Why interaction methods should be exposed and recognizable

Sylvain Malacria

Why interaction methods should be **exposed** and **recognizable**

Sylvain Malacria

Habilitation à diriger des recherches defended on October 25th 2023
at University of Lille, ED MADIS-631

Jury Members:

Caroline Appert, Reviewer
Directrice de Recherche, CNRS

Wendy Mackay, Reviewer
Directrice de Recherche, Inria

Duncan Brumby, Reviewer
Professor, University College London

Mira Dontcheva, Examiner
Principal Scientist, Adobe Research

Emmanuel Dubois, Examiner
Professeur des universités, University of Toulouse

Stéphane Huot, Garant
Directeur de Recherche, Inria

Lionel Seinturier, Président
Professeur des universités, University of Lille

ABSTRACT

Interaction with interactive systems mostly relies on command selection, enabling users to specify their intentions to the system. This habilitation manuscript discusses the design of command selection techniques on desktop and touch-based computers, which today require far too much from users: to know what they can do, to know how it can be done, and to improve by themselves on doing it. I argue for a design of Graphical User Interfaces (GUIs) that would rely more on users' recognition skills than their explicit memorization and recall skills. I present first the work I have conducted on the transition between a recognition-based method (e.g. selecting using a pointer in a menu) and a recall-based interaction mode (e.g. keyboard shortcuts) and reflect on how it might not be possible to perfect it. Therefore, I then explore whether it is possible to design highly efficient techniques with which users can still rely on recall whenever they want. Finally, I present work conducted on how GUIs could be refined to help uses discover that they can use certain input modalities with interactive systems.

ACKNOWLEDGMENTS

The work described in this habilitation was conducted over a period of twelve years. So much has happened in that time... I had the chance to live literally on the other side of the globe. And yet, as remote as I could be, I knew there would always be a place a could call home and come back to, thanks to my family. My parents, Gisèle and Max, have always been there for me. So has my sister, Anne, who in addition granted me with the absolute best niece, Gaïa. To all my family, thank you for your love, support, and acceptance. I know I can be hard to bear at times, but to be fair, so are you. I guess we are even :) Regardless, I will always be there for you, no matter what.

These twelve years have also largely been shared with someone who has been very important to me and, I suppose, will always be to some extent. Thank you for having tolerated me that long. I wish things may have ended differently, but this is how things had to go. Regardless of where you are and what you do now, I sincerely hope you are happy.

If by any chance, I have defined myself as a researcher in these twelve years, it has everything to do with senior researchers who were kind enough to teach me, explicitly or implicitly, during that time. To Eric Lecolinet who offered me the opportunity to discover the amazing world of HCI research and supported me for almost 4 years. To Andy Cockburn who supervised me in New Zealand and rapidly made me realize that research is first and foremost a human adventure in which you should never be afraid to try new things. To Carl Gutwin whose first discussion with enlightened me that a myriad of approaches can and should be employed to tackle a problem. To Géry Casiez, with whom I have collaborated, and learned from, a lot. I know we have our differences and disagreement, but I believe this is what makes our collaborations so enriching and successful. To Nicolas Roussel, former Mjolnir group leader, now busy with important things in the almost real world, who supported me and was kind enough to accept me in his research group. I will always be thankful for that. To Stéphane Huot, Loki group leader, perhaps soon busy with important things in the almost real world, who accepted to be my "garant" for this habilitation. I know you have a lot on your plate, yet you are always here when we really need you. And of course, to Edward Lank, who taught me so much during our collaboration. The best always go first. You are deeply missed.

I have always seen research as a team effort. As such, this work could not have been done without the help of my colleagues in Télécom, Canterbury,

London and Lille, and collaborators all around the world. I also made so many friends in the HCI community during these years. In particular, I simply cannot thank enough the BuDuFoDuCo members. Anne, thank you for having been the only one to call me when it mattered. I did not realize at that time how important it was. Thank you again. Dong-Bach and Aurélien, thank you for being so empathetic and patient. You are everything a friend can hope for. Thomas, thank you for always being so joyful. Your infectious good humor is important and compensates for my lack of it in our office. Gilles, I wish we could see each other more often but thankfully, we collaborate a lot which is the perfect excuse to have to see and call each other more than occasionally. Simon, thank you for your biting humour and very often having the 90s music references. Quentin, even if you are the "youngest" in this group, I am extremely happy you joined it and hope you'll stay around. But researchers based and met in Lille are not to be outdone. Mathieu, Fanny, Alix, Axel and Bruno, I am so happy we met, from a personal and professional perspective. We shared amazing moments and I know we will share so many more of them. Grégoire, thank you for having pushed for bouldering events. Even though I feel like I am more ruining my right hand rather than rehabilitating it. This is fun, almost as fun as Karaoke.

The most rewarding aspect of being an academic researcher is probably having the chance to help PhD students achieve their potential. Nicole, Axel, Damien, and Eva, my work is done now, and I know you're more than ready to stand on your own two feet. Constant, Raphaël, Vincent and Suliac, we still have some way to go, but I know your work will reach its final destination.

Life is more than research, and nothing would have been possible without the long-lasting support of my friends. Bunty and Wenwen, you have always been there for me and know you always will be. I hope you know I will always do my best to live up to your expectations. To the "Tontons" as well. Abde, Antoine and Joe, I am glad we are still that close. Even though the mafia killed breaking and mainstream events are incredibly boring nowadays, the BOTY is this year or never. I am also grateful to the amazing friends I made in Lille. Hanaë, Eulalie, Gaël, Vincent, Sara and Philippe, thank you for making me enjoy this city as soon as I arrived.

I would like also to thank the committee members for accepting to review this habilitation. Your work inspired my research on so many levels. It was an honor for me to talk about it in front of you and hope you enjoyed the presentation.

Finally, thank you to Maddy Makes Games for making Celeste, and to Mobius Digital for making Outer wilds. You should all play these games if you have not. They are life changers and very often on sale. You have no excuse.

“Only one thing makes a dream impossible to achieve: the fear of failure”
— Paulo Coelho

“Fear of failure is a poor reason not to try”
— Poke

TABLE OF CONTENTS

1 INTRODUCTION

Interaction with the computing systems that were introduced in the previous century strongly differ from how one can interact with a computer today. Indeed, when computers appeared around the time of the Second World War, they relied on an interaction paradigm inherited from, or to the least strongly influenced by, existing technologies that were used at the time for communication between humans: punch card machines and teletypes. Using the teletype, the operator could simply type in a line of text in the system console and then hit the return key to send that line to the computer. While not yet named that way, this interaction paradigm corresponded to a Command Line Interface (CLI), a type of user interface that processes commands to a computer program in the form of lines of specifically structured text inputs.

While extremely used at that time, CLIs suffered, and still do, from poor accessibility and usability [104], and require users to know beforehand the textual language required to communicate with the system. As a result, CLIs have very quickly been challenged by Graphical User Interfaces (GUIs), an other interaction paradigm relying mostly on visual presentation of interaction possibilities. By using graphical components displayed on-screen, it makes it possible for users to visually inspect the interface in order to recognize the controls they might want to interact with. Since then, CLIs and GUIs have cohabited to some extent in almost every major computing system, from desktop computer to touch-based devices, opposing two very distinct interaction paradigms: interaction techniques like CLIs that requires user to recall what actions are possible and how to perform them; and GUIs that rely on visual recognition to expose interaction possibilities and mediate user interaction. Graphical User Interfaces have however mediated most interactions between users and computing systems since. The success of GUIs can be easily explained by their direct support for users that are unfamiliar with the interface, as they can rely on visual search and direct manipulation[1] mechanisms to browse and interact with an interface they never interacted with. It is however often admitted, that what makes GUIs so great tends also to limit how performant users can be with these interfaces [106]. The simple task of selecting a command in a menubar is symptomatic of this situation. Even when

1 in the sense of Ben Shneiderman [110]

familiar with the interface, the mechanic of pointing is limited by a certain cap of performance corresponding to the user performance limit to operate the pointing device for that specific sequence of pointing actions [106]. In order to overcome this performance limit, alternative and theoretically more efficient methods have been proposed. For instance, in the context of command selection, *keyboard shortcuts* [16] allow a wide range of commands, each to be selected by holding one or more modifier keys (e.g. Ctrl, Alt, etc.) and subsequently hitting the key associated with the command (called a *hotkey*). This type of interaction method allow high performance ceilings but require users to have seen and memorized, beforehand, the corresponding key combination, thus coming back to an interaction paradigm less accessible and relying on recall.

These two paradigms have long been opposed and as a result, have impacted how user interfaces were designed since the 1980s. Typically, the above-mentioned example of keyboard shortcuts suggest that they might be recall-based "by nature", that is that they necessarily have to be seen and memorized beforehand, while they are actually recall-based "by design": is there any reason to *require* users to have to know the corresponding shortcut beforehand? Is it really impossible to provide a rapid command selection mechanism while still allowing users to mostly rely on visual inspection of the interface? Overall, my position on this question is that Interactive Systems still ask too much from the user, especially in terms of what they have to know and remember the interface offers.

In this document, I will report on my work conducted on recognition and recall based interaction techniques, mostly in the context of command selection.

In the first chapter, I will present work conducted on the transition from recognition-based to recall-based interaction modes, also known as the intermodal transition [36, 79, 106]. Two major problems arise from the intermodal transition. On one hand, a *performance dip* problem that occurs at the time of the transition [106], which is a temporary yet significant possible loss of performance resulting from the efforts needed to learn the *"shortcut"* method. On the other hand, regardless of the magnitude of the performance dip, transitioning to the best interaction technique will always require the user to decide at some point to switch to this novel interaction method. Therefore, I will first present a work focused on trying to encourage and accelerate this transition [HDR11]. Then, I will insist about the necessity to "recall", and describe work that "facilitate" this recall [HDR13], or try to alleviate recall difficulties [HDR5].

The intermodal transition issue and the above-mentioned two problems ironically yield from the fact that two modes co-exist: one based on recognition, one based on recall. These modes co-exist allegedly because it is often stated that interaction based on recognition cannot be of high efficiency, that is that it does not offer a high performance ceiling [67]. I will thus present in the second chapter work that I conducted on the design of *recognition-based* interaction mechanisms, that is they natively display information to guide users' interaction, while yet efficient. More interestingly, these techniques do not rely on explicit learning and recall from the user. I will present variants of such techniques for selecting commands via keyboard shortcuts on a desktop [HDR4, HDR10] but also on touch-based devices [HDR2, HDR3], as well as adapted to a famous unidirectional gesture-based menu system known as marking menu [HDR7].

In the third chapter, I reflect on the importance of making interaction possibilities visible and recognizable, regardless of their efficiency, in order to maximize the odds that they will be *discovered* by the users [77]. In other words, this means that the interface and interactive systems should be designed to *communicate* efficiently and effectively to users their underlying design intent and interactive principles [42]. In that respect, I will first present a work that explored how the design of the GUI could communicate force-based text selection input on a touch-based device [HDR6]. I will then present a study conducted in which we explored how different graphical designs of a simple button could communicate different *in-place touch inputs*, single finger contacts that do not move across the surface such as dwell, double-tap or forcepress [HDR9]. Finally, I will present a work exploring how *single-handed microgestures* could be visually and intelligibly presented to the user [HDR8].

Finally, I will conclude this manuscript by reflecting on why, in my humble opinion, interactive systems are simply asking too much from the user by requiring them to discover what is hidden and remember how to operate efficiently the systems, while their GUIs could be designed to encompass this information and visually convey them to the user.

2

RECOGNITION VS RECALL: THE CASE OF THE INTERMODAL TRANSITION

2.1 THE INTERMODAL TRANSITION FRAMEWORK

Similarly to any activity, interacting with an interactive system requires the acquisition of certain skills. Overall, skill acquisition (or performance level improvement) is complex, and far from being perfectly understood [50, 96, 107]. It has, however, often been simplified as being correlated with practice, with performance level over time that tends to follow a power function known as the power law of practice [91, 113, 114]. Basically, whether we're learning to play piano or tennis, performance improves over time, and moves from what we would call a novice level to an intermediate one, until ultimately becoming expert, where performance barely improves anymore.

But skill acquisition is not always that simple as it is often the result of sequences of different skill acquisition [46, 50, 116]. A good use case to illustrate this is to look at learning how to ride a bicycle. A common approach to teach how to ride a bicycle is to add training wheels at the back of the bicycle for stability, so the learner, most of the time a child, can pedal, steer, get used to the bicycle, and become more confident on it. However, these training wheels, that provide support, also limit performance while cycling. For example, it is impossible to tilt a bicycle equipped with training wheels, thus making it difficult to learn that a bicycle must lean in order to maintain balance in a turn. To the contrary, when equipped with training wheels, the balance of the bicycle is almost exclusively maintained by the training wheels themselves. Therefore, the child becomes an expert at cycling *with the training wheels*, but reaches a relatively low cap of performance at cycling.

In order to improve further, training wheels have to be removed at some point. Removing the training wheels leads to a different problem: the learner has not improved at all on one of the most important aspect of cycling, which is that velocity creates a gyroscopic force that plays a significant role in stabilizing the now bicycle. As a result, removing the training wheels significantly impacts user experience with it, and may imply a temporary, but substantial, reduction in performance, a *'performance dip'* (Figure 2.1), that is likely to encourage the learner to want the training wheels back and refuse to try again

without them. In addition, it is important to note that the decision of removing the training wheels is often made by the parents, not the child. Without an explicit decision from the parent, the child would probably continue to rely on these training wheels.

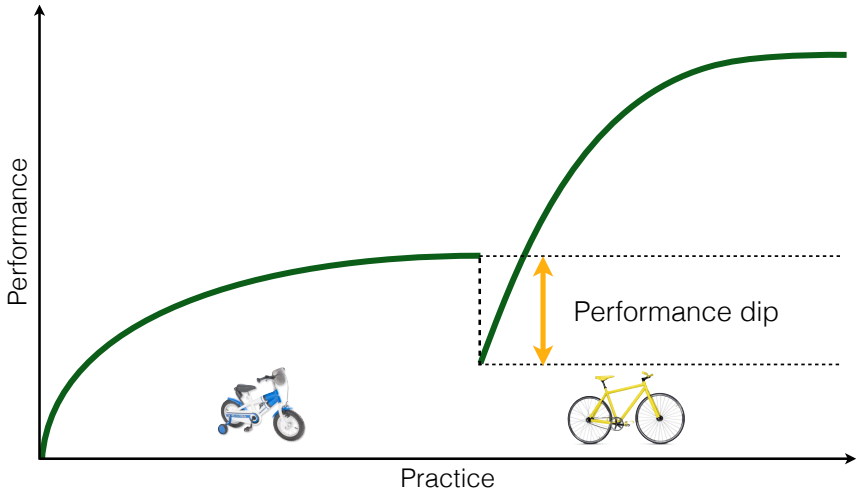

Figure 2.1: Characterization of a learner's performance when practicing cycling with training wheels, and then transitioning to cycling without the training wheels.

Douglas Engelbart frequently opposed a tricycle and a bicycle to illustrate the difference between the ease of use and the high-performance: a tricycle is easy to use, but requires efforts to travel even a short distance, while the bicycle requires more skill and practice, but with an effort-to-performance ratio that is dramatically higher [1].

Coming back to interactive systems, a parallel can be made between the above-mentioned phenomenon of learning cycling and interacting with Graphical User Interfaces. The simple task of selecting a command in a menubar is a great example. Unfamiliar users will browse the menu using the mouse pointer, visually inspect the interface to search and locate the target commands, and click to activate them. Once familiar with the interface, users have built their spatial memory of the interface and know where the interesting commands are located. They can directly move the pointer on them without having to visually inspect the interface and search for the commands, and are still able to visually confirm that the location is correct. The mechanic of pointing is however limited to a relatively low performance cap, resulting from

humans' biomechanics limit when operating the mouse pointer. In order to overcome this performance cap in the context of command selection, alternative and theoretically more efficient methods have been proposed. Keyboard shortcuts [16] are an example of such method. They allow a wide range of commands, each to be selected by holding one or more modifier keys and subsequently hitting the key associated with the command (the hotkey). Keyboard shortcuts allow a theoretical higher performance ceiling but require users to have seen beforehand and memorized the corresponding key combination.

In the end, in the context of desktop command selection, that is of mouse-based and hotkey command selections, the user is confronted to two distinct methods of command selection that simply oppose two philosophies of how one could interact with a system. The first is that GUIs are great because they are "recognizable", easy to discover, and efficient when interacting with an unfamiliar interface but performance with them is however limited to a relatively low ceiling. The second is that shortcuts can be used by "recalling" specific actions in order to more rapidly interact with the system. While less accessible, this is theoretically faster in execution, the word *shortcut* itself implicitly reminding that conventional GUIs relying on recognition are less efficient and do not offer the same performance ceiling.

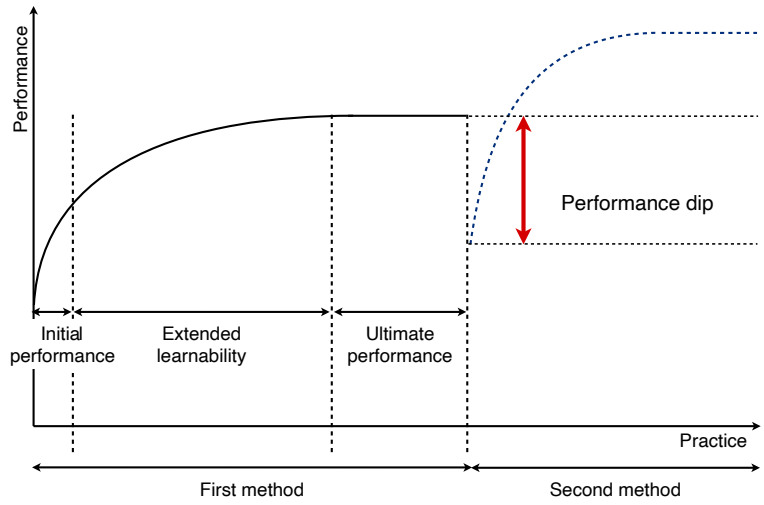

Figure 2.2: Characterization of the intermodal transition from one method (e.g. selecting a command by pointing its corresponding item in a menubar) to a second one more efficient (e.g. selecting it using its keyboard shortcut) [106]

Many interfaces support more than one interaction method for performing the same operation, and once a user approaches their performance limit with one method, performance improvement inevitably will require shifting to another method that offers a higher performance ceiling. In the context of command selection, the transition from pointer-based command selection to shortcut based command selection has been characterized in the intermodal transition framework illustrated Figure 2.2 introduced by Scarr et al. [106], a framework similar to the bicycle example detailed above. Basically, users progress with pointer-based selection until a certain point where they may decide (or not) to switch to a shortcut-based command selection mechanism. The transition from the pointer- to the shortcut-based mechanism implies a performance dip that might be the result of various factors, such as recall time or cost of errors. Research investigating this intermodal transition can be organized around two major axes: How to encourage users to switch to the most efficient method? How to minimize performance dip when that switch happens?

In this chapter, I will first detail a work I conducted on how reflective interface could encourage the transition to a more efficient method. I will then present a project in which we designed a gestural command selection technique that aimed to facilitate the memorization and recall of the shortcuts. Finally, I will present a work in which we tried to minimize the risks of errors when transitioning.

2.2 BACKGROUND

2.2.1 *Encouraging the transition*

In spite of the existence of shortcut methods that support a higher performance cap, users do not always make the decision to switch to this more efficient alternative [70, 112]. Several reasons can explain why users do not switch.

A first reason might simply be that users are not aware of this most efficient method. Indeed, as an example, previous research conducted on the adoption of keyboard shortcuts suggests that labels displayed in drop down menus might frequently be ignored by users [51]. In order to overcome this problem, several projects explored the use of feedback after command selection that would purposely emphasize hotkey existence. Krisler and Alterman's HotKeyCoach [65], which pops up a window that displays the corresponding hotkey and need to be explicitly dismissed every time the user selected a

command with the mouse, showed improved hotkey usage over time. Similarly, one of the most efficient method tested in Grossman et al.'s study on strategies for accelerating hotkey adoption [51] was *Audio feedback*, another feedback-based approach where both the command and hotkey are played using speech feedback when a command is selected using the mouse. Finally, the Blur interface [106] relies on what the authors call *calm notifications*, a transitory feedback revealing the command that is equivalent to each mouse initiated action. These projects all demonstrated that feedback provided after command selection with the theoretically less efficient modality, and emphasizing the theoretically more efficient one, increased usage of the efficient command selection method but do not guarantee adoption.

Indeed, another possible reason for users to not switch might however simply be that they opt to "satisfice" [112] with the interaction method they have been practicing with, often for a long time. The word *satisficing* is a portmanteau of satisfy and suffice, that more generally describes a decision-making strategy where users stick with strategies that offer a suboptimal, yet well known, result that reach an acceptability threshold. This is possibly why in their framework of interface expertise, Scarr et al. [106] insist that users' *perception* of a modality's efficiency is a critical factor influencing their decision of whether to use a modality. However, we know that a user's perception of performance is often unreliable [40]. Typically, in the context of command selection, several studies have shown that hotkeys offer a higher performance ceiling than mouse selection [70, 95], yet Tak et al. found that some participants failed to use hotkeys they already knew because they believed the mouse-based alternative, in their case toolbar buttons, to be faster [115].

Motivation also may play a critical role in the adoption of the most efficient input method. Interestingly, however, Tak et al. also found that an increased level of time pressure did not lead to increased hotkey use [115], suggesting that workload alone is not sufficient to encourage adoption of more efficient methods. This might be the result of the "Paradox of the active user" [25], a concept developed by Caroll and Rosson suggesting that users might be too engaged in their primary task to decide to dedicate cognitive effort into learning more efficient ways to accomplish that task. That being said, several studies suggest that social influence might trigger alternative behavior from the users. Typically, Banovic et al. [18] showed that it was a significant factor in encouraging tool palette customization, while Peres et al. [97] found that users were more likely to use keyboard shortcuts if they worked with others who do.

Finally, switching and adopting the recall-based mode of a technique requires users who are still in the associative phase of learning to 'launch out'

and trust their memory. However, the less practiced is the user, the more they are likely to make errors (e.g. by recalling an incorrect key combination in the case of keyboard shortcuts). Although recall errors are part of any memory-based learning process, the relative perceived cost of these errors can present a major barrier to development of expertise with any type of user interface. Many people are averse to making errors, especially when the cost of the error is high (e.g. performing the ⌘+Q keyboard shortcut that quits an application instead of ⌘+A that is used for the "select all" command). And even if the consequences of an error are not that important, users are likely to have a perception of the cost of its error that is much higher than the reality because of a negativity bias in which "negatives loom larger than positives" [64]. Therefore, the risk-taking approach required to switch combined with the perceived cost of potential error are likely to increase users perception of the performance dip occurring when switching.

2.2.2 *Minimizing performance dip*

The performance dip of switching can be of diverse magnitude and is impacted by various factors. In the case of the transition from mouse-based to hotkey command selection in desktop computer environments, the transition requires to a) memorize the corresponding hotkey (which can require significant recall time but also cause errors) and b) use a different input device which requires users to practice the typing of the key combination [16, 51].

2.2.2.1 *Mnemonic mappings*

A method to alleviate the associated cost of having to transition to a second modality is to rely on a mnemonic mapping between the first and the second modality. Semantically speaking, a mnemonic mapping is a mapping that is meant to *"aid one's memory"*[1]. In practice, this aid can be of diverse nature. Typically, in the context of keyboard shortcuts and already deployed in commercial system, it can be a key combination that relies on a character that is supposedly easier to memorize for the user such as the first letter of the command name (e.g. S for Save, C for Copy, P for Print or F for Find). In the context of gestural shortcut, it can be a symbol in line with the effect of the command, for instance using a star symbol to set an item as a favorite. Unfortunately, using a mnemonic mapping is not a safe-for-all solution as it still, in the end, relies on recall. Moreover, a given mapping cannot be applied to

1 https://www.collinsdictionary.com/dictionary/english/mnemonical

all commands since only a limited number of characters or symbols are available. As such, collisions are swiftly encountered when the set of commands is very large. It is thus required to use characters or symbols that do not rely on obvious mnemonics (e.g. X associated with Cut since C is already associated with Copy) or require pressing additional modifier keys (e.g. ⌘+⇧+F for "Find in File") making the execution of the shortcut more complicated and potentially slower.

2.2.2.2 *Principle of rehearsal*

Another method to minimize the performance dip can be found in the principle of rehearsal that characterize rehearasal-based interfaces. The principle of rehearsal was defined by Gordon Kurtenbach as the fact that *"guidance should be a physical rehearsal of the way an expert would issue a command"* [67].

This principle of rehearsal is at the design core of Marking Menus [66, 67], a command selection technique that relies on gestural interaction, as opposed to pointing operations in conventional GUIs. Marking Menus are radial menus that allow the user to select a command by "making a mark" in the direction of the desired menu item. These menus rely on two main modes of interaction. The first, called the *menu* mode (sometimes *novice* mode), supports command selection by pressing and holding down an input device (e.g. a mouse button), waiting for a certain delay for the menu to display (called press-and-wait), dragging the pointer in the direction of the desired command until being over it, and finally releasing the input device to activate the command. The second mode, called the *mark* mode (sometimes *expert* mode), supports command selection by directly drawing the mark corresponding to the desired command, without waiting for the menu to appear on-screen.

Because physical operations are similar in both modes, this design facilitates a "smooth" transition from menu to mark mode by simply repeating command selections – implementing the principle of "rehearsal". In other words, even though it relies on directional gestures that do not necessarily leverage any specific mnemonic mapping, users "rehearse" command selection in mark mode by repeatedly selecting commands in menu mode. Indeed, commands are selected using almost the same physical actions (a directional gesture) but with different dynamics, that is not waiting at the beginning in the case of mark mode at the cost of not having the visual aid. The principle of rehearsal might facilitate the intermodal transition, and thus reduce the corresponding performance dip, but users still need to memorize the corresponding gesture when selecting command without a visual aid. These marking menus were therefore not error-friendly as the user could not reveal

the menu once entered in mark mode. Kurtenbach's proposed solution was to introduce a *mark confirmation* mode (sometimes *intermediate* mode) [67], where the user can dwell for a predetermined delay after having begun the gesture and switch back to menu mode — thus displaying the menu on screen and allowing re-selection or a cancellation.

Marking menus have inspired a large body of related research [15, 47, 100, 122–124]. They also inspired the design of different command selection techniques implementing the principle of rehearsal (Rehearsal-Based Interfaces, or RBIs).

A notorious example is Octopus [19], that can be seen as a variant of Marking Menus. It is also a unistroke-gesture command selection mechanism using a delay as mode delimiter. That is, if users dwell for a certain time, the menu appears, but if they already know the gesture corresponding to the desired command, they can execute it directly. However, Octopocus supports arbitrary unistroke gestures, not only directional ones, and assist users in selecting these gestures via a guidance system in menu mode which continuously updates the elements of the menu that are displayed to progressively hide the ones that do not correspond to the gesture being performed.

Platform specific RBIs were also proposed. This is the case of MarkPad [48] and InOutPad [20], command selection techniques for trackpads that implement the principle of rehearsal, utilizing a delay prior to displaying the menu interface. A notorious example for touch-based handheld devices is FastTap [54], a command selection technique that displays commands in a spatially-stable grid-based overlay interface. Users can display the interface grid by dwelling on a button located in the bottom left corner of the screen and select a command by tapping an element of the grid without releasing the first finger. In the end, selecting a command with FastTap corresponds to perform a chord gesture on the touchscreen. Therefore, once the command location is known and memorized, users can select commands by performing a two-finger tap on the touch-screen. Finally, rehearsal-based command selection techniques for larger touch-based devices, such as tabletops [13, 49, 72] or interactive walls [52], have also been proposed.

Marking Menus have been highly influential since their introduction, and the principle of rehearsal has been replicated in many interaction techniques in the last 25 years, too many to detail them all [13–15, 47, 72, 100, 122–124].

2.3 SKILLOMETERS: REFLECTIVE WIDGETS THAT MOTIVATE AND HELP USERS TO IMPROVE PERFORMANCE

In the context of intermodal transition, it is not unreasonable for some users to not adopt the allegedly more efficient method. Indeed, adopting this novel interaction modality can require a significant investment of time and effort, that may overweight the potential efficiency gains. Given that the users have no way of estimating these associated costs and benefits, one should not be expected to switch. In the end, it is often users' perception of the novel interaction modality that matters [106], rather than its actual properties. Therefore, if a system could inform the user about the benefits of adopting a new mechanism or strategy, they might be more inclined to do so.

During my post-doctoral contract in New Zealand working with Prof. Andy Cockburn and collaborating with his PhD Student at the time Joey Scarr, with Prof. Carl Gutwin and with Dr Tovi Grossman, we investigated the use of reflective interfaces that could be used to encourage users to improve their interaction performance. More precisely we introduced the generic concept of *Skillometers*, graphical widgets designed to accelerate the development of UI expertise by allowing users to visualize their performance.

We defined a *Skillometer* as a reflective widget that is designed to motivate and help the user increase their level of performance with an interface, and supporting them in understanding meta-aspects of their interaction (meta-level aspects concerning not only the user's level of performance, but also others aspects such as the task strategy or how performance could be improved). In that sense, *Skillometers* are a generic type of widgets, that can be targeted at several distinct domains of user performance when interaction with an interactive system: Intramodal improvement (progression with one single modality), Intermodal improvement, Vocabulary Extension (raising awareness of available functionalities), and Strategic Improvement (help user to optimize their sequence of actions in order to complete a task, later renamed as Task Mapping in [36]).

Regardless of the targeted domain, there are four sub-goals that apply to the user interface of a Skillometer:

1. Clearly communicate to the user that their performance can be improved
2. Provide specific guidance about how that improvement could take place
3. Motivate the user to improve (using social influence, gamification, etc.)
4. Aim to provide information when it is most useful to the user, while still minimizing disruption of the user's normal workflow

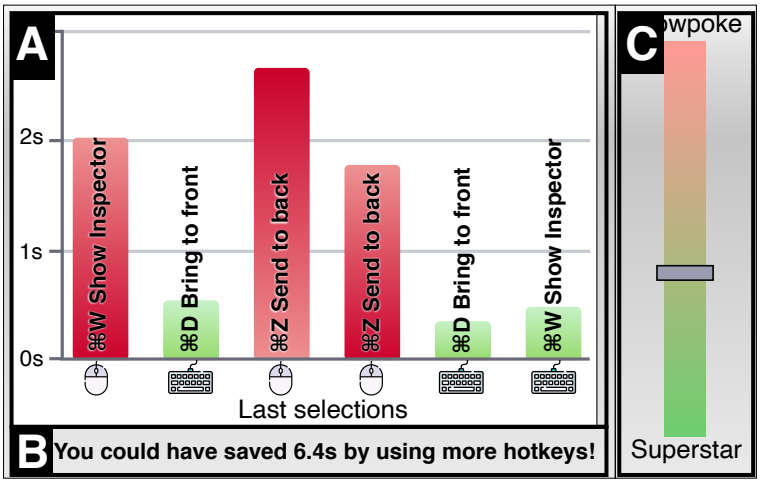

Figure 2.3: The visual design of the HotkeySkillometer (with A, B and C overlays for illustration purpose). Zone A shows a bar chart that shows the time taken to select the most recent commands. The height of the bar is proportional to the estimated command selection time, while its color depends on the modality used (red for mouse, green for keyboard shortcut). Each bar is overlaid with the name of the command and corresponding keyboard shortcut. Zone B shows motivational text with an estimation of how much time could have been saved by using more keyboard shortcuts (*hotkeys* on the figure). Zone C shows a meter that grades user's performance based on the modality used for last command selections.

The research publication accompanying the Skillometers project [HDR11] elaborates further the research domains a skillometer can apply to, as well as the different sub-goals of a skillometer. I invite the reader to refer to this publication if details are desired. In the following, I describe a specific instance of skillometer: the *HotkeySkillometer*.

2.3.1 *HotkeySkillometer: a skillometer for promoting hotkey use*

The HotkeySkillometer was designed, implemented and evaluated with the motivation of encouraging the intermodal transition in the context of desktop command selection. The main goal of the HotkeySkillometer was to motivate and assist users in transitioning from mouse-driven interaction methods (menu and toolbar selections) to keyboard shortcuts. More precisely, the HotkeySkillomoter monitors users' interactions with traditional WIMP appli-

cations, analyzing and depicting their level of performance, and displaying alternative and faster keyboard shortcut methods for executing commands whenever they are available. Its display consists of three parts, labelled A, B, and C in the figure 2.3.

Part A (left of Figure 2.3) uses a bar chart to show an estimation of the time taken to select each of the six most recently selected commands. When a new command is selected, an animation slides a new bar item into the view, scrolling all other items leftwards. Each bar is colored depending on the selection modality (red for mouse, green for keyboard shortcut), and an icon below each bar also depicts the modality that was used. Additionally, each bar is overlaid with a text displaying the name of the command and its corresponding keyboard shortcut.

Part B (bottom of Figure 2.3) shows a motivational text advising the user that keyboard shortcut selections are faster. It also estimates, across the past 6 selections, how much time could have been saved if the selections were performed with keyboard shortcuts instead of the mouse.

Part C (right of Figure 2.3) is a meter that grades the user's performance based on the modality used for the last 6 command selections – if all selections were made with keyboard shortcuts, the slider will be at the 'Superstar' end; if all selections were made using the mouse, it will be at the 'Slowpoke' end.

By default, the HotkeySkillometer is a transient window located at the bottom right of the display. As soon as the user selects a command or presses the ⌘ key, the window fades-in and remains until the user stops selecting commands. If the mouse pointer moves over the HotkeySkillometer, it fades-out revealing the widgets beneath it, enabling the user to interact with these underlying widgets.

More details about the implementation of the HotkeySkillometer can be found in the associated research paper [HDR11].

2.3.2 *Evaluating the HotkeySkillometer*

We conducted a laboratory study in order to better understand the impact of the HotkeySkillometer on adopting keyboard shortcuts. 24 participants were recruited from the University of Cantebury (20 male, 4 female). The experiment was performed on an Apple Macbook Pro running Mac OS 10.8, connected to a 24" external display running at 1920×1080 pixels that was used to display the experimental interface. Interaction was performed with a standard mouse and QWERTY keyboard.

The experimental task was conducted using Apple Keynote (with modified keyboard shortcuts) and a between-subject design with half of the participants having access to the HotkeySkillometer and the other half not (Control condition). Participants were pseudo-randomly assigned to either the Skillometer or control condition. They were then introduced to the Keynote software and to the experimental procedure, which consisted in performing sequences of requested formatting commands on a Keynote slide (10 repetitions of 8 commands each, for a total of 80 commands per participants), then selecting the 'next slide' command. Participants were instructed to perform these sequences of command selection as accurately and quickly as possible. In the Skillometer condition, participants were additionally given a brief description of the skillometer prior to the experiment. This description explained the visual presentation of parts A, B, and C shown in Figure 2.3 but not mentioning its overall goal in order to reduce the risk of introducing bias. Participants in both conditions were informed that keyboard shortcuts existed for all requested commands (with labels shown next to menu items). The whole experiment took participants 15 minutes on average.

The main goal of this study was to observe the impact of the HotkeySkillometer on keyboard shortcut usage, with our main dependent variables being the proportion of keyboard shortcut use measured as the proportion of correct command selection performed using a hotkey out of all the correct command selections. Figure 2.4 shows the proportion of command selections completed using keyboard shortcuts (*Hotkey* on the figure) for the control (left) and HotkeySkillometer (right) conditions across the ten repetitions. With HotkeySkillometer, the mean rate of keyboard shortcut usage across all repetitions was 50% compared to 28% for the control condition, giving a significant main effect of the interface and suggesting that the HotkeySkillometer increased the proportion of keyboard shortcuts used. There was also a significant interaction effect of Interface $\times$ Repetition, with participants making an earlier and more substantial switch to keyboard shortcuts in the HotkeySkillometer condition, with 80% of commands completed using keyboard shortcuts by the 10th repetition, compared to 42% in the control condition.

Altogether, these results suggest that users quickly learned and used more keyboard shortcuts in a commercial software product when presented with an accompanying skillometer. While extremely encouraging, the HotkeySkillometer only encourage users to transition to the second modality, but the transition and its resulting performance dip remain.

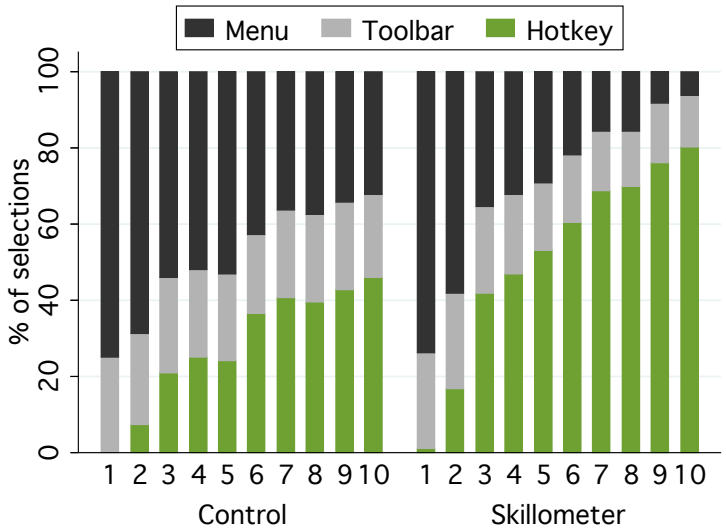

Figure 2.4: Percentage of selections per modality and per repetition, in both Control and Skillometer conditions

2.4 AUGMENTEDLETTERS: MNEMONIC GESTURE-BASED COMMAND SELECTION

The Marking Menus that were presented in section 2.2.2.2 offer a command selection method that is well suited to touch-based interaction. However, the nature of their gestures, directional marks, generally requires an arbitrary mapping between the commands and corresponding directions. While directions can be leveraged for some very specific cases (e.g. opposite directions for opposite commands such as open/close), users generally have to learn the mappings from scratch. The principle of rehearsal described earlier helps to minimize the performance dip associated with the potential recall time and errors, but it cannot completely compensate for it.

During my post-doctoral contract, I remotely worked with my PhD supervisor Eric Lecolinet, his new PhD student at the time Quentin Roy, and in collaboration with Dr. Yves Guiard and Dr. James Eagan. In this project, we investigated how a mnemonic mapping could be reintroduced in gesture-based menus such as Marking Menus in order to simplify recall. We proposed the Augmented Letters [HDR13], a new command selection technique where gestures consist of the initial of a command name affixed with a directional tail

to discriminate between several possible commands. The tail can be oriented in up to eight directions, so as to handle conflicts amongst commands that share the same initial. The aim behind using the initial of a command name is to simplify command memorization and to reduce the associated cognitive load. As such, Augmented Letters implement the principle of rehearsal while also leveraging a mnemonic mapping to facilitate users' recall.

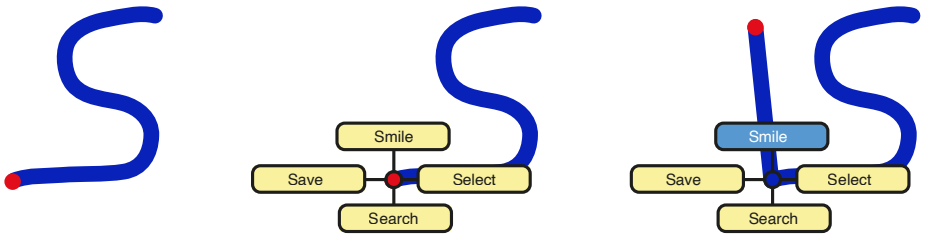

Figure 2.5: An example of AugmentedLetter that displays 8 possible items after the S unistroke letter has been drawn.

2.4.1 *AugmentedLetters*

An AugmentedLetter is a Graffiti-like unistroke letter [28] which is 'augmented' with a directional tail. The letter typically is the initial letter of the associated command, thus allowing a straightforward semantic mapping between the first part of the gesture and the command. On the other hand the tail makes it possible to differentiate between items that start with the same letter, hence handling name collisions. Up to eight commands can thus be specified with only one handwritten letter and a directional tail.

The augmented letters implement the principle of rehearsal and support two distinct modes of interaction. If the user does not know the gesture associated with the desired commands, he or she simply draws the first letter of the command name and wait for 500 ms for a marking menu to appear, showing the possible tails and the corresponding commands (Figure 2.5). When the user knows the entire augmented letter gesture that corresponds to the desired command, they can directly trace it without waiting for the menu to appear.

An AugmentedLetter can be augmented with a tail that may point in up to eight directions. Thus, it is theoretically possible to define up to $26 \times 8 = 208$ different commands (for the 26 letters of the Latin alphabet and 8 directional tails). However, there are a few conflicts that makes it difficult to disambiguate

the complete gesture, e.g. a left tailed C is similar to a left tailed G with just the length of the last segment of the gesture that might differ. We counted about six of them depending on the recognizer, leaving $208 - 6 = 202$ different commands.

2.4.2 *Studying the impact of AugmentedLetters on recall*

In order to assess whether AugmentedLetter can simplify the recall of gestural shortcuts for command selection, we compared the performance of AugmentedLetters (AL) relative to Marking Menus (MM) in a laboratory experiment. Our experimental protocol (schematized figure 2.6), inspired by previous research on the memorization of gesture-based command selection [19], relied on three repetitions of two blocks: one learning block immediately followed by one test block. During learning blocks, users could rely on the menu mode of both techniques. That is that they could, if they wanted to, dwell and wait for the menu to appear. A test block immediately followed each learning block. During these test blocks, users had to use mark mode. That means that they could not reveal the menu and had to recall the gestures from memory.

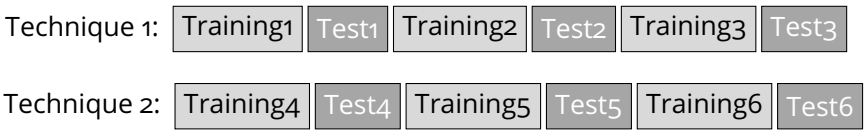

Figure 2.6: Organization of blocks in the experiment of AugmentedLetters

We recruited 12 participants (aged 21-48, 5 women). They were instructed to try to memorize as many gestures as possible. Each trial started by displaying the name of the target command at the top of the screen before to let the user attempt to select it, with accessibility to the menu mode or not depending on the type of block.

On average the recall rate for test blocks (Figure 2.7-left) was 24.5 points higher with AugmentedLetters (76.6%, 9.2 items) than with MarkingMenus gestures (52.1%, 6.25 items), an effect that was found to be statistically significant.

For its part, overall selection time (Figure 2.7-right) was the same in menu mode with AugmentedLetters and MarkingMenus (3.80s vs. 3.79s), with no significant effect found. This might be surprising given that the symbols to draw with AugmentedLetters are relatively larger and more complicated than the simple directional gestures from MarkingMenus. Interestingly, when com-

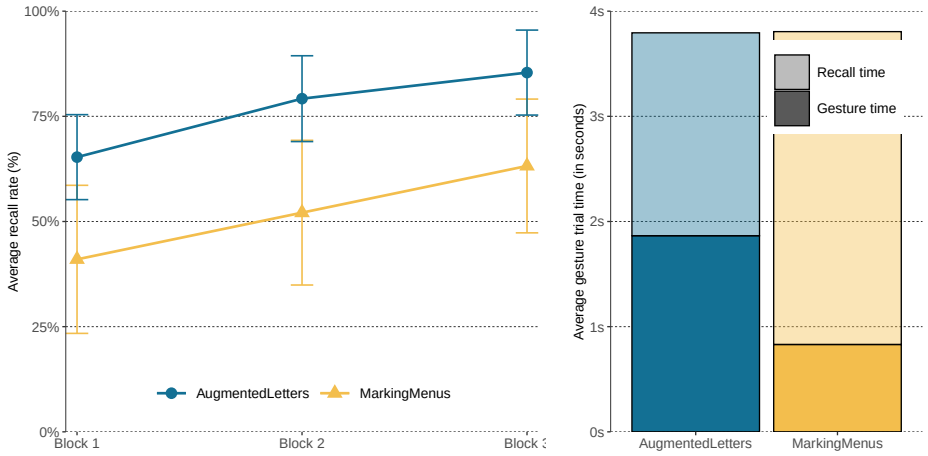

Figure 2.7: Left: average recall rate by test block for each technique; Right: average gesture trial time for each technique, split between recall and gesture times

paring reaction time (time between stimulus appearance and the beginning of gesture execution) and gesture execution time, we note that while it took participants significantly more time to execute AugmentedLetters than MarkingMenus gestures (1.86s vs. 0.83s), that difference is completely offset by a significantly shorter reaction time.

The latter finding might be the result of two things. First, with AugmentedLetters, users benefitted from the high familiarity of the cognitive path leading from a word, be it a country name or of computer commands, to its corresponding gesture that starts with the initial letter of the word. Second, they only had to recall a single direction for the tail instead of two with MarkingMenus. These two factors altogether probably explain why it was easier (and faster) to recall gestures with AugmentedLetters.

In addition, when looking at data collected during learning blocks where participants were free to use either menu or mark mode, we did observe over the last two learning blocks that the spontaneous usage of mark mode was 22.2 point more frequent with AugmentedLetters (66.0%) than with MarkingMenus (43.8%).

Altogether, these results suggest that Augmented Letters help to reduce the performance dip, thanks to an easier recall of command gestures, leading to fewer errors, at no noticeable speed cost. Moreover, we could observe that

users were likely to move away earlier from menu mode as suggested by the higher spontaneous use of mark mode with AL.

2.5 FASTTAPADJUST: THE IMPACT OF REDUCING ERROR AVERSION ON RECALL-BASED ADOPTION

In the section 2.2.2.2, we reviewed FastTap [54], a rehearsal-based interface that displays a hand-sized grid of items when a dedicated activation button in the lower left corner is pressed. With FastTap, inexperienced users can select commands in *menu* mode by first pressing the activation button (typically using their thumb), then waiting for a certain delay for the menu interface to appear, and finally tapping on the desired commands with another finger while still keeping the activation button pressed. Experienced users on the other hand can directly select the desired commands in *chord* mode, by performing the corresponding chord gesture that presses the activation button and desired command(s) without waiting for the menu interface to appear. The FastTap interface was extensively investigated, for instance on devices other than phones or tablets [20, 69]. More interestingly, the learnability and adoption rate of chord mode was also investigated in additional projects [53, 68].

Of particular interest, Gutwin et al. tested the rehearsal hypothesis, that is that rehearsal suffice to eventually adopt chord mode, in two different applications relying on a FastTap interface [53] and providing different levels of "urgency" (or incentive) to use chord mode. The first application was a game where the player had to 'slice' as many elements of different colors as possible in a given time. However, slicing could be performed only if the correct color was chosen (through a FastTap interface). As such, there was a high level of urgency to use chord mode as when used correctly, it guaranteed to slice more elements which would result in a higher score. The second application was a simple drawing application that used a FastTap interface to switch between tools. Participants were asked to reproduce, using this application, three drawings each week for ten weeks. This time, there was a low level of urgency to use chord mode as it did not bring any directly quantifiable reward to the user. The results of Gutwin et al.'s study revealed that while participants quickly switched to a sustained use of chord mode selections in the game applications, only few users used it in the drawing program, even at the end of the ten weeks. Notably, one of the reasons for not switching that participants reported was the perceived risk of making errors when selecting commands in chord mode.

Indeed, the seminal FastTap design activates the selected commands as soon as the second finger enters in contact with the touchscreen [54]. As such, in chord mode, it is impossible for users to visually confirm the command corresponding to a certain chord before its activation. Selecting commands in chord mode thus presented a high risk of error for users as they have to accurately recall the chord corresponding to the desired command.

Interestingly, I already mentioned in section 2.2.2.2 that the first description of Kurtenbach's marking menus also relied on only two modes: menu and mark modes. As such, similarly to FastTap, users could not visually confirm the command they were about to select when in mark mode. Kurtenbach attempted to mitigate this issue by introducing a mark-confirmation mode [67], where a user can dwell for a predetermined delay and switch back to menu mode – allowing re-selection or a cancellation. With Alix Goguey, Andy Cockburn and Carl Gutwin, we decided to investigate whether a similar mechanism for FastTap, called FastTapAdjust [HDR5], would assist users in switching to a sustained use of chords for selecting commands.

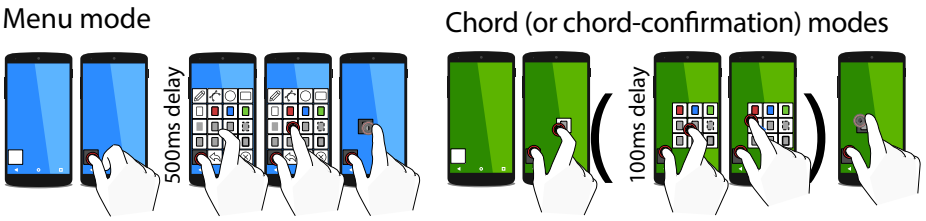

Figure 2.8: FastTapAdjust. In menu mode, users press the menu button and wait for the grid to appear, then select a command by tapping on a grid cell while holding the menu button. In chord mode, users tap the menu button and the command's location with a two-finger chord. Chord-confirmation mode uses the same chord-based invocation, but users can pause and display items around their selection, allowing them to adjust the chord before selecting

2.5.1 *The FastTapAdjust mechanism*

The rationale behind the FastTapAdjust mechanism is to offer an opportunity to users to visually confirm the command they are about to activate when this command was initiated in chord mode. For that, instead of activating the command as soon as the finger touches down the grid cell corresponding to that command, the selection happens when the finger lifts off from the sur-

face. A first difference with the regular FastTap interface here is that user can see the command that is going to be activated before it is actually activated (Figure 2.8). This provides an additional interaction state in which users can perform adjustments if needed. The command activated becomes the last grid cell the finger was in contact with before lift off. Therefore, when a user makes a two-finger chord in chord mode and then pauses for at least 500ms before to lift off, an area of the grid under the finger is displayed (Figure 2.8-right). That area could be of different sizes, up to the entire grid of commands. In aniticipation of an error, the user can then adjust the finger position and activate a different command than what was originally touched. If the finger moves outside the displayed area, no command is triggered and the grid is fully displayed (essentially defaulting to the menu-mode presentation). This "chord-confirmation" mode allows users to try recall-based chord selection, even when their recall of the grid is still imperfect, and thus at lower risk. In menu mode, FastTapAdjust works as a regular FastTap interface, that is, that the grid is revealed after the user dwells for a sufficient time (500ms) on the activation button.

Overall, we hypothesized that this chord-confirmation mode would provide users with a smoother skill acquisition when interacting with FastTap, by encouraging them to decrease the use of menu mode earlier in the learning process. In addition, we hypothesized that the additional feedback of the chord-confirmation mode would help to maintain a low error rate, and that the reduced number of choices will help to maintain rapid performance thus minimizing performance dip.

2.5.2 *Testing the efficiency of FastTapAdjust*

We investigated the impact of FastTapAdjust on switching away from menu mode in a controlled experiment where participants were asked to select commands as quickly and accurately as possible. We compared the original *FastTap* (*FT*) design [54] to two FastTapAdjust versions (differing in terms of the size of the grid that is uncovered): *FastTapAdjust* with a Moore neighborhood (i.e. the 8 surrounding cells, *FTA-8*), and *FastTapAdjust* with the full grid displayed as the neighborhood (*FTA-all*). For each technique, we tested different error-cost conditions induced by the number of attempts that participants could take.

We recruited 36 participants (mean age 26 years). The experiment used a $3\times3\times10$ hybrid design with Technique (FT, FTA-8 and FTA-all) as a between-subject factor, and MaxAttempts (1-, 2- and 5-tries) and Block (1-10) as within-

subject factors. Each combination of factors was repeated 24 times. Our main dependent variables were command selection time, error rate, and mode usage.

We found no significant effect of Technique on command selection time, thus not finding evidence that introducing the chord-confirmation mode would present any major barriers to rapid interaction.

For its part, error rate was computed as the proportion of trials where participants failed to activate the correct command given the number of attempts allowed divided by the total number of trials. Error rates were also relatively low and similar across the three different techniques, ranging between 4% for FastTap and 2% for the other two techniques. Unsurprisingly, however, error rates for the different cost conditions were more variable, with 7% for 1-try, 1% for 2-tries, and 0% for 5-tries. A significant Technique $\times$ MaxAttemps interaction effect was found illustrating that MaxAttempts impacted differently each Technique (see Figure 2.9). Notably, FastTap had a much higher error rate for 1-try than FTA-8 and FTA-all, whereas error rates for 2-tries and 5-tries were not significantly different between the three techniques.

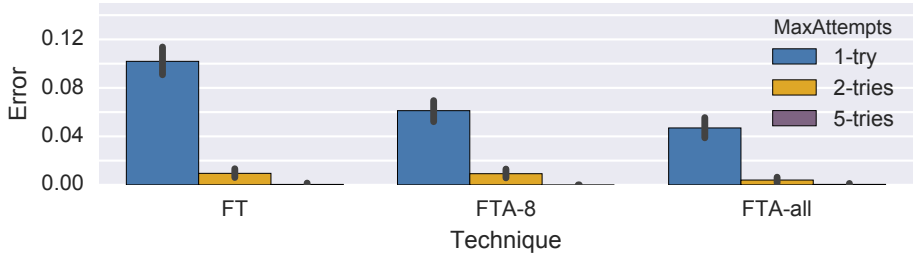

Figure 2.9: Error rates for FastTap and both FastTapAdjust versions for the three MaxAttempts conditions

Mode usage was assessed by first categorizing each correct command selection as *Menu*, *Confirmation* or *Chord* depending on the mode used to select it. Overall, usage of the menu mode decreased during the entire experiment. As shown in Figure 2.10, usage of menu mode decreased from 94% in the first block to 34% in the tenth block for FastTap, from 94% to 31% for FastTapAdjust-8 and from 81% to 21% for FastTapAdjust-all.

Our results also suggested that introducing the chord-confirmation mode does leads to an earlier switch away from use of menu mode (see Figure 2.10), that happened around block 2 for FTA-8 and FTA-all, versus between blocks 4 and 5 for FT. However, we also observed that the chord-confirmation mode

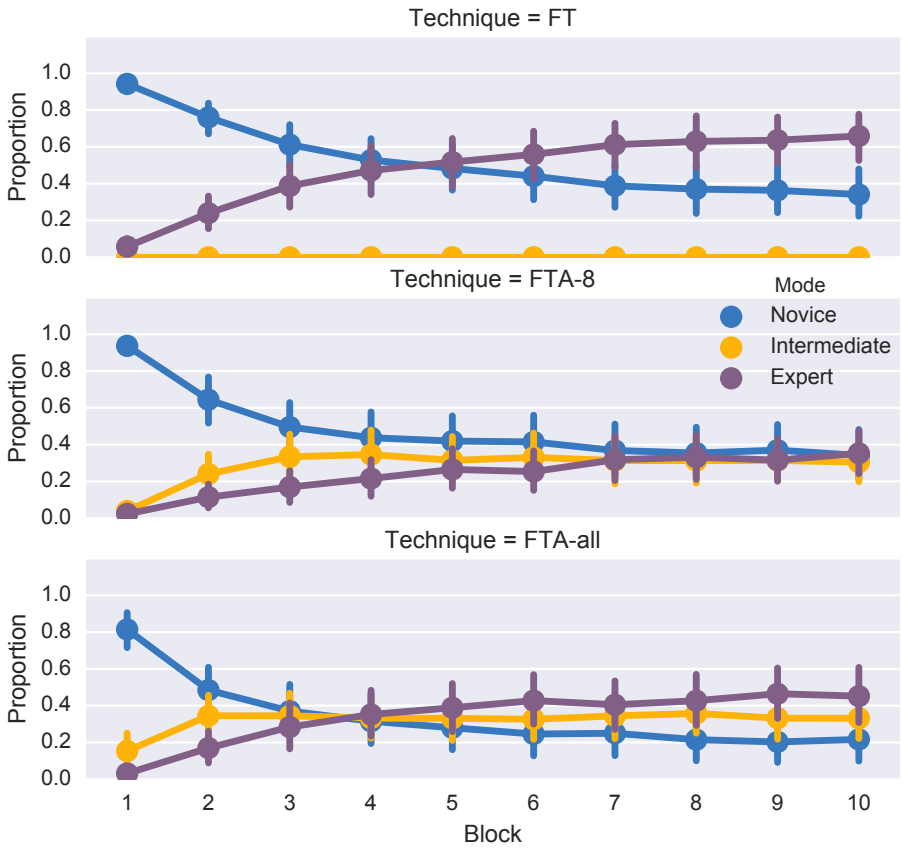

Figure 2.10: Overall rates of menu (novice), chord-confirmation (intermediate), and chord (expert) modes use for each technique, by block

drains some selections from the chord mode rather than from the menu mode. Looking at the final block for instance, we can see that chord mode usage is above 60% in the FT condition, while it is only about 40% for both of the FTA conditions. In the end, while the chord-confirmation mode decreases the use of menu mode, it also decreases the use of chord mode, and we still have a significant usage of menu-mode in the last block. Interestingly, in both of the FTA conditions, chord-mode never ends up with the absolute majority, as it never reaches a value higher than 50%.

In the end, the chord-confirmation mode was used frequently without impacting command selection time, resulting in an earlier switch away from menu mode. While it failed to help users to completely stop using the menu mode, it had the positive impact of reducing error rate in chord-mode compared to the conventional FastTap interface when only one attempt was allowed. In this way, it facilitates the intermodal transition thanks to an intermediate mode that users decided to rely on, at no speed cost. On the other hand, since usage of chord-mode was lower with the FTA interfaces than with the conventional FastTap technique, we must note that this confirmation mode also acted as a training wheel on which users might have relied too much. This training wheel, why smoothing the transition, might have the drawback of slowing it down, raising the question of whether having too many modes might, in the end, constrain more interaction rather than supporting it.

2.6 LESSONS LEARNED FROM TRYING TO FACILITATE AND ENCOURAGE THE INTERMODAL TRANSITION

The intermodal transition framework is often referred to when discussing command selection and menu techniques. Main reason is probably that these techniques have often been created with the conventional GUI design guidelines in mind, creating some tension between relying on *Recognition rather than recall* and *providing "shortcuts"* [89, 92, 93]. And the intermodal transition problem arises from the transition from one recognition based command selection mechanism, to a recall-based one.

In this chapter, we have summarized projects that were targeted at simplifying this transition, which can be approached in two ways: encouraging this transition and minimizing the associated performance dip. Our first study investigated the potential benefit of the HotkeySkillometer, a reflective widget designed to foster users' reflection on performance and encourage them to switch to the shortcut mechanism in the context of desktop command selection. The results of a controlled experiment conducted suggest that users

leveraged the information provided by this skillometer and adopted more rapidly the shortcut mechanism than when the Skillometer was not available. The second project investigated a novel type of gesture-based command selection mechanism that introduced a mnemonic mapping in MarkingMenus like gestural command selection, as an attempt to simplify gesture recall. The results suggest indeed an easier recall of command gestures, leading to less errors at no speed cost, as well as an earlier switch from menu to gesture mode. The third project investigated the impact of the introduction of a *confirmation* mode in the FastTap interface, a rehearsal-based command selection mechanism for touch-based devices. We introduced FastTapAdjust, an alteration of the seminal FastTap interface that lets users visually confirm the command they are about to invoke, and perform an adjustment if needed. The results of a controlled experiment comparing two FastTapAdjust interfaces to FastTap suggest that such a confirmation mechanism, while useful for reducing errors, might have the side effect of stopping users to rely on the recall-based mechanism.

These results, while positive since simplifying the intermodal transition, still remain somewhat limited. Indeed, while reducing error or minimizing the performance dip is undoubtedly beneficial, it remains insufficient when confronted to the intermodal transition problem. First, there was inevitably a phase where users relied on the recognition based method, no matter what. Second, the study conducted on FastTapAdjust suggests that users may choose to use more a recognition-based method than the recall-based one when they feel that the latter might be too risky.

Notably, this last result raises several important questions. Can we realistically hope that users will eventually move completely away from the recognition-based mechanism? And if not, can we design a command selection mechanism that would be close to as efficient as the recall-based mechanisms, while still providing instant recognition thus allowing users to "confirm" their selection at all time?

At the beginning of this chapter, in subsection 2.1, I was making an analogy with the problems faced during an intermodal transition and those faced when removing training wheels while learning to ride a bicycle. There is an idiom that says "(just) like riding a bicycle"[2] which means that something is easy to resume after a long break. But why is cycling something that is never forgotten? Possibly because this is a sensorimotor activity that is stored in our procedural memory [120], which is automatically retrieved when needed. But selecting commands with a shortcut, on the other hand, is not purely senso-

2 https://en.wiktionary.org/wiki/like_riding_a_bicycle

rimotor, with many cognitive aspects that come into play. In the end, it may rely more on associative memory, which is less robust. So while the parallel between riding a bicycle and selecting commands might work when it comes to learning and performance dip when transitioning from one condition to the other, it falls flat when it comes to remembering it.

Moreover, the problems inherent to the intermodal transition are emphasized when the two modes rely on different input modalities, thus creating an additional "distance", conceptual and physical, between these two modes which may require more efforts to the user. As an example, for command selection in desktop computers, the transition from the recognition-based method (pointing the commands in a menubar) to the recall-based one (using keyboard shortcuts) is complexified by the fact that these two modes require interacting with two different devices. On the contrary, marking menus simplify this transition thanks to a minimized distance between: since both modes rely on a gestural input, the main difference between modes remain in the fact that users can dwell to display the visual aid. But besides this visual aid, inputs are basically the same.

Interestingly, whether the visual aid had to be hidden by default was never questioned in these types of work. As an example, MarkingMenus appear after a delay because it allegedly could "be distracting" or "obliterate part of the screen" (without empirical evidence to back these assumptions up) [67]. However, from the different perspective, hiding the menu by default creates a massive cost to interaction: users have to recall and perform the correct gesture in order to rapidly select a command. In the next chapter, we will describe projects in which we attempted to design command selection techniques that would yield efficient performance, without *requiring* users to recall a gesture or key combination.

3 EFFICIENT RECOGNITION-READY COMMAND SELECTION

In the previous chapter, we discussed the context of the intermodal transition, a context in which a transition can occur between two different modalities that allow to perform the same operation. More precisely, we explored this transition applied to command selection mechanisms and presented several projects attempting to alleviate the problems inherent to this transition.

3.1 IS EXPLICIT RECALL NECESSARY FOR EFFICIENT COMMAND SELECTION?

A characteristic of this transition is that, by definition, it exists because at least two distinct modes co-exist. Interestingly, these modes leverage differently the motor and cognitive capabilities from the user. Some modes mostly rely on visual recognition, while others will require users to explicitly recall certain operations. This dichotomy can somewhat be seen as a design bias. Indeed, is really recall-based command selection the only available method for efficient command selection? Is recognition-based command selection necessarily capped at a relatively low performance level? And why do some methods such as keyboard shortcuts necessarily require an explicit recall from the user while nothing prevents them from leveraging visual recognition?

As mentioned in the previous chapter, the progression of a user with a given interface has been characterized as going through three stages: cognitive, associative and finally autonomous. This is, of course, assuming that the user is able to reach that autonomous phase. However, it might not always be the case as previous results suggest that even a number of repetitions as high as 1100 times was not the case for some tasks Card et al. [24]. Moreover, in the context of command selection, it would expect users to be able to reach automaticity for all required commands, which might remain a fantasy given that human memory is not infinite, be it in duration or capacity, and its mechanisms are still relatively poorly understood [62].

Therefore, it raises the question of whether recall-based command selection is necessary, and what its potential drawbacks and benefits are. Interestingly, looking at the selection of commands in hierarchical menus on desktop computers, pointer-based selection also leverages a certain type of user's memory, namely spatial memory. Spatial memory is swift to build, robust, and

helps users to rapidly visually locate a desired command [105]. Sometimes, users can even "mouse ahead" before to open a menu simply because they know where the command will eventually be displayed on screen.

This raises the question of the actual difference between recall-based and recognition-based interactions in the context of command selection, or to the least how different types of command selection mechanisms could be classified otherwise. In that respect, I find interesting to use the categorization used in the "Designing the User Interface - strategies for effective HCI" from Shneiderman and his colleagues [111]. In this book, command selection mechanisms are mostly referred in the chapter on "Fluid Navigation". The first subsection of this chapter, entitled is "Menu bars, pop-up menus, toolbars, palettes and ribbons" whereas the second is entitled "'Shortcuts and gestures for rapid interaction". These two subsection discriminate between interfaces operated by pointer navigation using the mouse, and "shortcuts" like keyboard shortcuts where "users can memorize the keystrokes for the menu items". Their categorization implies that hotkeys have to be a "shortcut" mechanism, thus referring they are a shorter path to reach a command. However, it is unclear why this "shorter path" would have to be memorized beforehand, and how.

Answering to this question is even more complicated when looking at the Marking Menus described in section 2.2.2.2. In this case, the menu mode can be seen as a navigation and recognition based mode: the user navigate through the different levels of the menu hierarchy and visually confirm their selection. The mark mode, on the other hand, can be seen as the shortcut and recall based mode: the user does not have to wait before to perform the mark and has to recall the gesture. Interestingly, we can see here that an artificial cost was added to the menu mode: the user **has to wait** before to be able to operate it. This necessarily limits user performance in menu mode to the time required to wait for the menu to appear plus the time needed to perform the gesture (with the latter that could mechanically be equivalent for both modes). We will get back later in this chapter on the rationale for this design, but without this artificial cost users would not have to memorize the gesture beforehand.

In this chapter, I investigate whether we can design efficient command selection mechanisms that offer the same performance limit than "shortcut" mechanisms (e.g. keyboard shortcuts) without requiring to rely on explicit recall. That is, can we design interface that would combine the usability of a recognition-based interface, with the efficiency of shortcuts mechanisms that can be performed swiftly, as illustrated on figure 3.1.

In the following, we will report on the design and evaluation of interaction techniques investigating the feasibility of recognition-based, yet efficient,

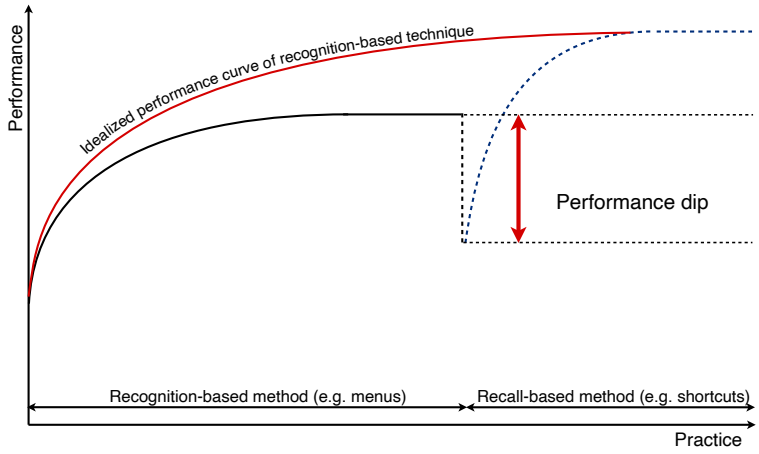

Figure 3.1: Idealized performance curve (in red) when using a single command selection method that uses a mechanic similar to the admitted efficient method (e.g. keyboard shortcuts) but still relying on recognition for training and execution, compared to the theoretical performance of the traditional recognition-based/recall-based combination.

command selection mechanisms. We will explore three different interaction paradigms: keyboard-based, touch-based, and gesture-based command selection.

3.2 FACILITATING KEYBOARD SHORTCUT USE WITH EXPOSEHK

Implementations of keyboard shortcuts in current operating systems suffer from several limitations. First, in order to select a command with one, users must be aware that this shortcut exists. Second, once aware of its existence, users have to know the corresponding key sequence and for that, to expose it which is typically done by moving the mouse cursor over the corresponding toolbar button or opening the corresponding menu. As such, they use pointer-based selection as the starting point for exposing the key combination (i.e., it is only shown after mouse-based operations). Consequently, users reinforce pointing even while trying to learn the faster pointerless method. Third, users must memorize the key sequence and recall it correctly next time they actually want to select the corresponding command using its keyboard shortcut.

The first project that I conducted after my PhD was to explore whether it would be possible to design a command selection mechanism that would

overcome these limitations. In particular, with Prof. Andy Cockburn, Dr Gilles Bailly, Prof. Carl Gutwin and Joel Harrison, we wanted to design a command selection mechanism relying on keyboard shortcuts, without requiring users to *have to* recall the corresponding key combination.

We have developed and evaluated ExposeHK (EHK) that addresses these issues [HDR10]. ExposeHK works very simply (illustrated Figure 3.2): it is activated by pressing a modifier key (e.g., the Ctrl or Command key) and remains as long as that key is held; when activated, it displays the keyboard shortcuts directly on the GUI of the application.

Figure 3.2: The user wants to open a new tab but is not sure of the hotkey. He visually locates the button in the toolbar (boxed on left), then presses the Command key (Cmd) to activate ExposeHK, which overlays toolbar items with available keyboard shortcuts (right). He completes the command by pressing Cmd+T.

ExposeHK presents the following benefits. First, it enables the browsing of keyboard shortcuts via the keyboard since they are displayed when a modifier key is pressed. Second, it supports rapid verification of keyboard shortcuts for experienced users that have partial knowledge of the interface and know approximately where the graphical component (toolbar button or menu item) associated with the command they want to activate is located. Third, it still supports a theoretically rapid command selection time because it relies on a stable flat hierarchy, and each command can be activated via a single key combination. Finally, as detailed below, ExposeHK is compatible with existing toolbar and ribbon designs, and it can be readily adapted to menus.

toolbar ExposeHK works in a very straightforward manner with Toolbars, as illustrated Figure 3.2, by overlaying the corresponding keyboard shortcut over each button of the toolbar when a modifier key is pressed and as long as it is held. Activation of the command can then be simply completed by performing a corresponding keyboard shortcut, or still by clicking on the button if the user feels like it.

menus Activation of ExposeHK with menus is similar to with toolbars – pressing a modifier key exposes all the first-level menus, thus displaying the already existing label for items that can be selected using a keyboard shortcut, as shown at the top of Figure 3.3. Command selections can be completed by pressing a keyboard shortcut or by using the pointer. However, in order to display all first-level menus, ExposeHK slightly modifies the visual layout of menus to remove spatial overlapping. This is necessary in order to support keyboard shortcut browsing while still help intermediate users exploit their spatial memory to some extent. Details about this adaptation and its consequences on the interface design, as well as discussion regarding access to second-level menu cascades can be found in the corresponding full research paper [HDR10].

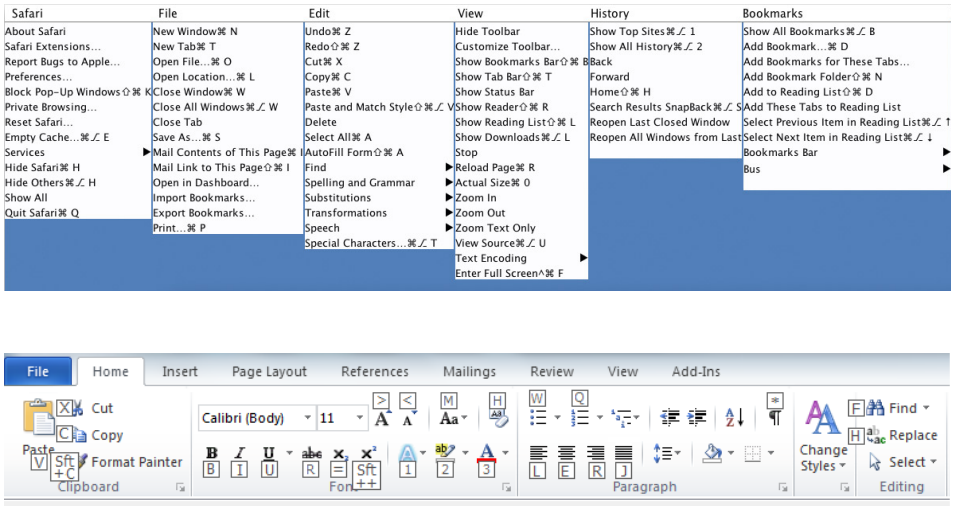

Figure 3.3: Instances of ExposeHK with menu bars (top) and a Microsoft Ribbon (bottom). With menu bars, ExposeHK reveals all first-level menus at once when a modifier key is pressed. With a Ribbon, ExposeHK overlays all buttons of current Ribbon tab with available keyboard shortcuts. The ribbon adaptation is very similar to the Toolbar one, but it additionally allows users to move between tabbed toolbars using the mouse scrollwheel (as does the current Microsoft Ribbon interface) or arrowkeys.

ribbon The ribbon adaptation of ExposeHK is visually similar to the toolbar one, but it additionally allows users to move between tabbed toolbars using the mouse scrollwheel (as does the current Microsoft Ribbon interface) or

arrow keys (thus preventing the use of keyboard shortcuts using arrow keys to activate commands). This adaptation remains challenging however, because each of several tabs places different items in the same spatial location.

3.2.1 *Investigating the usability and adoption of ExposeHK with toolbars and menus*

We first conducted two studies to investigate the usability and adoption of keyboard shortcuts with the toolbar and menu variants of ExposeHK. These studies were designed to answer several questions about the benefits of ExposeHK over traditional interfaces, among which:

- does ExposeHK result in earlier and higher levels of keyboard shortcut use?
- how does keyboard shortcut use with the test interfaces change with experience and frequency of item selection?
- does ExposeHK impact command selection performance?
- are errors affected by interface or selection mechanism?

Our experimental task involved selecting target commands as fast and accurately as possible, using the method selection that participants desired to use (mouse pointer or keyboard shortcuts). Each *trial* (selection of one target command) started with a stimulus[1] indicating the target command that participants had to select *as fast and accurate as possible using the modality of their choice*. Participants completed tasks with both available interfaces: ExposeHK and AudioFeedback[2]. In each study, participants completed six blocks of selecting multiple times the 12 same commands with each interface.

We recruited 36 participants, aged 20-45 (6 female). They all participated in both studies, completing first Study 1 and then Study 2. We refer to the corresponding research publication [HDR10] for more details regarding the experimental design decisions and description of the apparatus.

Regarding keyboard shortcut usage, both studies revealed a higher proportion of command selections completed using keyboard shortcuts with *ExposeHK* than with *Audio Feedback* (illustrated Figure 3.4). In study 1 (respectively study 2), participants completed 94% (respectively 99%) of correct se-

1 In study 1, voice-synthesis was used to pronounce the stimulus in order to ease risks of confounds stemming from factors such as visual pop-out effects due to exact matches between the stimulus (e.g., a cat icon) and its representation in the interface. In study 2, the stimulus was an icon representing the target command (similarly to Grossman et al.'s protocol [51])

2 In study 1, participants also tested a *tooltip* condition, that is not mentioned here for the sake of clarity. I refer to the corresponding research paper [HDR10] if interested in the results corresponding to this condition

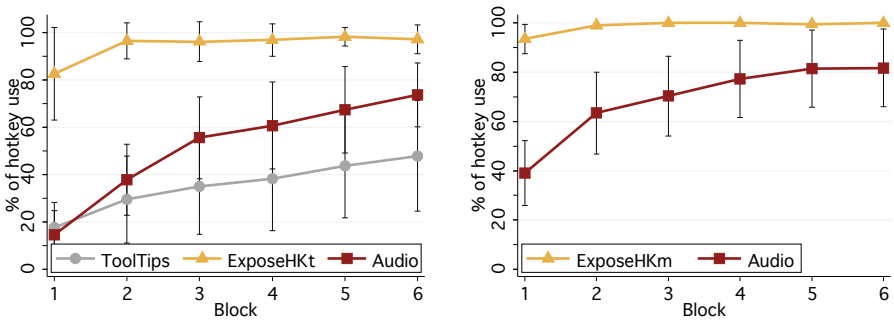

Figure 3.4: Average % of keyboard shortcuts (labelled 'hotkey' on the figure) for Study 1 conducted using a toolbar (left) and Study2 conducted using a menu (right)

lections with keyboard shortcuts in the *ExposeHK* condition, against 50% (respectively 64%) in the *Audio Feedback* condition. More interestingly, a significant Block $\times$ Technique effect was also found in both studies, suggesting that participants made an early and sustained switch to keyboard shortcuts with *ExposeHK*, contrasting with *Audio Feedback*'s gradual increase.

Regarding selection time, we found no significant effect in the first study (3s for EHK, 3.4s for audio). That being said, the mean time for pointer-based selections in the final block (for those who continued to use the mouse with any technique) was 3.53s, compared to 2.38s with hotkeys. In the second study, however, mean selection times with *ExposeHK* (2.78s) were found significantly shorter than with *Audio Feedback* (3.16s).

The results of these two studies suggest that toolbar and menu adaptations of ExposeHK improved on the state-of-the-art systems for promoting keyboard shortcut use. With ExposeHK, participants made an early and sustained switch to keyboard shortcuts, whereas with post-selection audio feedback they made a gradual, progressive and slower switch. I must stress however that participants, mostly university staff and students, were almost certainly aware of keyboard shortcuts as an interface mechanism, possibly increasing hotkey use for all conditions.

3.2.2 *Comparing the Performance of ExposeHK over hotkeys in a Ribbon interface*

The Microsoft Ribbon interface somehow merges toolbar and menu designs, using a tabbed 'ribbon' to separate multiple toolbars. Adapting *ExposeHK* to the ribbon is challenging because each of several tabs places different items in the same spatial location. Also, the ribbon supports the Access Keys mechanism (also known as Mnemonics outside the MS Office suite [16], and named *Alt Keys* in the original research publications of this project [HDR10]), that is superficially similar to *ExposeHK*. However, while relying on a modifier and associated keys, every selection with Access Keys is necessarily hierarchical, involving a sequential specification of the target tab and then the target item within the tab. This raises two questions: can *ExposeHK* be adapted to the Ribbon, and does it improve performance over the Access Keys technique?

Figure 3.5: Example of a ribbon used in our experiment, with the ExposeHK mode on

Each trial of this experiment involved selecting an item within a ribbon-like interface (Figure 3.5) in response to a text stimulus representing the target command. Unlike previous experiments, participants had to complete selections using the available interaction modality. Therefore, with *ExposeHK* and *Access Keys* they were required to use the keyboard and with *pointer* keyboard-based selections were impossible. Each block comprised 2 selections of the same 6 targets, allowing participants to gain expertise across blocks. Half of the selections in each block involved switching to a different tab, allowing us to inspect the impact on performance of tab-switching with the different interfaces. We recruited 18 participants (aged 21-45, 1 female). All participants completed the experiment using all techniques, in an order counter-balanced across participants.

Figure 3.6-left summarizes the selection time for the three *techniques* across *block* (left) and *tab switch* (right). The overall mean selection time with *ExposeHK* (2.86 s) was 8.6% shorter than pointer selections (3.13 s), and 36% faster than Access Keys. We also found a significant *technique*$\times$*block* interaction, with *ExposeHK* slightly slower than pointer in the first two blocks, but substantially faster (30%) than pointer by the final block. Note that the early

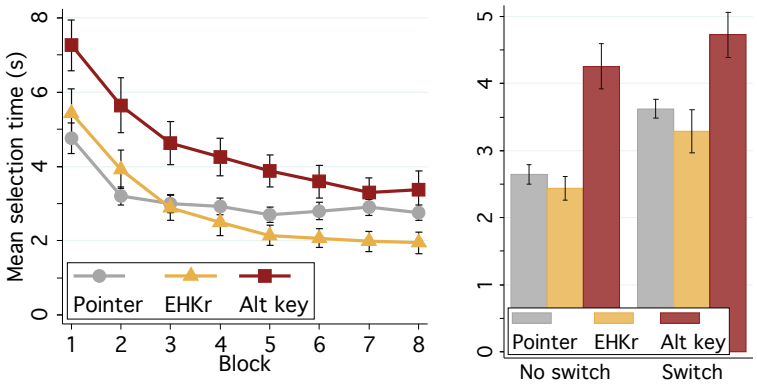

Figure 3.6: Mean selection time (s) per block (left) and condition (right)

asymptote of the pointer performance curve is a great illustration of the relatively low performance ceiling of pointer selection that is rapidly reached.

We also found a significant *technique*×*tab switch* interaction effect (illustrated Figure 3.6-right), which can be explained by the contrast between the Access Keys that required users to carry out identical hierarchical actions regardless tab switch state, whereas pointer and *ExposeHK* actions are different depending on the need to switch tabs.

3.2.3 *Investigating the continued use and memorization of keyboard shortcuts with ExposeHK*

One benefit of the ExposeHK technique is that it allows users to activate a command using a keyboard shortcut without having to memorize its corresponding key-sequence beforehand. A question we asked ourselves out of curiosity at that time was whether users would still be able to interact using keyboard shortcut in an interface where ExposeHK was available for then being disabled for any possible reason (e.g. an Operating System change or update).

We conducted a short pilot study investigating this question that relied on a simple text editor interface split in two, displaying a raw text next to a target formatted text (see figure 3.7). The top part of the text formatting interface displayed a toolbar containing buttons for each formatting commands. Each of these commands had associated keyboard shortcuts that were consistent with those used in Microsoft Word (e.g. Bold was Ctrl + B, Italics was Ctrl +

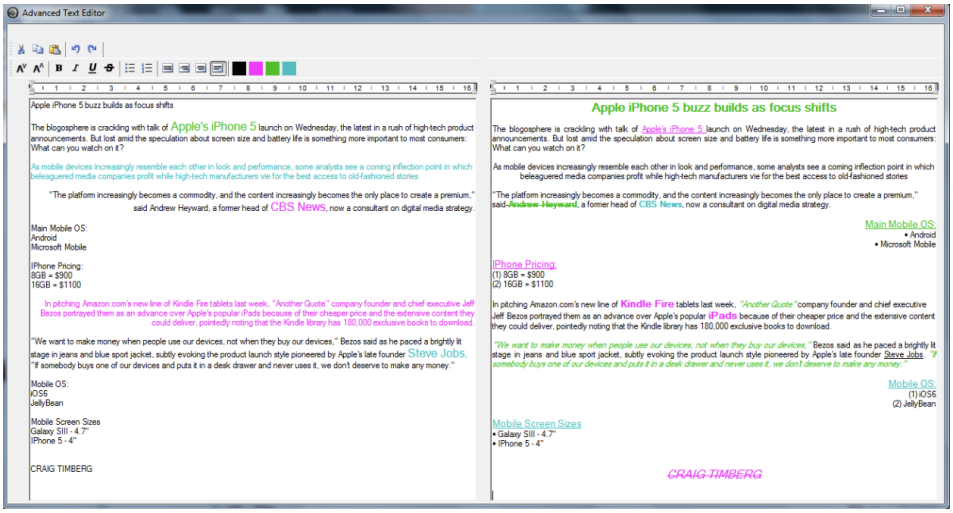

Figure 3.7: The experimental interface with a text editor on the left-hand side consisting of a text area and a toolbar of commands. The desired final format is displayed in the right-hand side of the text editor

I), thus allowing participants to use their pre-existing knowledge of keyboard shortcuts for commands and would not confuse the participants when learning new ones. Participants could also expose keyboard shortcuts through tooltips which would appear when the user hovered the mouse over a button of the toolbar.

Participants were instructed to perform three raw text formatting tasks. Formatting a portion of text was done by highlighting the text and then activating the desired formatting command, either by clicking on the corresponding button in the toolbar or using its associated keyboard shortcut. The first task was completed using this text editor with a simple toolbar (Baseline stage), that is, participants could use keyboard shortcuts but the only way to expose them was via their tooltips. The second task was then completed using the same text editor, but with ExposeHK enabled meaning that participants could expose keyboard shortcuts using ExposeHK (ExposeHK stage). Then participants completed a third task with a normal text editor again (post-ExposeHK stage), that is with ExposeHK removed from the toolbar. This text editor was the exact same interface used for the first text formatting task.

12 participants (ages 21-30, 2 female) completed the experiment. They were instructed not to try and go as fast as possible, but to just complete the task as they normally would with a text editor.

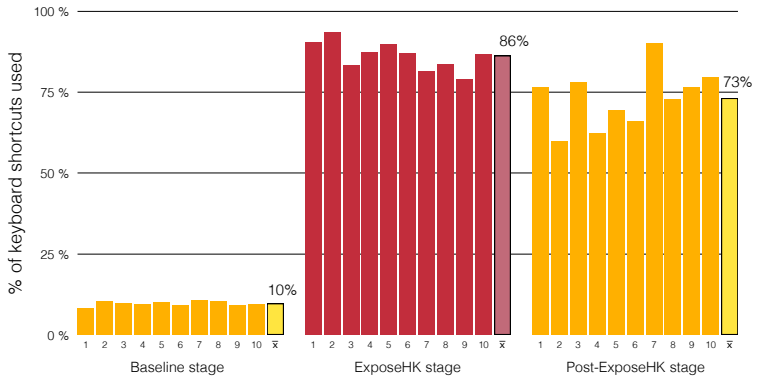

Figure 3.8: Proportion (in %) of selections performed using keyboard shortcuts during the different stages of the experiment.

We observed a large difference for each stage in terms of mean keyboard shortcut usage with 9.79%, 86.20%, and 73.09% for the Baseline, ExposeHK and post-ExposeHK stages, respectively. However, with that granularity, it is difficult to assess whether this evolution results from a simple learning effect or is affected by the possibility to use ExposeHK in the ExposeHK stage. To better understand this, we split each stage into 10 blocks over time (see Figure 3.8). This shows that the keyboard shortcut usage did not drastically increase overtime during the baseline stage, suggesting that users are not trying to learn new keyboard shortcuts but are just continuing with the ones that they currently know. For the ExposeHK stage, the figure suggests that keyboard shortcut usage is instantly very high and is consistent through the stage. Finally, we see that it remains relatively high in the post-ExposeHK stage, even though users cannot expose the keyboard shortcuts anymore and have to remember them, suggesting a certain memorization of the recently used keyboard shortcuts.

3.3 ICONHK: RECOGNIZE HOTKEYS IN TOOLBAR BUTTONS

When working with Gilles Bailly on the ExposeHK project described, we were somehow clueless about the best approach to present the keyboard shortcut when a modifier key was pressed. We ended up using different presentations (replacing the original icon of the button or overlaying a tooltip, showing the modifier keys or not, etc.). With Emmanouil Giannisakis, Dr Gilles Bailly and

Dr Fanny Chevalier, we explored in the IconHK project [HDR4] a novel perspective on the design of toolbar buttons that aims to blend visual cues that convey keyboard shortcut information directly within the toolbar buttons.

3.3.1 *IconHK in a nutshell*

IconHK is an approach that aims to visually describe a keyboard shortcut within a toolbar button. In the following, I will refer to button as an interactive control to execute a command, whose icon occupies a bounded physical space of a button (typically a toolbar button); hotkey as the character of a keyboard shortcut (which is generally a letter) and modifiers as its modifier key combination (respectively S and ctrl for the ctrl + S keyboard shortcut of Save) of a command; pictograph as the pictorial element of a button's icon that conveys the meaning of the command (e.g. the floppy disk for a Save command button); and symbol as the visually embedded hotkey letter (e.g. S for the ctrl + S keyboard shortcut of the Save command).

3.3.1.1 *Visually embedding keyboard shortcuts into toolbar buttons*

IconHK utilizes three strategies to incorporate a symbol representing the hotkey into the button's icon:


Empty space. Uses the empty space surrounding the pictograph on the button to display the symbol, typically by placing the symbol in the available space.


Positive space. Displays the symbol by utilizing the silhouette or the most prominent features, such as the edges, of the pictograph. By strategically incorporating the symbol within these elements, it becomes integrated into the overall design of the icon.


Negative space. Involves leveraging the open space around an object to disguise a letter representing the hotkey. This technique takes advantage of figure-ground ambiguity, where the visual perception affords two alternative viewpoints. By manipulating the background and foreground elements, a letter representing the hotkey can be hidden within the icon.

These three approaches offer a wide range of possibilities to embed the hotkey symbol into existing icons.

Embedding the sole hotkey symbol in an icon is sufficient when the keyboard shortcuts do not involve modifier keys (e.g, switching tools in Adobe Photoshop, or Apple Final Cut) or when the shortcuts consistently involve the same modifier, typically control or ⌘. When different modifiers are involved in the same application, further indications are necessary.

In order to communicate these modifiers to the user, we use the corners of the button. Each modifier of the keyboard (typically, shift, alt and control on MS Windows) is associated to one of the corner, following a spatial mapping based on its position on the keyboard. Then, each corner of a button is filled depending on whether the corresponding modifier is used in the keyboard shortcut or not. Using the example on Figure 3.9, both top-left (shift) and bottom-left (ctrl) corners are filled for the Ctrl+Shift+W keyboard shortcut. Finally, to help users understand this corner/modifier mapping, the color of a corner changes when the user presses the corresponding modifier key.

Figure 3.9: Example of an IconHK Button using the empty space strategy for the Ctrl+Shift+W keyboard shortcut

3.3.1.2 *The magnification continuum of IconHK*

While we assume that modifiers can be displayed at all times without impacting buttons' readability, a compromise might have to be found for symbols. Typically, in the context of empty space, one may not desire to have the letter permanently displayed (even though the writer of this thesis personally would not mind). In the context of positive or negative space, displaying the letter in permanence might affect aesthetic, or hinder pictograph recognition.

In order to overcome this issue, IconHK relies on a magnification continuum, a spectrum of representations resulting from a progressive morphing from the pictograph representation, to the symbol only (Figure 3.10). As we move along the magnification continuum, away from showing only the pictograph, the symbol is progressively revealed until it becomes fully legible by augmenting its saliency. This effect can be achieved with different transformations such as rotation, translation, scaling as well as manipulation of the

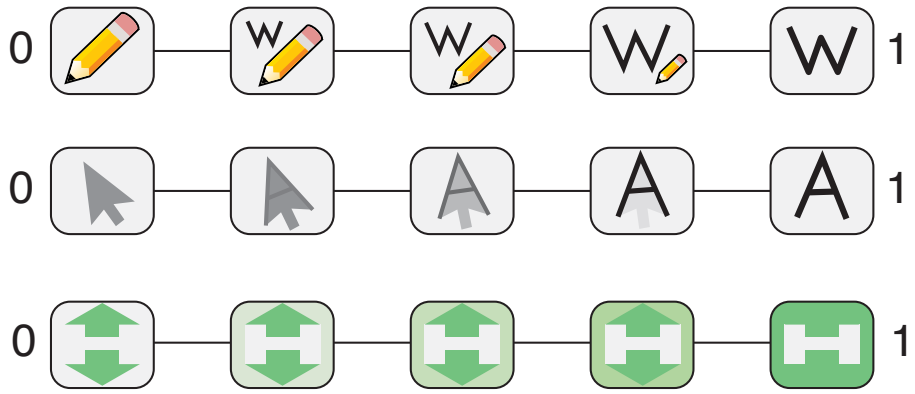

Figure 3.10: Variations of toolbar buttons from 0 to 1 on the IconHK continuum for the *Pencil*, *Move* and *Expand vertically* commands: *Pencil* scales down while a W scales up, exploiting the empty space; *Move* rotates and fades out the pointer while revealing a A overlaying the edges; *Expand vertically* changes the background color of the button to emphasize the H symbol in the negative space. (Animated Figure)

opacity or color of the foreground and background. The approaches depend on the magnification strategy and the designer's preference.

Finally, the level at which a given toolbar button decides to position itself on the IconHK magnification continuum can vary dynamically, especially as a result of users' interaction. For instance, an application could emphasize symbols of all toolbar buttons for a few seconds at launch up, before animating back to the more traditional pictographs. This could also be triggered every now and then to foster awareness and recall. Another possibility could be to modify the magnification level of icons depending on application usage. Basically, the more the user select a command by clicking on the toolbar button, the more it moves towards the hotkey end of the continuum. Conversely, the more the selection is performed using the keyboard shortcut, the more it moves towards the pictograph end. The dynamic behavior of icons could also vary depending on users' interactions with the system at a much lower level, as for instance, when hovering over the icon or when pressing a modifier key, similarly to ExposeHK. Finally,nother approach could be to jump to the hotkey end when the button is clicked, temporarily emphasizing the hotkey and then animating back to its default position along the continuum.

3.3.2 *Recognizing keyboard shortcuts thanks to IconHK*

We conducted a study to assess how efficiently users could retrieve a corresponding keyboard shortcut for each of the different strategies used. We also included a control condition in which keyboard shortcuts are never displayed on the toolbar button.

This experiment relied on the alternance of training and test phases. During the training phases, the experimental interface displayed a toolbar consisting of IconHK icons using all three strategies with the letter moderately emphasized (see Figure 3.11). In these phases, participants were free to select a command either by using the toolbar button or by keyboard shortcuts. Then during the test phases, the experimental interface hid the toolbar and displayed as a stimulus the icon of a toolbar button, but this time without emphasizing the letter (on the pictograph end of the magnification continuum then), and participants were instructed to select these commands using their corresponding keyboard shortcuts. Finally, we performed similar recall and retention tests 24 and 72 hours after the end of the experiment.

Figure 3.11: Toolbar used in our experiment

The results of this study are illustrated Figure 3.12 and clearly reveal two groups: a first group corresponding to the buttons implementing the Control condition and the Empty space strategy, where participants users had overall difficulties to retrieve the keyboard shortcut when the symbol was not emphasized; and a second group corresponding to the buttons implementing the Positive and negative space strategies, where participants were very quickly able to retrieve the keyboard shortcuts. More interestingly, for this second group, participants were still able to retrieve 100% of the shortcuts 72 hours later, against an average retention rate of 77% for the first group (71% for the control condition and 83% for the empty space strategy). Finally, it is important to note that participants spontaneously discovered the IconHK principle and that the experimenters did not explain it at all before the very end of the experiment (after the 72h retention test).

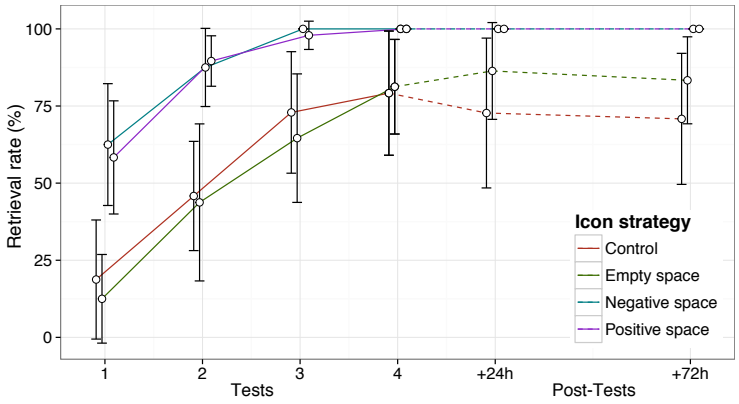

Figure 3.12: Percentage of correct recognition rates for each Icon strategy, for every block and retention tests (dashed)

3.3.3 *Discussing the feasibility, benefits and limitations of IconHK*

During the IconHK project, we explored whether a toolbar button could, within its boundaries, convey the keyboard shortcut associated to the command the button activates. In that respect, we defined a simple encoding for modifier keys and reflected on how the character key of the keyboard shortcut could be integrated with the pictograph of the button via three different strategies and a magnification continuum.

Interestingly, existing commands and buttons already employ a somewhat similar strategies. Obvious examples are the buttons for the Bold, Italic and Underline commands in most word processing applications, that display the B, I and U letters, respectively, on their button. But a more intriguing and interesting example is the *Cut* command, whose keyboard shortcut is Ctrl+X and the icon used conventionally a pair of scissors that look like a X. The toolbar button for activating the cut command can therefore be seen as an IconHK button, implementing a positive space strategy at the pictograph end of the continuum. That being said, we do not know if the character X was chosen for the Cut command because it looks like scissors, because the key is adjacent to the C letter on a QWERTY keyboard, for both reasons, or for another reason. But I must confess that, even if this is purely coincidental, I find this to the least amusing.

That being said, I must acknowledge that I know, the IconHK principle conflicts with the practice of many designers who simply do not care about com-

municating the keyboard shortcut via button design and will tend to prioritize minimalism is consistency for subjective aesthetic reasons over usability reasons. During the IconHK project, we re-designed the Photoshop tool palette, so it implements IconHK principles (see figure ref). This was mostly designed by myself and is, of course, questionable from many "graphics interface designer" perspectives but it, at least, shows that the palette can be easily adapted to implement the IconHK protocol with minimal changes. Moreover, IconHK is particularly interesting for keyboard shortcuts like the ones for switching between tools in Adobe Photoshop (or illustrator, or other) that are *keyboardless*. Indeed, in that case, the feedforward mechanisms that emphasize the keyboard shortcut when a modifier key is pressed seem less coherent to apply. Such strategy was therefore employed in an implementation of Histomages [34] that can be found on the animated transition website [2].

Finally, the findings of the memorization experiment we conducted are interesting on several aspects. First, it was positively surprising to see that participants spontaneously understood the principles behind the IconHK positive and negative space strategy. I also must note that after having understood the IconHK principles, some participants of our experiment were assuming more buttons implementing the IconHK principle and reported having created visual mnemonics to associate some keyboard shortcuts to the icons.

In the end, IconHK demonstrate that keyboard shortcuts do not need to systematically ask users to memorize and recall the corresponding key sequence, but that it can be encoded within the button design, so users can recognize it.

3.4 THE CASE OF SOFTWARE KEYBOARD SHORTCUTS

One conspicuous feature of the recent evolution of interactive devices is the importance taken by touch-sensitive surfaces in commercial products. In particular, smartphones have been a real tidal wave on the cell phone market. These devices do not rely on external devices, such as a keyboard or a mouse, to support indirect interaction anymore. Typically, the hardware keyboard that is used with desktop computers, always almost available, has been replaced with transient software keyboards that pop-in only when considered useful, that is, that the focus is in a text field (e.g. when typing a URL in a web browser bar or editing a text document). As a result, efficient command selection techniques such as keyboard shortcuts are not available anymore.

Yet, I believe that an efficient keyboard-based command selection technique could be designed for soft-based keyboards. All the more, I believe

that soft-keyboards are adapted to support an efficient recognition-based command selection technique. This is why, in the Soft-Keyboard Shortcuts project [HDR2, HDR3] (abbreviated as *Softcuts*), we advocated that despite the lack of physical keyboards on touch-based devices, keyboard shortcuts could be available using the soft-keyboards as leverage.

Indeed, soft keyboard are available on virtually every touch-based devices. Moreover, the fact that a software keyboard can on demand update what each keys display is actually an excellent feature to support keyboard shortcuts, as it won't require users to have to memorize the key correspondence beforehand: the key could simply update its appearance to reflect the command associated with it.

In this section, I will present the results of a work conducted during a remote collaboration with Katherine Fennedy and Dr Simon Perrault, her PhD co-supervisor at that time. I will first present the input technique of Softcuts and then review some studies we have conducted on the discoverability of Softcuts, as well as a study conducted regarding user performance depending on the type of mapping the commands rely on.

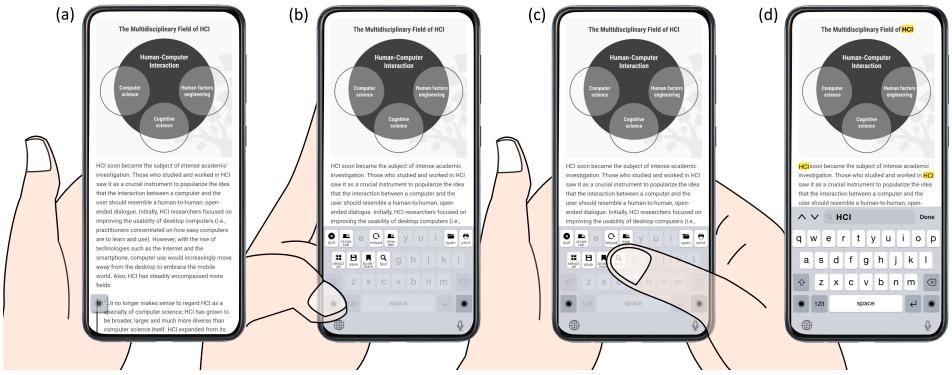

Figure 3.13: Web browsing scenario of soft keyboard shortcuts using Once method. (a) a semi-transparent modifier key is available in the browser, (b) when user taps it, all available shortcuts will be presented to facilitate browsing, (c) user then taps on the 'F' key (find command) and hence, (d) every occurrence of the "HCI" word (as typed by the user) is highlighted in yellow

3.4.1 *Selecting commands with SoftCuts*

Soft-Keyboard Shortcuts (abbreviated as *Softcuts*) are a simple interaction mechanism to support command selection on a touch-based device (Figure 3.13). It relies on a soft-keyboard that can switch to a command mode when a dedicated key is pressed, mode in which hitting another key activates the corresponding command instead of typing the corresponding character. To be more precise, selecting a command with a Softcut can be achieved in three different ways:
(1 Once mode:) The user taps the modifier key, which switches the mode of the keyboard subsequently displaying commands. The user then taps the key of the desired command which switches the mode of the keyboard back to default.
(2 User maintained mode:) The user presses and holds the modifier key. The keyboard remains in command mode as long as the modifier key is held and the user can tap the key of the desired command(s) to activate them.
(3 Swipe based:) The user presses the modifier key (which sets the keyboard to command mode), moves the finger on the surface to the key of the desired command, and lifts the finger to validate that command.

In our work [HDR2], we contrasted these three different ways in two studies that I will not detail in the present document, and refer the reader to the published research paper for more details.

3.4.2 *Discoverability of Softcuts*

We conducted a first study in order to investigate the discoverability of Softcuts when users are not aware of their availability. Conceptually, our intention was to see if users would notice the subtle addition of a modifier key on a soft-keyboard (that usually does not have it), and if it would result in them performing commands using Softcuts. This experiment was conducted using a crowdsourcing platform where participants were asked had to complete two consecutive text formatting tasks by applying various commands to the text content. No text entry was possible nor required. The first task could be completed by using commands that were all located in the conventional application menu, testing for the *spontaneous* discoverability of Softcuts. The second task, however, required to apply text-formatting commands that could only be activated by using their corresponding Softcuts, testing for the *enforced* discoverability of Softcuts. We then computed a *spontaneous* discovery rate (proportion of participants that did discover Softcuts during the first task), an

enforced discovery rate (proportion of participants that did discover Softcuts during the second task), and an *overall* discovery rate (proportion of participants that did discover Softcuts during this experiment, regardless of during which task).

The experimental interface presented to participants varied along two factors. First, it used two different designs for the modifier key, differing in terms of *familiarity*: a *Cmd/Ctrl* key (depending on participants' most familiar operating system) and an *abstract* key (displaying an arbitrary symbol). Second, the keyboard was either *permanently* displayed on-screen, or *transiently* displayed in which case participants would only see the modifier key in the bottom-left corner of the scree, thus varying the *saliency* of the modifier key and therefore the likeliness to press it. The four possible conditions of factors were counter-balanced across participants using a between-subject design. Finally, participants were asked to self-report their estimated frequency of usage of conventional keyboard shortcuts on a scale ranging from 1 (never) to 7 (always), as well as to report an estimation of how many of them they know and use.

We recruited 160 participants (77 were male, 81 were female, 1 of them non-binary, and 1 preferred not to disclose; age ranging between 19 and 75) that were assigned to one of the 4 conditions evenly. A total of 9 (5.6%) participants failed (or gave up) to complete the first task, 59 (36.9%) failed (or gave up) to complete the second task, 92 (57.5%) completed both tasks.

Overall, participants discovered Softcuts spontaneously at a similarly low rate of 18.1% (29 out of 160). There was no main effect of *familiarity* or *saliency* on spontaneous discovery rate. There was, however, a significant main effect of the number of keyboard shortcuts known/used, with participants who declared knowing/using 6-10 (25.9%), 21-30 (66.7%), and >30 keyboard shortcuts (36.4%), having a higher rate than those with only 1-5 keyboard shortcuts (4.5%). In addition, participants discovered Softcuts at a higher rate of 57.7% (71 out of 123) when they were enforced to do so (Figure 3.14). However, there was no main effect of *familiarity* nor *saliency*. Finally, participants discovered Softcuts overall at a proportion of 62.5% (100 out of 160).

3.4.3 *Learning of Softcuts*

Another question we were asking ourselves regarding Softcuts was how easily users can learn command mappings and if prior knowledge of command mapping does significantly impact this learning process. We therefore conducted a study relying on a simple task in which participants would have to

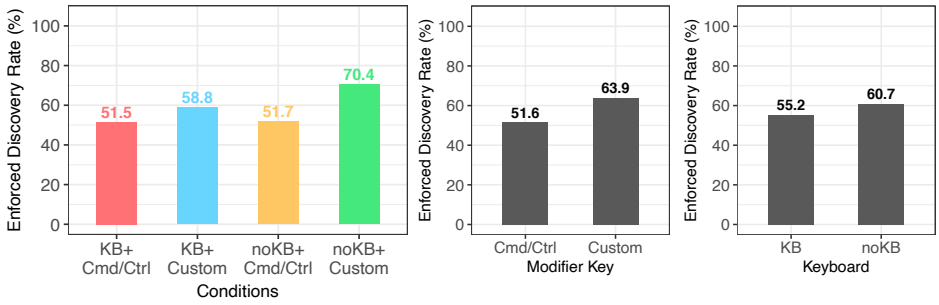

Figure 3.14: Enforced discovery rate of Softcuts (a) across all conditions, (b) aggregated by familiarity (i.e. type of modifier key used), and (c) aggregated by saliency (i.e. presence of keyboard)

repeatedly select commands displayed at the top of the screen using the corresponding Softcuts.

We tested two different command mappings of 16 command each: an *abstract* mapping consisting of the flags of 16 countries (Figure 16-b,c) arbitrarily associated with some keys, and a *realistic* mapping consisting of 16 existing keyboard shortcuts from Microsoft Word for Windows. For the need of the experiment, 8 commands per mapping were selected as targets the other acting as distractors. For each mapping, participants performed 9 experimental blocks (B1-B9). Each block consisted of 16 trials, where each of the 8 chosen target commands was repeated twice in random order. Finally, we measured retention rates on the 8 abstract and realistic commands at multiple phases: at the end of the study and 1, 3 and 7 days after the experiment. These post-experiment retention rates were tested by presenting an ordinary keyboard layout, and asking participants to map each of the target commands used during the experiment to keys of that keyboard.

Our main dependent measures were command selection time and retention rates. We recruited 12 participants (8 female, 4 male, all right-handed), aged 19 to 29.

First and foremost, command selection time demonstrated a very rapid learning effect for both mappings, with statistical tests demonstrating a significant difference between the first one and all the others. Therefore, we removed the first block for all analysis. We found that while participants selected on average commands significantly faster in the realistic mapping (1.283s) than the abstract one (1.352s), this difference remained marginal (69ms). Second, regarding retention rate (Figure 3.15-left), participants could recall the

spatial layout of commands of the realistic mapping (95.1%) significantly more efficiently than that of the abstract one (55.7%), which makes sense since participants were already familiar with them. However, a further analysis of the abstract mapping revealed that the magnitude of retention errors was actually low as we noticed that majority of these errors corresponded to the participants positioning a command on a key adjacent to the actual target (see Figure 3.15-right).

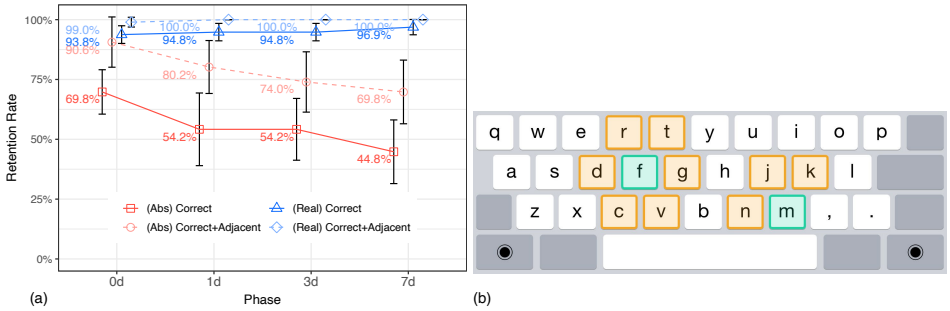

Figure 3.15: *Retention rate* by phase (at the end of the experiment (0d), after 1 day, 3 days, and 7 days) for each mapping. "Correct" rates indicate that the participant recalled the right key, while "Correct+Adjacent" also includes recalls on one of the adjacent keys. (b) Participants tended to misplace shortcuts from the target location (e.g. in green) to one of the keys adjacent to the target (e.g. in orange)

3.4.4 *Reflecting on software keyboard shortcuts*

First, regarding the discoverability of Softcuts participants did, somewhat unsurprisingly, discovered Softcuts more frequently when enforced to do so. What might be surprising, however, is the magnitude of the difference, since the discovery rate more than tripled, moving from a *spontaneous* discovery rate at 18.1% to an *enforced* discovery rate at 57.7%. It was also interesting to notice that the appearance and saliency of the modifier key does not seem to impact the discovery of Softcuts, whereas familiarity and usage of conventional keyboard shortcuts seems to do.

Then, regarding learning of Softcuts, we first observed a similar learning rate for both conditions, suggesting that Softcuts are rapid to learn with an abstract mapping. Second, even if participants were slightly faster with the realistic mapping, we can comfortably say that the abstract mapping still yielded

efficient command selection. Third, we were surprised to learn that with only about 15 minutes spent by each participant to complete the 9 blocks of trials in our study, participants could approximate the command location even after 1 week without rehearsal. From a recognition-ready command selection mechanism perspective, this is a positive result, as it suggests that participants have an approximately correct memory of the spatial location of the command, command that would be positioned where the user will look at when trying to activate this command.

Altogether, these results confirm that Softcuts are an efficient recognition-ready command selection mechanism that can reliably be used as a unified command selection mechanism across various applications on phones and tablets. Softcuts offer a robust multi-device performance with minimal training, are easily discoverable, and can be easily scaled and adapted to various applications.

3.5 ZERO DELAY MARKING MENU

The press-and-wait attribute of Marking Menus, typically a $1/3$ second delay in its original implementation [67], adds a cost to *menu mode* to distinguish, and encourage the use of, *mark mode*. In his doctoral thesis [67], Kurtenbach rationalizes displaying the menu after a certain delay on the basis that the menu *"can be distracting"*, *"can obliterate part of the screen"* and that *"displaying the menu takes time"*. However, despite the extensive study of marking menus, the duration of this delay has surprisingly received little attention.

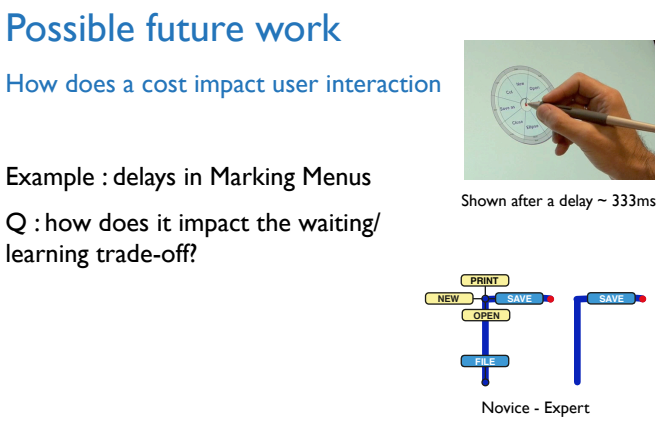

Figure 3.16: Slide that was included in a job interview presentations in 2013

When I was first interviewed for an academic position at Inria back in 2013, I had a slide (see Figure 3.16) to illustrate one possible project of my research agenda. It consisted in testing the impact of an artificial cost applied to an interaction method, and see how it would influence when learning to interact with a system. More precisely, I was planning to test the impact of the duration of the delay (the artificial cost) of a Marking Menu (the interaction method) on mark mode adoption. Once recruited at Inria, I started to work on that project and implemented an interactive playground to test different values of the delay. This interface had a slider to modify the value of the delay without having to restart the application, slider whose default value was set to zero. Launching that application for the first time made me think that I might have been wrong the whole time about this. Maybe the delay was not the solution to the above-mentioned problems introduced by Kurtenbach, but actually the source of another one which is that it creates a cost to novice interaction and forces them to memorize gestures, which from an interaction design perspective is questionable.

Interestingly, the question of the impact of the duration of the delay on mark mode adoption has recently been nicely explored by Lewis and Vogel [73, 74], but without considering the zero-second delay condition. Their results suggest that the longer the delay (that is, the longer the cost of interacting in menu mode), the more users try to select commands in Mark Mode. However, they also found that users make significantly more errors as the duration of the delay increase, but also that users never use mark mode for all command selection, even with a two seconds long delay.

Eventually, I decided to discuss the idea of a zero-second delay, which would basically mean having single-mode Marking Menus that are systematically and instantly displayed as soon as the user starts interacting with them, with my colleagues at Inria Dr Mathieu Nancel and Prof. Edward Lank. Edward was visiting our research group at that time and thought we should pitch this idea to one of his PhD student, Jay Henderson.

In this chapter, I report on the investigation the four of us conducted [HDR7]. Our motivations for doing this were two-fold. First, the delay in *menu mode* creates a cost for novice interaction. While this delay might act as an incentive to use *mark mode*, it is unclear by how much it accelerates learning in real use (i.e. is penalty an effective motivation [71] in context?). Second, while users may use *mark mode* for commonly selected commands, not every command is used frequently, thus *menu mode* interaction remains necessary for many commands, even for more experienced users.

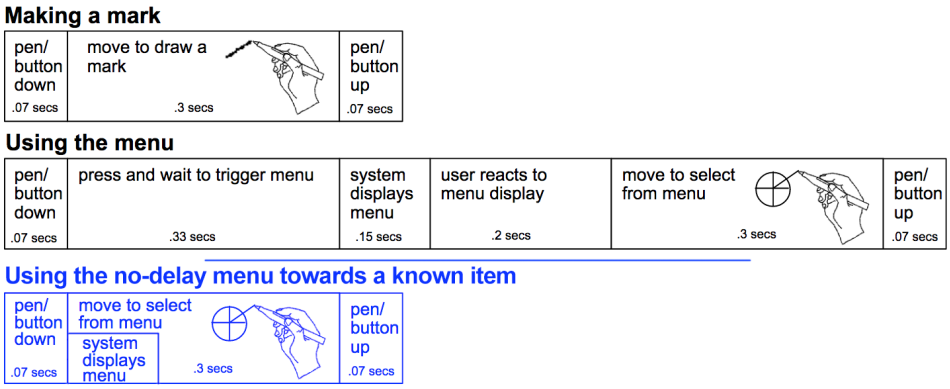

Figure 3.17: Top: Reproduction of figure 4.15 in [67]; Bottom: Our hypothesis on the time costs of a no-delay Marking Menu.

3.5.1 *Rationale for testing a single-mode Marking Menu*

In 1991, Kurtenbach proposed that *"even if a user did not have to pause to signal for the menu to be popped up, one would still have to wait for the menu to be displayed before making a selection. In many systems, displaying the menu can be annoyingly slow and visually disturbing"*. However, no study had been conducted to evaluate this visual disturbance, and, with advances in computing over the past 32 years, the likelihood that displaying a menu would be "annoyingly slow" is low, since the majority of systems display even the most complex user interfaces in milliseconds. Further, effective use of threading in GUI design can ensure that interfaces remain active even in the presence of costly computational tasks and users can act even before the menu appears, if displaying it is slow, as there has been anecdotal evidence that expert users avoid these issues by *"mousing ahead"* in pie menus [58], as cited in [66].

Looking more precisely the issues of mousing ahead and temporal costs, consider Figure 3.17. The top part is a reproduction from figure 4.15 in [67]; it illustrates users' expected behavior when using mark mode. The middle part is user's expected behavior in menu mode, where we can see additional costs to menu mode - including system display and user search costs. However, whether there is a cost to menu display is debatable. Consider the bottom part of Figure 3.17, our representation of an alternative hypothetical time costs of a no-delay marking menu. Given that the system can capture input immediately and continuously, does the user "most likely wait for the menu" (shown as *user reacts to menu display* time in Figure 3.17-middle) if they al-

ready know what gesture to perform? Or might the user simply begin to move, mouse ahead [66], removing the cost of menu display (bottom)? Assuming that after sufficient practice the user acquires knowledge of the visual layout, removing the delay (and therefore the *mark*-only mode) may reduce the temporal penalty without harming memorization. Other possible issues, such as distraction and occlusion, would remain to be investigated. The question then becomes whether we can determine the relative costs of these factors.

In the end, my perspective of imposing a delay to enter menu mode was the following:

- First, delay is a *cost* when performing command selections in *menu mode*, penalizing users if they are unfamiliar with the menu or with the specific command being invoked.
- Second, selecting a command in *mark mode* requires the user to memorize the corresponding mark beforehand, i.e. it leverages recall rather than recognition.
- Finally, with delay possibly acting as an incentive to use *mark mode*, the user may try to select commands via *mark mode* even if when not entirely sure of the correct mark, which might increase error rates even for practiced use [74, 74].

3.5.2 *Estimating the impact of delay*

As a first step in evaluating the relative costs and benefits of Delay versus NoDelay, we conducted a controlled experiment to contrast the impact of delay on the performance of marking menus. We recruited 16 participants who were instructed to select commands as fast and accurate as possible using the provided mouse controller. For each trial, the participant had to select a target command (displayed on top of the window) with a Marking Menu (Delay or NoDelay) of a given Layout (4, 8, 4x4, or 8×8 items). The Delay marking menu had a 333 ms delay to enter into menu mode, and a 200 ms delay to display the sub-menu. For every menu and layout, participants performed 10 Blocks of 8 command selections, except for the 4-item configuration where they performed 10 blocks of 4 command selections only.

The primary dependent measures were *Selection Time* (time from stimulus to correct command selection), *Execution Time* (time from the last mouse press to correct command selection) and *Error Rate*. Additional dependent measures were *Preparation Time* (time between the display of the target item and the first mouse-down event) and the proportion of *mark mode* usage in the Delay condition.

Regarding command selection, it was overall significantly shorter to select commands in the NoDelay (2042 ms) condition than in Delay (2294 ms) condition. Note that this results includes all command selections for both conditions, regardless of the mode they were selected in for the Delay condition. To the least, it suggests that in our experiment, the potential benefits of using mark-mode did not compensate the cost of having to wait in menu-mode, overall. Regarding error rate, participants made significantly more errors with Delay (4.2%) than with NoDelay (1.3%). We also found that Error Rate significantly increased with Blocks overall, but a Marking Menu $\times$ Block interaction effect revealed that Block levels are not different from each other in the NoDelay. In the Delay condition, however, the last two blocks had more errors, respectively 6.9% and 8.5%, as the result of more attempts to select commands in mark mode.

Finally, since there is one single mode of command selection in the NoDelay condition, it is complicated to contrast command selection for practice use the usual way which consists in comparing mark mode usage and command selection time. Therefore, we needed some measure of Practiced Use for the Nodelay. In order to properly compare practiced performance between conditions, we identify the Blocks in which Selection Time stabilized towards its minimum (using Hsu's HSB contrast), i.e. blocks 7 to 9. Regarding Selection time on these blocks, we no longer found an effect of Marking Menu condition nor any interaction effect. Finally, we broke down Selection time into execution time and preparation time (Figure 3.18). We found a significant effect of Marking Menu on Execution time, revealing that the Delay condition had a shorter one (948ms) than the NoDelay condition (1130ms). However, we also found an effect of Marking Menu on Preparation time, with the Delay condition having a longer one (830ms) than the NoDelay condition (569ms).

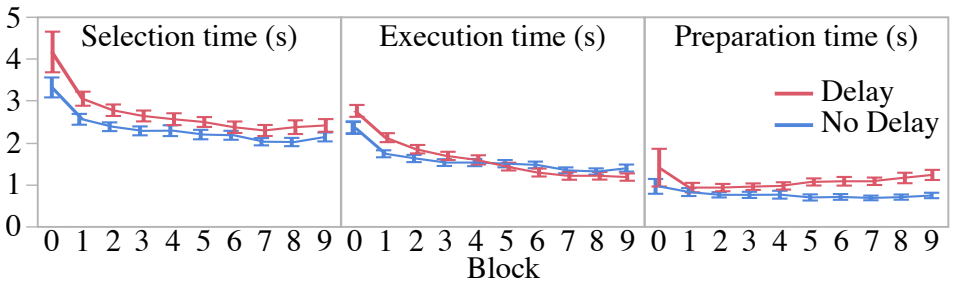

Figure 3.18: Average Selection, Execution, and Preparation Times per Block and Marking Menu condition

In summary, for practiced use, we find no significant difference between Delay and NoDelay in terms of Selection Time, and that while Execution Time is lower with Delay (182 ms difference), it is compensated by a lower Preparation Time (261 ms difference) for NoDelay.

3.5.3 *Focusing on the extremely practiced performance*

In the previous study, we did not find strong evidence that the Delay condition would significantly outperform the NoDelay in practiced use. Moreover, Delay suffered from higher error rates and reaction times, even when selecting commands consistently in mark mode. However, it may be that, with sufficient practice, users could reach a theoretical level of perfect expertise – autonomic response – that would result in an overall performance benefit. Thus, we designed a second study to balance the need for the "best case" of an autonomic response, versus the confound of anticipation, or the cost of deciding what selection to perform.

We recruited 8 participants who, similarly to the previous study, had to perform sequences of command selections using Marking Menus. The difference lies in the fact that they always had to select the same two commands within a Marking Menu of 64 items (8x8). Participants performed 8 blocks of 10 selections per command for each condition. At the end of these 8 blocks, we asked participants to select the two target commands 4 times each, using a directional arrow as instruction, to obtain a lowest possible command selection time.

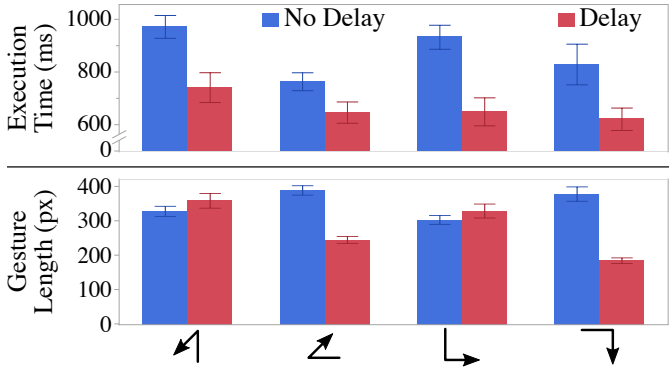

Figure 3.19: Effects of Menu condition on Execution time vs Gesture length for each item. Mind the non-zeroed Y-axis on top

Results are illustrated Figure 3.19. This time, we found a significant difference on gesture execution time, which was shorter in the Delay condition (mean 663 ms vs. 873 ms with NoDelay). This difference directly translated into a shorter Selection time (mean 1380 ms, vs. 1633 ms with NoDelay), i.e. a total improvement of about 250 ms. These results suggest that, after extensive training with two targets only, command selection in mark mode with Delay marking menus is shorter than with No Delay.

3.5.4 *Assessing the visual disruption of Marking Menus*

When Marking Menus were introduced, the rationale for not instantly displaying them was that they can *"be distracting"* and *"obliterate part of the screen"* [67].

In order to test these assumptions, we conducted a third experiment that relied on a simple yet realistic graphic arrangement task. 16 participants were instructed to recreate a reference image (cartoons and flow charts displayed in the top panel of our application), in a below panel by "spawning" and modifying graphical items using a marking menu and direct manipulation (see Figure 3.20 for an example). A right-button mouse press triggered a two-level marking menu containing the various available graphical items, organized in categories. Upon selection, the graphical item spawned onto the page under the cursor. The position of these items could be adjusted by drag-and-dropping them within the panel using the left mouse button. A right-button press on any of these items triggered a contextual marking menus containing item manipulation commands, such as "send to back", "rotate", etc. whose effects were applied upon selection.

To compare the subjective experience of Delay and No Delay, in addition to capturing spontaneous comments expressed during the tasks using a think-aloud protocol, participants were given after each condition a questionnaire with various questions (see Figure 3.21).

Regarding the results, first, note that in the Delay condition, 34% of command selections were performed entirely in mark mode, suggesting that approximately one selection out of three was without visual appearance of the menu. Despite this, subjective evaluations showed general patterns for both conditions. Performance-related questions with the already remembered items (a, b) earned neutral or favorable scores (4–7). Only one participant scored above 5 on question (g) (No Delay condition), indicating that participants were not generally affected by occlusion. Only two individuals, both for Delay, scored

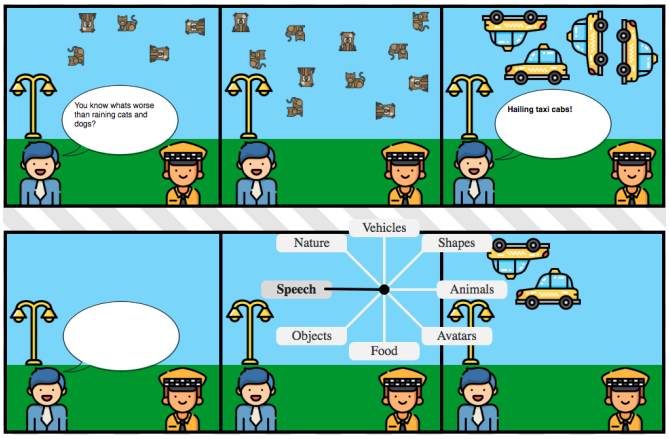

Figure 3.20: An example of a user interacting with the drag and drop application. On top is the figure to recreate. On the bottom is the participant's current figure with an ongoing menu selection. Icons in this figure were designed by Freepik and Smashicons from www.flaticon.com

lower than neutral (4) on the questions on memorizing (e) and real-world use (i).

Of particular interest, the participants reported having the impression they lost focus of the main task more in the Delay condition (mean score 4, most frequent scores 1, 3, 4, and 7 each with N=3) than in the No-Delay condition (mean 2.56, most frequent score 1 with N=7). All other differences were non-significant.

Finally, we did not observe a consensus among participants against Delay or No Delay marking menu in terms of disruption or disturbance. For example, P8 said they "didn't find any disturbance [in No Delay]" and P12 mentioned "I noticed no disturbance or disruption [in No Delay] because I could always control visibility of the marking menu by simply releasing the mouse button". Only one participant (P6) reported an issue with object occlusion in the No Delay condition, specifically when attempting to rotate an arrow image: "I cannot see which direction the arrow currently is, so I don't know how to rotate it. The arrow is hidden, the menu should come somewhere else without hiding the picture [...] The menu should not hide the existing element about to be manipulated". Nevertheless, this specific problem only affected one participant and would exist in both conditions for a person unfamiliar with the menu structure.

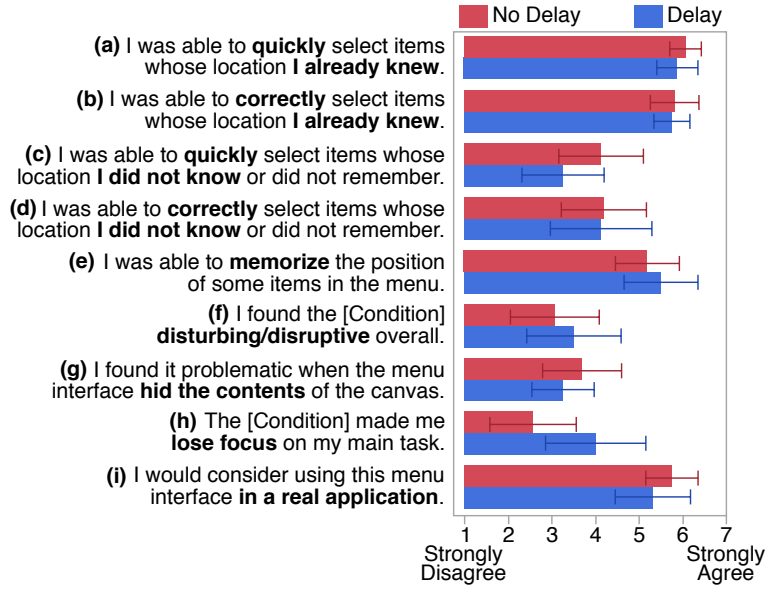

Figure 3.21: Results of the subjective assessments of participants between a standard and a zero-delay Marking Menus

3.5.5 *Reflecting on a zero-delay Marking Menu*

The findings of the first study suggest strong benefits for the No Delay during the learning phase, and therefore, for all commands a user might be unfamiliar with. They also suggest that users made fewer errors in absence of a delay. On the other hand, the results of study 2 demonstrate that users take less time to select commands in the Delay condition, but it required an extensive training suggesting that this small benefit might be present for only very few commands. However, it remains unclear whether users would be able to reach expertise with enough commands for the delay to be beneficial overall when novice, practiced, and expert behaviors are considered altogether. For sure, it is unrealistic to expect a user to remember a complete menu layout: even the best projections would be far from selecting all commands in *mark mode* after extensive use, reaching a proportion over 55% only when for extremely long delays (more than 2 seconds long) imposing a heavy cost on novice interaction and significantly increasing error rate [73].

The most surprising and controversial result in this work is the lack of subjective impact of visual disruption. In our third study, we note limited issues

around the alleged visual occlusion, even though *mark mode* was used 34% of the time during the Delay condition. In fact, we noted significantly more impact due to the cost of Delay than issues with occlusion. However, in hindsight it is questionable whether this result should be surprising: Previous models on command selection in linear menus [17, 35] include no temporal cost that results from visual disruption, and still correlate perfectly with user performance. Regardless, visual disruption may also be an over-stated issue: while one way to eliminate visual disruption caused by occlusion is to avoid displaying the menu, partial transparency of the menu may also allow users to continue to see underlying content without the need to include novice mode penalty and increased error rate in marking menus.

Overall, this work opens similar questions regarding other command selection techniques that rely on delay-separated modes and paves the way to explore alternative adaptations of Marking Menus or other mode delimiters.

3.6 LESSONS LEARNED FROM THE DESIGN OF RECOGNITION-READY COMMAND SELECTION MECHANISMS

In this chapter, I presented several interaction techniques for command selection that are *recognition-ready*, that is that they were designed with the premise that they should show the commands to the user and mostly trust their visual perception. Yet, all these command selection mechanisms remain efficient from a command selection time perspective compared to their recall-based counterpart. More interesting, we did not find evidence that the visual aid that was provided to users in several projects had actually any negative effect on users' subjective opinion, unlike what had been claimed evidencelessly in previous work, and repeated several times since by the community, including myself.

These work make me posit the hypothesis that explicit memorization, recall and modes **are not necessary** for efficient command selection. The myth that it would be might come from the fact that so-called shortcuts introduced in the 80s, typically keyboard shortcuts as an alternative to pointer-based command activation, rely on two different input modalities that simply cannot compete from an execution time perspective. A direct command selection using a key-chord, once learned and practiced, will always be faster than a hierarchical pointer-based command selection. Therefore, the question should be how can we train and assist users in keyboard-based command selection, rather than whether it is a shortcut or not. Indeed, we did see in several projects that even though these techniques do not need users to explicitly

memorize a gestural shortcut or key-chord correspondence, other implicit memorization mechanism might come into play and be a side effect of using the recognition-based interaction, even though users systematically have a visual aid. We could witness this in the ExposeHK usage and SoftCuts memorization tests.

On very long term, maybe these recognition-ready techniques might be negatively impacting the best command selection time one might reach, but such evidence should be provided as well. And even if it is the case, shall the designs negatively affect users for all commands they are not highly familiar with, and continuously during their familiarization process, just for the benefits of very few commands that will be repeated so many times that not having a visual aid systematically displayed would save a few milliseconds?

4

COMMUNICATING INTERACTION POSSIBILITIES

4.1 INTERACTION MECHANISMS NEED TO BE DISCOVERED

In the previous section, we saw the specific case of interaction with several modalities, the problem(s) associated with having two modalities and their associated transitions, and showed efficient interaction does not necessarily need multiple modalities or modes. Another problem inherent to the existence of two distinct modalities is the fact that users need to be *aware* of the alternative methods in order to be able to adopt it. This issue was particularly salient to me in 2013 when I did work on the ExposeHK project. When pilot studying the experimental procedure, several participants did not use a single keyboard shortcut, and as such, did not realize they could expose keyboard shortcuts by pressing a modifier key. To overcome this issue, all three studies included explicit instruction and practice with ExposeHK prior to experimental tasks. But the question of how it would be discovered and used in practice remained. I briefly mentioned this issue of not discovering a modality in the seminal publication of the Skillometers [79] project, when I described the intermodal improvement performance domain, emphasizing the need of *Awareness of other modalities*. Interestingly, this resonates with one comment that we received from Gordon Kurtenbach when we interviewed him by e-mail for the ZeroDelay Marking menus project[1], described section 3.5, where he noted that with a "very small delay, some users never used the marks", but also that "many people have to be told about mark mode. They don't seem to discover it". Complementary to this, Nicole Pong, a PhD student I supervised, investigated the interaction with "signifier-less" user interfaces, and more specifically *Swhidgets* (portmanteau for *Swipe-revealed hidden widgets*), widgets that by default are entirely hidden under another interface element or the screen bezels [HDR12]. In her work, she conducted a study that revealed that even though users are aware of a moderate number of these widgets that must be explicitly revealed by the user, they never know all of them. In addition, they very rarely figure out the existence of these controls directly via the interface, but mostly through external sources, be it because they witnessed someone else using it or watched a tutorial video [99].

1 I refer the reader to the corresponding publication for details about this exchange [HDR7]

I posit that when an interaction possibility is not explicitly (visually) communicated to the user, it is therefore harder to recognize and in the end, to discover. Event though interactive systems design guidelines almost systematically highlight how important it is for the user to be able to "figure out what actions are possible and where and how to perform them" [94], and how interactive systems inputs and features should be ***discoverable***. Yet, the introduction and validation of novel input methods and features, be it in research papers or commercial products, almost systematically **ignore how the system could foster the discovery of these mechanisms** besides by *"providing help and documentation"* [92]. How such input methods could be discovered in practice are usually limited to a paragraph, or a subsection at best, in the discussion sections of the paper, just like I did in the ExposeHK paper [HDR10].

The poor discoverability of an input method can significantly impact the adoption of an interactive system. This problem can be typically illustrated with the introduction of force input that was available in touch-based devices (e.g. Apple iPhones, Huawei Mate S) not so long ago: research in HCI has repeatedly demonstrated the potential benefits of using force input for increasing users' bandwidth [6, 37, 55]. Yet, while supported, force-input remains restricted in current touch-based OS, and analysts suggest Apple did remove it from their most recent iPhones [2] because of the cost of embedding force-sensing technologies into smartphones for something that *"nobody is really using"*.

I argue that when available in modern smartphones, if force-based input is barely used it is not because it is not useful, but simply because users do not know when it is supported, do not know what they could achieve with it and are not taught how to use it.

This problem is not restricted to force-input, and exists regardless of the computing system considered, including gestural interaction (What are the available gestures and how can I make sure that the system will properly recognize them?) or more exotic environments such as Mixed Reality (What can I do in a world where my avatar has virtually endless capabilities?).

In this chapter, I will reflect on the discoverability and communicability of interaction. First, I will summarize a work that reviews the different definitions of discoverability and clarify the concept. Then, I will describe how providing a simple visual feedback on force-based input can help to discover this input capabilities. After, I will report on a study investigating how in-place touch inputs could be visually communicated by interface widgets. Finally, I will de-

2 https://www.theverge.com/circuitbreaker/2018/8/27/17787938/apple-new-iphone-3d-touch-pencil-support-missing

scribe a work attempting to communicate micro-gestural interaction visually to the user, including for interaction in a mixed reality context.

4.1.1 *Discoverability and Communicability*

Discovering that a specific interaction is possible is the first step towards engaging and utilizing an interactive technology. Such *initial* discoveries rely on users' having the ability to perceive relevant parts of a system, comprehend them as such and interpret their functionality correctly.

Being able to perceive the availability of interaction possibilities is of particular importance in modern devices, which tend to support more diverse input methods. This is exacerbated as visual guidance in the interface is at the same time reduced, possibly to minimize visual clutter [98, 99]. With this initial discovery harder to achieve, users can completely miss an interaction possibility that would help to use a system to its full potential [4, 61].

In order to investigate the persistence of discovery problems and possible solutions, Eva Mackamul, a PhD student I co-supervise with Géry Casiez, did survey the concept of discoverability when referring to user interaction with systems. Its definitions vary, but generally describes users encountering something new that they were not previously aware of, therefore *discovering* it. The term was popularized by Don Norman who describes it through the following question:

"Is it possible to even figure out what actions are possible and where and how to perform them?" [94]

However, this definition remains relatively high level and ambiguous. In the more specific context of HCI, we split discoverability into three focus areas:

- **System discoverability** is concerned with whether users notice and recognize a novel system as something they can interact with, for instance that they can interact with a public display in a mall.
- **Interaction discoverability** is concerned with whether users can identify and successfully use novel interaction and input methods when they encounter them without instruction or guidance, for instance that an iOS icon can be force pressed to reveal a menu.
- **Feature discoverability** is concerned with users' ability to discover features or functionalities they were previously unaware of while using a system, for instance that a command to comment a block of text is available in a programming IDE.

While this separation between different discoverabilities has been implicit with research specifying a focus on interaction discoverability [33] or feature discoverability [108], these focus areas have not been collated and considered in relation to one another. This is unfortunate given that, even though they are separate considerations, these different discoverabilities are interconnected.

In an unpublished work (that I will not describe in this document) we review, among others, the use of Discovery in HCI and clarify how the concept connects to other important design properties. Of particular interest is the notion of *Communicability*, introduced within Human-Computer Interaction in the book *The semiotic engineering of human-computer interaction* by Clarisse de Souza [42]. She describes communicability as *"the distinctive quality of interactive computer-based systems that communicate efficiently and effectively to users their underlying design intent and interactive principles"*. In other words, communicability presents a system perspective of the discovery and comprehension process but without actually focusing on whether users will perceive these principles. Instead, it focuses on the system's ability to communicate its purpose and interactivity to the user.

Back in the years, paper manuals were provided when one was buying an interactive system or software. For manifold reasons, such as because they cost to print or to avoid waste, instruction manuals are now less and less provided. But another possible reason might be that readers simply do not read them [22]. These instruction manuals have been moved to manufacturers' websites that users may still not read [22], or have been replaced by online tutorials or demonstrations in keynotes or advertisements [87] that only the most tech-savvy users would watch.

Regardless, in an increasingly complex technological world, the first condition for allowing one to operate with interactive systems is the ability to detect and properly interpret what we are confronted with. My main concern does not lie in the fact that manuals are not provided anymore, they have always been away from where and when the help is needed, and requiring users to explicitly read them. It lies in the fact that, I believe, the graphical interface itself should try to communicate input possibilities to the user. Indeed, the main approach interaction designers use to instruct about a system and communicate interaction possibilities directly from the system itself is via in-app tutorials that are displayed on first launch. These *tour guide* or *onboarding* [63] tutorials are rich and useful. However, they expose the user to too much information and at a time it is not directly needed.

4.2 COMMUNICATING FORCE-BASED TOUCH-INTERACTION FOR TEXT SELECTION

As discussed above in section 4.1, previous iPhone models used to support force-based input, that is, that they sensed how strong a finger in contact with a surface was pushing that surface. Force-sensing capabilities had always motivated the design of novel input techniques that would increase user interaction possibilities (e.g. [6, 37]). Of course, these work always took users' awareness of the input modalities for granted and never considered how this could be communicated to the user. Like many before us, motivated by the force-sensing capabilities of certain models of iPhone, Alix Goguey, Carl Gutwin and I worked on how force-based input could be leveraged to improve text selection on smartphones.

Indeed, text selection on touch-based devices can be a difficult task for users. The difficulties arise because letters and words are small targets, while our fingers are relatively large, making both targeting inaccurate and visual occlusion. As a result, since direct touch alone is too imprecise and difficult to rely on it exclusively for text selection, mobile platforms embed several additions to overcome this problem: magnifier views [117], selection handles, or word snapping.

With the introduction of force-sensitive touchscreens on iPhone 6S, yet another augmentation had been added to text selection. It made use of force, in the sense that user could perform a sequence of hard presses, as a way to switch between word-level, sentence-level, and paragraph-level text selection. However, the force capabilities in this technique had no visual signifiers to inform users that they could rely on this feature.

It is, however, important for infrequent users that the system exposes the features of the technique, especially that the device force-sensing capabilities of the device are mapped to text-selection granularity. In that respect, visual signifiers should invite the user to explore the possibilities of the technique, and should not hide information about the interaction mechanisms. For example, force-based selection in iOS enables sentence-level and paragraph-level selection features, but these are hidden from the user – most users probably discover this feature by accident, through social suggestion or by reading online documentation. In contrast, the visual signifiers of handles probably emphasize that interactions with handles are possible.

4.2.1 *The force-based text selection techniques*

In this work, we proposed two novel force-based text-selection techniques: ForceSelect (FS) and ForceSelect High-Pressure (FSHP).

Both techniques, as well as the default iOS text selection technique that could be used on iPhone XS and 6S under iOS 10, exploited force sensed input to modify the magnitude of the text selection. Typically, ForceSelect (Figure 4.1-left) maps the force sensed by the touchscreen to the granularity of text selection: the higher is the force, the more text is selected. FSHP also maps force directly to the granularity of text selection, but will eventually validate a granularity only if force was maintained at that level for a given time (300ms). Finally, the NORMAL iOS technique also uses force, but does not map it directly to the granularity of text selection. Instead, the user can increase and decrease the force applied (simulating a physical *press* action) which results in selecting the word located below the finger and displaying two cursors right before (respectively after) the first (last) character of the word. Moving the magnifier around snaps one of the text selection handles to the nearest word while the other cursor stays at its initial position. If the user performs another *press* action, the cursors disappear and moving the magnifier highlights the word under the finger. Additional presses on the screen cycle through these two modes. Three rapid presses enable a similar snapping mode at the sentence level, and four enables snapping at the paragraph level. We refer the reader to the original publication [HDR6] for details on how force is exploited in detail by each technique.

That being said, how force is exploited by each technique to modify the granularity of text selection is somewhat irrelevant in the context of this manuscript. What is not, however, is the fact that these techniques all use a magnification lens (see Figure 4.1-right) to assist user's in text selection, and that the magnification lens of FS and FSHP have been specifically tailored to assist users in their interaction and communicate the fact that the techniques exploit force-sensing. With FS and FSHP, The top semicircle around the magnifier shows a *mode gauge* whose background is segmented into five colored sections. The gauge represents the amount of force applied by the finger. If the force remains in the light-pressure region of the gauge (colored blue), moving the finger moves a cursor which is snapped to the closest character. Pressing harder (moving the gauge's indicator above the blue region) fixes the initial cursor and starts the manipulation of a second cursor Returning to the light-pressure region fixes the second cursor and enters a clutch mode: the magnifier can be moved around without changing the cursors' positions. Pressing

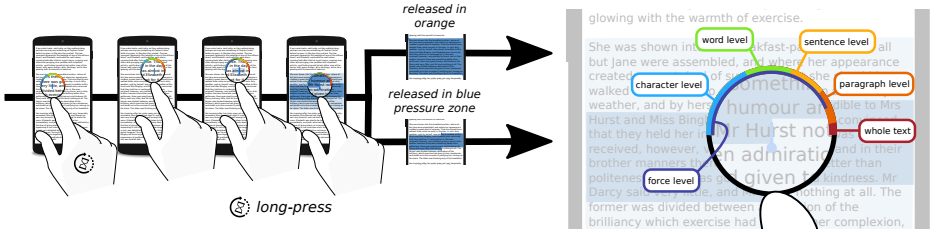

Figure 4.1: Left: Example of text selection using ForceSelect. The user performs a long-press that displays the callout magnifier. Keeping the force in the character level, the user adjusts its position by moving her finger. She then presses harder to start the manipulation of the second cursor. If she releases her finger while the force is at the sentence level of the "mode gauge", she will select the whole paragraph. If she releases her finger while the force is at the character level, she will only select what is between the two cursors; Right: Close-up of the "mode gauge". Also, notice in the background two types of text highlighting: dark highlighting covers between both handles and light highlighting acts as a feedforward of which portion of text will be selected if the user released her finger (here the whole paragraph)

harder again captures the closest cursor for manipulation, until the clutch mode is entered again.

4.2.2 *Evaluating the discoverability of these techniques*

In the research paper summarizing our work on Force-based text selection on touchscreen [HDR6], we conducted two studies — one that examined performance on a variety of selection tasks, and one that examined the discoverability of the novel techniques. In this section, I summarize the latter.

We recruited daily users of touchscreen as participants, none that had used a force-sensitive screen before. We tested both ForceSelect techniques (FS and FSHP), as well as the apple system baseline. For each technique, participants were first presented with a "playground" application, so they could explore the text-selection technique. Participants were informed that the touchscreen had force sensing capabilities, that tapping on the screen would reset the selection, and that a long press would trigger the selection mode. However, no information was given regarding how the techniques worked or how the force applied to the touchscreen is mapped the range of text selected. Participants were invited to notify the experimenter once they felt comfortable with the technique after what, the participant was asked to complete 2

blocks of various text selections. After that, the experiment moved on to the next technique. The experiment concluded with an open interview.

For FS, all participants discovered the main force-based principle of the technique, even though they did not all understood the specific mechanics of the technique afterwards. Notably, 4 of them did not understand that long pressing back on the screen would refine the selection of one handle only, (i.e. physical clutch), but they all suggested in the interview, possibly presumptuously, that they would have found this functionality eventually. For FSHP, all participants but one discovered the force-based principle of the technique, but three did not discover the physical clutching mechanism (i.e. lifting off the finger and long-pressing back on the screen); two others discovered the physical clutching mechanism but not the high-pressure clutch (i.e. keeping the finger in contact and pressing to the red region to unlock the gauge). For the iOS baseline, however, results were mixed. While all participants simulated a button press while in contact with the screen, which would highlight the word underneath the finger, two did it by accident and were not able to reproduce it without help from the experimenter (that reminded that the touchscreen had force-sensing capabilities), and one had issues reproducing it consistently. Only two participants discovered that an additional force-press while in contact would select at the word level (one by frustration and one while demonstrating interaction to the experimenter), and none that a triple force-press would adapt to the word level.

The primary contribution of the ForceSelect project lied in the novel force-based technique as well as the continuous feedback that the gauge offered to the user to understand how the technique worked. We compared the novel ForceSelect techniques to the iOS baseline as it was the state-of-the-art at that time. It is important to note, however, that the ForceSelect techniques differed from the baseline on two main aspects. First, with the ForceSelect techniques, text-selection granularity was continuously mapped to force input, while it was discretely mapped to it for the iOS Baseline. Second, the ForceSelect techniques had a visual feedback about force-sensing whereas the iOS baseline had note. In our study, participants understood more easily how force was mapped to text-selection granularity than with the iOS baseline. However, this study was conducted with 6 participants only. Moreover, the above-mentioned difference in terms of how force input is mapped to text-selection granularity is a confounding factor on whether the principles of the techniques were better communicated thanks to the gauge, or if the techniques were simply easier to apprehend. Regardless, I believe that once again, the iOS default technique is a typical example of minimalist design where interactive principles could be communicated to the user but are not without

any real reasons. The gauge that we decided to display for the ForceSelect techniques occupies barely any additional screen space and a similar feedback could be included with virtually any text-selection technique.

4.3 COMMUNICATING IN-PLACE TOUCH-INPUT

In the previous section, I described a work conducted on force-based text selection technique, but force sensing has been used in commercial devices to perform other operations than selecting text. For example, it was used as a way to activate a menu on home screen buttons on the iPhone 6S. Many research projects also explored how force input could be used to improve interaction [6, 37, 55]

Note that force input is only one of the possible parameter of a touch-input. Others, like dwell (long-press) or double-taps are also used to increase the expressiveness of a direct touch input. We refer to these inputs as *in-place touch inputs*, single finger contacts that enrich the expressiveness of the interaction without moving across the surface. Just like force-input, these inputs are usually barely signified to users, and without visual signifiers (i.e. depictions signalling what can be done), these potentially useful inputs remain unknown or underutilized. In the absence of visual signifiers aiding the perception of input opportunities, it is left to the users guesswork which input method is available to them at any given point. For example, although most users can be expected to be familiar with the input of dwelling on a widget, they may fail to deduce when the interaction is meaningful in a given context. Typically, users can dwell on application buttons on home-screens or messages in messaging applications to access shortcut menus. But they can also dwell on their camera feed, gallery button, quoted tweets, Instagram stories and slack reactions to name a few. This creates uncertainty as input is not systematically used or available, leaving a heavy burden on the user to accurately anticipate when and where a given input is available. The absence of visual signifiers places a high burden on users to discover and recall when which input method is available to allow full access to features.

With Eva Mackamul, a PhD student co-supervised with Géry Casiez, we decided to focus specifically on expanding the perception of possible inputs beyond the conventional 'Tap' on UI elements in touch interfaces, and to explore how *in-place touch inputs* could be communicated to the user directly by the interface design.

4.3.1 *Design space for visually signifying in-place touch inputs*

To explore how visual signifiers could communicate specific interaction possibilities, we propose a design space (see Figure 4.2) focused on the following characteristics:

Timing. Describes when and for how long a signifier is displayed by the interface. The main separation considered in this respect is between signifiers that are *permanently* displayed and those *temporarily* displayed (concurrently to interaction). We consider signifiers to be permanent if displayed before, during, and after the interaction; and temporary if displayed during and immediately after the interaction. A finer consideration between different levels of temporary visibility exceeds the scope of this study but may be worth considering in future work.

Proximity. We consider proximity to the point of interaction, which can influence the perception of a signifier. It can highlight the connection of the signifier to the widget in question by being close as well as increase the likelihood of the signifier being noticed by being displayed within the area of the users focus. On the contrary, close proximity can also lead to the signifier being occluded by the touch interaction. We propose three levels of proximity. *On* the widget: the visual signifier is superimposed on the widget in question, in our case the button. *Vicinity* of the widget: the signifier is displayed in the area immediately surrounding the widget, but not the widget itself. *Away* from the widget: the signifier is visible elsewhere on the display but not connected to the widget (e.g. the edges of the screen).

Explicitness. Describes how clearly the interaction is communicated to the user and, consequently, how much interpretation, and possibly mental effort, is expected. We propose 5 levels of explicitness: *Not at all* explicit (signifiers have no link to the interaction intended to be signified.), *Mirror (established) digital* signifiers (commonly understood digital signifiers, e.g. progress bars), *Mirror physical* affordances (e.g. controls designed to resemble physical buttons or switches), *Descriptive* (show how the system is considering the input but without instruction, e.g. countdowns that may show that the system is recording the progression of time) and *Explicit and Instructive* (unambiguously communicate interaction possibilities to the user and instruct them on how to use them, typically by using text).

We used an approach inspired by the research through design methodology [125] in which we produce various designs to explore and reflect on the characteristics of the design space. More specifically, we created signifier designs for the three selected interaction methods (Double Tap, Dwell, Force Press) combining different levels of explicitness, timing and proximity. The

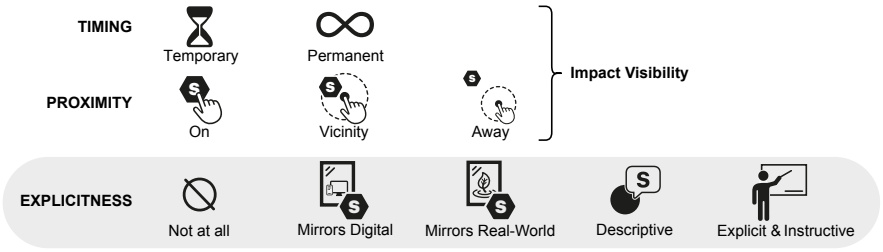

Figure 4.2: Visual representation of the signifier design space featuring the characteristics Explicitness, Timing and Proximity. Each characteristic then includes the levels considered for it which are represented through text as well as small graphical representations. Where characteristics are considered to be influencing the visibility of a signifier is indicated through the curly bracket

created Designs are instances of a signifier, featuring characteristics explored in the design space. To limit confounding factors, we used buttons, which are widely used and can be expected to be familiar to most users. The design space was filled semi-systematically (see corresponding publication for details [HDR9]) and we created a broad group of designs (36) which are presented in Figure 4.3, covering the design space broadly and fulfilling the majority of possibilities. The only designs which have been previously implemented or tested, are 01 (default button design from most touch-based OS systems), 24 (found in the mobile application Adobe Fresco Version 4.4.2 where they signify a possible dwell by a small triangle, mirroring the same functionality and signifier in Adobe desktop applications), and 15 and 25 (which are adapted from existing research on signifying touch-based interaction [10, 41]).

4.3.2 *Investigating the communicability of proposed designs*

We conducted two studies in order to investigate how the proposed designs would actually signify and communicate efficiently the corresponding in-place touch inputs.

We first conducted a remote online survey to establish which input participants perceive as possible with a given Design. The survey relies on videos that present each Design implemented in a button, placed on a grid similar to a smartphone home screen, and being 'tapped' by a finger. For each Design, participants are invited to select from a list all inputs they consider to be pos-

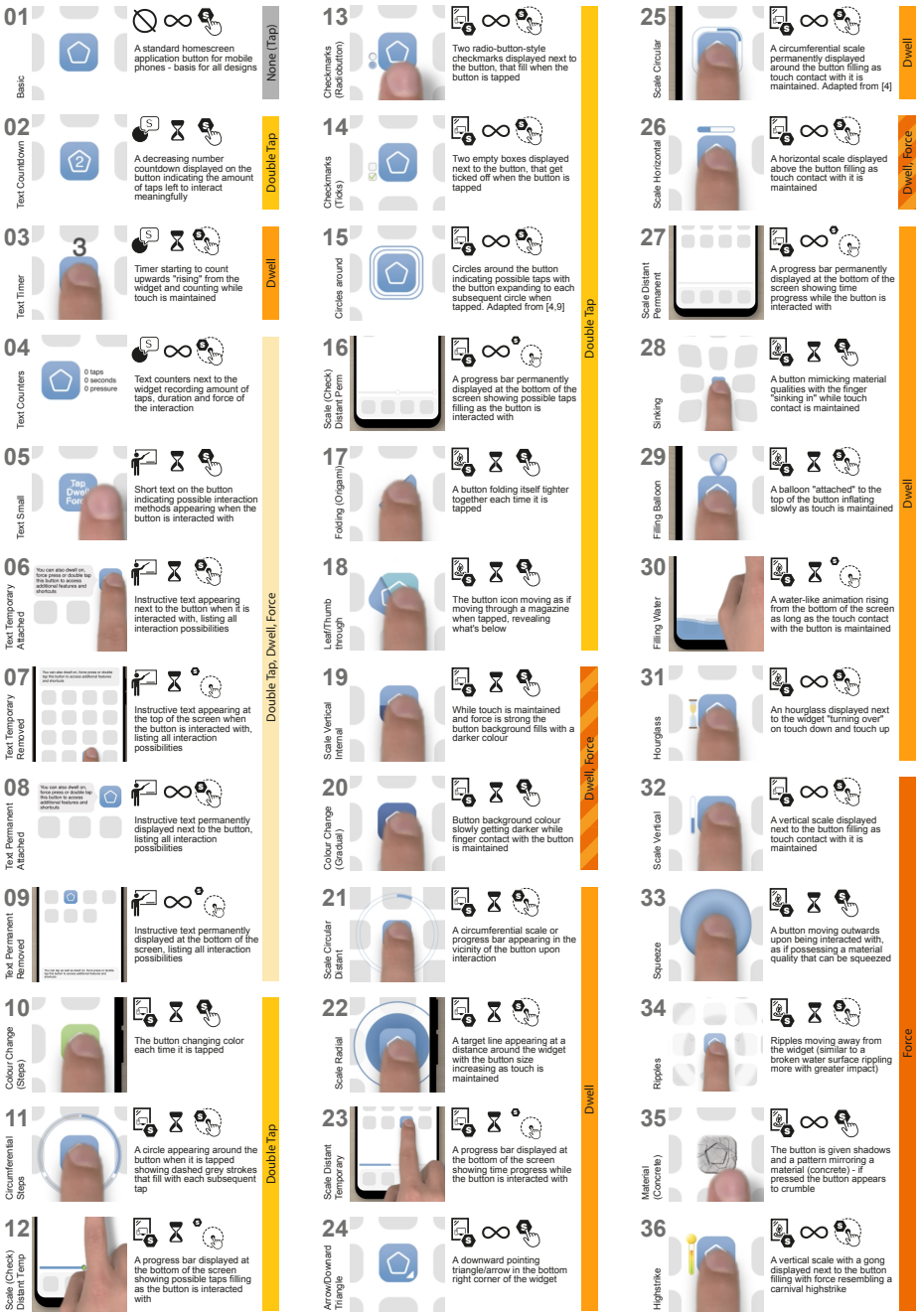

Figure 4.3: Table of designs filling the design space. Each design is displayed alongside the design space icons that apply to it and a text description. The vertical strip on the right displays the assumed signified interaction

sible with the button. 32 participants recruited via direct message and aged 22 to 58 years completed this online survey.

We then conducted an in laboratory interactive user study. Participants had to interact with the experimental software that consisted of a mobile application featuring a home screen grid populated with gray squares with one Design featured in the middle, mirroring the online study. At the top, a text is displayed which states "You can start" until the participant starts interacting with the button. If the input performed was the specific input supported by the design, the experiment moved to the next design. When any other input was performed on the button, it was counted as an error. The experiment automatically moved to the next design after 6 errors. Participants were instructed to progress through all Designs while making the least errors possible, and encouraged to comment on their thought process throughout. They were told that one of the inputs they just indicated familiarity with would achieve this and that it may vary between Designs, but were given no further guidance on how to achieve this beyond the visual signifiers in the Designs themselves. The experiment was completed by 24 participants aged 20 to 59.

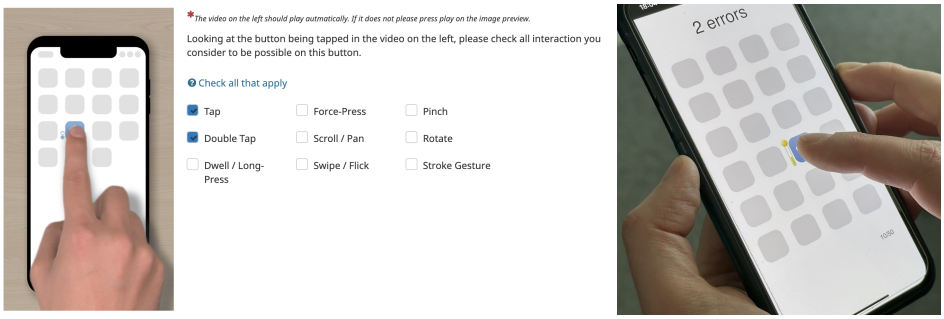

Figure 4.4: Experimental interfaces used during our experiments. Left: the online web-based interface used during the survey; Right: a participant interacting with the software platform used for the laboratory study

Results of both studies suggested that in general, visual signifiers do increase the perception of available inputs. Basically, when examining which inputs were assumed to be available in study 1, the vast majority of designs created for 'Double Tap', 'Dwell' and 'Force Press' resulted in participants selecting more possible inputs than the baseline '01 Basic'. When considering which inputs participants attempted first in study 2, out of the 1200 total interactions, the first input performed was 'Dwell' (400 times), 'Tap' (393 times),

'Double Tap' (173 times), 'Force Press' (155 times) and 'Swipe' (79 times), showing that participants frequently attempted different inputs than 'Tap'.

Regarding whether the Designs convey the intended inputs, looking at study 2 and using an arbitrary threshold of 75% to consider that a Design communicates the assumed input, which means that above 75% of participants considered this input as possible to perform with the corresponding Design, we found 22 Designs that communicate at least one of the assumed inputs. Looking in detail for each input, we found 9/16 Designs for 'Double Tap', 16/20 for 'Dwell', 9/14 for 'Force Press'. Note that looking at these assumed inputs is not the ultimate aim of our study as, in the end, what matters is what inputs participants did report suggesting which affordances were perceived. Considering the same 75% threshold and only when participants attempted the correct input for each Design, only 2 Designs. If examining only the first interaction performed by each user, only 8 Designs pass the 75% threshold, but it rapidly increases to 21 Designs when extending this to the first two interaction attempts of each user.

Finally, of particular interest, participants generally preferred the inclusion of signifiers over a baseline without them. While most Designs received varying preference ratings, they were widely preferred over a complete absence of signifiers.

While preliminary, these studies were still informative on various aspects. First, users prefer added visual signifiers, at least in the context of such experiment. This may go against current trend in graphical design that tends to rely on minimalists visuals, but at least, it suggests that added visual signifiers may not repel users. Moreover, despite the lack of significant difference in performance for some Designs and inputs in study 2, all but 2 Designs were perceived as less mentally demanding. This suggests that these Designs can still improve the subjective experience when these inputs should be communicated. This is further supported by participants indicating a significant preference for the Designs with added visual signifiers over the baseline except the aforementioned 2 Designs. Overall, this work suggests the feasibility and usefulness of visually communicating input possibilities for improving user experience with a widget.

4.4 COMMUNICATING SINGLE-HAND MICROGESTURES

Single-hand microgestures are a subset of gestures that can be performed by subtly moving one or more fingers from one hand without needing to move the arm nor the wrist. They present various advantages, such as versatility in

the context of use (eye-free, hand-free or not), ease of execution (avoidance of the "Gorilla arm" effect [56]), and a better social acceptance than larger scale gesture-based interactions thanks to their subtlety [31, 32, 75].

The research area of microgestural interaction has focused on elaborating relevant sets of microgestures through elicitation studies [31, 75, 78], evaluating the end-users' performances [23, 60, 109] or designing and implementating systems and devices that sense and recognize such microgestures [101, 119]. However, as for any other gesture-based interactions, using microgestures as inputs requires to be aware of which gestures are available and which commands they may activate. And similarly to other gesture-based interactions, very little work had been dedicated to this question. As a result, the diversity of visual representations used for microgestures can be confusing. Practitioners encounter similar issues as they tend to use crib-sheets along with video tutorials to depict gesture-based interaction [9, 80].

Vincent Lambert, a PhD student at Université Grenoble-Alpes that I remotely co-supervise with Prof. Laurence Nigay and Dr Alix Goguey, decided to investigate how microgestures could intelligibly be *presented*. More precisely, we compared distinct visual characteristics which constitute the basis of microgesture representations for both static (e.g. in a crib-sheet) and dynamic image (e.g. in video tutorials). For that, we produced 21 different "families" of representations, each family representing 4 microgestures with similar visual designs in static and in dynamic (for a total of 8 visual designs per family). These families have been assessed in an online experiment, and we kept the 4 bests to test them in context using an Augmented Reality (AR) headset with the ambition of refining our quantitative results by qualitative information.

4.4.1 *Producing visual representations of microgestures*

In order to produce a comprehensive set of microgestures families, we reviewed previous work on microgesture in order to analyze how researchers had represented microgestures (e.g. [29, 30, 59, 78, 109]), as well as reviewed more general work on how to visually represent gestures [8, 88]. In particular, this work builds on top of the work of Axel Antoine [HDR1], a PhD student I co-supervised with Gery Casiez who worked on the design and production of illustrative figures of interaction [5]. During his PhD, Axel proposed a taxonomy of such illustrations in which he classified techniques used by the HCI community for visually representing interaction scenarios, notably motion, interactors and dynamic [8].

Expanding Axel's taxonomy and combining it with other taxonomies and existing visual representations (e.g. [21, 88]), we focused on 4 categories that can be used to represent different parts of a microgesture: the **trajectory**, its **direction**, the **actuator** and **receiver**, and **emphasis tools**.

Trajectory is mainly represented by different kinds of Lines. Curves and broken lines are used mainly to represent symobilic microgestures, e.g. circle or square symbols, and also movements of the tip of a finger. Another strategy to depict the trajectory in static images, is the use of a *stroboscopic* effect to display multiple positions through time [39, 88].

Direction is mainly indicated by Arrows. Associated with Lines or more rarely used as a pattern to shape a chevron [121], arrows indicate the direction of a microgesture. Motion lines[3] can also be used.

Actuator and receiver are often shown through Contact shapes. These shapes can vary along a continuum from shapes that exactly match the outline of the finger or hand part described, to generic geometric shapes above or next to the part of the finger or hand described.

Emphasis tools are Effects that modify the aspect of a visual representation. Examples of aspects of a visual representation include the line type, the filling pattern, the color, and the level of transparency.

Similarly to the work described section 4.3, we had a design space too large to be systematically explored. We therefore produced a set of 21 families trying to pave the entire design space with sufficient diversity. These families are illustrated Figure 4.5.

4.4.2 *Investigating communicability of the proposed designs*

The methodology was similar to the methodology used in the previous section. We first conducted an online study, followed by a laboratory study. We deployed an online form presenting the 21 families, in which respondents were asked to score how efficiently each graphical representation communicates the corresponding microgesture. We recruited 45 unpaid volunteers whom age ranged from 13 to 76 years old. Participants were asked to evaluate the ability of each representation to efficiently communicate a given microgesture on a 6-point Likert item (Very bad, Bad, Quite bad, Quite good, Good, Very good).

This first study evaluated the perceived efficiencies to graphically represent microgestures on a computer screen with our 21 families, and provided insights on which would work best. However, microgestures are frequently

3 See *Motion lines* [Wikipedia]

Figure 4.5: Static representations of the 21 families for the tap, hold, swipe and flex microgestures

deemed promising for Augmented Reality [45,101,119,126]. Therefore, in order to refine the results of our first study, we conducted a qualitative controlled laboratory study using an Hololens 2 Augmented Reality headset. However, in order to keep the study relatively short to minimize the risks of discompfort due to AR [76] we decided to only test a subset of 4 families that had both the best scores and distinct designs among the 8 bests families of the online experiment. For that second study, we recruited 16 unpaid volunteers whom age ranged from 15 to 49 years old.

Given the number of representations tested, I will not report specific results discussing each of these representations. Instead, I remain at a relatively "high" level and refer the reader to the corresponding research paper for details [HDR8].

Regarding *what* information of a microgesture should be represented, the results of both studies suggest possibly unsurprisingly that the more information is provided, the better. In general, representations that showed more details (trajectory, direction, actuator and receiver) were preferred and more efficient. That being said, results also suggest that what seems to matter the most is to present the actuator finger as we observed that every representation that did not obtain negative results. Then, the most useful features seemed to be the trajectory and/or the direction that should be shown when possible. For cases where displayed information has to be limited, it is preferable to clearly show the trajectory of a microgesture. Indeed, explicitly representing the direction can help to disambiguate out which finger is the actuator and which is the receiver

Regarding *how* information should be presented, arrows or partially transparent fingers should be preferred to show directions. Indeed, our results suggest that using arrows with a line is a good practice which allows great flexibility in design, as all 4 families using arrows with a line were appreciated by our participants. On the other hand, designers should be careful with visual associations, as how the actuator and receiver are represented can yield negative results if the trajectory is not explicitly represented. As an example, one of the family using red and blue circles with symbols was relatively low scored by our participants in the online experiment, while a similar one but using empty/filled matching shapes was one of the preferred. Finally, our results suggest that animations were preferred by respondents in the online study, but did not result in a better understanding of the microgestures in the second one. Therefore, animations can be used to increase satisfaction, but are not needed to improve recognition of the available microgestures.

4.5 WHY INTERACTION POSSIBILITIES SHOULD BE SHOWN

In this chapter, I reviewed work that was conducted on the communicability and discoverability of input techniques. Discoverability and communicability of interaction, while being deemed as important in virtually every design guidelines and textbook for interactive system design, is extremely frequently overlooked by the research community.

These work have convinced me that this is a difficult problem that should be a group effort from the community, and not remain buried in discussion sections because users will eventually "figure it out". Two main factors actually make this problem difficult. First, there is no clear terminology on what is discoverability making it problematic to approach as a scientific concept [3, 57]. Eva Mackamul tried to improve on this aspect in her work.

Then, there is no accepted evaluation method or protocol for testing discoverability. Moreover, it is extremely complicated as one must found participants that are not aware of what is being tested, evaluate current users' knowledge of the tested interaction methods, and implement a system that is convincing and robust enough to conduct a fictional task to test it. Therefore, it is extremely complicated to test discoverability of interaction, and easier to test how it can be efficiently communicated to the user, which is what Eva Mackamul and Vincent Lambert tested in their work.

Regardless, what emerged is that in order to efficiently communicate interaction possibilities, these interaction possibilities must be **explicitly shown** to the user. Once again, it means displaying more information to the user, but information that is important and needed to efficiently use a system, in order to reduce the distance between where the information on how to interact with a system is needed and where it is currently provided (mostly on websites and tutorial videos).

5 REFLECTION AND CONCLUSION

5.1 MY TWO CENTS ON INTERACTION AND INTERFACE DESIGN

5.1.1 *Interactive systems are expecting too much from the user*

The initial intentions behind the introduction of Graphical User Interfaces were commendable. They heavily simplified the interaction with a computer system and made it more accessible. My impression is, however, that their success resulted in more and more software developed, each relying on more and more features, a growth that original GUI designs could simply not cope with.

As a result, commercial interactive systems and their GUIs do, in my opinion, expect too much from the user. Users need to know what they can do with the system and how they could do it. This requires users to know beforehand what is achievable, to go online to figure out the interaction possibilities the systems fail to communicate, and they expect users to share this information with each others (via tutorials, observations, trainings, online tweets and blog posts, etc.). But they also expect users to *remember* all this information, and to recall it for later use. The reasons for this over-expectation are possibly manifold.

A possible one is that these systems usually all follow the same design process. They are first introduced with a relatively small number of application and features, few enough for being able to display all of it and not rely on hidden information that need users to magically discover it or require to remember it. They then grow to the point where so many features are included that a reorganization is necessary, often taking the form of building (deep) hierarchies and hiding information. However, once the foundations of an operating system have been set, usually to support only a few features, it is simply too costly to build again. This phenomenon could be witnessed with desktop computers, again with the modern touch-based devices such as iPhone and Android phones, and is likely to happen again with future computing systems if we do not provide the product manufacturers with the theoretical and conceptual knowledge to anticipate this.

Another possible reason might be more structural and lie in the fact that interface design nowadays is mostly driven by commercial product designers with various design objectives that might negatively impact usability at times. For instance, they may put too much emphasis on aesthetic and apparent simplicity of a product because they are mostly interested in product sales, which need users to rapidly feel like they can use the system, even if that means that they don't know how to do many other things as they may not realize it anyway. A typical example of the opposition between system design and product design might be found when in 2012 Tim Cook decided to restructure Apple, dismiss Scott Forstall [43], promote Jonathan Ive as head of Human Interface division at Apple [38], resulting in the release of iOS 7 and introduction of flat design in the interface. From this time, the approach of iOS was articulated around the experience with the product, and not its usability, possibly for better commercial success.

Regardless, as an academic researcher, I don't believe this is what we should be interested in. I, personally, am (obviously) not interested in product sales. I am not interested in providing guidelines and immediate actionable items that designers can reproduce afterwards either. Design trends will always be driven by big actors anyway (material design, UI guidelines). I am interested in exploring hypothesis, testing hypothesis, and see if alternatives are possible. And I believe that alternative interfaces and system designs exist that would be usable while expecting much less from the users. The work I presented in this document is an example of such design applied to GUIs for command selection.

5.1.2 *Interaction design is trapped in a local optimum because of a replication crisis*

At the beginning of my research career I have designed interaction techniques that were inspired by previous work and reimplemented certain design characteristics that "made sense" without further questioning. The stereotypical case is the implementation of the rehearsal principle when working on the Augmented Letters [HDR13], FastTap [54] and FastTapAdjust [HDR5].

I fear that I may not be the only HCI researcher that tends to re-implement existing design characteristics without wondering whether alternative designs could be better. Even worse (in my opinion), I had the impression that reviewers of the ZeroDelay Marking Menu papers [HDR7] had difficulties to acknowledge that there might be value in an alternative design, and claimed that if this had been replicated for decades, then this must be because this is how

this should be done. In that respect, I wonder whether the interaction design might not about to get trapped in a local optimum because we tend to replicate what has been done, and claimed to work, without questioning further. We tend to replicate things and guidelines without questioning whether they are actually correct and worth replicating, and what the sources of the problems that led to these recommendations are.

Interestingly, there are situations where these replications "clash" and result in replicating one design characteristic over another. A great example of this is the non adaptation of transfer functions for touch-based and VR input techniques, because direct manipulation using a 1:1 mapping is the norm. Transfer functions, that are highly efficient for indirect desktop interaction [27], could be employed on these platforms as well, and yet, this is extremely rare, possibly because designers might consider that a 1:1 mapping is a must. But why? Is this necessary? Would it bother the user if it wasn't the case? And if yes, wouldn't it be possible to design a transfer function that would improve interaction performance without being noticeable from users? I am glad and happy that a few HCI researchers like Dan Vogel, Edward Lank and their colleagues since decided to explore this alternative and demonstrated the benefit of such transfer functions [11, 118].

Finally, users relationship with interactive systems evolve over time. What might have been useful at one time may not be so now, because technologies have evolved, new use cases have emerged, users have changed in terms of skills and habits, and more generally because it has changed at a societal level. Once again, touch-based interfaces are a good example of this: a direct translation of the MS Windows user interface to touch-based devices failed because it was not adapted to the technology (small screens, direct input) and use cases (mobility); since then, different interfaces have been proposed and this is now the norm in our society.

5.1.3 *Learning and discoverability of interaction should be a standard for interaction design publications*

Research introducing novel interaction techniques usually validate these novel interaction techniques by picking one or two metrics and providing evidence that in a certain context, the novel interaction technique improves over the state-of-the-art regarding these metrics. These metrics most of the time focus on the peak performance that a user can reach typically in terms of execution time or precision. As an example, several work presenting variations of Marking Menus focused their validation on the time required to perform the ges-

ture to select a command [72,122,123], ignoring a possible reaction time, visual search, or the whole learning process. We have seen in recent years more and more studies interested beyond performance, and observing learning rates thanks to block based experimental designs (e.g. [13, 15, 54]).

Learnability is not a standard yet when it comes to validating a novel interaction technique, but it is slowly getting there. Discoverability, on the other hand, remains largely ignored. At best, the question of interaction discoverability is hidden in a discussion section and pushed back as a possible future work. Sometimes, it is claimed that users would eventually figure it out (without backing it up with any reference), and users themselves seem to be overconfident about the fact that they would eventually figure it out (possibly because one cannot be aware of not having discovered something you do not know the existence of).

I believe that the discoverability remains overlooked by the HCI community, while it should be a standard of validation. That being said, conducting discoverability studies is more difficult that it may seem, and I suspect that more than one research project actually wanted to run such study but eventually gave up on it. But this is why this should be a group effort, at the community level, as this is required to better understand what makes an interaction technique more easily discoverable, and design future systems that will better foster the discovery of their interaction principles. Otherwise, we will remain in an elitist and survivalist model where mastering a system requires searching and accessing online resources, or being helped by your social network, thus creating some sort of social interaction determinism when it comes to interacting with computing systems.

5.1.4 *On the difficulty of conducting long term studies*

Possibly the main limitation of the work I present in this manuscript might be that it is backed up almost exclusively by results of laboratory studies. Unfortunately, I did not manage to run a long term study that would evaluate one of the proposed interaction technique, or evaluate the long term efficiency of recognition ready command selection interfaces. Truth is that I have tried on few occasions.

Typically, the HotkeySkillometer presented in section 2.3 had, from the beginning, been implemented as a working background application running under macOS. It combined the use of global action listeners to be notified when a command from the menubar is selected by the user, which happens when the command is selected thanks to its keyboard shortcut or by clicking the

menu item in the menubar, with the macOS accessibility API to try to detect when the user clicks on another GUI element, typically a toolbar button. The problem was that the vast majority of software, including Apple native ones, implement their own GUI components that made the HotkeySkillometer fail to detect clicks on some toolbar buttons or other GUI components in many occasions, thus missing many command activations. Moreover, even when managing to detect these command activations, these GUI components rarely encapsulate the corresponding keyboard shortcut, when existing. Sometimes it can be accessed via the accessibility API because it displays the shortcut in its tooltip, but it is rarely the case anymore. Therefore, in order to display the command in the HotkeySkillometer, it is still necessary to find a way to tie a toolbar button to its corresponding menu item in order to retrieve its keyboard shortcuts. The laboratory study conducted to evaluate the benefits of the HotkeySkillometer was run over Apple Keynote, for which I manually coded all these connections for the commands we asked participants to use. But this would have been extremely long to do for all commands of the application, and virtually impossible for several applications. I still deployed a Skillometer on some users computers to try to gather some feedback about it in real use, but it was rapidly confirmed that it was pointless without supporting toolbar buttons.

I also made few attempts to implement the ExposeHK interface presented in section 3.2, so it could be studied on long term. The main one was done in 2015 in collaboration with Jonathan Aceituno who had been a PhD student at Inria in Lille. The idea was to integrate ExposeHK for toolbars directly in macOS application by injecting code at runtime using method swizzling (similar to what Scotty [44] and Fscript [90] do). While making good progress, Jonathan's implementation was regularly impacted by macOS updates, that were getting more and more frequent, and required to systematically update the code. We gave up after several of these updates. I must confess that we may have not tried enough, but continuing this effort was simply too difficult without manpower (and money) dedicated to it.

5.1.5 *GUI programming and OS APIs should consider how the user interact with the interface*

As just described above, as much as I would love to run long term studies, implementing novel interaction techniques to be able to run such studies often require accessing information that systems were simply not designed to provide. I suspect that most of the academic HCI researchers have faced the same

problem. Nowadays, implementing interaction techniques like ExposeHK or Skillometers systematically require hacks and heuristics to infer some aspects of system design and users' intentions. Ideally, future OS and GUI APIs will facilitate the implementation of such techniques, so we can evaluate them on the long run.

Reflecting on this question is one of the objective of the Loki research group I currently belong to. Thibault Raffaillac and Stephane Huot have reflected on what researchers need when implementing novel interaction techniques [103], and we have also started to think how GUIs and their APIs could rely on different paradigms [26, 102] Notably, we recently published a simple demonstration at the AFIHM French HCI conference IHM 2023 where we present Signifidgets [26], a reflection on how programming APIs could (and should?) allow programmers to add different possible user inputs to a component, that would modify its behavior and appearance to signify and react to them, but also be able to introspect to know which inputs it supports.

5.2 PERSPECTIVE AND FUTURE WORK

5.2.1 *Continue to work on recognition ready interfaces*

The work presented in this manuscript was advocating for User Interfaces that show users what they can do with the system, either to require less recall or make it easier to discover interaction possibilities.

While I will probably explore less the context of the intermodal transition, I plan to continue to explore how interfaces can visually convey interaction. First, I plan to conduct additional studies where an interface instantly pops-up on demand when users want to interact, and test whether users are actually bothered by this or if they manage to rapidly bypass it. Indeed, it is frequent to design interfaces where the UI fades-in as a way to avoid a big visual pop-up, but is the pop-up in itself really a problem? Second, I also plan to continue to work on interaction with interactive components that are either hidden by default, or simply never displayed. As an example, I will investigate how users apprehend Swhidgets [98] and other GUI components hidden by default, and in particular try to better understand how they build a mental model of an interface that has off-screen components, and if this mental model can translate and facilitate the discovery of such off-screen components in another interface.

All in all, I believe that interaction possibilities must be shown to the user, that APIs should facilitate the implementation of GUIs that show these inter-

action possibilities, and that we should evaluate how much an interface can show to the users before it negatively impacts usability.

5.2.2 *Studying how we communicate about interaction*

Overall, my research interests lie in the core of interaction and how interaction is *"communicated"* at diverse levels.

First, communicated *from the user to the system*. Over my young career, I have worked on various *input* techniques, some detailed in this manuscript [HDR2, HDR3, HDR5, HDR10, HDR13] but others not [6, 12, 54, 81], that facilitate how users can specify their intention to the system.

Then, communicated *from the system to the user*. In the sections 3.3 and 4.1, I have presented work that specifically investigated how GUIs could communicate interaction possibilities differently to the user. While researchers have vastly explored data visualization and overall how information could be presented to users, less work has been targeted at interaction possibilities.

Finally, communicated *between users*. I currently supervise the PhD of Raphaël Perraud who investigates how software tutorials could be adaptable to specific platforms, user interfaces and user preferences. I also plan to design and evaluate interfaces that can be used by users to share computing skills between them, both at a small-scale and community of practice level.

The framework above, that encompass what an interaction technique is, how we can implement them, and how it can be communicated, has always been the core of HCI research as well as encompassed my research.

5.2.3 *Studying interaction for visual communication*

Finally, in a very egocentric fashion, I have always been interested in how we, HCI researchers and more generally scientists, visually communicate about our work. In that respect, I recently co-supervised two PhD students on related topics. First, I co-supervised the PhD of Axel Antoine on knowledge and tools to produce illustrative figures about interaction. Notably, Axel did elaborate a taxonomy of visual strategies used to communicate interaction on static supports [8] and designed Esquisse [7], a software tool that combines 3D staging with a non-photerealistic rendering algorithm to produce vector based trace figures. Then, I co-supervised the PhD of Damien Masson on making scientific documents polymorphic, so readers can change their form to offer more adapted, dynamic and interactive representations of information to accommodate readers skills and needs. Damien notably applied this

concept to data visualization [82, 86], statistics [83], and developed a method to integrate interactive figures within any static document [84, 85]. I remain strongly interested in this research area, and plan notably to expand my research to scientific and pedagogical presentation supports (such as keynotes or MOOCs) and work on knowledge and software tools to help their authoring, re-utilization, and while giving the presentation.

5.3 FINAL WORDS

Research in HCI is in my humble opinion a unique job that offers the opportunity to work on the two things that drive me the most: technology and people. The HCI research domain is fantastic on that aspect, as I have the impression that we can literally investigate any research question we want, and from the perspective we want. Attending once the flagship conference ACM CHI is enough to realize the broad spectrum of contributions that the HCI community has to offer every year. And while I have been mostly contributing to the interaction techniques and systems sub-communities, I love that as an HCI researcher, I have the feeling that I could decide to strive my research on more design or methodological aspects.

Finally, I have met countless interesting and benevolent people in this community, which resulted in various collaborations, even long distance ones, that have always been positive and enriching. I hope that when I collaborate with other researchers or supervise students, I manage to share at least a fraction of the excitement and motivation these collaborations are giving me. Research is sharing.

BIBLIOGRAPHY

[1] Tricycle vs. bicycle. https://www.dougengelbart.org/content/view/224/#6a. Accessed: 2023-07-02.

[2] Try histomages! http://tan.lille.inria.fr/tan.html#histomages. Accessed: 2023-07-02.

[3] Bruce G. Allen and Elizabeth Buie. What's in a word? the semantics of usability. *Interactions*, 9(2):17–21, mar 2002.

[4] Rui Alves and Nuno Jardim Nunes. Ceiling and threshold of paas tools: the role of learnability in tool adoption. In *Human-Centered and Error-Resilient Systems Development*, pages 335–347. Springer, 2016.

[5] Axel Antoine. *Études des stratégies et conception d'outils pour la production de supports illustratifs d'interaction*. PhD thesis, Université de Lille, Jan 2021.

[6] Axel Antoine, Sylvain Malacria, and Géry Casiez. Forceedge: Controlling autoscroll on both desktop and mobile computers using the force. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, page 3281–3292, New York, NY, USA, 2017. Association for Computing Machinery.

[7] Axel Antoine, Sylvain Malacria, Nicolai Marquardt, and Géry Casiez. Esquisse: Using 3d models staging to facilitate the creation of vector-based trace figures. In David Lamas, Fernando Loizides, Lennart Nacke, Helen Petrie, Marco Winckler, and Panayiotis Zaphiris, editors, *Human-Computer Interaction – INTERACT 2019*, pages 496–516, Cham, 2019. Springer International Publishing.

[8] Axel Antoine, Sylvain Malacria, Nicolai Marquardt, and Géry Casiez. Interaction illustration taxonomy: Classification of styles and techniques for visually representing interaction scenarios. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (2021)*, CHI '21, 2021.

[9] Apple. Trackpad gestures for iPad, 2022.

[10] Liv Arleth, Emilie Lind Damkjær, and Hendrik Knoche. "but wait, there's more!" a deeper look into temporally placing touch gesture signifiers. In

Anthony Brooks and Eva Irene Brooks, editors, *Interactivity, Game Creation, Design, Learning, and Innovation*, pages 290–308, Cham, 2020. Springer International Publishing.

[11] Jeff Avery, Mark Choi, Daniel Vogel, and Edward Lank. Pinch-to-zoom-plus: An enhanced pinch-to-zoom that reduces clutching and panning. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, page 595–604, New York, NY, USA, 2014. Association for Computing Machinery.

[12] Jeff Avery, Sylvain Malacria, Mathieu Nancel, Géry Casiez, and Edward Lank. Introducing transient gestures to improve pan and zoom on touch surfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–8, New York, NY, USA, 2018. Association for Computing Machinery.

[13] Gilles Bailly, Eric Lecolinet, and Yves Guiard. Finger-count & radial-stroke shortcuts: 2 techniques for augmenting linear menus on multi-touch surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, page 591–594, New York, NY, USA, 2010. Association for Computing Machinery.

[14] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. Wave menus: improving the novice mode of hierarchical marking menus. In *IFIP Conference on Human-Computer Interaction*, pages 475–488. Springer, 2007.

[15] Gilles Bailly, Eric Lecolinet, and Laurence Nigay. Flower menus: A new type of marking menu with large menu breadth, within groups and efficient expert mode memorization. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '08, page 15–22, New York, NY, USA, 2008. Association for Computing Machinery.

[16] Gilles Bailly and Sylvain Malacria. *Command Selection*, pages 1–35. Springer International Publishing, Cham, 2020.

[17] Gilles Bailly, Antti Oulasvirta, Duncan P. Brumby, and Andrew Howes. Model of visual search and selection time in linear menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, page 3865–3874, New York, NY, USA, 2014. Association for Computing Machinery.

[18] Nikola Banovic, Fanny Chevalier, Tovi Grossman, and George Fitzmaurice. Triggering triggers and burying barriers to customizing software. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*,

CHI '12, page 2717–2726, New York, NY, USA, 2012. Association for Computing Machinery.

[19] Olivier Bau and Wendy E. Mackay. 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, page 37–46, New York, NY, USA, 2008. Association for Computing Machinery.

[20] Mathieu Berthellemy, Elodie Cayez, Marwan Ajem, Gilles Bailly, Sylvain Malacria, and Eric Lecolinet. Spotpad, locipad, chordpad and inoutpad: Investigating gesture-based input on touchpad. In *Proceedings of the 27th Conference on l'Interaction Homme-Machine*, IHM '15, New York, NY, USA, 2015. Association for Computing Machinery.

[21] Jacques Bertin. *Semiology of Graphics: Diagrams, Networks, Maps*. Esri Press, 2011.

[22] Alethea L Blackler, Rafael Gomez, Vesna Popovic, and M Helen Thompson. Life Is Too Short to RTFM: How Users Relate to Documentation and Excess Features in Consumer Products. *Interacting with Computers*, 28(1):27–46, January 2016.

[23] Roger Boldu, Alexandru Dancu, Denys Matthies, Pablo Cascon, Shanaka Ransir, and Suranga Nanayakkara. Thumb-In-Motion: Evaluating Thumb-to-Ring Microgestures for Athletic Activity. In *Proceedings of the Symposium on Spatial User Interaction*, pages 150–157. Association for Computing Machinery, October 2018.

[24] Stuart K. Card, Allen Newell, and Thomas P. Moran. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., USA, 1983.

[25] John M. Carroll and Mary Beth Rosson. *Paradox of the Active User*, page 80–111. MIT Press, Cambridge, MA, USA, 1987.

[26] Géry Casiez, Sylvain Malacria, and Eva Mackamul. Signifidgets : What you see is what widget ! In *IHM 2023 - 34e Conférence Internationale Francophone sur l'Interaction Humain-Machine*, Troyes, France, April 2023. AFIHM and Université de Technologie de Troyes.

[27] Géry Casiez and Nicolas Roussel. No more bricolage! methods and tools to characterize, replicate and compare pointing transfer functions. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, page 603–614, New York, NY, USA, 2011. Association for Computing Machinery.

[28] Steven J. Castellucci and I. Scott MacKenzie. Graffiti vs. unistrokes: An empirical comparison. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, page 305–308, New York, NY, USA, 2008. Association for Computing Machinery.

[29] Adrien Chaffangeon, Alix Goguey, and Laurence Nigay. µglyphe: Une notation graphique pour décrire les microgestes. IHM '22, New York, NY, USA, 2022. Association for Computing Machinery.

[30] Adrien Chaffangeon Caillet, Alix Goguey, and Laurence Nigay. µglyph: A microgesture notation. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery.

[31] Edwin Chan, Teddy Seyed, Wolfgang Stuerzlinger, Xing-Dong Yang, and Frank Maurer. User Elicitation on Single-hand Microgestures. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 3403–3414, San Jose California USA, May 2016. ACM.

[32] Liwei Chan, Rong-Hao Liang, Ming-Chang Tsai, Kai-Yin Cheng, Chao-Huai Su, Mike Y. Chen, Wen-Huang Cheng, and Bing-Yu Chen. FingerPad: private and subtle interaction using fingertips. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*, pages 255–260, St. Andrews Scotland, United Kingdom, October 2013. ACM.

[33] Victor Cheung. Improving interaction discoverability in large public interactive displays. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, ITS '14, page 467–472, New York, NY, USA, 2014. Association for Computing Machinery.

[34] Fanny Chevalier, Pierre Dragicevic, and Christophe Hurter. Histomages: Fully synchronized views for image editing. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, page 281–286, New York, NY, USA, 2012. Association for Computing Machinery.

[35] Andy Cockburn, Carl Gutwin, and Saul Greenberg. A predictive model of menu performance. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, page 627–636, New York, NY, USA, 2007. Association for Computing Machinery.

[36] Andy Cockburn, Carl Gutwin, Joey Scarr, and Sylvain Malacria. Supporting novice to expert transitions in user interfaces. *ACM Comput. Surv.*, 47(2), November 2014.

[37] Christian Corsten, Simon Voelker, Andreas Link, and Jan Borchers. Use the force picker, luke: Space-efficient value input on force-sensitive mobile touchscreens. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–12, New York, NY, USA, 2018. Association for Computing Machinery.

[38] Katie Cotton and Steve Dowling. Apple announces changes to increase collaboration across hardware, software and services, October 2012.

[39] James E Cutting. Representing Motion in a Static Image: Constraints and Parallels in Art, Science, and Popular Culture. *Perception*, 31(10):1165–1193, October 2002. Number: 10.

[40] Mary Czerwinski, Eric Horvitz, and Ed Cutrell. Subjective duration assessment: An implicit probe for software usability. In *IHM-HCI 2001 Conference*, volume 2, pages 167–170, September 2001.

[41] Emilie Lind Damkjær, Liv Arleth, and Hendrik Knoche. "i didn't know, you could do that" - affordance signifiers for touch gestures on mobile devices. In Anthony L. Brooks, Eva Brooks, and Cristina Sylla, editors, *Interactivity, Game Creation, Design, Learning, and Innovation*, pages 206–212, Cham, 2019. Springer International Publishing.

[42] Clarisse Sieckenius De Souza, Bonnie A Nardi, Victor Kaptelinin, and Kirsten A Foot. *The semiotic engineering of human-computer interaction*. MIT press, 2005.

[43] Daniel Dilger. Scott forstall's leadership of ios, siri, maps and user interface revoked, October 2012.

[44] James R. Eagan, Michel Beaudouin-Lafon, and Wendy E. Mackay. Cracking the cocoa nut: User interface programming at runtime. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, page 225–234, New York, NY, USA, 2011. Association for Computing Machinery.

[45] Barrett Ens, Aaron Quigley, Hui-Shyong Yeo, Pourang Irani, Thammathip Piumsomboon, and Mark Billinghurst. Counterpoint: Exploring Mixed-Scale Gesture Interaction for AR Applications. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–6, Montreal QC Canada, April 2018. ACM.

[46] Paul Morris Fitts and Michael I Posner. Human performance. 1967.

[47] Jérémie Francone, Gilles Bailly, Eric Lecolinet, Nadine Mandran, and Laurence Nigay. Wavelet menus on handheld devices: Stacking metaphor for novice mode and eyes-free selection for expert mode. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, page 173–180, New York, NY, USA, 2010. Association for Computing Machinery.

[48] Bruno Fruchard, Eric Lecolinet, and Olivier Chapuis. Markpad: Augmenting touchpads for command selection. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 5630–5642, New York, NY, USA, 2017. Association for Computing Machinery.

[49] Emilien Ghomi, Stéphane Huot, Olivier Bau, Michel Beaudouin-Lafon, and Wendy E. Mackay. Arpège: Learning multitouch chord gestures vocabularies. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '13, page 209–218, New York, NY, USA, 2013. Association for Computing Machinery.

[50] Wayne D. Gray and John K. Lindstedt. Plateaus, dips, and leaps: Where to look for inventions and discoveries during skilled performance. *Cognitive Science*, 41(7):1838–1870, 2017.

[51] Tovi Grossman, Pierre Dragicevic, and Ravin Balakrishnan. Strategies for accelerating on-line learning of hotkeys. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, page 1591–1600, New York, NY, USA, 2007. Association for Computing Machinery.

[52] François Guimbretière, Maureen Stone, and Terry Winograd. Fluid interaction with high-resolution wall-size displays. In *Proceedings of the 14th Annual ACM Symposium on User Interface Software and Technology*, UIST '01, page 21–30, New York, NY, USA, 2001. Association for Computing Machinery.

[53] Carl Gutwin, Andy Cockburn, and Benjamin Lafreniere. Testing the rehearsal hypothesis with two fasttap interfaces. In *Proceedings of the 41st Graphics Interface Conference*, GI '15, page 223–231, CAN, 2015. Canadian Information Processing Society.

[54] Carl Gutwin, Andy Cockburn, Joey Scarr, Sylvain Malacria, and Scott C. Olson. Faster command selection on tablets with fasttap. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, page 2617–2626, New York, NY, USA, 2014. Association for Computing Machinery.

[55] Jaehyun Han and Geehyuk Lee. Push-push: A drag-like operation overlapped with a page transition operation on touch interfaces. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology*, UIST '15, page 313–322, New York, NY, USA, 2015. Association for Computing Machinery.

[56] Jeffrey T. Hansberger, Chao Peng, Shannon L. Mathis, Vaidyanath Areyur Shanthakumar, Sarah C. Meacham, Lizhou Cao, and Victoria R. Blakely. Dispelling the Gorilla Arm Syndrome: The Viability of Prolonged Gesture Interactions. In *Virtual, Augmented and Mixed Reality*, volume 10280, pages 505–520. Springer International Publishing, Cham, 2017. Series Title: Lecture Notes in Computer Science.

[57] Rex Hartson. Cognitive, physical, sensory, and functional affordances in interaction design. *Behaviour & Information Technology*, 22(5):315–338, 2003.

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

[59] Da-Yuan Huang, Liwei Chan, Shuo Yang, Fan Wang, Rong-Hao Liang, De-Nian Yang, Yi-Ping Hung, and Bing-Yu Chen. Digitspace: Designing thumb-to-fingers touch interfaces for one-handed and eyes-free interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 1526–1537, New York, NY, USA, 2016. Association for Computing Machinery.

[60] Renate Häuslschmid, Benjamin Menrad, and Andreas Butz. Freehand vs. micro gestures in the car: Driving performance and user experience. In *2015 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 159–160, March 2015.

[61] Samia Ibtasam, Hamid Mehmood, Lubna Razaq, Jennifer Webster, Sarah Yu, and Richard Anderson. An exploration of smartphone based mobile money applications in pakistan. In *Proceedings of the Ninth International Conference on Information and Communication Technologies and Development*, ICTD '17, New York, NY, USA, 2017. Association for Computing Machinery.

[62] Jeff Johnson. *Designing with the mind in mind: simple guide to understanding user interface design guidelines*. Morgan Kaufmann, 2020.

[63] Alita Joyce. Mobile-app onboarding: An analysis of components and techniques, June 2020.

[64] Daniel Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

[65] Brian Krisler and Richard Alterman. Training towards mastery: Overcoming the active user paradox. In *Proceedings of the 5th Nordic Conference on Human-Computer Interaction: Building Bridges*, NordiCHI '08, page 239–248, New York, NY, USA, 2008. Association for Computing Machinery.

[66] Gordon Kurtenbach and William Buxton. Issues in combining marking and direct manipulation techniques. In *Proceedings of the 4th Annual ACM Symposium on User Interface Software and Technology*, UIST '91, pages 137–144, New York, NY, USA, 1991. ACM.

[67] Gordon Paul Kurtenbach. *The design and evaluation of marking menus.* PhD thesis, University of Toronto, 1993.

[68] Benjamin Lafreniere, Carl Gutwin, and Andy Cockburn. Investigating the post-training persistence of expert interaction techniques. *ACM Trans. Comput.-Hum. Interact.*, 24(4), August 2017.

[69] Benjamin Lafreniere, Carl Gutwin, Andy Cockburn, and Tovi Grossman. Faster command selection on touchscreen watches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 4663–4674, New York, NY, USA, 2016. Association for Computing Machinery.

[70] David M Lane, H Albert Napier, S Camille Peres, and Aniko Sandor. Hidden costs of graphical user interfaces: Failure to make the transition from menus and icon toolbars to keyboard shortcuts. *International Journal of Human-Computer Interaction*, 18(2):133–144, 2005.

[71] Kris M.Y. Law, Victor C.S. Lee, and Y.T. Yu. Learning motivation in e-learning facilitated computer programming courses. *Computers and Education*, 55(1):218–228, 2010.

[72] G. Julian Lepinski, Tovi Grossman, and George Fitzmaurice. The design and evaluation of multitouch marking menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, page 2233–2242, New York, NY, USA, 2010. Association for Computing Machinery.

[73] Blaine Lewis. Longer delays in rehearsal-based interfaces increase expert use. Master's thesis, University of Waterloo, 2019.

[74] Blaine Lewis and Daniel Vogel. Longer delays in rehearsal-based interfaces increase expert use. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 27(6):1–41, 2020.

[75] Guangchuan Li, David Rempel, Yue Liu, Weitao Song, and Carisa Harris Adamson. Design of 3D Microgestures for Commands in Virtual Reality or Augmented Reality. *Applied Sciences*, 11(14):6375, January 2021. Number: 14 Publisher: Multidisciplinary Digital Publishing Institute.

[76] Sungjin Lim, Hosung Jeon, Minwoo Jung, Chulwoong Lee, Woonchan Moon, Kwangsoo Kim, Hwi Kim, and Joonku Hahn. Fatigue-free visual perception of high-density super-multiview augmented reality images. *Scientific Reports*, 12(1):2959, February 2022. Number: 1.

[77] Eva Mackamul. Improving the discoverability of interactions in interactive systems. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI EA '22, New York, NY, USA, 2022. Association for Computing Machinery.

[78] Nathan Magrofuoco, Jorge-Luis Perez-Medina, Paolo Roselli, Jean Vanderdonckt, and Santiago Villarreal. Eliciting Contact-Based and Contactless Gestures With Radar-Based Sensors. *IEEE Access*, 7:176982–176997, 2019.

[79] Sylvain Malacria, Joey Scarr, Andy Cockburn, Carl Gutwin, and Tovi Grossman. Skillometers: Reflective widgets that motivate and help users to improve performance. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, page 321–330, New York, NY, USA, 2013. Association for Computing Machinery.

[80] Mamaylya. HoloLens 2 gestures (for example, gaze and air tap) for navigating a guide in Dynamics 365 Guides - Dynamics 365 Mixed Reality, November 2022.

[81] Damien Masson, Alix Goguey, Sylvain Malacria, and Géry Casiez. Whichfingers: Identifying fingers on touch surfaces and keyboards using vibration sensors. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, page 41–48, New York, NY, USA, 2017. Association for Computing Machinery.

[82] Damien Masson, Sylvain Malacria, Géry Casiez, and Daniel Vogel. Charagraph: Interactive generation of charts for realtime annotation of data-rich paragraphs. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery.

[83] Damien Masson, Sylvain Malacria, Géry Casiez, and Daniel Vogel. Statslator: Interactive translation of nhst and estimation statistics reporting styles in scientific documents. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, New York, NY, USA, 2023. Association for Computing Machinery.

[84] Damien Masson, Sylvain Malacria, Edward Lank, and Géry Casiez. Bringing interactivity to research papers and presentations with chameleon. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI EA '20, page 1–4, New York, NY, USA, 2020. Association for Computing Machinery.

[85] Damien Masson, Sylvain Malacria, Edward Lank, and Géry Casiez. Chameleon: Bringing interactivity to static digital documents. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[86] Damien Masson, Sylvain Malacria, Daniel Vogel, Edward Lank, and Géry Casiez. Chartdetective: Easy and accurate interactive data extraction from complex vector charts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery.

[87] Sven Mayer, Lars Lischke, Adrian Lanksweirt, Huy Viet Le, and Niels Henze. How to communicate new input techniques. In *Proceedings of the 10th Nordic Conference on Human-Computer Interaction*, NordiCHI '18, page 460–472, New York, NY, USA, 2018. Association for Computing Machinery.

[88] Erin McAweeney, Haihua Zhang, and Michael Nebeling. User-Driven Design Principles for Gesture Representations. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, Montreal QC Canada, April 2018. ACM.

[89] Rolf Molich and Jakob Nielsen. Improving a human-computer dialogue. *Commun. ACM*, 33(3):338–348, mar 1990.

[90] Philippe Mougin. F-script: Smalltalk scripting for the mac os x object system. In *Companion of the 18th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, OOPSLA '03, page 84–85, New York, NY, USA, 2003. Association for Computing Machinery.

[91] Allen Newell and Paul S Rosenbloom. Mechanisms of skill acquisition and the law of practice. *Cognitive skills and their acquisition*, 1(1981):1–55, 1981.

[92] Jakob Nielsen. *Usability engineering*. Morgan Kaufmann, 1994.

[93] Jakob Nielsen. Ten usability heuristics. 2005.

[94] Donald A. Norman. *The design of everyday things: Revised and expanded edition*. Basic books, 2013.

[95] Daniel L. Odell, Richard C. Davis, Andrew Smith, and Paul K. Wright. Toolglasses, marking menus, and hotkeys: A comparison of one and two-handed command selection techniques. In *Proceedings of Graphics Interface 2004*, GI '04, page 17–24, Waterloo, CAN, 2004. Canadian Human-Computer Communications Society.

[96] Stellan Ohlsson. *Computational Models of Skill Acquisition*, page 359–395. Cambridge Handbooks in Psychology. Cambridge University Press, 2008.

[97] S. Camille Peres, II Franklin P. Tamborello, II Michael D. Fleetwood, II Phillip Chung, and II Danielle L. Paige-Smith. Keyboard shortcut usage: The roles of social factors and computer experience. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 48(5):803–807, 2004.

[98] Nicole Ke Chen Pong and Sylvain Malacria. Awareness, usage and discovery of swipe-revealed hidden widgets in ios. In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces*, ISS '19, page 193–204, New York, NY, USA, 2019. Association for Computing Machinery.

[99] Nicole Ke Cheng Pong. *Interacting with signifier-less designs - the case of swhidgets*. Theses, Université de Lille (2018-....), Oct 2020.

[100] Stuart Pook, Eric Lecolinet, Guy Vaysseix, and Emmanuel Barillot. Control menus: Excecution and control in a single interactor. In *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '00, page 263–264, New York, NY, USA, 2000. Association for Computing Machinery.

[101] Manuel Prätorius, Dimitar Valkov, Ulrich Burgbacher, and Klaus Hinrichs. DigiTap: an eyes-free VR/AR symbolic input device. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology - VRST '14*, pages 9–18, Edinburgh, Scotland, 2014. ACM Press.

[102] Thibault Raffaillac and Stéphane Huot. Polyphony: Programming interfaces and interactions with the entity-component-system model. *Proc. ACM Hum.-Comput. Interact.*, 3(EICS), jun 2019.

[103] Thibault Raffaillac and Stéphane Huot. What do researchers need when implementing novel interaction techniques? *Proc. ACM Hum.-Comput. Interact.*, 6(EICS), jun 2022.

[104] Harini Sampath, Alice Merrick, and Andrew Macvean. Accessibility of command line interfaces. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery.

[105] Joey Scarr, Andy Cockburn, and Carl Gutwin. Supporting and exploiting spatial memory in user interfaces. *Foundations and Trends® in Human–Computer Interaction*, 6(1):1–84, 2013.

[106] Joey Scarr, Andy Cockburn, Carl Gutwin, and Philip Quinn. Dips and ceilings: Understanding and supporting transitions to expertise in user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2741–2750, New York, NY, USA, 2011. ACM.

[107] Richard A Schmidt, Timothy D Lee, Carolee Winstein, Gabriele Wulf, and Howard N Zelaznik. *Motor control and learning: A behavioral emphasis*. Human kinetics, 2018.

[108] Ross Shannon, Aaron Quigley, and Paddy Nixon. Software considerations for automotive pervasive systems. *Workshop on Software Engineering Challenges for Ubiquitous Computing*, 01 2006.

[109] Adwait Sharma, Michael A. Hedderich, Divyanshu Bhardwaj, Bruno Fruchard, Jess McIntosh, Aditya Shekhar Nittala, Dietrich Klakow, Daniel Ashbrook, and Jürgen Steimle. SoloFinger: Robust Microgestures while Grasping Everyday Objects. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–15, Yokohama Japan, May 2021. ACM.

[110] Ben Shneiderman. Direct manipulation: A step beyond programming languages. *SIGSOC Bull.*, 13(2–3):143, may 1981.

[111] Ben Shneiderman, Catherine Plaisant, Maxine S Cohen, Steven Jacobs, Niklas Elmqvist, and Nicholas Diakopoulos. *Designing the user interface: strategies for effective human-computer interaction*. Pearson, 2016.

[112] Herbert A Simon. Rational choice and the structure of the environment. *Psychological review*, 63(2):129, 1956.

[113] George S Snoddy. Learning and stability: a psychophysiological analysis of a case of motor learning with clinical applications. *Journal of Applied Psychology*, 10(1):1, 1926.

[114] K. M. Swigger. Review of "the psychology of human-computer interaction by stuart k. card, thomas p. moran and alan newell", lawrence erlbaum associates, hillsdale, nj, 1983. *SIGCHI Bull.*, 15(4):26, April 1984.

[115] Susanne Tak, Piet Westendorp, and Iris van Rooij. Satisficing and the use of keyboard shortcuts: Being good enough is enough? *Interacting with Computers*, 25(5):404–416, 2013.

[116] Caitlin Tenison and John R Anderson. Modeling the distinct phases of skill acquisition. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 42(5):749, 2016.

[117] Daniel Vogel and Patrick Baudisch. Shift: A technique for operating pen-based interfaces using touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, page 657–666, New York, NY, USA, 2007. Association for Computing Machinery.

[118] Johann Wentzel, Greg d'Eon, and Daniel Vogel. Improving virtual reality ergonomics through reach-bounded non-linear input amplification. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–12, New York, NY, USA, 2020. Association for Computing Machinery.

[119] Eric Whitmire, Mohit Jain, Divye Jain, Greg Nelson, Ravi Karkar, Shwetak Patel, and Mayank Goel. DigiTouch: Reconfigurable Thumb-to-Finger Input and Text Entry on Head-mounted Displays. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):1–21, September 2017. Number: 3.

[120] Daniel B Willingham, Mary J Nissen, and Peter Bullemer. On the development of procedural knowledge. *Journal of experimental psychology: learning, memory, and cognition*, 15(6):1047, 1989.

[121] Katrin Wolf, Anja Naumann, Michael Rohs, and Jörg Müller. A Taxonomy of Microinteractions: Defining Microgestures Based on Ergonomic and Scenario-Dependent Requirements. In *Human-Computer Interaction – INTERACT 2011*, volume 6946, pages 559–575. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. Series Title: Lecture Notes in Computer Science.

[122] Shengdong Zhao, Maneesh Agrawala, and Ken Hinckley. Zone and polygon menus: Using relative position to increase the breadth of multi-stroke marking menus. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, page 1077–1086, New York, NY, USA, 2006. Association for Computing Machinery.

[123] Shengdong Zhao and Ravin Balakrishnan. Simple vs. compound mark hierarchical marking menus. In *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, page 33–42, New York, NY, USA, 2004. Association for Computing Machinery.

[124] Jingjie Zheng, Xiaojun Bi, Kun Li, Yang Li, and Shumin Zhai. M3 gesture menu: Design and experimental analyses of marking menus for touchscreen mobile interaction. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 1–14, New York, NY, USA, 2018. Association for Computing Machinery.

[125] John Zimmerman, Jodi Forlizzi, and Shelley Evenson. Research through design as a method for interaction design research in hci. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, page 493–502, New York, NY, USA, 2007. Association for Computing Machinery.

[126] Klen Čopič Pucihar, Christian Sandor, Matjaž Kljun, Wolfgang Huerst, Alexander Plopski, Takafumi Taketomi, Hirokazu Kato, and Luis A. Leiva. The Missing Interface: Micro-Gestures on Augmented Objects. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–6, Glasgow Scotland Uk, May 2019. ACM.

[HDR1] Axel Antoine, Sylvain Malacria, Nicolai Marquardt, and Géry Casiez. Interaction illustration taxonomy: Classification of styles and techniques for visually representing interaction scenarios. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery.

[HDR2] Katherine Fennedy, Sylvain Malacria, Hyowon Lee, and Simon Tangi Perrault. Investigating performance and usage of input methods for soft keyboard hotkeys. In *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '20, New York, NY, USA, 2020. Association for Computing Machinery.

[HDR3] Katherine Fennedy, Angad Srivastava, Sylvain Malacria, and Simon T. Perrault. Towards a unified and efficient command selection mechanism for touch-based devices using soft keyboard hotkeys. *ACM Trans. Comput.-Hum. Interact.*, 29(1), jan 2022.

[HDR4] Emmanouil Giannisakis, Gilles Bailly, Sylvain Malacria, and Fanny Chevalier. Iconhk: Using toolbar button icons to communicate keyboard shortcuts. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 4715–4726, New York, NY, USA, 2017. Association for Computing Machinery.

[HDR5] Alix Goguey, Sylvain Malacria, Andy Cockburn, and Carl Gutwin. Reducing error aversion to support novice-to-expert transitions with fasttap. In *Proceedings of the 31st Conference on l'Interaction Homme-Machine*, IHM '19, New York, NY, USA, 2019. Association for Computing Machinery.

[HDR6] Alix Goguey, Sylvain Malacria, and Carl Gutwin. Improving discoverability and expert performance in force-sensitive text selection for touch devices with mode gauges. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–12, New York, NY, USA, 2018. Association for Computing Machinery.

[HDR7] Jay Henderson, Sylvain Malacria, Mathieu Nancel, and Edward Lank. Investigating the necessity of delay in marking menu invocation. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[HDR8] Vincent Lambert, Adrien Chaffangeon Caillet, Alix Goguey, Sylvain Malacria, and Laurence Nigay. Studying the visual representation of microgestures. *Proc. ACM Hum.-Comput. Interact.*, 7(MHCI), 9 2023.

[HDR9] Eva Mackamul, Géry Casiez, and Sylvain Malacria. Exploring visual signifier characteristics to improve the perception of affordances of in-place touch inputs. *Proc. ACM Hum.-Comput. Interact.*, 7(MHCI), 9 2023.

[HDR10] Sylvain Malacria, Gilles Bailly, Joel Harrison, Andy Cockburn, and Carl Gutwin. Promoting hotkey use through rehearsal with exposehk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, page 573–582, New York, NY, USA, 2013. Association for Computing Machinery.

[HDR11] Sylvain Malacria, Joey Scarr, Andy Cockburn, Carl Gutwin, and Tovi Grossman. Skillometers: Reflective widgets that motivate and help users to improve performance. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, page 321–330, New York, NY, USA, 2013. Association for Computing Machinery.

[HDR12] Nicole Ke Chen Pong and Sylvain Malacria. Awareness, usage and discovery of swipe-revealed hidden widgets in ios. In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces*, ISS '19, page 193–204, New York, NY, USA, 2019. Association for Computing Machinery.

[HDR13] Quentin Roy, Sylvain Malacria, Yves Guiard, Eric Lecolinet, and James Eagan. Augmented letters: Mnemonic gesture-based shortcuts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, page 2325–2328, New York, NY, USA, 2013. Association for Computing Machinery.

[F1] J. Aceituno, S. Malacria, P. Quinn, N. Roussel, A. Cockburn, and G. Casiez. The design, use, and performance of edge-scrolling techniques. *International Journal of Human-Computer Studies*, 97:58–76, 2017.

[F2] Jessalyn Alvina, Andrea Bunt, Parmit K. Chilana, Sylvain Malacria, and Joanna McGrenere. Where is that feature? designing for cross-device software learnability. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference*, DIS '20, page 1103–1115, New York, NY, USA, 2020. Association for Computing Machinery.

[F3] Axel Antoine, Sylvain Malacria, and Géry Casiez. Controlling autoscroll on trackpads using the force. In *Actes de La 28ième Conference Francophone Sur l'Interaction Homme-Machine*, IHM '16, page 264–270, New York, NY, USA, 2016. Association for Computing Machinery.

[F4] Axel Antoine, Sylvain Malacria, and Géry Casiez. Forceedge: Controlling autoscroll on both desktop and mobile computers using the force. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 3281–3292, New York, NY, USA, 2017. Association for Computing Machinery.

[F5] Axel Antoine, Sylvain Malacria, and Géry Casiez. Turbomouse: End-to-end latency compensation in indirect interaction. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI EA '18, page 1–4, New York, NY, USA, 2018. Association for Computing Machinery.

[F6] Axel Antoine, Sylvain Malacria, and Géry Casiez. Using high frequency accelerometer and mouse to compensate for end-to-end latency in indirect interaction. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–11, New York, NY, USA, 2018. Association for Computing Machinery.

[F7] Axel Antoine, Sylvain Malacria, Nicolai Marquardt, and Géry Casiez. Esquisse: Using 3d models staging to facilitate the creation of vector-based trace figures. In David Lamas, Fernando Loizides, Lennart Nacke, Helen Petrie, Marco Winckler, and Panayiotis Zaphiris, editors,

Human-Computer Interaction – INTERACT 2019, pages 496–516, Cham, 2019. Springer International Publishing.

[F8] Axel Antoine, Sylvain Malacria, Nicolai Marquardt, and Géry Casiez. Interaction illustration taxonomy: Classification of styles and techniques for visually representing interaction scenarios. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery.

[F9] Axel Antoine, Sylvain Malacria, Daniel Vogel, and Géry Casiez. Étude de l'influence de la taille des sphères virtuelles de contrôle sur les rotations 3d: Studying the influence of the size of a virtual trackball on 3d rotations. In *Proceedings of the 33rd Conference on l'Interaction Humain-Machine*, IHM '22, New York, NY, USA, 2022. Association for Computing Machinery.

[F10] Jeff Avery, Sylvain Malacria, Mathieu Nancel, Géry Casiez, and Edward Lank. Introducing transient gestures to improve pan and zoom on touch surfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–8, New York, NY, USA, 2018. Association for Computing Machinery.

[F11] Mathias Baglioni, Sylvain Malacria, Eric Lecolinet, and Yves Guiard. Flick-and-brake: Finger control over inertial/sustained scroll motion. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, page 2281–2286, New York, NY, USA, 2011. Association for Computing Machinery.

[F12] Gilles Bailly and Sylvain Malacria. Menuinspector: Outil pour l'analyse des menus et cas d'étude. In *Proceedings of the 25th Conference on l'Interaction Homme-Machine*, IHM '13, page 103–106, New York, NY, USA, 2013. Association for Computing Machinery.

[F13] Gilles Bailly and Sylvain Malacria. *Command Selection*, pages 1–35. Springer International Publishing, Cham, 2020.

[F14] Gilles Bailly, Sidharth Sahdev, Sylvain Malacria, and Thomas Pietrzak. Livingdesktop: Augmenting desktop workstation with actuated devices. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 5298–5310, New York, NY, USA, 2016. Association for Computing Machinery.

[F15] Mathieu Berthellemy, Elodie Cayez, Marwan Ajem, Gilles Bailly, Sylvain Malacria, and Eric Lecolinet. Spotpad, locipad, chordpad and inoutpad: Investigating gesture-based input on touchpad. In *Proceedings of the 27th Conference on l'Interaction Homme-Machine*, IHM '15, New York, NY, USA, 2015. Association for Computing Machinery.

[F16] Yuan Chen, Géry Casiez, Sylvain Malacria, and Ed Lank. Exploring the effects of intended use on targeting in virtual reality. In *Proceedings of the Graphics Interface 2023 Conference (2023)*, GI '23, 1 2023.

[F17] Andy Cockburn, Carl Gutwin, Joey Scarr, and Sylvain Malacria. Supporting novice to expert transitions in user interfaces. *ACM Comput. Surv.*, 47(2), nov 2014.

[F18] Marios Constantinides, John Dowell, David Johnson, and Sylvain Malacria. Exploring mobile news reading interactions for news app personalisation. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, page 457–462, New York, NY, USA, 2015. Association for Computing Machinery.

[F19] Marios Constantinides, John Dowell, David Johnson, and Sylvain Malacria. Habito news: A research tool to investigate mobile news reading. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, MobileHCI '15, page 598, New York, NY, USA, 2015. Association for Computing Machinery.

[F20] John Dowell, Sylvain Malacria, Hana Kim, and Edward Anstead. Companion apps for information-rich television programmes: representation and interaction. *Personal and Ubiquitous Computing*, page 14, 0 2015.

[F21] Katherine Fennedy, Sylvain Malacria, Hyowon Lee, and Simon Tangi Perrault. Investigating performance and usage of input methods for soft keyboard hotkeys. In *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '20, New York, NY, USA, 2020. Association for Computing Machinery.

[F22] Katherine Fennedy, Angad Srivastava, Sylvain Malacria, and Simon T. Perrault. Towards a unified and efficient command selection mechanism for touch-based devices using soft keyboard hotkeys. *ACM Trans. Comput.-Hum. Interact.*, 29(1), jan 2022.

[F23] Christian Frisson, Sylvain Malacria, Gilles Bailly, and Thierry Dutoit. Inspectorwidget: A system to analyze users behaviors in their applications. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, page 1548–1554, New York, NY, USA, 2016. Association for Computing Machinery.

[F24] Bruno Fruchard, Cécile Avezou, Sylvain Malacria, Géry Casiez, and Stéphane Huot. A case study on the design and use of an annotation and analytical tool tailored to lead climbing. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI EA '23, New York, NY, USA, 2023. Association for Computing Machinery.

[F25] Bruno Fruchard, Sylvain Malacria, Géry Casiez, and Stéphane Huot. User preference and performance using tagging and browsing for image labeling. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery.

[F26] Emmanouil Giannisakis, Gilles Bailly, Sylvain Malacria, and Fanny Chevalier. Iconhk: Using toolbar button icons to communicate keyboard shortcuts. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 4715–4726, New York, NY, USA, 2017. Association for Computing Machinery.

[F27] Alix Goguey, Sylvain Malacria, Andy Cockburn, and Carl Gutwin. Reducing error aversion to support novice-to-expert transitions with fasttap. In *Proceedings of the 31st Conference on l'Interaction Homme-Machine*, IHM '19, New York, NY, USA, 2019. Association for Computing Machinery.

[F28] Alix Goguey, Sylvain Malacria, and Carl Gutwin. Improving discoverability and expert performance in force-sensitive text selection for touch devices with mode gauges. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–12, New York, NY, USA, 2018. Association for Computing Machinery.

[F29] Carl Gutwin, Andy Cockburn, Joey Scarr, Sylvain Malacria, and Scott C. Olson. Faster command selection on tablets with fasttap. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, page 2617–2626, New York, NY, USA, 2014. Association for Computing Machinery.

[F30] Raiza Hanada, Damien Masson, Géry Casiez, Mathieu Nancel, and Sylvain Malacria. Relevance and applicability of hardware-independent

pointing transfer functions. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, UIST '21, page 524–537, New York, NY, USA, 2021. Association for Computing Machinery.

[F31] Jay Henderson, Sylvain Malacria, Mathieu Nancel, and Edward Lank. Investigating the necessity of delay in marking menu invocation. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[F32] Vincent Lambert, Adrien Chaffangeon Caillet, Alix Goguey, Sylvain Malacria, and Laurence Nigay. Studying the visual representation of microgestures. *Proc. ACM Hum.-Comput. Interact.*, 7(MHCI), 9 2023.

[F33] Eva Mackamul, Géry Casiez, and Sylvain Malacria. Exploring visual signifier characteristics to improve the perception of affordances of in-place touch inputs. *Proc. ACM Hum.-Comput. Interact.*, 7(MHCI), 9 2023.

[F34] Sylvain Malacria, Jonathan Aceituno, Philip Quinn, Géry Casiez, Andy Cockburn, and Nicolas Roussel. Push-edge and slide-edge: Scrolling by pushing against the viewport edge. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, page 2773–2776, New York, NY, USA, 2015. Association for Computing Machinery.

[F35] Sylvain Malacria, Gilles Bailly, Joel Harrison, Andy Cockburn, and Carl Gutwin. Promoting hotkey use through rehearsal with exposehk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, page 573–582, New York, NY, USA, 2013. Association for Computing Machinery.

[F36] Sylvain Malacria, Alix Goguey, Gilles Bailly, and Géry Casiez. Multi-touch trackpads in the wild. In *Actes de La 28ième Conference Francophone Sur l'Interaction Homme-Machine*, IHM '16, page 19–24, New York, NY, USA, 2016. Association for Computing Machinery.

[F37] Sylvain Malacria and Eric Lecolinet. Espace de caractérisation du stylo numérique. In *Proceedings of the 20th Conference on l'Interaction Homme-Machine*, IHM '08, page 177–184, New York, NY, USA, 2008. Association for Computing Machinery.

[F38] Sylvain Malacria and Eric Lecolinet. U-note: Classe augmentée et stylo numérique. In *Proceedings of the 21st International Conference on Associ-*

ation Francophone d'Interaction Homme-Machine, IHM '09, page 255–258, New York, NY, USA, 2009. Association for Computing Machinery.

[F39] Sylvain Malacria, Eric Lecolinet, and Yves Guiard. Clutch-free panning and integrated pan-zoom control on touch-sensitive surfaces: The cyclostar approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, page 2615–2624, New York, NY, USA, 2010. Association for Computing Machinery.

[F40] Sylvain Malacria, Thomas Pietrzak, Aurélien Tabard, and Éric Lecolinet. U-note: Capture the class and access it everywhere. In Pedro Campos, Nicholas Graham, Joaquim Jorge, Nuno Nunes, Philippe Palanque, and Marco Winckler, editors, *Human-Computer Interaction – INTERACT 2011*, pages 643–660, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[F41] Sylvain Malacria, Joey Scarr, Andy Cockburn, Carl Gutwin, and Tovi Grossman. Skillometers: Reflective widgets that motivate and help users to improve performance. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, page 321–330, New York, NY, USA, 2013. Association for Computing Machinery.

[F42] Damien Masson, Alix Goguey, Sylvain Malacria, and Géry Casiez. Whichfingers: Identifying fingers on touch surfaces and keyboards using vibration sensors. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, page 41–48, New York, NY, USA, 2017. Association for Computing Machinery.

[F43] Damien Masson, Sylvain Malacria, Géry Casiez, and Daniel Vogel. Charagraph: Interactive generation of charts for realtime annotation of data-rich paragraphs. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery.

[F44] Damien Masson, Sylvain Malacria, Edward Lank, and Géry Casiez. Bringing interactivity to research papers and presentations with chameleon. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI EA '20, page 1–4, New York, NY, USA, 2020. Association for Computing Machinery.

[F45] Damien Masson, Sylvain Malacria, Edward Lank, and Géry Casiez. Chameleon: Bringing interactivity to static digital documents. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*,

CHI '20, page 1–13, New York, NY, USA, 2020. Association for Computing Machinery.

[F46] Damien Masson, Sylvain Malacria, Daniel Vogel, Edward Lank, and Géry Casiez. Chartdetective: Easy and accurate interactive data extraction from complex vector charts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA, 2023. Association for Computing Machinery.

[F47] Simon Perrault, Sylvain Malacria, Yves Guiard, and Eric Lecolinet. Watchit: Simple gestures for interacting with a watchstrap. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, page 1467–1468, New York, NY, USA, 2012. Association for Computing Machinery.

[F48] Thomas Pietrzak, Sylvain Malacria, and Gilles Bailly. Ctrlmouse et touchctrl: Duplicating mode delimiters on the mouse. In *Proceedings of the 26th Conference on l'Interaction Homme-Machine*, IHM '14, page 38–47, New York, NY, USA, 2014. Association for Computing Machinery.

[F49] Thomas Pietrzak, Sylvain Malacria, and Éric Lecolinet. S-notebook: Augmenting mobile devices with interactive paper for data management. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, page 733–736, New York, NY, USA, 2012. Association for Computing Machinery.

[F50] Nicole Ke Chen Pong and Sylvain Malacria. Awareness, usage and discovery of swipe-revealed hidden widgets in ios. In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces*, ISS '19, page 193–204, New York, NY, USA, 2019. Association for Computing Machinery.

[F51] Philip Quinn, Sylvain Malacria, and Andy Cockburn. Touch scrolling transfer functions. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, page 61–70, New York, NY, USA, 2013. Association for Computing Machinery.

[F52] Quentin Roy, Sylvain Malacria, Yves Guiard, Eric Lecolinet, and James Eagan. Augmented letters: Mnemonic gesture-based shortcuts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, page 2325–2328, New York, NY, USA, 2013. Association for Computing Machinery.

[F53] Joey Scarr, Andy Cockburn, Carl Gutwin, and Sylvain Malacria. Testing the robustness and performance of spatially consistent interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, page 3139–3148, New York, NY, USA, 2013. Association for Computing Machinery.

[F54] Philippe Schmid, Sylvain Malacria, Andy Cockburn, and Mathieu Nancel. Interaction interferences: Implications of last-instant system state changes. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, UIST '20, page 516–528, New York, NY, USA, 2020. Association for Computing Machinery.

[F55] Shaishav Siddhpuria, Sylvain Malacria, Mathieu Nancel, and Edward Lank. Pointing at a distance with everyday smart devices. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–11, New York, NY, USA, 2018. Association for Computing Machinery.