the keyboard, as long as she already knows the command name. She need not learn any special commands beyond the execute gesture.

We designed the execute gesture specifically so that it would not interfere with the Google Keyboard's technique for cancelling gestures. The user cancels the current word by sliding her finger into the space above the keyboard and releasing it. By contrast, COMMANDBOARD's execute gesture is designed to move up and then down, explicitly change direction, to reduce the risk of issuing unintended commands.

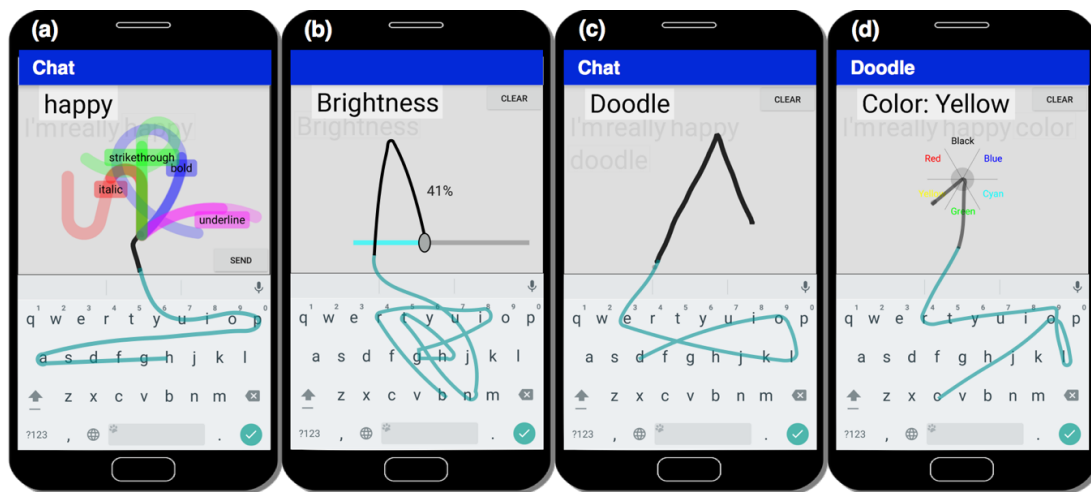


Figure 6.4: CommandBoard creates a new *command gesture input space* above a soft keyboard. Users can: a) type 'happy' and use a dynamic guide to style it as **bold**; b) type 'brightn', draw an execute gesture and adjust the brightness slider; c) type 'sans', choose 'sans mono' and draw an execute gesture to change the font; d) type 'color', select yellow in the marking menu to change the brush color.

I provide some applications of COMMANDBOARD's TYPE-AND-EXECUTE technique, to better illustrate how the users can use them in different context.

TEXT EDITOR APPLICATION

Most TEXT EDITOR applications for mobile devices, such as Google Docs, offer only a limited number of commands. The process is also cumbersome: Selecting a menu command

requires hiding the keyboard, navigating to and executing the command, then closing the menu and bringing back the keyboard, all before continuing to type.

The TYPE-AND-EXECUTE technique simplifies command selection for text editors. The user can type the name of any menu item, as if it were a search word, and then execute it directly. For example, Figure 6.4c shows the user typing the word ‘sans’, then sliding her finger above the keyboard to perform the execute gesture, at which point TYPE-AND-EXECUTE applies the SANS MONO font to the selected text.

COMMANDBOARD’s TYPE-AND-EXECUTE technique also lets users type sub-menu names, and display their items in the command bar located above the suggestion bar. This is particularly useful when the menu item cannot be typed, for example the numbers shown in Figure 6.5. The user types ‘line’ and a preview for the LINE SPACING sub-menu appears (Figure 6.5a). The menu items then appear on the command bar. She sets the LINE SPACING value to 1.2 by crossing through it in the command bar and then performing the execute gesture (Figure 6.5c).

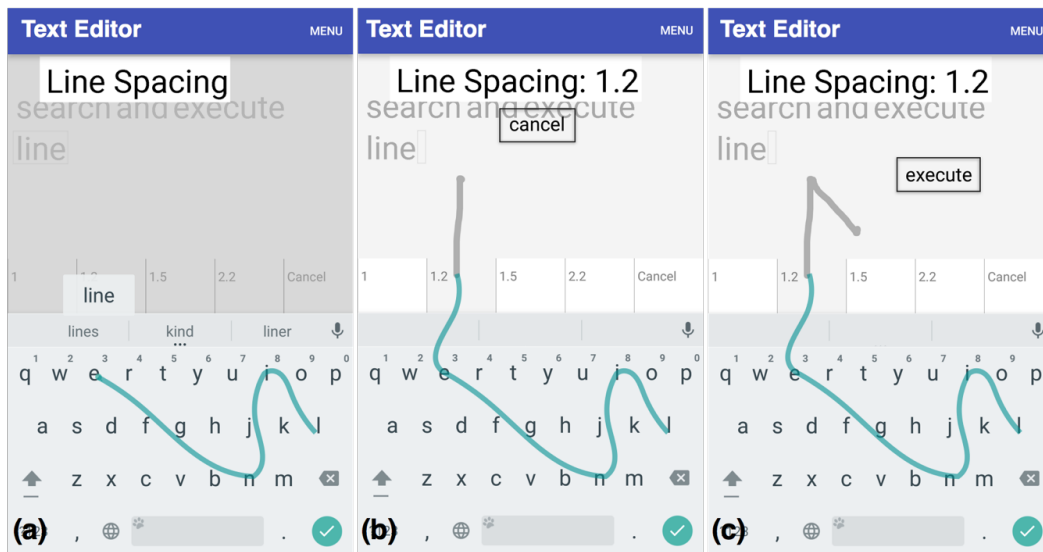


Figure 6.5: After typing an existing command name, a) a preview appears. Since “Line Spacing” is a sub-menu, its menu items appear in the command bar. After sliding across the command bar, b) performing a | gesture will cancel writing, whereas c) performing a ^ gesture will execute the command.

DOODLE APPLICATION

COMMANDBOARD's TYPE-AND-EXECUTE technique lets users display and invoke commands from a contextual menu such as marking menu [44]. The user types the command name, and the corresponding menu appears as she slides up to the upper space. For example, many mobile applications, such as iMessage and SnapChat, let users 'doodle' on their messages. With COMMANDBOARD's TYPE-AND-EXECUTE, users can specify brush properties, such as changing the color or brush type, with a marking menu. In Figure 6.4d, the user types 'color'. She slides her finger upward to reveal the COLOR marking menu, which brings up various brush color, and then moves down-left to select YELLOW.

Note that a challenge in combining COMMANDBOARD with a marking menu is deciding when a gesture should be interpreted as a 'mark'. One solution is to require users to begin from the middle of the screen, which, given the phone's limited screen real estate, would ensure sufficient space to move in all directions.

ACCESSING OS-LEVEL COMMANDS

COMMANDBOARD can also access OS-level commands such as changing the brightness level or volume. For example, in Figure 6.4c, the user begins gesture-typing the 'brightness' command. Because the keyword search is already successful by the time she writes 'brightn', she can immediately slide to the command gesture input space. Upon performing the /\ gesture, a slider appears, with the handle under her finger. Moving along the x-axis (left and right) adjusts the screen's brightness, whereas moving along the y-axis (up and down) moves the slider's position on the screen.

6.2.2 INLINE GESTURE SHORTCUT

COMMANDBOARD's INLINE GESTURE SHORTCUTS let users invoke gesture shortcuts from the keyboard *as they type*. Instead of typing the command name, the user types the object of the command, for example, the text to be styled. The user then slides her finger above the keyboard, pausing to bring up the dynamic guide that shows the current set of possible commands (see Figure 6.4a). Users can benefit from motor memory to recall these gestures. As they become experts, they can perform the command gesture directly, without pausing for the guide.

I provide some applications of COMMANDBOARD's INLINE GESTURE SHORTCUTS technique to illustrate how the users can use them in different context.

CHAT APPLICATIONS

Although soft keyboards are not specifically designed to support command input, some *chat applications* e.g. WhatsApp or Slack lets user use markdown languages to issue text styling commands. For example, typing an asterisk before and after a word (`_help!_`) produces (*help!*). Markdown commands are effective keyboard shortcuts when using physical keyboards, because users avoid lifting their hands to move the mouse. Unfortunately, on soft keyboards, markdown languages force users to switch from the alphabetic to the symbolic keyboard, disrupting their writing flow. Issuing styling commands as text can be cumbersome, especially if done often or multiple times in a row.

COMMANDBOARD's INLINE GESTURE SHORTCUTS offer a more efficient alternative, by executing a specialized gesture directly from the keyboard. In Figure 6.4a, the user wants to style the word 'happy'. After writing it, she moves into the upper area, pauses to see several styling alternatives, and then follows the pigtail to execute the bold command. As before, the dynamic guide offers a path to help users develop their motor memory, becoming expert over time.

COMMANDBOARD's markdown language is designed to be similar to those in existing applications, where the user writes a symbol before and after the word. Thus, writing 'news' followed by a pigtail gesture generates '`_news_`' on the text field buffer, which then will be rendered as *news*. This enables users to style more than one word, by moving the caret in between the markdown symbols and insert more words.

CONTACT APPLICATIONS

COMMANDBOARD's INLINE GESTURE SHORTCUTS lets users issue commands from within the search bar, as soon as the desired result appears. For example, most phones have a *contacts* application that lets users tap on a contact to view the person's details and then call, send an SMS, or use another communication app to communicate with that person. Users can also access a person's details by typing her name in the search bar.

COMMANDBOARD's INLINE GESTURE SHORTCUTS let users issue commands from within the search bar, as soon as the desired result appears. For example, if 'Mom' exists in the contacts

list, the user can gesture-type ‘Mom’, then slide up to the upper space and draw a pigtail gesture to call her. If the search produces multiple contacts the command bar displays the alternatives. For example, Figure 6.2 shows two contacts: *Alice Brooke* and *Alice Waltz*. Here, the user crosses through the *Alice Brooke* contact and then draws a pigtail gesture to call her.

6.3 GESTURE CHUNKING WITH COMMANDBOARD

Buxton [16] introduced the notion of “chunking”, i.e. grouping task elements into a larger inseparable whole. Novice users, during the initial learning process, often divide a task into smaller chunks that they perform sequentially. In the case of gesture typing, this happens as novices perform each gesture segment as a separate act, i.e. a line from “h” to “e”, and then another line to “y” when writing the word “hey”. On the other hand, in expert performance, the size of the chunk is larger. Gesture chunking improves motor efficiency, however it also increases the cognitive and learning burden on the users.

By interpreting each segment of a gesture to invoke a command, COMMANDBOARD enables gesture chunking, i.e. the command object and the command verb to be *chunked* in one single continuous stroke. The INLINE GESTURE SHORTCUTS technique involves two levels of chunking: the first segment on the keyboard area is interpreted as a word (command object) and the segment above the keyboard area as a command verb. We demonstrate three levels of chunking with the TYPE-AND-EXECUTE technique, where the command verb space is extended to multi layers submenu. For example, as illustrate in Figure 6.5, the first segment on the keyboard area is interpreted as the menu, the second on the command bar as the option, and the last above the keyboard area to execute the command.

6.4 LEARNABILITY ASPECTS

Both TYPE-AND-EXECUTE and INLINE GESTURE SHORTCUTS are designed for efficiency, and rely on an experienced user’s ability to either *recall* the command name, or the associated gesture. By and large, experts want to enter commands as efficiently as possible, intermittent users need to be reminded of commands, and novices need incremental help when learning the commands. Expert users must not only know that a command exists, but must also be able to recall either the command name, or the associated gesture shortcut.

Each technique provides scaffolding to help novice users discover the functionality and learn how to perform the gesture chunking. The TYPE-AND-EXECUTE technique has a *com-*

mand preview function, that displays a preview on the upper space if users gesture type a command name. Users may discover the contextual command-gesture input space accidentally while they gesture type. In the beginning, the preview has a high transparency. If the user move her finger towards the upper space, the preview becomes more apparent, i.e. the transparency is getting lower. This may help users to discover that they need to slide up to invoke the command.

We also provide several types of dynamic guides to help novices learn, and to help intermittent users when they forget. For TYPE-AND-EXECUTE technique, these may appear as command options in the command bar. Users can access a hierarchical command by typing the category name from any level, either the branch (e.g. “Style”) or the leaf (e.g. “Title”), and the command items reveal all of the command items. For INLINE GESTURE SHORTCUTS technique, a marking menu [44] or OctoPocus free-form gestures [8] appear if users pause their gesture in the command-gesture input space. Marking Menu or OctoPocus then display all available commands and users are guided to finish the gesture invocation. Experts users who already know how to invoke the desired command can just perform the gesture without pausing.

6.5 COGNITIVE AND MOTOR EFFICIENCY

One of the key benefits of graphical user interfaces such as menu or toolbar is their accessibility to novices. Invoking commands via buttons, menus and dialog boxes requires users to simply *recognize* the corresponding commands. In contrast, interfaces designed for more expert users, such as the Linux command-line interface or markup languages such as LaTeX, require the user to *recall* the command name, as well as its syntax. Many experts prefer the efficiency of command-line interfaces, even though they require learning and subsequent recall of command names and syntax. Some mobile chat applications such as WhatsApp or Slack already include command-line interface, e.g. markdown language that lets users write bold or italic text by adding symbols before and after the text. For example, writing `_hello_` will produce italicized *hello*. This approach is efficient on a physical keyboard, since it avoids leaving the keyboard to move the mouse, but requires two keyboard swaps on a soft keyboard. Worse, users have no easy way to learn the symbol mappings.

One of the goals of COMMANDBOARD is to bridge the gap between these two approaches, by supporting both recognition- *and* recall-based interaction, with a smooth transition between novice and expert use. With COMMANDBOARD’s INLINE GESTURE SHORTCUTS technique, users can apply text formatting to the current written word by drawing a gesture short-

cut on the upper space at the end of the word gesture input. Novices can pause to trigger the dynamic guide and just follow the guide to finish the gesture. Once they remember how to do it, they can just draw the gesture shortcut without pausing. `COMMANDBOARD` benefits the most when users are able to remember the command shortcut. Nevertheless, I argue that it also offers a cognitive advantage, that they do not have to switch the keyboard view to the symbol view nor open a pop-up menu, which may disturb the flow of typing.

To investigate further how much cognitive and motor efficiency `COMMANDBOARD` offers, I conduct an experiment to compare `COMMANDBOARD` to a command invocation technique on mobile devices.

6.5.1 EXPERIMENT DESCRIPTION

For users to invest their time in learning an interaction technique, we need to show that it offers more power and both cognitive and motor efficiency than other alternatives. We begin by examining “expert” behavior, with a focus on the cognitive and motor efficiency of the technique. We use a common experimental strategy for simulating expert performance: we show the participant the correct action so that we measure only performance, not confounded by unmeasured memory issues.

We sought an ecologically valid domain for testing `COMMANDBOARD`’s ability to support both recognition and recall. We chose the markdown commands available in chat applications such as *WhatsApp* and *Slack*, since users can style their text by typing markdown symbols before and after the text (recall), with a “cheatsheet” in the menu if they forget the symbols (recognition). For evaluating expert behaviour, `MARKDOWN SYMBOLS` offer a fairer, more realistic comparison than standard pull-down menus, which would be even slower. In the `INLINE GESTURE SHORTCUTS` condition, users write a word and then draw a command gesture directly from the keyboard to style it, whereas in the `MARKDOWN SYMBOLS` condition, users type markdown symbols before and after the word to be styled. Although not a primary goal, we are also interested in whether or not users begin to learn gesture-command mappings, simply by using the technique. Our research questions include:

1. Are `COMMANDBOARD`’s `INLINE GESTURE SHORTCUTS` faster and more accurate than text-based `MARKDOWN SYMBOLS`?
2. Do users prefer `COMMANDBOARD`’s `INLINE GESTURE SHORTCUTS`?

6.5.2 METHOD

We conducted a two-part study, using a within-participants design, to compare COMMANDBOARD's INLINE GESTURE SHORTCUTS technique to MARKDOWN SYMBOLS (see Figure 6.7). Part A is a one-factor experiment that compares speed and accuracy of expert users using these two techniques. Part B is a qualitative study designed to assess participants' preferences as well as incidental learning with respect to each technique. Part B follows Part A, with the same participants, hardware and software.

6.5.3 PARTICIPANTS

We recruited 12 right handed participants (4 women, 8 men), aged 23-41. All use mobile phones daily. Two gesture-type daily; the others are non-users. Three sometimes use mark-down symbols in existing chat applications; the rest do not.

6.5.4 HARDWARE AND SOFTWARE

We used two LG Nexus 5X (5.2" display) smartphones, running Android 7.1. I implemented COMMANDBOARD as an Android application that lets users issue text-styling commands with INLINE GESTURE SHORTCUTS, using the native Android gesture recognizer. The INLINE GESTURE SHORTCUTS technique requires the user to draw through the letters of the indicated word on the keyboard. COMMANDBOARD recognizes the word, and renders it on the screen. If the user continues the stroke above the keyboard, a semi-transparent overlay appears and the stroke is interpreted as a command gesture. The overlay displays an OctoPocus-like [8] dynamic guide indicating the gestures associated with possible styling commands. Lifting the finger applies the recognized gesture-command to the word output and the overlay disappears. Note: We removed OctoPocus' dwell delay in the experiment to avoid confounding time measures. I also implemented the MARKDOWN SYMBOLS technique, which requires the user to type a specified symbol before and after the word to be styled.

COMMAND-SET DESIGN I created a command set consisting of six text-styling commands: *underline*, *monospace*, *big*, *small*, *outline*, and *gradient color* and mapped them to INLINE GESTURE SHORTCUTS and MARKDOWN SYMBOLS. The INLINE GESTURE SHORTCUTS set consists of six gestures chosen from [5] (see Figure 6.6). I ensured that these gestures do not overlap when displayed together in an OctoPocus-style dynamic guide using [54]. The MARKDOWN

SYMBOLS set consists of six characters chosen from the second row of the symbol keyboard: @, #, \$, %, &, and +. I ensured that none overlap with existing chat symbols from, e.g., *WhatsApp* and *Slack*. Mappings between gestures and markdown symbols are counter-balanced across participants using a Latin square.

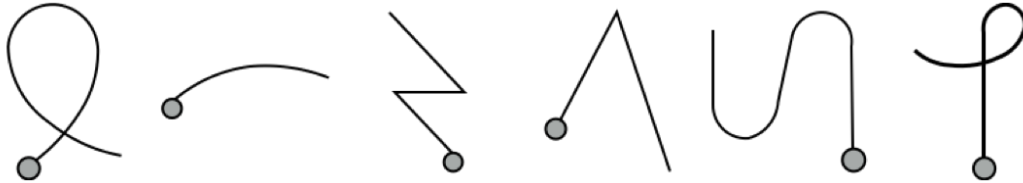


Figure 6.6: Gesture set: Grey circles indicate where to begin drawing.

PHRASE SET DESIGN We constructed two sets of 24 three-word phrases drawn from the Oxford Dictionary*. The middle words are each four-five letters long, and end in 24 different letters of the alphabet (we exclude ‘j’ and ‘q’), to ensure gesture starting points are distributed evenly across the keyboard. I also balanced angles between stroke segments across the sets, to avoid unwanted performance effects [62, 2]. Eight words include acute angles, e.g. “men”; eight include at least one obtuse angle, e.g. the ‘agi’ in “magic”; and eight include only 0° or 180° angles, e.g. “power”.

We used the 24 middle words to create two sets of 24 three-word phrases. We created two phrases around each middle word, using three-to-six letter surrounding words that make sense when read together as a phrase. For example, the first set includes ‘play *video* games’, and the second set includes ‘some *video* clips’. We distributed the first set of 24 phrases across the practice and experimental conditions of the experiment, and distributed the second set across the pre- and post-test conditions of the study. We counter-balanced for order within and across participants using a Latin square.

6.5.5 PROCEDURE

Figure 6.7 shows the study design. Part A consists of four conditions, each comprised of two blocks of six trials, grouped by technique. Part B consists of a single recomposition task where users can freely choose the desired technique.

*<https://en.oxforddictionaries.com/>

Condition	PART A			PART B	PART A
	Practice	Experimental	Pre-test	Recombination Task	Post-Test
Feedback?	yes	yes	no	yes	no
Gestures	6 trials	6 trials [x3]	6 trials	12 trials	6 trials
Symbols	6 trials	6 trials [x3]	6 trials	(user chooses technique)	6 trials

Figure 6.7: Part A (Experiment): Each condition (Practice, Experiment, Pre-test, and Post-test) includes two blocks of six trials, one per technique, with three replications in the experimental condition. Part B (Study): Participants recompose 12 of their own messages, with free choice of technique.


P1-EXP-o-g-1  the menu today	(a)	P1-EXP-o-s-8 they #think# fast	(b)	P1-TEST-o-s-9 with gesture what idiom works	(c)
the menu today		they think fast		what idiom works	

Figure 6.8: Each trial presents instructions above the line, and the result below the line. Practice and Experimental conditions present a) INLINE GESTURE SHORTCUTS to draw, or b) MARKDOWN SYMBOLS to type, to issue the specified styling command. Pre- and Post-Test conditions present c) the styled text to reproduce with the specified technique, with no feedback.

PART A: TRIAL DESCRIPTION

Each trial begins by displaying a three-word phrase, with a styled middle word, e.g. play *video* games. The participant presses *START*, then retypes the phrase, using the indicated technique to style the middle word. This simulates the process of issuing styling commands during the flow of writing. To simulate “expert” behavior, each trial includes explicit instructions as to how to execute the command, removing the need for recall memory. Participants may preview styling results.

Practice and experimental trials display the correct styling command, either the gesture to draw (Figure 6.8a, INLINE GESTURE SHORTCUTS condition) or the symbols to type (Figure 6.8b, MARKDOWN SYMBOLS condition). This simulates expert performance by eliminating errors due to forgetting a gesture shape or markdown symbol. Conditions are separated by short breaks.

PRACTICE CONDITION Participants are exposed to two practice blocks, one per TECHNIQUE (INLINE GESTURE SHORTCUTS and MARKDOWN SYMBOLS). Each block involves typing

six three-word phrases, and styling the middle word. Each trial shows which `INLINE GESTURE SHORTCUTS` or `MARKDOWN SYMBOLS` to use. In the `INLINE GESTURE SHORTCUTS` condition, the gesture template appears as soon as the participant's finger leaves the keyboard. Participants can retype phrases as often as they like, until they are comfortable performing the task quickly and reliably. An error message appears if they forget to apply the style or make a typing or styling error. Pressing `CLEAN` restarts the trial; `DONE` moves to the next trial.

EXPERIMENTAL CONDITION Participants are exposed to two six-trial blocks, one per `TECHNIQUE`(`INLINE GESTURE SHORTCUTS` and `MARKDOWN SYMBOLS`), for a total of 12 trials. Experimental trials are identical to practice trials, except that participants retype and style each three-word phrase three times (three replications), to provide a stable performance measure.

PRE- AND POST-TEST CONDITIONS Participants begin with two blocks of six trials, one for each `TECHNIQUE`(`INLINE GESTURE SHORTCUTS` and `MARKDOWN SYMBOLS`), counter-balanced for order within and across participants. Each trial displays the phrase to be typed including the styled the middle word (see Figure 6.8c). Participants reproduce the styled phrase with each technique, with no feedback. This serves as a baseline measure of styling command recall.

The pre- and post-test conditions are identical, but use phrases from the alternate phrase set. The pre-test offers an initial assessment of learning, how much they remember immediately after their first exposure to each technique. The post-test offers a second assessment, based on more extensive practice during the recomposition task.

PART B: RECOMPOSITION TASK

After completing the Pre-Test condition in Part A, participants are asked to perform a more open-ended set of tasks, in order to assess their overall preferences for each technique. For greater ecological validity, we asked participants to check their smart phones and choose 12 recent messages to retype, avoiding ones they felt were too personal. Participants were free to change the text as they liked. We then asked them to recompose these 12 messages, using either technique to style at least one word. We provided a 'cheat sheet' with the relevant markdown symbols for the `MARKDOWN SYMBOLS` technique, and displayed a dynamic guide with the relevant gestures for the `INLINE GESTURE SHORTCUTS` technique.

6.5.6 MEASURES

INPUT TIME We measure input time in seconds for the phrase and each word-output, referred to as: WO_1 , WO_2 , and WO_3 . Note that WO_2 includes inserting the two markdown symbols. This measure allows us to assess the gesture-typing time for both **INLINE GESTURE SHORTCUTS** and **MARKDOWN SYMBOLS**.

GESTURE-TYPING AND COMMAND SELECTION TIME The participant must gesture-type the middle word and style it using **INLINE GESTURE SHORTCUTS** or **MARKDOWN SYMBOLS** (i.e. WO_2). We capture the times spent in each sub-activity. We measure Command Selection Time (command time) and Gesture-Typing Time (typing time).

INLINE GESTURE SHORTCUTS: We measure the time spent leaving the keyboard and drawing the gesture (command time). If a participant crosses the top border of the keyboard, below the suggestion bar, at $event_k$, then command time and typing time are as follows:

$$\text{command time} = t(event_N) - t(event_k) \quad (6.1)$$

$$\text{typing time} = t(event_k) - t(event_o) \quad (6.2)$$

MARKDOWN SYMBOLS: We measure the time spent writing the symbols before and after the word (command time) for WO_2 . Given an input I is a sequence of touch *events*, where $I = \langle event(x, y, t, \text{action})_{o\dots N} \rangle$, if a participant starts gesture-typing the word at $event_i$ (tagged as **TOUCH_DOWN**) and lifts her finger at $event_j$ (tagged as **TOUCH_UP**) in WO_2 , then command time and typing time are as follows:

$$\text{command time} = t(event_i) - t(event_o) + t(event_N) - t(event_j) \quad (6.3)$$

$$\text{typing time} = t(event_j) - t(event_i) \quad (6.4)$$

GAP TIME We assess how long participants spend switching from writing a regular word (WO_1) to a styled word (WO_2) and back again (WO_3). Given that an input I is a sequence of touch *events*, $I = \langle event(x, y, t, \text{action})_{o\dots N} \rangle$ where t is the timestamp, we measure gap time between each word-output as follows:

$$\text{gap}(WO_i, WO_{i+1}) = t(WO_{i+1}.event_o) - t(WO_i.event_N) \quad (6.5)$$

ERRORS We count three types of error: typographical errors (typing errors), incorrect symbols or gestures (styling errors), or forgetting to style the middle word (missing errors). Note that typing errors and styling errors can occur in the same trial. A trial is considered correct when it has no errors.

6.5.7 DATA COLLECTION

I log all touch events and the recognized word output for each trial. I tag each touch event with one of five actions: SHIFT, TAP, TOUCH_DOWN, TOUCH_MOVE, and TOUCH_UP. TAP involves pressing a key and SHIFT involves holding down the keyboard shift key. The remaining actions identify the start (TOUCH_DOWN), drawing phase (TOUCH_MOVE) and completion (TOUCH_UP) of a gesture. These measures allow us to compute speed, movement time and errors for each technique.

Participants answer a five-point Likert-style questionnaire to assess their perceived accuracy, speed, ease-of-use, confidence, comfort, and enjoyment of each technique. We also take observational notes and debrief participants, with a particular focus on what the participants liked and disliked about the techniques and their strategies for styling their text.

6.5.8 RESULT: EXPERIMENT

We collected a total of 432 experimental trials (12 participants \times 2 TECHNIQUE \times 6 trials \times 3 replications). We removed one trial (P4) who gave up after repeated typing errors on the third word of one phrase. After determining we had no unwanted significant effects from the word sets, we ran a one-way repeated-measures analysis of variance for factor TECHNIQUE, followed by Tukey HSD tests for post-hoc comparisons.

INPUT TIME

The overall input time (trial completion time) is significantly affected by TECHNIQUE ($F_{1,11}=86.9$, $p<0.0001$). This is due primarily to styling the middle word (WO_2), as shown in Figure 6.9.

GESTURE-TYPING AND COMMAND SELECTION TIME

On average, participants spent significantly more time styling words with MARKDOWN SYMBOLS (mean 6.3s) than with INLINE GESTURE SHORTCUTS (3.3 seconds), with $F_{1,11}=71.1$, $p<0.0001$.

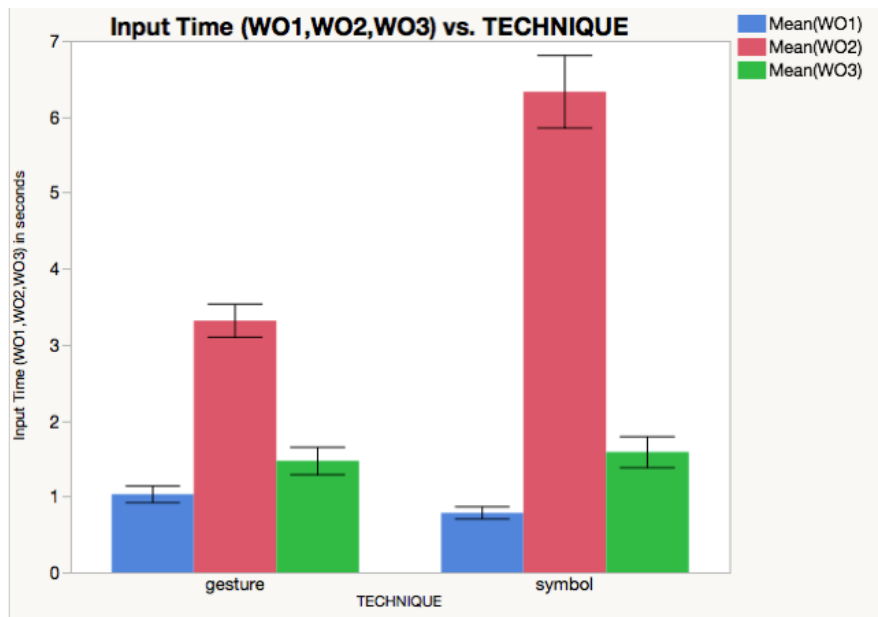


Figure 6.9: Average time spent entering each word. WO_2 is the styled word. Using INLINE GESTURE SHORTCUTS is significantly faster: almost double.

When we break apart input time for WO_2 into time to select the command (command time) and time to gesture-type it (typing time), we find that participants spend significantly longer writing symbols (mean command time=5.8s) than drawing gestures (mean command time=1.5s) [$F_{1,11}=177.6$, $p<0.0001$] (Figure 6.10). This highlights the motor efficiency of COMMAND-BOARD – users can invoke the command almost twice faster than MARKDOWN SYMBOLS.

However, they spend more time gesture-typing the styled word when using INLINE GESTURE SHORTCUTS (mean typing time=1.8s) than MARKDOWN SYMBOLS (mean typing time=0.6s) [$F_{1,11}=68.3$, $p<0.0001$]. This may be an artifact of the experimental design, since participants slowed down to check that they had gesture-typed the correct word, before drawing the styling gesture. In the long run, this may actually benefit the INLINE GESTURE SHORTCUTS technique, because slowing down improves the recognition process with gesture keyboards [42]. Recognized words are less likely to change when users slide into the command gesture input space.

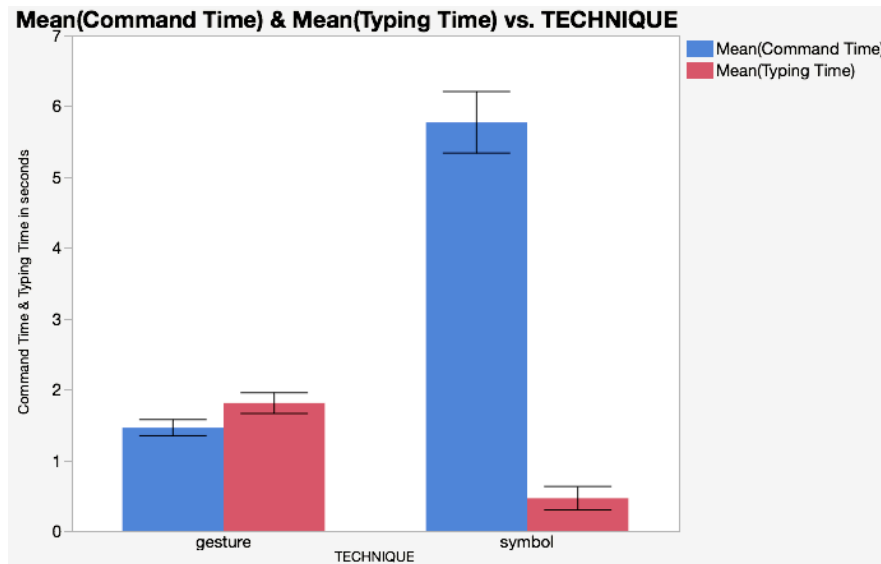


Figure 6.10: Average time spent gesture-typing (typing time) and issuing the command (command time). Participants drew quickly with INLINE GESTURE SHORTCUTS, but took significantly longer inserting MARKDOWN SYMBOLS.

GAP TIME

When the participants switch from writing the first word to applying a styling command to the second word, the gap duration ($\text{gap}(WO_1, WO_2)$) is significantly longer for MARKDOWN SYMBOLS (mean=1.9s) than for INLINE GESTURE SHORTCUTS (mean=1.2s) [$F_{1,11}=49.7, p<0.0001$]. This suggests that participants needed more time to consider which key to press when selecting markdown symbols, i.e. searching and pre-planning. COMMANDBOARD brings a cognitive benefit, since users can just continue gesture typing right after they are done invoking the command.

However, when participants finish applying the styling command to the middle word, they spend significantly less time writing the third word when using MARKDOWN SYMBOLS (mean $\text{gap}(WO_2, WO_3)$ 0.9s) than when using INLINE GESTURE SHORTCUTS (mean $\text{gap}(WO_2, WO_3)$ 1.5s) [$F_{1,11}=128.4, p<0.0001$]. In the MARKDOWN SYMBOLS condition, they can already see if they have applied the correct command as they press the SPACE bar, whereas with INLINE GESTURE SHORTCUTS, they must check again after releasing their finger. This would be improved by displaying a progressive preview at the end of the dynamic guide, but was not made available during the experiment.

ERRORS

Participants made significantly fewer styling errors with `INLINE GESTURE SHORTCUTS` (mean styling errors = 0.09) than with `MARKDOWN SYMBOLS` (mean styling errors = 0.36), [$F_{1,11}=13.7$, $p=0.0035$]. However participants using `INLINE GESTURE SHORTCUTS` were somewhat more likely to forget to actually style the word – `INLINE GESTURE SHORTCUTS` (mean missing errors = 0.3) versus `MARKDOWN SYMBOLS` (mean missing errors = 0.04), [$F_{1,11}=26.7$, $p=0.0003$]. This is probably an artifact of the experimental setting, since in actual use, users would not ‘forget’ to style a word if they wanted to. We did not find a significant effect of `TECHNIQUE` on accuracy [$F_{1,11}=49.7$, $p=0.47$], which suggests that using gestures to style text does not interfere with typing accuracy.

6.5.9 RESULT: PREFERENCES STUDY

PRE- AND POST-TEST RESULTS

We ran a one-way repeated measures analysis of variance for factor `TECHNIQUE` to compare styling errors during the Pre- and Post-test conditions. We found a significant interaction effect [$F_{1,11}=4.4$, $p=0.0375$] for styling errors. In the Pre-test, the average styling errors for `INLINE GESTURE SHORTCUTS` and `MARKDOWN SYMBOLS` are 0.52 and 0.32, respectively. In the Post-test, the average styling errors for `INLINE GESTURE SHORTCUTS` and `MARKDOWN SYMBOLS` are 0.35 and 0.38, respectively.

Prior to the pre-test condition, participants had practiced both techniques, but always with a direct indication of how to perform the gesture or what symbols to type. The pre-test was the first time that participants had ever tried executing the commands without help. Participants remembered half the gestures and two thirds of the symbols from the previous practice and experiment condition. The post-test was given after participants had experimented with their choice of technique to recompose their own text, and participants remembered almost two thirds of the gestures. This suggests that we should study longer term use of `COMMANDBOARD`’s `INLINE GESTURE SHORTCUTS` technique, to see how well it supports incremental learning over time.

RECOMPOSITION TASK RESULTS

Although given a choice between using `MARKDOWN SYMBOLS` or `INLINE GESTURE SHORTCUTS`, all participants chose gestures. They ignored the cheat-sheet showing all markdown sym-

bols and their resulting styles. P11 was the exception, but he only looked at the cheat-sheet to get inspiration from the style examples. We observed three strategies when styling words with gestures: thinking of a style first, and then using OctoPocus to follow the corresponding gesture; activating OctoPocus first, and then deciding on a style from the options; and performing a learned gesture to apply a style with no hesitation.

A few participants explained the rationale behind their styling. P2 recomposed a text message to his wife with a shopping list, and he used all available styles to highlight the ingredients they had to buy for a salad. P8 associated word categories to styles: big meant positive or a lot, small meant negative or uncertain, underline was important or certain, outline and gradient were for special words. P12 also assigned meanings to different styles: gradient for opinions, outline for time-related words, underline for important words, and big for emphasis in general: “*Big is the most useful.*” P11 on the other hand cared less about the different styling options, and mostly focused on emphasizing important words: “*I think I didn’t really want to choose a specific [style], I just wanted to add an effect on it so it looks different from other words.*”

SELF-REPORTED QUANTITATIVE MEASURES

Participants were asked to rate six statements on a 5-point Likert scale, from strongly disagree (1) to strongly agree (5). The statements asked whether the current technique helped them to style text: a) accurately, b) quickly, c) easily, d) confidently, e) comfortably, and f) enjoyably. Table 6.2 lists the medians of each question for both techniques. An analysis using a Friedman test showed that participants reported significantly stronger agreement for INLINE GESTURE SHORTCUTS compared to MARKDOWN SYMBOLS on five statements: ACCURATELY ($p=0.34$, $\chi^2(1) = 4.5$), QUICKLY ($p=0.007$, $\chi^2(1) = 7.36$), EASILY ($p=0.11$, $\chi^2(1) = 6.4$), COMFORTABLY ($p=0.002$, $\chi^2(1) = 10$) and ENJOYABLY ($p=0.001$, $\chi^2(1) = 11$).

USER PREFERENCES AND DEBRIEFING

The final questionnaire asked participants to rate their preference between the two techniques on a 5-point scale (from strong preference for MARKDOWN SYMBOLS to strong preference for INLINE GESTURE SHORTCUTS). All participants preferred INLINE GESTURE SHORTCUTS: 10 indicated a strong preference, 2 indicated some preference.

Six participants expressed their preference in terms of *typing flow*, explaining that INLINE GESTURE SHORTCUTS best supported styling without interrupting their text composition pro-

Statement	MARKDOWN SYMBOLS	INLINE GESTURE SHORTCUTS
ACCURATELY*	2.5	4.0
QUICKLY*	2.0	4.0
EASILY*	2.0	4.0
CONFIDENTLY	2.5	4.0
COMFORTABLY*	2.0	4.0
ENJOYABLY*	2.0	4.5

Table 6.2: Participant ratings of how each technique helped them to style text (median values; * indicates a significant difference). Participants significantly preferred gestures in all categories except ‘confidently’.

cess. P2 commented *“I didn’t use the symbols at all in the chat. It’s troublesome to have to switch the keyboard, doing it in the beginning and at the end. It really breaks the flow of the writing. While with the gesture, it’s always there, I can pick what I want on the go.”* P9 wrote *“It’s enjoyable to use and in coherent with using gestures to type words.”*

Participants differed with respect to recognition and recall. Four participants found INLINE GESTURE SHORTCUTS easier to recall than MARKDOWN SYMBOLS: *“I used big, small, underline in the recomposition task, so I remember them”* (P1). However, four participants had difficulty recalling the INLINE GESTURE SHORTCUTS mappings: P9 said *“the paths of gestures are difficult to link with their meanings”*, and P6 said *“If the gestures are well designed or designed by the user himself, it could be quite natural.”* Two participants felt more comfortable creating mnemonics for MARKDOWN SYMBOLS rather than INLINE GESTURE SHORTCUTS, despite their overall preference for INLINE GESTURE SHORTCUTS: *“It’s easier to remember the symbols for each type (+ for big; \$ for the underlined because of the line in the S).”* Three participants also appreciated the convenience of recognizing gestures with OctoPocus rather than always having to recall them: *“this is nice, I don’t have to remember and just follow [the OctoPocus guideline].”*

Finally, we asked participants to suggest other applications for INLINE GESTURE SHORTCUTS. Four participants suggested using gestures to add emojis: *“I have 5-10 smileys that I always use, so I think it’d be nice if I can use the gesture to get it. Because it’s bothersome having to change to another keyboard view (emoticon), so if I can do it with the gesture it’d be cool.”* (P3). Two thought of command shortcuts: *“If you like a webpage, you could do a special gesture to bookmark it. To refresh the page, you could use a circular gesture, etc.”* (P2). Other suggested applications were changing lines, replacing the enter key, taking notes and changing fonts.

6.6 DISCUSSION

Although we expected that `COMMANDBOARD`'s `INLINE GESTURE SHORTCUTS` would perform better than current markdown commands, we were surprised by the size of the effect (approximately twice as fast) and by how much the participants preferred it over standard markdown commands. We believe this is because users can fluidly style their text without interrupting the flow of their typing, that reduces the cognitive and motor load. Users not only avoid switching modes, but also avoid selecting text, the most time-consuming aspect of text editing [25].

The pre- and post-test results from the experiment indicate that users can easily learn gesture commands simply through the process of using them. We expected relatively low post-test scores, since users had only limited experience with the gestures during the practice and experimental conditions. Even so, users clearly made fewer errors in the post-test, which suggests that even limited experience can improve gesture recall. We should be able to further reduce learning time and enhance the transition from novice to expert performance by letting users define their own memorable, yet recognizable gestures [57], e.g. with [54]. In future work, we plan to conduct a longitudinal field study of `COMMANDBOARD`, in order to more thoroughly investigate this transition from novice to expert.

The experiment restricted `COMMANDBOARD`'s `INLINE GESTURE SHORTCUTS` to styling one word at a time. For example, the 'happy'+pigtail gesture generates *happy*. However, sometimes users want to apply a style to multiple words. One option would be to combine `COMMANDBOARD` with other advanced text selection techniques, such as selecting a phrase with a two-finger gesture on top of the keyboard [25]. Gesture grammars can also combine command gestures with selection-scope gestures. For example, in `TYPE-AND-EXECUTE`, after sliding her finger to the input space above the keyboard, the user could specify the scope of the selection with a marking menu that includes last word, last sentence, last paragraph, and select all.

6.7 SUMMARY

Since many mobile activities involve typing, I believe letting users invoke a command from the keyboard is very beneficial. With `COMMANDBOARD`'s `TYPE-AND-EXECUTE` technique, users can type the command name and execute it right away from the keyboard. They do not have to search through a long menu with many intermediate steps that may disturb the flow of

typing. The command-gesture input space only appears if the user slides up to the upper space and/or have gesture typed a command name before. Thus, `COMMANDBOARD` offers a seamless, efficient way to invoke gestures while the users are typing. It also tackles the common issue of designing a gesture-based command invocation technique, that is discriminating a command gesture from a direct-manipulation gesture [48]. Since `COMMANDBOARD` takes advantage of the recognition engine of the gesture keyboard, the command name is recognized as soon as users gesture type, and we can provide a progressive feedback. Users must learn how to gesture type and then slide up to perform the command gesture, making the interaction more complex than just gesture-typing, however they get access to far more powerful functionality.

`COMMANDBOARD` is mostly beneficial for expert users, who can *recall* the command name or the associated command gesture. In Experiment 4, I show that users can a invoke command almost twice as faster than markdown shortcuts. Nevertheless, the design of `COMMANDBOARD` supports the transition from novice to expert users. If a novice users forgets the command name or wants to discover the full command space, she can open the menu and invoke the command from there, since `COMMANDBOARD` co-exists with traditional menu or toolbars. With `COMMANDBOARD`'s `INLINE GESTURE SHORTCUTS`, a dynamic guide appears whenever she hesitates and pauses on the command-gesture input space. `COMMANDBOARD` is a useful technique that can be used in conjunction with existing command invocation techniques.

All gesture-based menu systems, including Marking Menus and OctoPocus, run into visual overload problems when forced to display more than about 16 menu items at a time. This is commonly addressed by creating hierarchical menus or by restricting the command set to a more limited context. `COMMANDBOARD` faces these same limitations, but they can be partially mitigated when `COMMANDBOARD` is used in conjunction with other recognition-based techniques. On the other hand, using `COMMANDBOARD` to type commands on the keyboard and then select a parameter in the gesture-input space can help users access the full range of available commands. The `INLINE GESTURE SHORTCUTS` technique is also most useful when the current context significantly limits the command space, for example a set of user's "favorite" commands.

Note that we do not seek to define a single 'best' method of issuing commands, since different commands perform better in different contexts [52], but rather to create a keyboard that offers users a choice, based on their cognitive and motor skills, as well as the size and organization of the current command set. `COMMANDBOARD` exists in harmony with exist-

ing command-generation techniques, such as menus and buttons, but also offers novices the opportunity to transition into power users, to execute commands fluidly at their fingertips.

7

Conclusion

7.1 TRANSFORMING MOBILE DEVICES INTO POWERFUL, PERSONALIZED TOOLS

Mobile devices have a very limited screen space. Appert and Zhai argued that a soft keyboard takes the valuable screen space, and thus, a free form gesture-based command selections may be preferable [5]. However, soft keyboards can be displayed in any mobile applications: communication app, web service, contacts, calendar, games, text editor, etc. Users already have a lot of experience with typing, and invest a lot of time interacting with their mobile phones. Thus, why don't we empower the keyboard, by leveraging users' experience, motor control, and cognitive ability? How about we let users generate different kind of rich output from the keyboards to enrich communications? How about we let them access a large command space from the keyboards? How about we let them appropriate the system to fit their needs and preference?

In this thesis, I describe my attempt in rethinking access to powerful functionality through gesture keyboards. My focus is on increasing *expressivity* and *efficiency*, while supporting learnability and appropriability. By building upon the gesture keyboard, I leverage its powerful machine-learning algorithms, and offer an easy way to incorporate successful gesture-based interaction techniques from the research literature.

Through an experiment, I show that users naturally vary their gesture input due to biomechanics, personality, activities, environments, etc. EXPRESSIVE KEYBOARD “recycles” users'

otherwise-unused gesture variation to create rich output under the users' control, without sacrificing accuracy. Thanks to their extremely fine motor control, users can also control aspects of their gesture deliberately, in order to generate desirable rich output. I found that users are more successful when they focus on output characteristics (such as red) rather than input characteristics (such as curviness). This highlights the importance of feedback, in this case, the color output, both to better reveal how the mapping between gesture characteristics and color components to the users, and to apprise users of their performance.

However, I also observed a trade off between effort and feature accuracy (i.e. how accurate users can generate specific output properties). Since all participants were necessarily novice users, a more longitudinal study of EXPRESSIVE KEYBOARD is needed to determine whether accuracy improves with experience. Additional factors may also affect user performance when using EXPRESSIVE KEYBOARD, for example, finger v.s. stylus, which fingers the users gesture type with, etc. Furthermore, in real usage, users should be able to vary the sensitivity of the output variation or turn it off completely [6], especially in cases where the user prefers to generate more formal output. Users should also be able to (re)design their own feature detectors, text-rendering properties, and mappings linking the two. Nevertheless, the fact that the users were still in the "initial learning" phase did not prevent the users to appropriate EXPRESSIVE KEYBOARD in various different contexts.

To enable command selection from soft keyboards, Kristensson and Zhai [43] proposed Command Strokes where a CONTROL button is added to the keyboards, so that users can perform CONTROL+C as they can do in a physical keyboard. I propose COMMANDBOARD that pushes the idea further: users can invoke hierarchical commands from the keyboard in different ways. Analogically, it is like typing a command name on a *search bar* of a desktop application. The application progressively searches and displays command candidate(s) based on the typed keyword, and executing the command from the search bar gives exactly the same effect as if it is done through the menu.

By transforming the area above the keyboard into an alternate gesture-input space, users can benefit from a variety of new command entry techniques, using text, gestures or both. This is a useful technique that can be used in conjunction with existing command invocation techniques. We can make it *discoverable* by offering in-context dynamic guides when the user pauses [8, 44], depending upon the task, the user's cognitive and motor skills, and the size and structure of the current command space.

With TYPE-AND-EXECUTE, if the user knows the command name, but not the associated gesture, she can always type the command name, followed by the *execute* gesture. If the

user types a command category, we can display the most likely alternatives in the suggestion bar. For example, if the user types `style+ /` bold, italic, and an underline will appear in the suggestion bar. It can also handle parametric commands, such as typing `'volume' + /` to bring up a volume slider, with continuous control of the sound level.

COMMANDBOARD can also support commands that relate to the current typing activity, such as writing a word, and then applying a styling gesture to that word. With INLINEGESTURE SHORTCUTS, if she does not remember or only knows the first part of a free-form gesture shortcut, she can pause to see the remaining available commands arrayed around her finger, and move in the indicated direction.

Both of the proposed techniques, COMMANDBOARD and EXPRESSIVE KEYBOARD, can co-exist and support each other, for example to support better customization. When combined together, users can parameterize the command by manipulating the gesture variation when gesture-typing the command name, and then execute it by performing the execute gesture on the upper space. Users can toggle the rich output, or change the output properties they want to use in the current context through the keyboard.

Ideally, both EXPRESSIVE KEYBOARD and COMMANDBOARD should exist on the mobile operating system (OS) level, and mobile application developers should be able to use them as a service. The OS provides a library where designers can use predefined gesture characteristics, gesture shortcuts, and/or command sets, and they can add more.

7.2 CONTRIBUTIONS

This thesis describes my attempts on increasing the expressive power of gesture-based interactions on mobile devices, by leveraging the variations in human gesture to produce rich output, to improve user control, and to facilitate appropriation in different contexts of use. It contributes to the HCI literature in several ways.

First, this work contributes to fundamental research by exploring the use of gesture variability to express different concepts, opening up a wide range of research directions. I conducted a series of controlled experiments, that show that users *do* vary their gesture input. The gesture variations are significantly different across and within users, and the recognition engine already anticipates these variations. Users can also deliberately control the variations of their gesture input *as they gesture type*, while retaining the recognition accuracy. I show that we can “recycle” these otherwise-unused variations, quantify and then transform them into various kind of rich output that users can easily appropriate in different contexts. For

this purpose, I designed, implemented, and evaluated EXPRESSIVE KEYBOARD to support an expressive channel between users and the systems. I have explored how to appropriate this expressive channel to enrich inter-personal communications in different forms, for example to generate dynamic fonts or parametric emoticons. With EXPRESSIVE KEYBOARD, users can gesture type just as they usually do, but the generated rich output can express different concepts such as personal styles, implicit communication of mood, activity, context, etc. Once they better learn to control aspects of their gestures, they can deliberately generate rich output to explicitly communicate emphasis, sarcasm, humor, excitement, etc. This approach had not been explored in research on continuous text-input techniques.

Secondly, I have explored command invocation from soft keyboards that offers cognitive and motor efficiency. I designed COMMANDBOARD, which lets users gesture-type commands directly from a gesture keyboard. The novelty relies on the idea that we transform the space *above* the keyboard into an alternate, gesture-based input space. I proposed two techniques that address different trade-offs. Users can gesture-type a known command name followed by an *execute* gesture; or move from the gesture-type keyboard directly to the *command-gesture input space* above, to execute a unique command gesture. COMMANDBOARD benefits from extended learning where users are able to recall the command names. Thus, the design includes supports to *discoverability* and *learnability*, by offering two types of dynamic guides. The command bar offers suggested command names that users can select by crossing through, and the dynamic guide offers progressive feedforward, to suggest alternative command-gesture mappings. I implemented several applications of each technique to illustrate the variety of ways that it can be incorporated into different context.

Altogether, I demonstrate how we can extend the interaction space of a widely-adopted system like gesture keyboards to empower its functionality while preserving its accuracy, simplicity, and accessibility. Nevertheless, the gesture keyboard is a means to an end to build simple but powerful mobile interactions, and the proposed techniques are not limited to employing gesture keyboards. I view my work as a strategy for transforming mobile devices into powerful, personalized tools, with which users can benefit from a variety of new gesture-based interaction techniques, based on what they can do and their personal needs.

7.3 PERSPECTIVE

Finally, I reflect on my work, and use generative design tools to demonstrate the directions we can take in the future: *reciprocal co-adaptive instrument* and the notion of *substrate*.

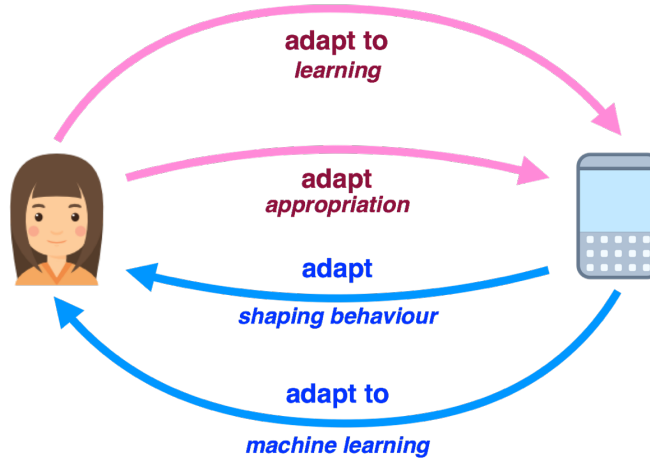


Figure 7.1: Reciprocal co-adaptation focuses on the human-computer partnership in which both the user and the system learn from and adapt each other in order to give more power to the users.

7.3.1 BUILDING HUMAN-COMPUTER PARTNERSHIP

Reciprocal co-adaptation extends the principal of co-adaptation [51], emphasizing the notion of human-computer partnership. We can see the partnership from *user's* point of view and the *computer's* point of view (see Figure 7.1).

With users as the subject, they *adapt to* the systems, by understanding how the system works. This process is related to *learnability*, and elevated through progressive feedback i.e. *discoverability*. In EXPRESSIVE KEYBOARD, users can discover how the additional output properties e.g. color change as soon as they gesture type, along with the intermediate word output. This mechanism helps users to understand and learn better how the mapping between the gesture characteristics and the output properties works. With COMMANDBOARD, that especially benefits from extended learning, users can “accidentally” discover the existing commands just by gesture typing, and the progressive dynamic guides help them to learn how to perform the gestures. COMMANDBOARD is also *discoverable*, by offering in-context dynamic guides when the user pauses, such as OctoPocus or Marking Menu.

Users also *adapt* the systems, by appropriating the system to fit their needs. We see how users start to come up with different uses of EXPRESSIVE KEYBOARD the first time they use it. In Experiment 3 (Section 5.3), the participants started appropriating EXPRESSIVE KEYBOARD within the current context i.e. text-based communication. Nevertheless, it did not take long for them to come up with ideas on how to use it in different context, such as text-to-speech and command selection. COMMANDBOARD can also be appropriated in different

context of use, for example by offering dynamic guides for generating recognizable, but memorable gestures [54]. We can make it *expressive* by allowing gestures to dynamically modify command parameters, for example combining it with EXPRESSIVE KEYBOARD.

In the future, I am interested in expanding the different forms of output that can be mapped to user gesture, including parametric animated emoji, handwriting, and nuanced speech synthesis. I am particularly interested in studying the use of the combined techniques (EXPRESSIVE KEYBOARD and COMMANDBOARD) in ecological settings to see how it is appropriated for providing more powerful functionality such as customization.

With the system as the subject, it *adapts* to the users, for example via machine learning algorithm. The system learns user behaviour, such as personal dictionaries, gesture-typing styles, gesture features, etc. The more users use the system, the better the machine learning algorithm in “guessing” users’ intention. Users then can do more powerful interaction with relatively less effort. This is advantageous especially for novices, when they are still learning to get used to the system. However, it may also disturb the input flow once the users become experts, since at this point the machine learning algorithm cannot anticipate the user behaviour anymore. There is also the danger of trapping the users in arbitrary behaviour, overlooking that humans are creative beings with rich cognitive and motor skills. For future work, we can improve the machine learning algorithm to anticipate deliberate gesture variations.

More interesting is how the system then *adapts* user behaviour, for example to increase accuracy. In Experiment 3, while using EXPRESSIVE KEYBOARD, some participants became aware of their gesture typing performance, and put efforts in changing the way they performed. In this case, perhaps they just wanted their output to be black – however, the word gesture generating a “black” text is actually what the gesture keyboard’s algorithm expect. For future work, we can control this factor to “seamlessly train” the users to gesture type more accurately. We hypothesize that if we always let users to type “sloppily”, then at some point they may surpass the recognition algorithm’s tolerance and thus fail to gesture type accurately. Cockburn et al [21] also suggested that explicitly increasing the mental effort of interaction may actually support better learning of an interaction technique. Testing this hypothesis requires support from the machine learning algorithm, in which the “sloppiness tolerance” changes depending on user performance, and should be done in a longitudinal study. This also includes training users to be able to control aspects of their gestures more independently and reliably.

7.3.2 SCREEN SPACES AS DIFFERENT SUBSTRATES

The screen space's divisions in `COMMANDBOARD` can be explained with the notion of *substrate*. Each screen space described in Figure 6.2 is a substrate. Thus, the gesture input is chunked as it enters a different substrate, and each chunk is interpreted differently. The text input space, i.e. the keyboard, uses the `QWERTY` layout as its structure, and interprets the gesture drawn on top of it as a word gesture. The suggestion bar is another substrate that displays word candidates produced by the keyboards.

`EXPRESSIVE KEYBOARD` uses the structure of the keyboard substrate, yet it increases the granularity of the interpretation, i.e. a curvy “hello” word gesture. The generated rich output e.g. a text view does not have a structure, however it is a first-class object that contains rich information e.g. the gesture features, the accuracy score, etc. Users can potentially inspect, manipulate, or reuse these data by directly manipulating the output view. For example, when a user dwells on the output view, a Marking Menu appears, and she can choose to perform an action (i.e. invoking a command) such as copy, paste, or store the properties for future (re)use. A possible scenario for reusing the output properties is for example to save a generated dynamic font that the user likes so that she can reuse it in the future. She can also apply the same properties to other type of output, for example to parametric emoticons. However, as with spatially-dependent gestures/menus, the challenge is how to support the discoverability [78].

The upper space, in normal uses, is yet another substrate that displays the output from the keyboard, informations, etc. With `COMMANDBOARD`, we add two additional substrates: the command bar and the command-gesture input space. These two substrates are contextual, meaning they only appear when users gesture type a command name and/or drawn above the keyboard area. Application designers can decide what kind of structure they would like to apply to a command-gesture input space, and the gesture chunk will be interpreted based on it. This concept explains how `COMMANDBOARD` turns the upper space into a general-purpose command-gesture space. Future work should improve the customization aspects, for example to let users define their own gesture shortcut to elevate the memorability of gesture shortcuts.

8

Appendix A: Accessing Hierarchical Commands from COMMANDBOARD

Display and Parameterize Command from Widgets COMMANDBOARD's TYPE-AND-EXECUTE technique can also be combined with widgets e.g. sliders, color picker, etc to parameterize commands. Figure 8.1a shows an example of choosing a color from a color picker. The user gesture-types 'color', slides up and draws the execute gesture on the upper space, and COMMANDBOARD displays a color picker. The user can drag their finger to choose the desired color. In Figure 8.1b, the user inserts a table with specified number of row and column: the user gesture-type 'table', slides up and draws the execute gesture on the upper space, and COMMANDBOARD displays a widget that lets her insert a table as well as define how many rows and columns in the table.

Specify Target Scope Users can specify the scope of object to which the command will be applied, e.g. text selection scope. If the user wants to activate a command, she performs the execute gesture after gesture-typing the command name. For example, 'bold'+execute will activate BOLD function to the upcoming text she is going to write. She can also apply the BOLD function to the previous already-written text. For example, 'bold'+pigtail gesture makes the previous word bold, 'bold'+zigzag makes the all of the written text bold.

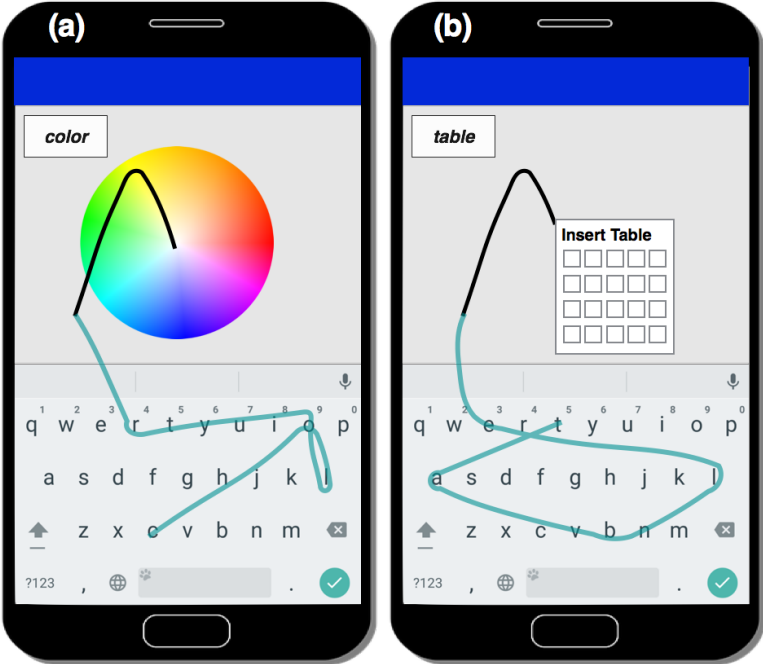


Figure 8.1: CommandBoard enables users to display and parameterize command from widgets: a) type 'color' draw an execute gesture and pick a color from the color palette; b) type 'table', draw an execute gesture and specify how many rows and columns in the inserted table.

References

- [1] Alvina, J., Griggio, C. F., Bi, X., & Mackay, W. E. (2017). Commandboard: Creating a general-purpose command gesture input space for soft keyboard. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17 (pp. 17–28). New York, NY, USA: ACM.
- [2] Alvina, J., Malloch, J., & Mackay, W. E. (2016). Expressive keyboards: Enriching gesture-typing on mobile devices. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16 (pp. 583–593). New York, NY, USA: ACM.
- [3] Andersen, T. H. & Zhai, S. (2008). Writing with music: Exploring the use of auditory feedback in gesture interfaces. *ACM Trans. Appl. Percept.*, 7(3), 17:1–17:24.
- [4] Appert, C. (2017). *From Direct Manipulation to Gestures: Moving the Expressive Power from the Displays to the Fingers*. Habilitation à diriger des recherches en informatique – spécialité interaction homme-machine, Université Paris-Sud XI. 172 pages.
- [5] Appert, C. & Zhai, S. (2009). Using strokes as command shortcuts: Cognitive benefits and toolkit support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09 (pp. 2289–2298). New York, NY, USA: ACM.
- [6] Arif, A. S., Kim, S., Stuerzlinger, W., Lee, G., & Mazalek, A. (2016). Evaluation of a smart-restorable backspace technique to facilitate text entry error correction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16 (pp. 5151–5162). New York, NY, USA: ACM.
- [7] Azenkot, S. & Zhai, S. (2012). Touch behavior with different postures on soft smartphone keyboards. In *Proc. ACM MobileHCI 2012* (pp. 251–260).: ACM Press.
- [8] Bau, O. & Mackay, W. E. (2008). Octopocus: A dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology*, UIST '08 (pp. 37–46). New York, NY, USA: ACM.
- [9] Bevan, N. & MacLeod, M. (1994). Usability measurement in context. *Behaviour & Information Technology*, 13(1-2), 132–145.

- [10] Bi, X., Azenkot, S., Partridge, K., & Zhai, S. (2013a). Octopus: evaluating touchscreen keyboard correction and recognition algorithms via “remulation”. In *Proc. ACM CHI 2013* (pp. 543–552).: ACM Press.
- [11] Bi, X., Li, Y., & Zhai, S. (2013b). Fitts law: Modeling finger touch with fitts’ law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ’13* (pp. 1363–1372). New York, NY, USA: ACM.
- [12] Brown, B., McGregor, M., & McMillan, D. (2014). 100 days of iphone use: Understanding the details of mobile device use. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices & Services, MobileHCI ’14* (pp. 223–232). New York, NY, USA: ACM.
- [13] Burgbacher, U. & Hinrichs, K. (2015). Modelling human performance of stroke-based text entry. In *Proc. ACM CHI 2015* (pp. 137–143).: ACM Press.
- [14] Buschek, D., De Luca, A., & Alt, F. (2015). There is more to typing than speed: Expressive mobile touch keyboards via dynamic font personalisation. In *Proc. ACM MobileHCI ’15* (pp. 125–130).: ACM Press.
- [15] Butler, K. A. (1985). Connecting theory and practice: A case study of achieving usability goals. *SIGCHI Bull.*, 16(4), 85–88.
- [16] Buxton, W. (1986). Chunking and phrasing and the design of human-computer dialogues (invited paper). In *IFIP Congress* (pp. 475–480).
- [17] Cao, X. & Zhai, S. (2007). Modelling human performance of pen stroke gestures. In *Proc. ACM CHI 2007* (pp. 1495–1504).: ACM Press.
- [18] Caramiaux, B., Bevilacqua, F., & Tanaka, A. (2013). Beyond recognition: using gesture variation for continuous interaction. In *ACM CHI EA 2013* (pp. 2109–2118).: ACM Press.
- [19] Caramiaux, B., Donnarumma, M., & Tanaka, A. (2015). Understanding gesture expressivity through muscle sensing. *ACM Trans. Comput.-Hum. Interact.*, 21(6), 31:1–31:26.
- [20] Carroll, J. M. & Rosson, M. B. (1987). : chapter Paradox of the Active User, (pp. 80–111). Cambridge, MA, USA: MIT Press.
- [21] Cockburn, A., Kristensson, P. O., Alexander, J., & Zhai, S. (2007). Hard lessons: Effort-inducing interfaces benefit spatial learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ’07* (pp. 1571–1580). New York, NY, USA: ACM.

- [22] Dix, A. (2007). Designing for appropriation. In *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...But Not As We Know It - Volume 2*, BCS-HCI '07 (pp. 27–30). Swindon, UK: BCS Learning & Development Ltd.
- [23] Felleisen, M. (1990). On the expressive power of programming languages. In *Science of Computer Programming* (pp. 35–75).: Springer-Verlag.
- [24] Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 74, 381–391.
- [25] Fuccella, V., Isokoski, P., & Martin, B. (2013). Gestures and widgets: Performance in text editing on multi-touch capable mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13 (pp. 2785–2794). New York, NY, USA: ACM.
- [26] Garcia, J., Tsandilas, T., Agon, C., & Mackay, W. (2012). Interactive paper substrates to support musical creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12 (pp. 1825–1828). New York, NY, USA: ACM.
- [27] Garcia, J., Tsandilas, T., Agon, C., & Mackay, W. E. (2014). Structured observation with polyphony: A multifaceted tool for studying music composition. In *Proceedings of the 2014 Conference on Designing Interactive Systems*, DIS '14 (pp. 199–208). New York, NY, USA: ACM.
- [28] Ghomi, E., Huot, S., Bau, O., Beaudouin-Lafon, M., & Mackay, W. E. (2013). Arpège: Learning multitouch chord gestures vocabularies. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '13 (pp. 209–218). New York, NY, USA: ACM.
- [29] Giaimo, C. (2016). the hidden messages of colonial handwriting. <http://www.atlasobscura.com/articles/the-hidden-messages-of-colonial-handwriting>. Accessed: 2017-06-29.
- [30] Godøy, R. I. & Leman, M., Eds. (2009). *Musical Gestures: Sound, Movement, and Meaning*. Routledge.
- [31] Goldberg, D. & Richardson, C. (1993). Touch-typing with a stylus. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93 (pp. 80–87). New York, NY, USA: ACM.
- [32] Grossman, T., Fitzmaurice, G., & Attar, R. (2009). A survey of software learnability: Metrics, methodologies and guidelines. In *Proceedings of the SIGCHI Conference on*

- Human Factors in Computing Systems*, CHI '09 (pp. 649–658). New York, NY, USA: ACM.
- [33] Guimbretière, F. & Winograd, T. (2000). Flowmenu: Combining command, text, and data entry. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*, UIST '00 (pp. 213–216). New York, NY, USA: ACM.
- [34] Gunawardana, A., Paek, T., & Meek, C. (2010). Usability guided key-target resizing for soft keyboards. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*, IUI '10 (pp. 111–118). New York, NY, USA: ACM.
- [35] Holz, C. & Baudisch, P. (2010). The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10 (pp. 581–590). New York, NY, USA: ACM.
- [36] Holz, C. & Baudisch, P. (2011). Understanding touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11 (pp. 2501–2510). New York, NY, USA: ACM.
- [37] Hunt, A., Wanderley, M. M., & Paradis, M. (2003). The importance of parameter mapping in electronic instrument design. *Journal of New Music Research*, 32(4), 429–440.
- [38] Iwasaki, K., Miyaki, T., & Rekimoto, J. (2009). Expressive typing: A new way to sense typing pressure and its applications. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09 (pp. 4369–4374). New York, NY, USA: ACM.
- [39] J. Santos, P. & Badre, A. (1995). *Discount Learnability Evaluation*. Technical report, Georgia Institute of Technology, USA, GVU Center Technical Reports.
- [40] Kim, S. & Lee, G. (2016). Tapboard 2: Simple and effective touchpad-like interaction on a multi-touch surface keyboard. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16 (pp. 5163–5168). New York, NY, USA: ACM.
- [41] Klokmoose, C. N., Eagan, J. R., Baader, S., Mackay, W., & Beaudouin-Lafon, M. (2015). Webstrates: Shareable dynamic media. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15 (pp. 280–290). New York, NY, USA: ACM.
- [42] Kristensson, P.-O. & Zhai, S. (2004). Shark2: A large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th Annual ACM Symposium*

- on User Interface Software and Technology*, UIST '04 (pp. 43–52). New York, NY, USA: ACM.
- [43] Kristensson, P. O. & Zhai, S. (2007). Command strokes with and without preview: Using pen gestures on keyboard for command selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07 (pp. 1137–1146). New York, NY, USA: ACM.
- [44] Kurtenbach, G. & Buxton, W. (1993). The limits of expert performance using hierarchic marking menus. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93 (pp. 482–487). New York, NY, USA: ACM.
- [45] Lee, J. Y., Hong, N., Kim, S., Oh, J., & Lee, J. (2016). Smiley face: Why we use emotion stickers in mobile messaging. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, Mobile-HCI '16 (pp. 760–766). New York, NY, USA: ACM.
- [46] Lee, S. & Zhai, S. (2009). The performance of touch screen soft buttons. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09 (pp. 309–318). New York, NY, USA: ACM.
- [47] Lesaffre, M., Leman, M., & Instituut voor Psychoakoestiek en Elektronische Muziek (Ghent, B. (2013). *The Power of Music: Researching Musical Experiences : a Viewpoint from IPEM*. Acco.
- [48] Li, Y. (2010). Gesture search: A tool for fast mobile data access. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, UIST '10 (pp. 87–96). New York, NY, USA: ACM.
- [49] Long, Jr., A. C., Landay, J. A., Rowe, L. A., & Michiels, J. (2000). Visual similarity of pen gestures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '00 (pp. 360–367). New York, NY, USA: ACM.
- [50] Lottridge, D. & Mackay, W. E. (2009). Generative walkthroughs: To support creative redesign. In *Proceedings of the Seventh ACM Conference on Creativity and Cognition*, C&C '09 (pp. 175–184). New York, NY, USA: ACM.
- [51] Mackay, W. E. (2000). Responding to cognitive overload: Co-adaptation between users and technology. *Intellectica*, 30(1), 177–193.
- [52] Mackay, W. E. (2002). Which interaction technique works when?: Floating palettes, marking menus and toolglasses support different task strategies. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '02 (pp. 203–208). New York, NY, USA: ACM.

- [53] MacKenzie, I. S. & Soukoreff, R. W. (2003). Phrase sets for evaluating text entry techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03 (pp. 754–755). New York, NY, USA: ACM.
- [54] Malloch, J., Griggio, C. F., McGrenere, J., & Mackay, W. E. (2017). Fieldward and pathward: Dynamic guides for defining your own gestures. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17 (pp. 4266–4277). New York, NY, USA: ACM.
- [55] Markussen, A., Jakobsen, M. R., & Hornbæk, K. (2014). Vulture: A mid-air word-gesture keyboard. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14 (pp. 1073–1082). New York, NY, USA: ACM.
- [56] Maudet, N., Jalal, G., Tchernavskij, P., Beaudouin-Lafon, M., & Mackay, W. E. (2017). Beyond grids: Interactive graphical substrates to structure digital layout. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17 (pp. 5053–5064). New York, NY, USA: ACM.
- [57] Nacenta, M. A., Kamber, Y., Qiang, Y., & Kristensson, P. O. (2013). Memorability of pre-designed and user-defined gesture sets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13 (pp. 1099–1108). New York, NY, USA: ACM.
- [58] Nardi, B. A. & Miller, J. R. (1990). An ethnographic study of distributed problem solving in spreadsheet development. In *Proceedings of the 1990 ACM Conference on Computer-supported Cooperative Work*, CSCW '90 (pp. 197–208). New York, NY, USA: ACM.
- [59] Neumann, P., Tat, A., Zuk, T., & Carpendale, S. (2007). Keystrokes: Personalizing typed text with visualization. In *Proceedings of the 9th Joint Eurographics / IEEE VGTC Conference on Visualization*, EUROVIS'07 (pp. 43–50). Aire-la-Ville, Switzerland, Switzerland: Eurographics Association.
- [60] Nielsen, J. (1993). *Usability Engineering*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.
- [61] Omanson, R. C., Miller, C. S., Young, E., & Schwantes, D. (2010). Comparison of mouse and keyboard efficiency. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 54(6), 600–604.
- [62] Pastel, R. (2006). Measuring the difficulty of steering through corners. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06 (pp. 1087–1096). New York, NY, USA: ACM.

- [63] Pirzadeh, A., Wu, H.-W., Bharali, R., Kim, B. M., Wada, T., & Pfaff, M. S. (2014). Designing multi-touch gestures to support emotional expression in im. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '14 (pp. 2515–2520). New York, NY, USA: ACM.
- [64] Reyat, S., Zhai, S., & Kristensson, P. O. (2015). Performance and user experience of touchscreen and gesture keyboards in a lab setting and in the wild. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15 (pp. 679–688). New York, NY, USA: ACM.
- [65] Rosner, D. & Bean, J. (2009). Learning from ikea hacking: I'm not one to decoupage a tabletop and call it a day. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09 (pp. 419–422). New York, NY, USA: ACM.
- [66] Roy, Q., Malacria, S., Guiard, Y., Lecolinet, E., & Eagan, J. (2013). Augmented letters: Mnemonic gesture-based shortcuts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13 (pp. 2325–2328). New York, NY, USA: ACM.
- [67] Salvucci, D. D., Taatgen, N. A., & Borst, J. P. (2009). Toward a unified theory of the multitasking continuum: From concurrent performance to task switching, interruption, and resumption. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09 (pp. 1819–1828). New York, NY, USA: ACM.
- [68] Simon, M. (2015). Macworld the hidden messages of colonial handwriting. <https://goo.gl/mFFDBZ>. Accessed: 2017-06-29.
- [69] Spelmezan, D., Appert, C., Chapuis, O., & Pietriga, E. (2013a). Controlling widgets with one power-up button. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13 (pp. 71–74). New York, NY, USA: ACM.
- [70] Spelmezan, D., Appert, C., Chapuis, O., & Pietriga, E. (2013b). Side pressure for bidirectional navigation on small devices. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '13 (pp. 11–20). New York, NY, USA: ACM.
- [71] Tsandilas, T., Appert, C., Bezerianos, A., & Bonnet, D. (2014). Coordination of tilt and touch in one- and two-handed use. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14 (pp. 2001–2004). New York, NY, USA: ACM.
- [72] Tu, H., Ren, X., & Zhai, S. (2015). Differences and similarities between finger and pen stroke gestures on stationary and mobile devices. *ACM Trans. Comput.-Hum. Interact.*, 22(5), 22:1–22:39.

- [73] Tulving, E. (2000). *The New Cognitive Neurosciences*, chapter Introduction to Memory. Cambridge, MA: MIT Press, 2nd ed.: London, England.
- [74] Vertanen, K. & Kristensson, P. O. (2014). Complementing text entry evaluations with a composition task. *ACM Trans. Comput.-Hum. Interact.*, 21(2), 8:1–8:33.
- [75] Viviani, P. & Terzuolo, C. (1982). Trajectory determines movement dynamics. *Neuroscience*, 7(2), 431–437.
- [76] Wickens, C. D., Hollands, J. G., Banbury, S., & Parasuraman, R. (2015). *Engineering psychology & human performance*. Psychology Press.
- [77] Woods, W. A. (1983). What's important about knowledge representation? *IEEE Computer*, 16(10), 22–26.
- [78] Zhai, S., Kristensson, P., Appert, C., Andersen, T., & Cao, X. (2012a). Foundational issues in touch-surface stroke gesture design: An integrative review. *Found. Trends Hum.-Comput. Interact.*, 5(2), 97–205.
- [79] Zhai, S., Kristensson, P., Appert, C., Andersen, T., & Cao, X. (2012b). Foundational issues in touch-surface stroke gesture design: An integrative review. *Found. Trends Hum.-Comput. Interact.*, 5(2), 97–205.
- [80] Zhai, S. & Kristensson, P.-O. (2003). Shorthand writing on stylus keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03 (pp. 97–104). New York, NY, USA: ACM.
- [81] Zhai, S. & Kristensson, P. O. (2008). Interlaced qwerty: Accommodating ease of visual search and input flexibility in shape writing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08 (pp. 593–596). New York, NY, USA: ACM.
- [82] Zhai, S. & Kristensson, P. O. (2012). The word-gesture keyboard: Reimagining keyboard interaction. *Commun. ACM*, 55(9), 91–101.
- [83] Zhao, S. & Balakrishnan, R. (2004). Simple vs. compound mark hierarchical marking menus. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04 (pp. 33–42). New York, NY, USA: ACM.

Titre : Augmenter le Pouvoir d'Expression de l'Interaction Gestuelle sur les Appareils Mobiles

Mots clés : interaction mobile, communication expressive, interaction gestuelles, sélection de commande, co-adaptation

Résumé : Les interfaces mobiles actuelles permettent aux utilisateurs de manipuler directement les objets affichés à l'écran avec de simples gestes, par exemple cliquer sur des boutons ou des menus ou pincer pour zoomer. Pour accéder à un espace de commande plus large, les utilisateurs sont souvent forcés de passer par de nombreuses étapes, rendant l'interaction inefficace et laborieuse. Des gestes plus complexes sont un moyen puissant d'accéder rapidement à l'information ainsi que d'exécuter des commandes plus efficacement [5]. Ils sont en revanche plus difficiles à apprendre et à contrôler. Le "Gesture Typing" (saisie de texte par geste) est une alternative intéressante au texte tapé: il permet aux utilisateurs de dessiner un geste sur leur clavier virtuel pour entrer du texte, depuis la première jusqu'à la dernière lettre d'un mot.

Dans cette thèse, j'augmente le pouvoir d'expression de l'interaction mobile en tirant profit de la forme et la dynamique du geste et de l'espace d'écran, pour invoquer des commandes ainsi que pour faciliter l'appropriation dans différents contextes d'usage. Je conçois *Expressive Keyboard*, qui transforme la variation du geste en un résultat riche et je démontre plusieurs applications dans le contexte de la communication textuelle. Et plus, je propose *CommandBoard*, un clavier gestuel qui permet aux utilisateurs de sélectionner efficacement des commandes parmi un vaste choix tout en supportant la transition entre les novices et les experts. Je démontre plusieurs applications de *CommandBoard*, dont chacune offre aux utilisateurs un choix basé sur leurs compétences cognitives et moteur, ainsi que différentes tailles et organisations de l'ensemble des commandes.

Ensemble, ces techniques donnent un plus grand pouvoir expressif aux utilisateurs en tirant profit de leur contrôle moteur et de leur capacité à apprendre, à contrôler et à s'approprier.

Title : Increasing The Expressive Power of Gesture-based Interaction on Mobile Devices

Keywords : mobile interaction, expressive communication, gesture-based interaction, command selection, co-adaptation

Abstract : Current mobile interfaces let users directly manipulate the objects displayed on the screen with simple stroke gestures, e.g. tap on soft buttons or menus or pinch to zoom. To access a larger command space, the users are often forced to go through long steps, making the interaction cumbersome and inefficient. More complex gestures offer a powerful way to access information quickly and to perform a command more efficiently [5]. However, they are more difficult to learn and control. Gesture typing [78] is an interesting alternative to input text: it lets users draw a gesture on soft keyboards to enter text, from the first until the final letter in a word.

In this thesis, I increase the expressive power of mobile interaction by leveraging the gesture's shape and dynamics and the screen space to produce rich output, to invoke commands, and to facilitate appropriation in different contexts of use. I design *Expressive Keyboard*, that transforms the gesture variations into rich output, and demonstrate several applications in the context of text-based communication. As well, I propose *CommandBoard*, a gesture keyboard that lets users efficiently select commands from a large command space while supporting the transition from novices to experts. I demonstrate different applications of *CommandBoard*, each offers users a choice, based on their cognitive and motor skills, as well as the size and organization of the current command set. Altogether, these techniques give users more expressive power by leveraging human's motor control and cognitive ability to learn, to control, and to appropriate.

