Study and design of interaction techniques to facilitate object selection and manipulation in virtual environments on mobile devices

Siju Wu

NNT : 2015SACLE023

THESE DE DOCTORAT
DE L'UNIVERSITE PARIS-SACLAY,
préparée à l'Université d'Evry Val-d'Essonne

ÉCOLE DOCTORALE N°580
STIC : Sciences et technologies de l'information et de la communication

Spécialité de doctorat : Informatique

Par

Mr Siju Wu

Study and design of interaction techniques to facilitate object selection and manipulation in virtual environments on mobile devices

Thèse présentée et soutenue à Evry, le 30 novembre 2015 :

Composition du Jury :

M, Merienne, Frédéric, Professeur, Institut Image - ENSAM, Examinateur, Président
M, Casiez, Géry, Professeur, Université de Lille 1, Rapporteur
M, Duval, Thierry, Professeur, IMT - Télécom Bretagne, Rapporteur
M, Arnaldi, Bruno, Professeur, INSA de Rennes, Examinateur
M, Dumas, Cédric, Maître de conférences, Ecole des Mines de Nantes, Examinateur
M, Otmane, Samir, Professeur, UEVE, Directeur de thèse
M, Moreau, Guillaume, Professeur, Ecole Centrale de Nantes, Co-directeur de thèse
M, Chellali, Amine, Maître de Conférences, UEVE, Co-encadrant

Résumé

Les avancées dans le domaine des NUIs (interfaces utilisateur naturelles) permettent aux concepteurs de développer de nouvelles techniques efficaces et faciles à utiliser pour l'interaction 3D. Dans ce contexte, les interfaces mobiles attirent beaucoup d'attention sur la conception de techniques d'interaction 3D pour une utilisation ubiquitaire.

Nos travaux de recherche se focalisent sur la proposition de nouvelles techniques d'interaction pour faciliter la sélection et la manipulation d'objets dans des environnements virtuels s'exécutant sur des interfaces mobiles. En effet, l'efficacité et la précision de la sélection des l'objets sont fortement affectés par la taille de la cible et la densité de l'environnement virtuel.

Pour surmonter le problème d'occlusion du bout des doigts sur les Smartphones, nous avons conçu deux techniques de sélection reposant sur le toucher. Nous avons également conçu une technique hybride à main levée pour la sélection à distance de petits objets. Pour effectuer une manipulation d'objets contraints sur les Tablet-PC, nous avons proposé une technique bimanuelle basée sur un modèle asymétrique. Les deux mains peuvent être utilisés en collaboration, afin de spécifier la contrainte, déterminer le mode de manipulation et de contrôler la transformation. Nous avons également proposé deux autres techniques de manipulation à une seule main en utilisant les points de contacts identifiés.

Les évaluations de nos techniques démontrent qu'ils peuvent améliorer l'expérience des interactions utilisateurs sur des interfaces mobiles. Nos résultats permettent aussi de donner quelques lignes directrices pour améliorer la conception de techniques d'interactions 3D sur des interfaces mobiles.

Mots clés : interaction 2D/3D, interface naturelle utilisateur, environnement virtuel, interface mobile

Abstract

The advances in the field of NUIs (Natural User Interfaces) can provide more and more guidelines for designers to develop efficient and easy-to-use techniques for 3D interaction. In this context, mobile devices attract much attention to design 3D interaction techniques for ubiquitous usage.

Our research work focuses on proposing new techniques to facilitate object selection and manipulation in virtual environments on mobile devices. Indeed, the efficiency and accuracy of object selection are highly affected by the target size and the cluster density.

To overcome the fingertip occlusion issue on Smartphones, we have designed two touch-based selection techniques. We have also designed one freehand hybrid technique for selection of small objects displayed at a distance. To perform constrained manipulation on Tablet-PCs, we have proposed a bimanual technique based on the asymmetrical model. Both hands can be used in collaboration, in order to specify the constraint, determine the manipulation mode, and control the transformation. We have also proposed two other single-hand manipulation techniques using identified touch inputs.

The evaluations of our techniques demonstrate that they can improve the users' interaction experience on mobile devices. Our results permit also to give some guidelines to improve the design of 3D interactions techniques on mobile devices.

Keywords: 2D/3D interaction, natural user interfaces, virtual environment, mobile devices

Acknowledgements

This PhD work was carried out in the Interaction, Augmented Reality and Ambient Robotic (IRA2) research group, at the laboratory of Informatics, Integrated Biology and Complex systems (IBISC) associated with the University of Evry Val d'Essonne, member of the University of Paris Saclay.

First, I would like to thank my three thesis supervisors: Mr. Samir OTMANE, professor at the University of Evry Val d'Essonne; Mr. Guillaume MOREAU, professor at Ecole Centrale de Nantes; Mr. Amine CHELLALI, assistant professor at the University of Evry Val d'Essonne. Thanks to them, I got the opportunity to do my PhD research work for three years. I would like also thank to Mr JinSong Yu, associated professor at BeiHang University, who proposed me this opportunity.

I am sincerely grateful to the members of the jury, Mr. Géry CASIEZ, professor at the University Lille 1; Mr. Thierry DUVAL, professor at IMT-TÉLÉCOM Bretagne; Mr. Bruno ARNALDI, professor at INSA de Rennes; Mr. Cédric DUMAS, assistant professor at Ecole des Mines de Nantes and Mr. Frédéric MERIENNE, professor at Image Institute-ENSAM, who accepted to evaluate this work, more particularly the two reporters who reviewed this manuscript. I address the most sincere gratitude to Mr. Samir OTMANE, Mr. Guillaume MOREAU and Mr. Amine CHELLALI, who taught me and helped me a lot during the three years. Without your support and patience, it would have been much more difficult to obtain all the achievements of today. I really appreciate your generous giving.

I want also to thank all my colleagues, the professors, the engineers, as well as the secretaries. Thanks for their company and selfless help. I acknowledge China Scholarship Council who gave me the financial support for 48 months, which provided me good living conditions in France.

At last but the most important gratitude should be addressed to my parents, my whole family and all of my friends who have supported me from the beginning to the end, who have shared my joys and sorrows, whom I have been loving forever.

Contents

List of figures

List of Tables

Introduction

Introduction

After the dominance of the WIMP paradigm for GUIs during 30 years, here comes the era of the next generation of interfaces. In fact, since Apple has launched the iPhone in 2007, the touch-based paradigm has been introduced. This quickly reshaped the interaction techniques of mobile devices such as smartphones and tablePCs. After the emergence of the touchscreen, other new interaction devices such as the Kinect, the Leap Motion Controller and the Google Glass appeared one after another. Because there is a great difference between these new interaction devices and the mouse, they are not adapted for use with traditional GUIs. Natural User Interfaces (NUIs) have gained momentum during the last decade and are considered as the next generation of User Interfaces. Indeed, they are expected to provide a more intuitive and seamless interaction experience than traditional UIs by taking advantage of learnt skills of human beings.

3D interaction is one of the domains that are being reshaped by the development of NUIs. In many interaction scenarios, it is inconvenient or even impossible to use the mouse and keyboard for some tasks. The emergence of new input and output devices brings the hopes that new interaction techniques, more adapted for this context of use, can now be designed and developed. The advances in the field of NUIs can provide more and more guidelines for designers to develop efficient and easy-to-use interaction techniques for 3D interaction. In this context, mobile devices attract much attention to design 3D interaction techniques for ubiquitous usage. For instance, architects and industrial designers may find it necessary to make quick sketches on tabletPCs in outdoor environments when inspiration comes. Moreover, augmented reality applications require efficient and walk-up-and-use techniques to manipulate augmented content in the real space.

Our research work focuses on proposing new techniques to facilitate object selection and manipulation in virtual environments for mobile devices. On touchscreens, object selection is highly affected by the fingertip occlusion problem. Although a lot of techniques have been proposed to improve the selection precision, the accuracy is acquired at the cost of efficiency. Our first objective was then to design new touch-based selection techniques that can improve the efficiency without sacrificing the accuracy.

Besides touch-based selection, it also turns out to be difficult to select small objects displayed at a distance using traditional 3D selection techniques. In fact, because many 3D selection techniques rely on the use of handheld devices or freehand movements in air, there is a lack of physical support to stabilize the selection tool in the virtual environment. Our second objective was then to design and develop a freehand hybrid selection technique that aims to facilitate the selection of small targets from a dense cluster. The designed technique can switch to the appropriate modality automatically according to the cluster density and the target size in the proximity of the selection tool. It allows also using natural hand translation to specify the target simply from a set of preselected objects.

For object manipulation, although some techniques are designed to make non-constrained

manipulation, only a few touch-based techniques allow making constrained transformation. Thus, our third objective was to propose a bimanual technique based on the asymmetrical model in order to improve the efficiency and fluency of constrained manipulation on TabletPCs. The designed technique allows using the non-dominant hand to specify the constraint and the dominant hand to determine the manipulation mode and control the transformation. To further explore the interaction possibilities, two other single-hand manipulation techniques using identified touch inputs were proposed. With the help of identified touch inputs, the vocabulary of classic 2D gestures was extended and each new gesture was mapped to a set of constrained manipulation operations.

Our last objective was then to evaluate and validate our design choices through conducting several experimental user studies. The evaluation studies demonstrate that they can improve the users' interaction experience on mobile devices. This permits to give some guidelines to improve the design of 3D interactions techniques on mobile devices.

The manuscript is divided into two parts: 'State of the art' and 'contributions '. The first part consists of three chapters. The first chapter presents some basic concepts and definitions required to understand the work of this thesis. The two other chapters present a State of the art on selection and manipulation techniques. The second part (contributions) consists also of three chapters. These chapters summarize all of our contributions to facilitate objects selection and manipulation in virtual environments on mobile devices.

Below, we summarize the content of each chapter.

In the first chapter, we will first present relative concepts of NUIs and mobile device interaction. We will give our own definition of NUIs, and present potential modalities that can be used to provide a natural interaction experience. Then we will present previous works which have been conducted to enrich the interaction possibilities on mobile devices.

In the second chapter, we will present the state-of-the-art techniques which are proposed for object selection. Selection techniques are divided into two groups according to the input dimension: 2D and 3D selection techniques. In the first group, we mainly present mouse-based and touch-based selection techniques proposed to simplify the selection of small objects and avoid the occlusion problem. Then, in the second group, we present 3D selection techniques which allow using the motion of handheld devices or freehand movements to select objects at a distance.

In the third chapter, we will present the state-of-the-art techniques for object manipulation. First, we will present how different 2D manipulation techniques can overcome the inconsistency between 2D inputs and 3D outputs. Then, we will present how 3D freehand movements or handheld devices can be utilized to manipulate objects in intuitive ways.

In the fourth chapter, we will present LayerStroke and LayerMenu, two selection techniques that are designed to make precise selection on Smartphones. The main contribution of these techniques is to reduce the cluster density by dividing the targets space into different layers. To make a selection, users should first select the layer in which

the target is located. LayerStroke allows drawing a stroke on the screen to specify a layer while LayerMenu permits calling out a menu of layer for layer selection. After the layer is selected, Voronoi tessellation is to be generated for all the objects inside the layer. The target can be selected by tapping simply its corresponding tile. A grouping algorithm is proposed to ensure that the tile area of each target is large enough for tap selection without occlusion. An experimental study was conducted to compare our techniques with four recent state-of-the-art acquisition techniques regarding the selection speed and accuracy. Results show that our techniques decrease both the selection time and errors rate.

In the fifth chapter, we will present HorizontalDragger, a freehand 3D selection technique which is designed to select objects at a distance. When the cluster density in the proximity of the cursor is low, our techniques can be used as the Bubble Cursor to select the target rapidly. However, if the density is higher than a threshold value and objects around the cursor are of small size, our techniques can automatically set a region of interest and allow dragging the hand horizontally, to specify the target from all the objects covered by the region. Because the effective width of each potential target is set to a constant value, the user can switch the focus in the region of interest in a predictable way. We have conducted an experimental study to compare our techniques with two state-of-the-art techniques. Although our technique did not provide the shortest completion time, it significantly decreased the errors rate.

In the sixth chapter, we will present our touch-based manipulation techniques which are designed to make constrained manipulation. We first present the design of the Constraint Menu technique and explain how two hands can be used simultaneously to manipulate objects. One advantage of this technique is that the non-dominant hand can be used without changing the holding posture. Moreover, object manipulation can be made without touching the target. Therefore, the performance is less affected by the object position, orientation and scale. We have made a controlled user study to compare the performance of this technique with two state-of-the-art techniques. The results show that our technique outperforms a Widget-based technique regarding both efficiency and fluency. Then, we propose two single-hand manipulation techniques, Classic3D and Single3D, which are developed based on the usage of identified touch inputs. In a user study, we evaluated the learnability of all three techniques. On the one hand, Constraint Menu required significantly less effort to understand its usage. On the other hand, participants commented that with the help of visual guides, Classic3D and Single3D could be learnt without facing great difficulties.

PART I: CONCEPTS AND STATE-OF-THE-ART

Chapter 1: Natural User Interface & Mobile Device interaction

Chapter 1: Natural User Interface & Mobile Device interaction

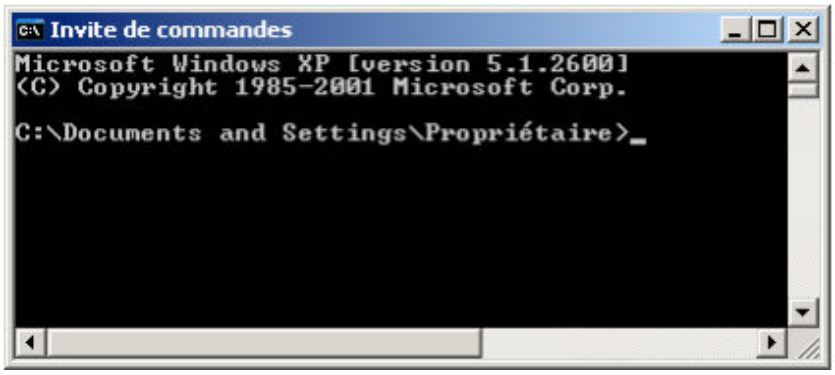
1 Introduction

The invention of the computer in the last century has completely changed the way people live and significantly boosted the development of the world. The evolution of computers during the last 50 years has contributed to reshape both the inside and outside of these machines. Moreover, the computing power increases continuously while the cost and size decreases without stopping. This transformation comes from the rapid development of technology that can be described by the Moore's law proposed in 1956:

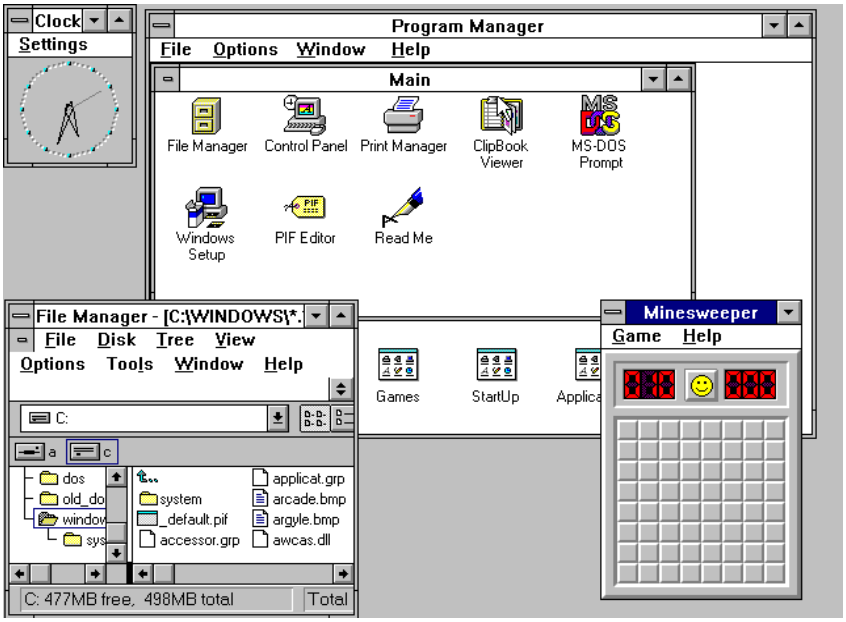
"The number of transistors that can be placed inexpensively on an integrated circuit will double approximately every two years."

Although computers were invented originally for scientists and researchers, nowadays they have already transcended national and cultural boundaries and they contribute to many aspects of our lives. The use of computers is not only restricted to the domain of scientific calculation and simulation of physical phenomenons, but also includes web browsing, playing video games, watching movies, etc. In addition, the emergence of laptop computers and mobile devices such as smartphones and tablePCs expands the usage of computers in terms of space. Now, it is possible to get access to enormous amount of data anywhere in the world.

In this context, while the computing power increases year after year, the development of user interfaces to communicate with computers seems to evolve far more slowly. As mentioned in (Wigdor, & Wixon, 2011), the development of interfaces has passed through phases. The evolution of the user interfaces began from the phase of the command-line interface (CLI) and then entered the era of the graphical user interface (GUI) (Figure 1). A CLI is a means of interaction where the user issues commands to the program in the form of successive lines of text while a common-use GUI relies on a set of user interface elements commonly referred to as WIMP: windows, icons, menus and pointers. GUIs significantly simplified computer interaction and acquired the dominance in the computer landscape quickly for several reasons. Indeed, based on the use of the mouse, GUIs reduce the barriers between users and computers by enabling users to directly manipulate digital data through the use of a cursor. Moreover, by leveraging the desktop metaphor to organize contents, the interaction process becomes less abstract. Furthermore, clicking an icon to open a file or start an application also turns out to be easier than typing commands.



(a)



(b)

Figure 1: Interface. (a) CLI; (b) GUI

The augmentation of computing power and the emergence of GUIs permitted to build interactive 3D graphics in computers. This powerful medium allows users to perform a lot of tasks in a digital 3D environment. For instance, simulation of real world can be done in computers for research purposes. In the domain of industry, products design largely relies on CAD software. Complex interactions within a digital 3D environment benefit from the high accuracy of the mouse. However, the mouse is a 2D input device and this led to problems of how to map 2D inputs to 3D operations. With the help of 3D transformation manipulators and keyboards shortcuts, it is still possible to perform basic 3D interaction tasks on GUI-based desktop computer. Commercial CAD software such as Maya (Autodesk), Blender (Blender Foundation) and Sketchup (Google) are all developed based on the mouse-based interaction paradigm. Besides CAD software, the same paradigm is used in 3D video games such as World of Warcraft (Blizzard Entertainment).

After the dominance of GUIs during almost 30 years, here comes the era of the next generation of interface. Indeed, since Apple launched the iPhone in the market in 2007, the touch-based paradigm was introduced and quickly reshaped the interaction techniques for mobile devices such as smartphones and tablePCs. After the emergence of the touchscreen, other new interaction devices such as the Kinect, the Leap Motion Controller and the Google Glass appeared one after another. Because there is a great difference between these new interaction devices and the mouse, they are not well suited for use with traditional GUIs.

Natural User Interfaces (NUIs) have gained momentum during the last decade and are considered as the next generation of User Interfaces. Indeed, they are expected to provide a more intuitive and seamless interaction experience than traditional UIs by taking advantage of learnt skills of human beings.

3D interaction is one of the domains that are being reshaped by the development of NUIs. In many interaction scenarios, it is inconvenient or even impossible to use the mouse and keyboard for some tasks. The emergence of new input and output devices brings the hopes that new interaction techniques, more adapted for this context of use, can now be designed and developed. For instance, in some immersive environments, users can use freehand gestures to control 3D objects, while in some outdoor environment users may need to use a touchscreen for interaction. The advances in the field of NUIs can provide more and more guidelines for designers to develop efficient and easy-to-use interaction techniques for 3D interaction.

Nowadays, much attention is attracted to build 3D interaction on mobile devices. For instance, architects and industrial designers may find it necessary to make quick sketches on tabletPCs in outdoor environments when inspiration comes. Moreover, augmented reality applications require efficient and walk-up-and-use techniques to manipulate augmented content in the real space.

Because many sensing devices have already been embedded into mobile devices, interaction designers can combine a set of interaction modalities to design advanced manipulation techniques.

In the following sections of this chapter, we will present the definition of NUIs, mobile devices, and also discuss related aspects of these two concepts.

2 Natural User Interface (NUI)

The concept of NUIs was first presented in the work of Steve Mann in the 1990s (Mann, 2001). By using interaction skills involved in communication with the real world, he developed a number of user-interface strategies as substitution of CLIs and GUIs. Although NUIs are widely discussed and mentioned, the term of NUI is often confusing and misinterpreted. Through the literature, several definitions and discussions about NUIs can be found.

Blake gave a definition of NUIs in his book "Natural User Interface in .NET" (Blake, 2011):

"A natural user interface is a user interface designed to reuse existing skills for interacting directly with content."

Another definition of NUIs is given by Buxton (Buxton, 2010):

"An interface is natural if it exploits skills that we have acquired through a lifetime of living in the world."

Valli (Valli, 2005) gave his own definition about the research work of natural interaction:

"Research work in natural interaction is to invent and create systems that understand these actions and engage people in a dialogue, while allowing them to interact naturally with each other and the environment."

Both the definition of Blake and Buxton mention the reuse of skills that people have already acquired and are also familiar with. Skills that are used by NUIs can be divided into two kinds: innate skills and postnatal skills. Innate skills include those born abilities that people get from heredity while postnatal skills are those learnt from the natural and social environment through growth. Since performing innate and postnatal skills requires much less cognitive effort than unfamiliar skills, reusing them for interaction can help users concentrate on the interactive content.

One misunderstanding of NUIs is that natural interaction can be provided only by assembling familiar skills haphazardly. Blake emphasizes the specific planning efforts for the design of the interface (Blake, 2011). Skills reused for interaction should be appropriate with contents. Several aspects of interaction such as efficiency, directness and simplicity should all be considered during the design phase. Another misunderstanding related to NUIs comes from the fact that natural interfaces can be provided once some specific technologies are used. The technologies used do not define the nature of a system. Technology is just a tool for creating communicative space or artifacts. A technology which is appropriate for one scenario may not ensure natural experience in another case. Natural interaction is more about how users feel about the interaction experience (Wigdor, & Wixon, 2011). A user interface cannot be considered natural only if natural perception of human beings is involved.

Based on the definitions mentioned above, we can give a new definition of NUI:

"A natural user interface is a user interface which maps learnt skills to interaction operations in an appropriate way."

According to this definition, NUIs should have two main characteristics. First, as mentioned above, learnt skills should be used. Chosen skills should be easy to reproduce without paying high mental and physical cost. For example, when designing hand-based

gestures, the anatomy structure of the hand should be considered in order to avoid proposing tiring and uncomfortable gestures. Second, an appropriate metaphor should be constructed between reused skills and operations. The metaphor of interaction signifies that an object or a real concept is used as a virtual tool to interact with the virtual environment (Sternberger, 2006). An interaction metaphor regroups a set of interaction techniques which use the same virtual tool or the same concept to interact with the virtual world. If the semantic meaning of the chosen skills corresponds to that of mapped operations, users can reproduce and memorize the operations with low learning costs. For instance, if the 1-finger dragging gesture is mapped to the object translation, it is an appropriate metaphor because the finger movement is the same as that of the object. However, the 1-finger rotating gesture is a less appropriate metaphor for translation because it is less intuitive to map circular finger movement to object translation. It may require more mental efforts and more time for users to get familiar with this mapping.

Although user interaction is always used as a synonym of user interface, there is a slight different between these two terms. A natural user interface is a set of hardware and software that enables users to interact with an interactive system in a natural way. It is in fact, a medium between the user and the digital content. However, a user interaction technique defines the process that users can follow to use the interface and interact with the system. The interaction technique determines how users can make inputs and how machines can return outputs back to users. It describes the relationship between people and the machines they use. Thus, we can define a natural user interaction technique as:

"A natural user interaction technique defines how learnt skills are mapped to operations and how users can use them to interact with a natural user interface."

Besides mapping acquired skills to interaction operations, the interface should have some characteristics to provide natural experience. Valli states that one purpose of a natural interaction framework is to **remove any level of mediation between the person and the object of his action** (Valli, 2005). Unlike GUIs, NUIs should be more invisible and more seamless in order to reduce the barrier between users and computers. Traditional devices such as the mouse and the keyboard are less preferred for natural interaction. The user himself becomes the interface and multiple types of perception are used to communicate with the computer. For example, the body can serve as a spatial reference and hand and foot gestures can be used to control digital contents. People can manipulate digital contents in the same way they experience in the real world.

Another characteristic is that **users can have more liberty when using NUIs**. One reason that GUIs are more user-friendly than CLIs is that users can control the cursor in a free way to translate across the working desk or to open a file. Users are not forced to retain plenty of commands to manipulate digital data. In addition, GUIs provide more visual and auditory feedback while outputs of CLIs are displayed only after a command is entered. In the same way, NUIs should provide more liberty to interact with the computer. Users should not be forced to learn a complex set of constraints and the interaction methods could be learnt easily after few trials. It is preferred that multiple interaction modalities can be provided so

users can choose the most favourable and the appropriate way for interaction. Furthermore, abundant feedback should be provided to encourage and also guide users to find out the right way to enjoy the interaction.

2.1 Natural Interaction Modalities

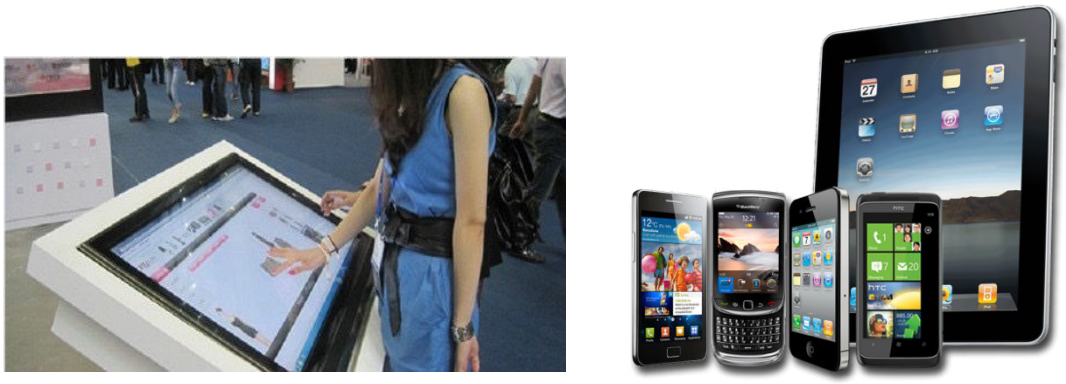
Unlike GUIs, which are mostly implemented in desktop computers, GUIs can be developed based on multiple modalities and have various forms under different contexts and interaction tasks (Djelil et al., 2013). However, there is no modality that can be adapted to all interaction scenarios. For example, touch-based interaction paradigms are appropriate when playing video games on a smartphone, while it may distract users' attention when driving. How to choose the appropriate technology relies on both the interaction environment and the scenario. Each modality relies on difference perceptional abilities so that they can be used for different purposes.

2.1.1 Tactile interaction

To make inputs, tactile interaction allows the use of direct hand contact on a touchpad, which can be a digital touchscreen or a physical augmented surface. Common digital touchscreens in the market include resistive, capacitive and infrared touchscreens (Bhalla & Bhalla, 2010). There are also touchscreens based on acoustic pulse recognition. In the research field, physical surfaces can also serve as touchscreens with the help of external image sensors (Wilson, 2010). Both planar surfaces and curved surfaces of the arm or even physical objects with more complex form can be transformed into a touchpad (Harrison, Benko, & Wilson, 2011).

When a hand or fingers are pressed on a touchpad, information such as the contact shape, the count, and the position of contact points can be detected. This set of abundant information can be used to design interactive gestures so that users can perform pre-defined gestures for various operations. Two types of contact were identified by Ghomi et al.,: static contact regarding configuration of fingers or hands held on the touchpad, and dynamic contact, which is described by finger and hand movements (Ghomi et al., 2010). In additional to interaction by fingertips, different finger parts and even the whole hand can be used to interact with the system (Wu & Balakrishnan, 2003).

Tactile interaction has several advantages. First, because gestures are commonly involved in physical objects manipulation or human-human communication, appropriate metaphors can be constructed between familiar gestures and interaction operations. Second, gestures allow users to control digital content in a more direct way. Hands and fingers serve as the interface so that it is unnecessary to use any additional device as an interactive medium. Third, multiple gestures can be designed by leveraging abundant information captured by the touchpad. Nowadays, digital touchscreens are widely used as guiding tools in public places such as shopping malls and hospitals (Figure 2 (a)). They are also used on mobile devices such as smartphones and tabletPCs as well (Figure 2 (b)).



(a) (b)

Figure 2: Touchscreens.

2.1.2 Tangible interaction

Tangible interaction enables users to take control of digital information through the physical environment. The purpose of tangible interaction development is to empower collaboration, learning, and design by giving physical forms to digital information, thus taking advantage of human abilities of grasp and manipulate physical objects and materials (Ishii, 2008). Because people are familiar with and sensitive to the physical representation of the environment, coupling the physical world with the digital realm enables users to access digital information in a more concrete way.

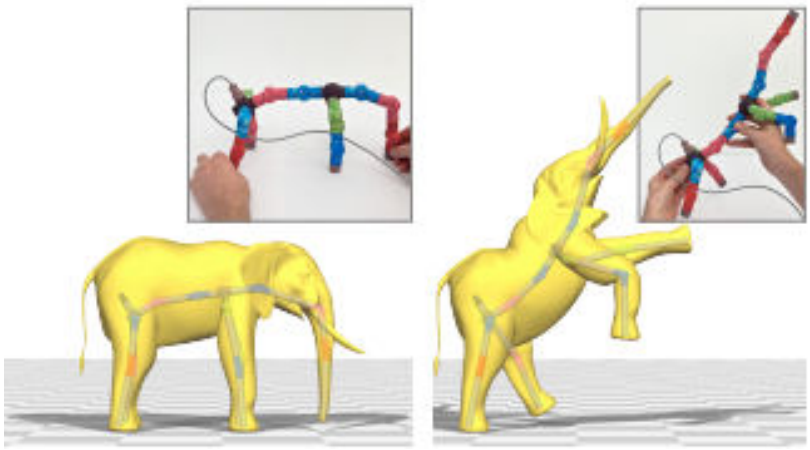


Figure 3: The elephant character can be manipulated by using a set of connected tangible devices (Jacobson et al., 2014).

One way to design tangible interfaces is to use the interactive widget as a proxy of the manipulated context. Jacobson et al. have proposed a tangible and modular input device for

character articulation (Jacobson et al., 2014). As shown in Figure 3, a 3D elephant character consists of many body parts and each of them can be controlled independently. In this work, the user can put together a set of hot-pluggable devices to represent the elephant and each of the devices serves as a proxy of one body part. When the position and orientation of one device is changed, the same transformation is reproduced synchronically on the body part of the character.

Besides manipulating virtual 3D objects, changes of the state of physical widgets can be transferred as inputs for many other operations. Weiss et al. have developed SLAP widgets which can be manipulated on touchscreens to bring the gap between virtual and physical controls (Weiss et al., 2009). As shown in Figure 4, a SLAP Knob user interface can be rotated on the surface to select an item in a circular menu, adjust values, or step frames in videos. Because people are familiar with using a knob to adjust the volume of the radio, reusing the same gesture offers a user friendly method to interact with the surface. The use of tangible interfaces also make the interaction more perceivable with the help of haptic feedback.

Figure 4: SLAP Knob user interface. a) Selecting image property from menu; b) Setting continuous value; c) Relative navigation for frame stepping in videos (Weiss et al., 2009).

2.1.3 Freehand interaction

Freehand interaction enables users to use freehand gestures in air to make interaction. The common point among freehand interaction, tactile interaction and tangible interaction is that they are based on using hand poses and motions. However, unlike the two other modalities, freehand interaction is not restricted by a touchpad or an interactive widget. Hands motions can be done freely in the sensing field of the sensor that is used for hand tracking.

On the one hand, freehand interaction provides a larger design space since hand movements in all 3 dimensions can be used. On the other hand, it brings more technical and ergonomically challenges to design natural experiences. The first issue is the lack of haptic feedback. Without a physical support, it is more difficult to make precise inputs using freehand gestures than simply dragging the fingers on a touchscreen. More concentration of users is needed in order to control the hand motion and reduce the influence of tremble. One possible solution is to use filters to remove the input noises (Vogel & Balakrishnan,

2005, Casiez et al., 2012). Another approach consists of adjusting the Control-display ratio when precise control is necessary (Nancel et al., 2011a, Nancel et al., 2013). The second issue is the difficulty of gesture recognition. A gesture on the surface begins when contacts are detected and ends when hands or fingers are released. However, freehand gestures do not have such explicit delimiters and there is a lack of efficient ways to segment meaningful gestures from hand trajectories. Some studies solve this issue by adding a device held in the dominant or the non dominant hand. Users can use the physical buttons on the device as a gesture delimiter. Chen et al. proposed using freehand gestures to interact with smartphones (Chen et al., 2014c). They used a touch input performed before or after a freehand gesture as an explicit delimiter. The third issue is the hand tracking precision.

(a) (b) (c)

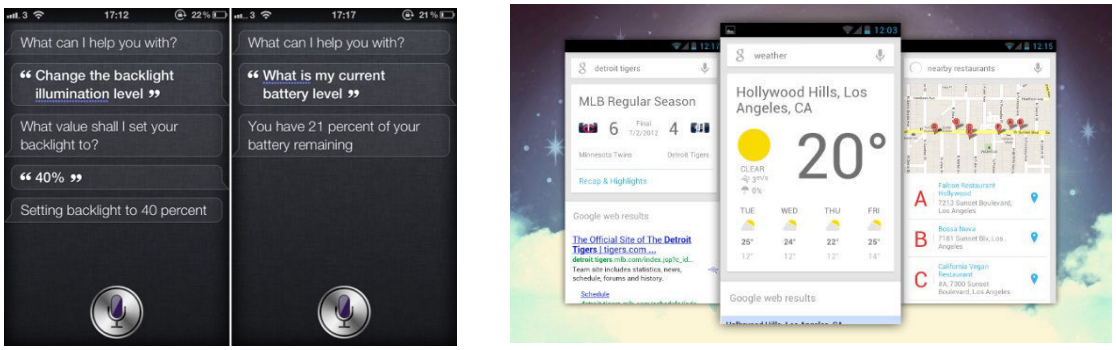
Figure 5: Gesture detecting devices. (a) Kinect; (b) Leap Motion Controller; (c) MYo.

Common solutions in early studies used motion sensor based or optical based embedded tracking devices for hand tracking. Using embedded tracking devices can provide accurate result but the user experience is degraded. Hand tracking can also be done using techniques of computer visions, but the result is always limited by the occlusion problem. Recently, some low-cost devices such as the Kinect (Microsoft) and the Leap Motion Controller permitted to track hand motions more accurately by using stereoscopic camera techniques (Figure 5 (a) (b)). MYO is an armband that can capture hand gestures by measuring micro currents passing through muscles (Figure 5 (c)). With the help of all these new devices it is now possible to design promising freehand interfaces.

2.1.4 Voice

Voice interaction enables users to communicate with the computer by speaking. As human-human communication is frequently done through verbal communication, if computers can understand natural languages of users and even respond in a similar way as human beings, the interaction experienced can be highly efficient and natural as well. Some early studies that explored voice interaction only provided a vocabulary for interaction (Cohen et al., 1997, McGlashan, 1995, Nugues et al., 1996). However, instead of natural language, these studies only allow using restricted language to communicate with machines (McGlashan, 1992). Nowadays, a big leap of voice recognition can be seen and some more intelligent voice assistance interfaces, such as Apple's Siri (Figure 6 (a))

and Google Now (Figure 6 (b)) are already launched in the market. It is possible to use natural language to give an instruction and the interface can either search for responses of the question or just speak with the user in natural language as well. Voice interfaces can be very useful when hands are already used for other tasks or when the meaning of the instruction can be only expressed by a complex phrase.



(a) (b)

Figure 6: Voice interface. (a) Siri; (b) Google Now.

2.1.5 Gaze

The gaze can also be used for interaction. When performing interaction with the computer, the gaze is frequently used for visual searching of interesting displayed content or to ensure that a manipulation is done correctly. For example, when a cursor is moved on the screen, before moving the mouse, users often first change the gaze to locate visually the cursor destination.

Interaction through the gaze is one of the most interesting natural modalities, especially for disabled persons (Lim & Kim, 2012). For people suffering motor and language disorders, the gaze can be used in substitution of handheld devices (Pfeiffer, 2008). Some other studies have been done to explore the gaze for text typing (Ding et al., 2009). Besides assisting disabled persons, the gaze can also be used for selection and manipulation of digital content displayed in a large display located at a distance.

One potential issue when using the gaze for active interaction is that it is hard to use the eye movement as triggers of specific operations. Although the eye can be closed and opened, it is not an ideal option because blinking is a natural behavior that cannot be avoided. Lee et al. tried using the gaze to select digital contents in an augmented reality environment (Lee et al., 2014). For object selection, an algorithm was proposed to detect half blink and this eye motion is used as a trigger. Some other projects enable users to move the cursor on the display by the gaze, but other modalities are used to spark operations (Fares et al., 2013; Turner et al., 2014).

Some studies have exploited the use of the gaze in a passive way by combining it with

other modalities. The gaze is used mainly for predicting users' actions to provide better real-time rendering or assistance for interaction while other more explicit modalities are used to improve the determinism of interaction (Oviatt & Cohen, 2000, Zhai et al., 1999). In the work of (Turner et al., 2011), if users set the gaze at one object when touch inputs are made, the object can be picked up. After setting the gaze at another position, tapping the screen permits to drop the object to that position (Figure 7).

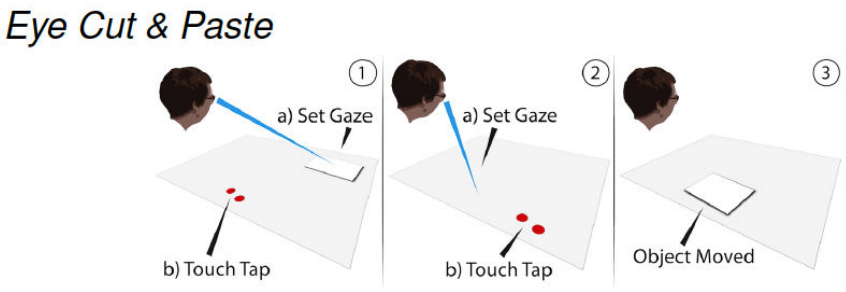


Figure 7: Gaze interaction (Turner et al., 2011).

2.1.6 Body movements

Similar to hand movements, body movements can also be used as inputs for interaction. Interaction based on body movements is widely exploited in virtual reality and video games. All parts of the user's body, mainly the head, hands and feet are tracked to manipulate objects in the virtual realm, or to reproduce the movement of the body through an avatar (Wren et al., 1997a). The work in (Hasenfratz et al., 2004) can capture body movements in real time to reconstruct and animate virtual characters and virtual objects corresponding to physical objects manipulated in the real world. Interaction tasks such as navigation and application control can also be performed using body movements (Wren et al., 1997b).

Actually, the body itself can be used as an interface to communicate with the machine. In the work of (Wagner et al., 2013), a body-centric design space has been explored. Eighteen body targets have been grouped into five categories and a user study has been conducted to examine the comfort and efficiency of using different body parts as interfaces. The space around the body can also be used to extend the interaction space of mobile devices (Chen et al., 2012). For example, a specific operation can be triggered automatically by moving the mobile device close to a body part.

Some new applications of body movements have also been explored. The work in (Schwarz et al., 2014) tried using the body pose, gaze and gestures of users to capture their attention of the interface (Figure 8). The authors have proposed a technique to recognize whether a body gesture is made for interaction or just for human-human communication. By combining body information with the pattern of a wave gesture, an algorithm is

proposed in (Hayashi et al., 2014) to identify the user in front of the camera. It enables each user to use the same gesture to sparse self-defined operations.

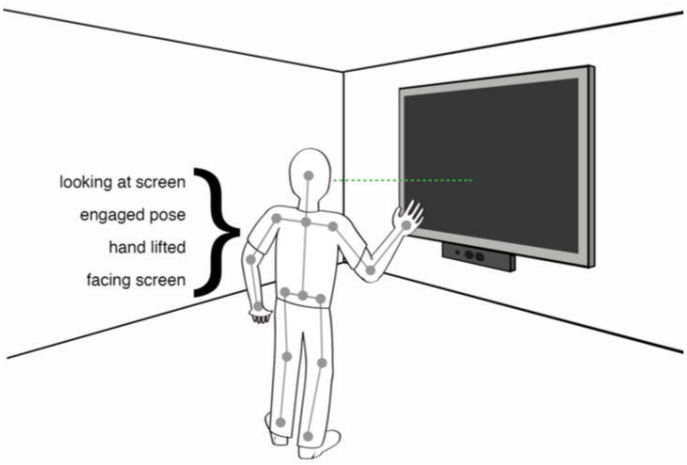


Figure 8: The intension of a user to interact with a vision-based interface can be found by the considering a set of features (Schwarz et al., 2014).

2.1.7 Lip movements

The lip movement is not used to develop an entire set of interaction methods independently. Indeed, it is mostly used to improve recognition of the speech signal, especially in noisy environments (Bregler et al., 1997). However the lip movement can still be applied to build a powerful virtual actors animation engine (Blake & Isard, 1994). In this context, a system prototype of avatars animation which exploited lip movements, has been established (Bondy et al., 2001).

A Japanese research team had also shown that this modality can provide a very powerful means of interaction for elderly and disabled persons. An interface prototype based on lip movements has been tested for robot control (Figure 9).

Figure 9: Command of a robot through the mouth movement.

2.1.8 Brain activities

Sending commands through the brain activity is another means of natural interaction in virtual reality. Several studies have shown that through a brain-computer interface, it is possible to move, select and manipulate virtual objects (Lotte et al., 2006). This is possible thanks to an electroencephalograph device placed on the scalp of the user. This device can measure micro-currents in the brain to determine, for instance, concentration levels of the user. This was used in a game called Mindflex where the user can move a ball on a plate by varying the level of mental concentration (O'Hara et al., 2011).

Figure 10: Simulation of control of a wheelchair in a virtual environment with the help of a brain-computer interface (Leeb et al., 2012)

The use of this interface in virtual reality has been experienced by Friedman et al, where two tasks were performed in two separate virtual environments (Friedman et al., 2004). The first task was to change the point of view of the user in a virtual bar, and the second consisted of moving in a virtual road. The user had to imagine the movement of his feet to move in the street, and the movement of his right hand to stop.

One of the typical applications that rely on this interface is the video game Mind-Balance which was used to control the movement of a 3D character in a virtual scene (Lécuyer et al., 2008). Another application was designed for controlling the simulation of a wheelchair in a virtual street populated with avatars (Figure 10) through imagination of feet movements (Leeb et al., 2012). During the simulation, the person stopped to listen to each avatar met on his way.

2.1.9 Facial expressions

Facial expressions are used in several applications of human-computer interaction, simulation of human expressions by virtual avatars is one of these applications (Wei & Hu, 2011). Facial expression was also combined with the speech signal in order to study its

influence on the voice modality (Busso & Narayanan, 2007). It was thus demonstrated that the expression of the face has a large influence on the content of the voice message (Figure 11).

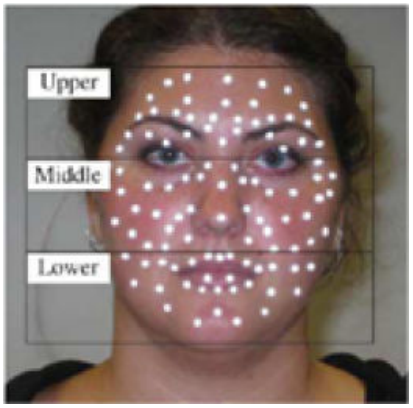


Figure 11: Subdivision of the facial surface in three levels, the expression is tracked using markers attached to the face (Busso & Narayanan, 2007).

3 Mobile devices interaction

Before the release of iPhone in the market, interaction with most of the mobile devices such as mobile phones and Personal Digital Assistants (PDAs) relied on the use of keyboards, keypads or styluses (Figure 12). The emergence of capacitive touchscreen reshaped mobile device interaction and also had a knock-on effect on smartphone form factors. The touchscreen enables users to select and manipulate digital contents on mobile devices in a more direct way. It also helped Apple and Samsung overthrow the dominance of early mobile device producers such as Nokia and Motorola.

Although touchscreens promise a more intuitive interaction experience, they also have some inherent problems. The first one is that the accuracy of touch-based interaction is limited. Compared with mouse-based interaction, it is more difficult to tap a finger precisely on a specific position on the screen. When a finger is pressed on the screen, the tap position is occluded by the fingertip so that users cannot visually check whether the finger is located in the desired position. One option could be to have larger size for UI widgets on mobile devices to ensure selection precision. However, this remains a challenge because the size of screen is very limited. The second issue is the limited gesture vocabulary. On keyboard-based and stylus-based mobile devices, to switch among different operations, users can either select a desired option in an operation menu or press a specific key on the keyboard. However, relying excessively on menus and buttons goes against the touch-based interaction paradigm since it is desirable to provide more direct interaction experience. As touchscreens can capture touch information such as count and positions of fingers, it is possible to design gestures using different hand and finger motions performed on the screen. For example, gestures such as 1-finger drag, 2-finger rotate and 2-finger pinch gestures are widely used to pan, rotate and zoom in/out a map.

However, the number of gestures supported by commercial applications is limited. This is a design tradeoff between functionality and simplicity. Although it is possible to map more gestures to numerous functions, it is inevitable to increase the mental cost of users to learn and memorize a larger gesture glossary. For general users of mobile devices, existing fundamental gestures seem to be powerful enough for common usage and a large glossary may weights too much for them. Nevertheless, it is still worth exploring how the vocabulary can be enriched and how touch-based interaction can be used for more complex work. A lot of studies have been done for this purpose. We will present them in the following section.

(a) (b) (c)

Figure 12: Traditional mobile device interaction. a: Keyboard; b: Keypad; c: stylus.

3.1 Gesture

3.1.1 2D touch-based gesture

Although a lot of studies have been done to design gestures that provide natural interaction, it seems that most of the work do not provide a clear definition of gestures and the authors just employ an implicit notion of this term. A few studies provide an insight into gestures used for interaction. Baudel and Beaudouin-Lafon state that a gestural command can be described by a start position, a dynamic phase and an end position (Baudel & Beaudouin-Lafon, 1993). The start and end positions refer to positions of a hand in the space and also include the notion of finger/wrist configurations. However, their concept only concerns gestures performed with dynamic hand motion but not related to steady ones.

Wu et al. propose a set of design principles for constructing multi-touch gestures: gesture registration, gesture relaxation and reuse of gestures and tools (Wu et al., 2006). Similar to the idea of Baudel and Beaudouin-Lafon, the gesture registration phase is used to set the context for the dynamic and end phases. Gesture registration serves as delimiters when a tool is switched from one interaction style to another. It is pointed out that gesture registration can take two forms: it can be done either by holding the hand static or using

both the hand configuration and dynamic actions that occur after the posture is recognized. The principle of gesture relaxation suggests that after registering a gesture, the hand configuration can be altered so that the gesture can be performed more freely. To reduce the memory burden, the authors suggest compounding primitives such as hand configuration and dynamics to design gestures. They also mention that a gesture can be continuous or discrete, probably referring to both dynamic and static gestures.

Table 1: Taxonomy of surface gestures (Wobbrock et al., 2009).

Taxonomy of surface gestures		
Form	Static pose	Hand pose is held in one location.
	Dynamic pose	Hand pose changes in one location.
	Static pose and path	Hand pose is held as hand moves.
	Dynamic pose and path	Hand pose changes as hand moves.
	One-point touch	Static pose with one finger.
	One-point path	Static pose & path with one finger.
Nature	Symbolic	Gesture visually depicts a symbol.
	Physical	Gesture acts physically on objects.
	Metaphorical	Gesture indicates a metaphor.
	Abstract	Gesture-referent mapping is arbitrary.
Binding	Object-centric	Location defined w.r.t. object features.
	World-dependent	Location defined w.r.t. world features.
	World-independent	Location can ignore world features.
	Mixed dependencies	World-independent plus another.
Flow	Discrete	Response occurs after the user acts.
	Continuous	Response occurs while the user acts.

Instead of proposing a clear definition of gestures, Wobbrock et al. (Wobbrock et al., 2009) establish a taxonomy of surface gestures to describe the gesture design space. Gestures are classified according to: form, nature, binding and flow (Table 1). The form considers whether a hand gesture is static or dynamic regarding both the hand pose and the hand motion. The nature describes whether a gesture visually depicts a symbol, acts physically on object, indicates a metaphor, or the gesture-referent mapping is arbitrary. The binding describes the gesture location with respect to the object or world features. The flow can be discrete or continuous, which means the response occurs after or while the gesture is performed.

Isenberg and Hancock distinguish between postures, quasi-postures and gestures (Isenberg & Hancock, 2012). They define a touch gesture as:

"a way to invoke manipulations in a direct-touch environment that is started by touching the surface in a well-defined initial configuration and that is continued for some time in a

well-defined motion pattern (incl. the null motion) during which the configuration may change."

Previously mentioned notions such as configuration and a dynamic phase (Baudel & Beaudouin-Lafon, 1993, Wobbrock et al., 2009) as well as gesture registration and relaxation (Wu et al., 2006) are involved in this definition. They consider that gestures are used mainly for discrete operations whose responses occur after gestures are recognized. For operations which can be done in a more continuous way, they argue that postures and quasi-postures are more favorable. In the literature, the notion of posture is always used to indicate static hand pose. However, different from its common use, in this work, a touch posture is defined as:

"a way to invoke manipulations in a direct-touch environment that is characterized by touching the surface in a well-defined initial configuration whose effect can be parameterized by a subsequent dynamic action."

The essential difference between gestures and postures is whether the dynamic action characterizes the manipulation or not. For both gestures and postures, the concept of configuration includes the count of fingers and their position, the contact shape, and touch IDs. Inspired by the concept of binding in (Wobbrock et al., 2009), touchable widgets such as buttons or even specific screen areas can also be considered as parts of a configuration. And a quasi-posture is described as:

"a posture whose initial configuration is augmented with a brief initial dynamic action but where this action's continuation is also used to parameterize the effect."

3.1.2 3D freehand gestures

Besides touch-based gestures, 3D freehand gestures can also be used for interaction. Similar to touch-based gestures, 3D freehand gestures map hand movements in space to specific operations to perform in the computer. Because freehand gestures are widely used to manipulate physical objects and animate speeches, it is a natural step to explore their usage for mobile device interaction. However, compared to touch-based gestures, it is more difficult to determine the beginning and end of a freehand gesture and to distinguish a given gesture from another or just from casual movements (Ren & O'Neill, 2013). To solve the issue of gesture delimiters, several solutions have been proposed. In some studies, specific hand postures (Song et al., 2012) or the pinch posture (Grossman et al., 2004, Guimbretière & Nguyen, 2012) are used as explicit gesture delimiters. Several studies take use of buttons on a handheld device as gesture clutch (Jones et al., 2012, Nancel et al., 2011a). Spatial hand movements are considered as inputs only when the clutch button is pressed. Chen et al. have designed a similar strategy to use a touch event before or after the freehand gesture, as a delimiter (Chen et al., 2014c).

The definition of touch-based 'gestures' and 'postures' proposed by Isenberg and Hancock can be extended for 3D freehand gestures (Isenberg & Hancock, 2012). We define a "3D

gesture" as:

"a way to invoke manipulations in a freehand interaction environment that is started by making a well-defined initial configuration and that is continued for some time in a well-defined motion pattern (incl. the null motion) during which the configuration may change."

Similarly, a "3D posture" can be defined as:

"a way to invoke manipulations in a freehand interaction environment that is characterized by making a well-defined initial configuration whose effect can be parameterized by a subsequent dynamic action."

For 3D freehand gestures, the concept of the initial configuration includes all the behaviors that can be used as a gesture delimiter. For example, the initial configuration can be the performance of a specific clutch hand or finger pose or pressing a button on a hand held device. Other modalities such as head movements or voice commands can also be used as the initial configuration of a 3D gesture and posture.

Table 2: Taxonomy of motion gestures for mobile interaction (Ruiz et al., 2011).

Taxonomy of motion gestures		
Gesture mapping		
Nature	Metaphor of physical	Gesture is a metaphor of another physical object
	Physical	Gesture acts physically on object
	Symbolic	Gesture visually depicts a symbol
	Abstract	Gesture mapping is arbitrary
Context	In-context	Gesture requires specific context
	No-context	Gesture does not require specific context
Temporal	Discrete	Action occurs after completion of gesture
	Continuous	Action occurs during gesture
Physical characteristics		
Kinematic impulse	Low	Gestures where the range of jerk is below $3m/s^3$
	Moderate	Gestures where the range of jerk is between $3m/s^3$ and $6m/s^3$
	High	Gestures where the range of jerk is above $6m/s^3$
Dimension	Single-Axis	Motion occurs around a single axis
	Tri-Axis	Motion involves either translational or rotational motion, not both
	Six-Axis	Motion occurs around both rotation and translational axes
Complexity	Simple	Gesture consists of a single gesture
	Compound	Gesture can be decomposed into simple gestures

3.1.3 Device motion gesture

Besides the 3D freehand gestures, the device motion can be considered as an input. Nowadays, it is common to have an accelerometer, a gyroscope and a GPS inside a mobile device. This set of sensors permits to capture the position and orientation transition of the device and promises a new way for interaction. Similar to the work of Wobbrock et al. (Wobbrock et al., 2009), Ruiz et al. have made a study by asking several participants to design motion gestures for mobile interaction (Ruiz et al., 2011). Through the experiment, they have proposed a motion gesture taxonomy. Motion gestures can be described by two kind of features: gesture mapping and physical characteristics (Table 2). Gesture mapping describes how motion gestures are mapped to device commands regarding three features: the nature, context and temporal dimension. While these three dimensions correspond the nature, binding and flow dimensions in the taxonomy shown in Table 1. Physical characteristics include the spatial aspects of gestures: kinematic impulse, dimension and complexity.

3.2 Interaction possibilities

In this section, we will present several strategies for extending interaction possibilities on mobile devices.

3.2.1 Symbol-based gestures

One common way to enrich the touch glossary is to use symbol-based gestures as operation shortcuts. Users can press the finger on the screen and draw a pre-defined symbol to trigger a corresponding operation. A symbol can be a letter, a number, a geometrical graph or even an abstract figure. Wobbrock et al. have proposed $1 Recognizer (Wobbrock et al., 2007) (Figure 13) and $N Recognizer (Anthony & Wobbrock, 2012) for uni-stroke and multi-stroke gesture recognition.

Vatavu et al. have presented another technique which is called $P Recognizer (Vatavu et al., 2012). Unlike the two previous ones, this technique represents a gesture as an unordered point cloud. Li has proposed Gesture Search which allows users to search data by drawing gestures on the screen (Li, 2010). It is similar to a mouse-based technique which uses stroke gestures as shortcuts to menu selection (Appert & Zhai, 2009). The fact that Gesture search was assessed positively by users demonstrates that using gesture can facilitate users' access to data.

Wu et al., have proposed a visual query language for geographic information system (GIS) (Wu et al., 2013). Their interface allows users to draw some symbolic graphical objects and use their topological relationship to express the query intention. Compared with traditional SQL language, this visual query language is less abstract and can be learnt quickly even for novice users of GIS.

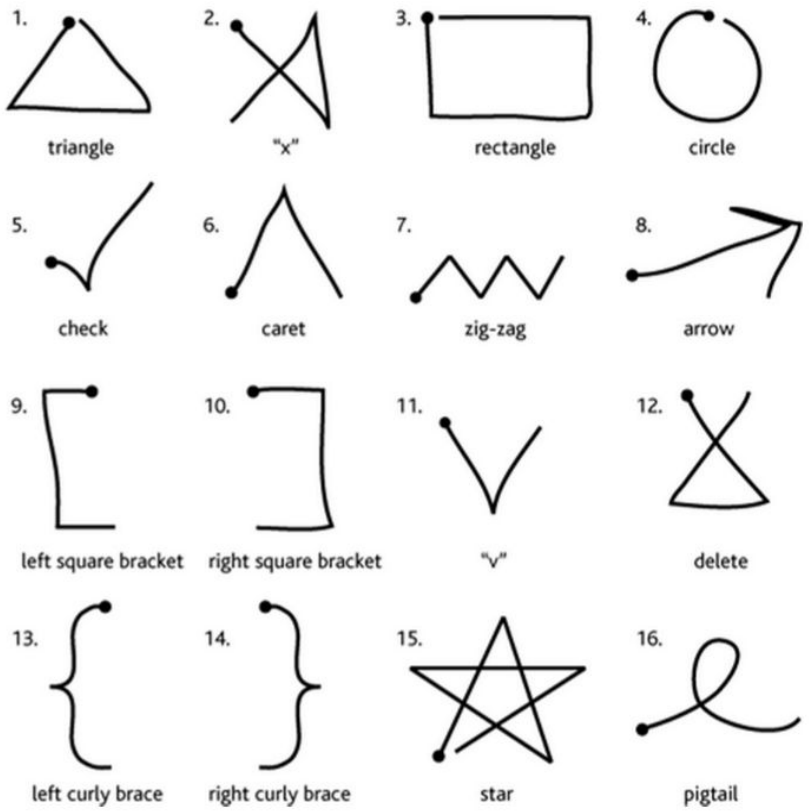


Figure 13: Example of gestures that can be recognized by $1 Recognizer (Wobbrock et al., 2007).

3.2.2 Augmented thumb gestures

In some interaction scenario, users have to hold and interact with the mobile device by only one hand. As a result, only the thumb of the hand holding the device can be used for interaction. To make the thumb more productive, Roudaut et al. have proposed MicroRoll gestures to enrich the vocabulary (Roudaut et al., 2009). This technique allows users to perform tiny thumb movements to trigger different operations. This set of gestures can be precisely distinguished from traditional dragging and swiping gestures by analyzing the thumb trajectory on the screen. A user study demonstrated that these gestures can be performed without difficulties and can simplify the interaction process. Similarly, ThumbRock, a micro gesture that consists in rolling the thumb back and forth on the screen, was proposed (Bonnet et al., 2013). This gesture can be used as a substitute of the long tap gesture or other gestures which require the use of two fingers to improve the interaction efficiency. Boring et al. have proposed a similar technique which is called Fat Thumb (Boring et al., 2012). When the contact size of the thumb is small, dragging the finger can pan the map. However, the map can be zoomed in around the contact point when the contact size increases (Figure 14).

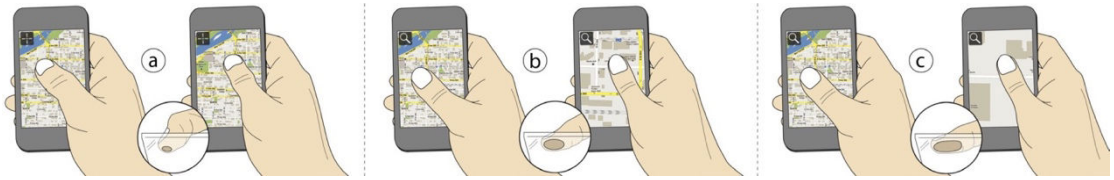


Figure 14: Walkthrough of pan and zoom example: movement with small contact size pans the map (a), increasing contact size switches to zoom (b), and further increasing the contact causes faster zoom operation (c) (Boring et al., 2012).

3.2.3 Finger pose based gestures

In addition to finger position, finger pose can also be used for gestures design. By using a vision-based touchscreen, Wang et al. have developed a technique to detect finger orientation (Wang et al., 2009). When the user is tapping a finger on the screen, the change of the contact region is analyzed dynamically to infer the orientation. Some interaction techniques which rely on finger orientation were proposed. For example, fingers can be rotated around the contact point to make selection in a circular menu or adjust the value of an orientation-sensitive button. Rogers et al. have proposed an algorithm for detecting both finger yaw and pitch rotation on a capacitive prototype (Rogers et al., 2011). The authors have also imagined some applications of finger orientation (Figure 15). Private messages can only be displayed when fingers are tilted at a specific angle, and finger rotation can be used to make 3D map navigation. Kratz et al. have installed a depth sensor on a mobile device and used the captured point cloud to reconstruct the finger model (Kratz et al., 2013). Two evaluation studies have shown that their technique was reliable for detecting finger rotation and tilt on the screen.

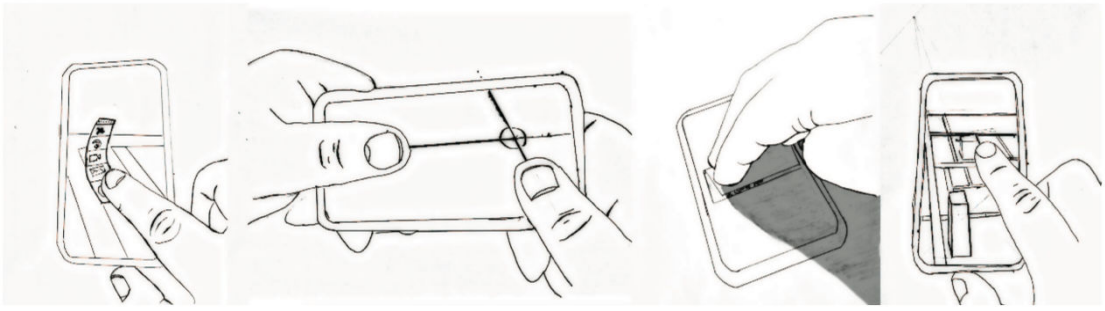


Figure 15: Interaction concepts using tilt-sensitive touch sensing on mobile devices. Left-to-right: (a) Rolling context menus; (b) High-precision bimanual pointing; (c) Peeking through message flaps using whole hand pitch angle detection; (d) 3D map navigation (Rogers et al., 2011).

3.2.4 Consecutive tap gestures

Besides proposing new gestures using different finger movement trajectories, the use of consecutive taps was also explored in several studies. Ghomi et al. have proposed using rhythmic tapping patterns as an input method (Ghomi et al., 2012). An algorithm was

developed to distinguish short tap and long tap and these two primitive gestures can be combined to from different rhythmic tapping patterns (Figure 16). An evaluation study has shown that tapping patterns can serve as efficient application shortcuts. Boland and Murray-Smith have developed a casual way to make music selection by tapping a rhythm or tempo on the mobile device (Boland & Murray-Smith, 2013). After a tempo is tapped, the algorithm can compare it with the rhythm of vocals or that of each instrument to filter the music. These two techniques focus on exploring only the usage of temporal features of consecutive taps, while Heo et al. considered using both the temporal features and the spatial ones (Heo et al., 2014). Unlike the traditional double-tap gesture, they have proposed using two consecutive taps performed at different locations on the screen as new input methods. When consecutive distant taps are made, a specific operation can be triggered depending to the direction of the vector determined by the two taps.

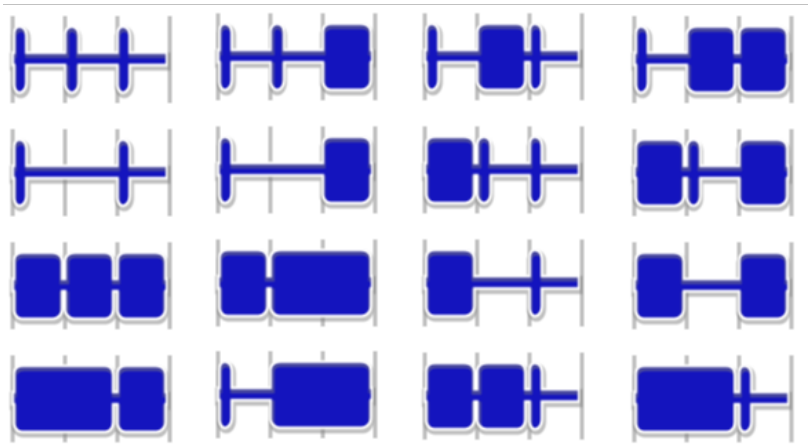


Figure 16: The 16 three-beat patterns defined by our rules. Each rectangle represents a tap. The thin gray lines show the beats (Ghomi et al., 2012).

3.2.5 Multi-finger chord gestures

In some studies, multi-finger chords are used as input methods. To perform a multi-finger chord gesture, it is necessary to tap multiple fingers on the screen in a well-defined configuration. Au and Tai have developed a multitouch finger registration technique that can identify the hand and fingers pressed on the screen (Au & Tai, 2010). In one application of this technique, which is called PalmMenu, after making a registration, users can make a multi-finger chord to select an option in the menu. Ghomi et al. have made a study to acquire some guidelines for building a large chord vocabulary (Ghomi et al., 2013). Through one evaluation, they have found that users preferred relaxed chords to tense chords, chords with fewer fingers and chords with fewer tense fingers (Figure 17).

In another study of multi-finger chords, Wagner et al. have proposed a novel hand-centric approach to recognize multi-finger chords performed on hand-held tablets (Wagner et al., 2014). To find an effective way to reduce the learning cost and also the necessary

information to retain for memorization, they have compared 'random' to a 'categorized' chord-command mapping and found that grouping chords by categories can significantly minimize the amount of information to retain. Participants who tried chords grouped by categories could recall chord-command mapping with less errors than those who tried random mapping after one week.

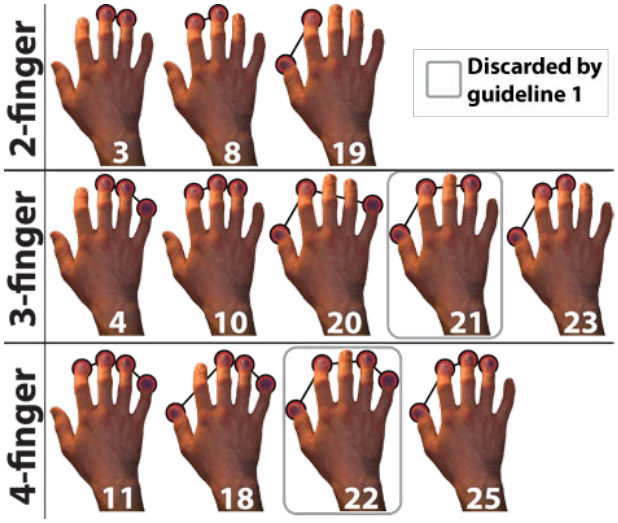


Figure 17: The set of preferred relaxed chords from the pilot study of (Ghomi et al., 2013).

3.2.6 Physic-based gestures

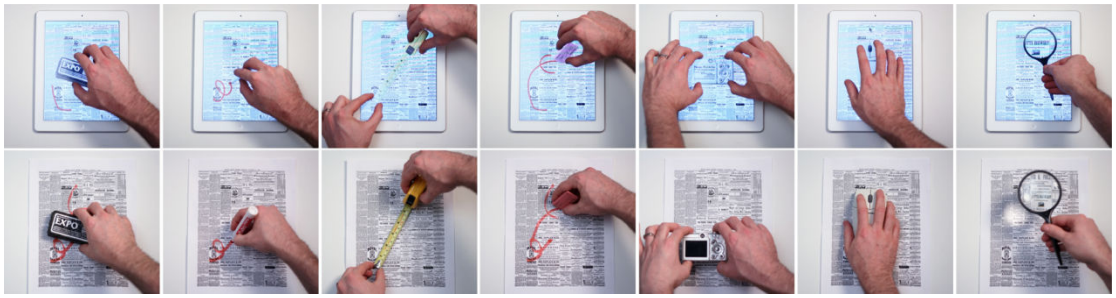


Figure 18: Example suite of TouchTools (top row; real tools shown below). From left to right: whiteboard eraser, marker, tape measure, rubber eraser, camera, mouse, magnifying glass. Contemporary multitouch interactions, such as single finger panning and two-finger pinch are unaffected (Harrison et al., 2014).

Some touchscreens are developed based on the computer vision technologies. Unlike common capacitive screens, some vision-based touchscreen can capture the shape of the contact region. This ability provides a larger design space to extend the touch vocabulary. Wilson et al. have tried using the contact shapes to make physics simulation (Wilson et al., 2008). With the help of a physic engine, users were able to manipulate virtual objects displayed on the screen as physical objects. Furthermore, Wilson has also developed an algorithm to simulate the grasping behavior in the same system (Wilson, 2009). A similar technique called ShapeTouch has been proposed by Cao et al. to make physic simulation on

the surface (Cao et al., 2008). In this technique, three kinds of forces such as pressing, colliding and friction can be simulated according to the shape on the contact region on the virtual object. The size of the force changes proportionally with the size of the contact area.

Because people have a large experience in using various tools in everyday life, a natural step is to use this experience for gesture design. Bartindale et al. have simulated a virtual mouse which can follow the hand movement on the screen (Bartindale et al., 2011). Users can perform left/right mouse button clicking and wheel scrolling on the virtual mouse to interact with the machine. Inspired by this work, Harrison et al. have developed TouchTools to leverage familiar skills with physical tools to augment touch interaction (Harrison et al., 2014) (Figure 18). Singlehanded or bimanual gestures involved when using physical tools are used as metaphors for various operations. For example, performing a gesture of holding a magnifier can zoom in digital contents while mimicking the gesture of holding a whiteboard eraser can wipe off user-drawn sketches. One advantage of these two techniques is that reusing known gestures can augment the touch vocabulary without increasing greatly the learning and memory cost.

3.2.7 Finger identification based gestures

Although touchscreens are able to detect the touch position, most of them are not able to infer the source of touch and cannot tell the difference among touch inputs made by different hand parts. Sugiura and Koseki have developed a prototype by using a fingerprint scanner to identify fingertips and have proposed mapping different operations to each finger to enrich interaction possibilities (Sugiura & Koseki, 1998). Colley and Häkkilä have made a similar approach on a smartphone which allows using fingers as shortcuts for some common operations (Colley & Häkkilä, 2014). Besides identifying fingertips, some studies have been made to distinguish different hand parts. Harrison et al. have presented a technique that can identify whether a contact is made by the nail, tip, pad or knuckle of a finger by classifying sounds made on the screen (Harrison et al., 2011) (Figure 19). In one study about how to make joint interactions between a smartphone and a smartwatch, Chen et al. have explored the use of the smartwatch for identifying which finger part is tapped on the smartphone (Chen et al., 2014b). Because the smartwatch is equipped with an accelerometer, it is possible to detect the hand orientation when a touch event occurs on the screen of the smartphone. The touch of pad, side and knuckle of the finger can be detected.

Figure 19: Left: a directory of files; items can be opened or dragged using the traditional finger pad tap. Center: user alt-clicks a file with a knuckle tap, triggering a contextual menu. Right: User pad taps on the

print option (Harrison et al., 2011).

3.2.8 Back-side touch inputs

When the finger is tapped on the screen, a small part of the screen is occluded. Due to this occlusion, it is difficult to make fine-grain interaction on the surface. To solve this issue, Wigdor et al. have developed an interactive table which can accept touch inputs on both sides (Wigdor et al., 2006). This table supports multi-touch inputs made simultaneously on both sides by multiple users. Inspired by this work, a see-through mobile device called LucidTouch was developed to augment the interaction space of mobile devices (Wigdor et al., 2007) (Figure 20). This prototype fixes a camera behind it for tracking back side touch inputs. The benefit of making touch inputs on the back side of the device is that interaction can be made without changing the holding pose. Several applications such as text typing, object manipulation and map navigation have been developed to show the productivity of this prototype. Based on the same idea, Bausch and Chu have developed a small size touch device to prove the benefit of using the backside for interaction (Baudisch & Chu, 2009). Their user study has shown that even for devices whose screens diagonal is shorter than 1 inch, precise selection still can be made by using backside touch inputs.

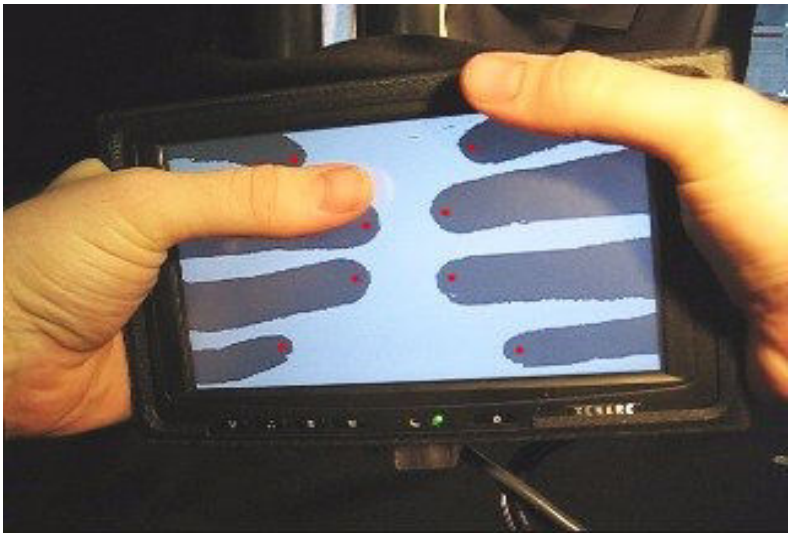


Figure 20: The LucidTouch prototype (Wigdor et al., 2007).

3.2.9 Bezel gestures

According to the taxonomy of Wobbrock et al. (Wobbrock et al., 2009), a touch gesture can be used for different purposes if the binding feature varies. To extend the touch vocabulary, some studies explore the use of bezels of the mobile device to reuse pre-defined gestures. Roth and Turner have proposed the technique of Bezel Swipe to solve the problem of gesture conflict (Roth & Turner, 2009). If a finger is tapped on the bezel and dragged to the display, the operation called by this gesture is different from that called by the dragging

gesture triggered in the display (Figure 21). Moreover, the operation varies when the gesture is started from different bezels of the device. In this study, Bezel Swipe gestures are used to select multiple photos and to mark the beginning and the end of the text to be selected. A similar study is made by Chen et al. and the bezel drag gesture is used in substitution of the long tap gesture for triggering text selection (Chen et al., 2014a). Bragdon et al. found that the speed and accuracy of bezel gestures were not significantly affected by the environment which can has different motor activity and distraction level, and some of them can be articulated eyes-free (Bragdon et al., 2011). Jain and Balakrishnan have leveraged bezel drags from eight different bezel positions of the device to trigger marking menus (Jain & Balakrishnan, 2012). After the menu appears, users can continue dragging the finger in different directions to select desired options in the menu. Bezel-tap gestures proposed by Serrano et al. allow users to call a menu for several operations (Serrano et al., 2013). After performing the bezel-tap gesture, the finger can be tapped on one operation to display its hierarchical extension and then it can be dragged in one direction to select one sub-operation.

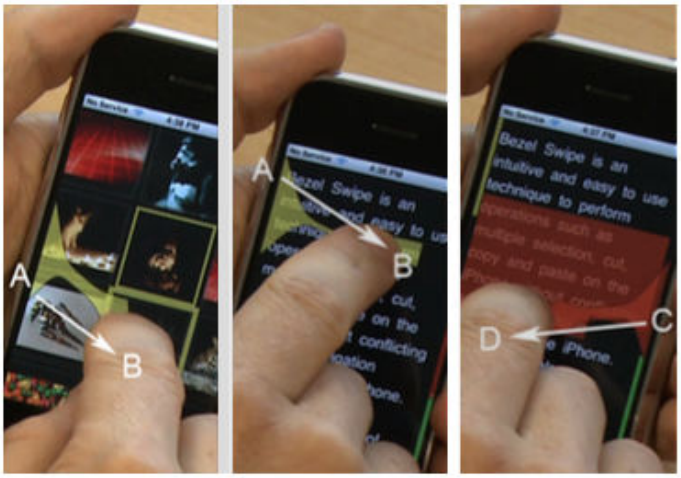


Figure 21: The Bezel Swipe prototype in action. Left panel: A swipe from A to B selects an image. Middle panel: A swipe from A to B marks the beginning of a text selection. Right panel: A swipe from C to D marks the end of the text selection (Roth & Turner., 2009).

3.2.10 Force-based & motion-based gestures

When a finger is tapped on the screen, the mobile device is under pressure and makes tiny movements. Some studies have tried measuring the pressing force by using embedded or instrumented tools and explored using this additional information to acquire different pressing patterns. Heo and Lee measured the absolute value of Z-axis accelerations in a small time window when a touch event occurs to distinguish normal tap from ForceTap (Heo & Lee, 2011b). They have proposed using ForceTap either as an alternative touch input to magnify the view, for displaying a context menu, or as expressive touch input for playing virtual instruments or force-sensitive video games. They have also built a prototype which can measure both the normal and tangential forces when a touch input

occurs (Heo & Lee, 2011a). This prototype can distinguish five gestures: tap, press, pivot, slide and drag. With the help of sensing devices, some other information such as the touch source or the holding pose can also be inferred. Sepia and Devlin used both the microphone and the gyroscope of the device to infer whether a touch input made on the back or side of the device is generated by the index, middle of thumb finger (Seipp & Devlin, 2014). Similarly, Goel et al. tried using built-in sensors to infer hand postures including one- or two-handed interaction, use of thumb or index finger, or use on a table (Goel et al., 2012). In addition, the force exerted on the screen can also be measured.

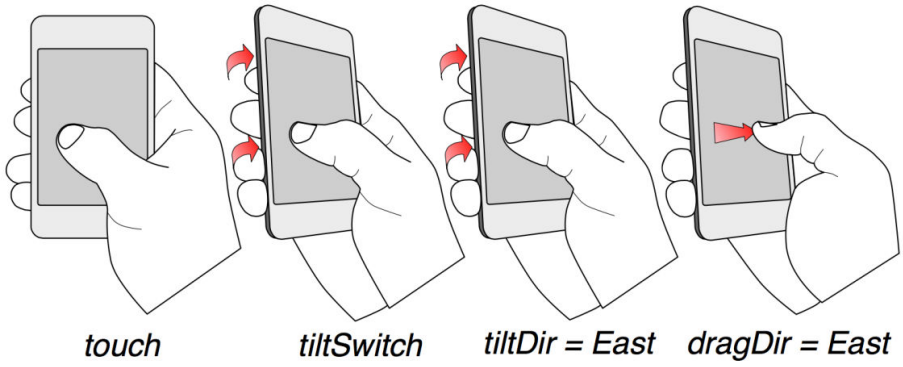


Figure 22: <tiltDir, dragDir> = <East, East> gesture performed with one hand. Here, the user starts with a tilt followed by a touch action (Tsandilas et al., 2014).

One problem of using motion gestures is how to segment meaningful motions from casual movements. Ruiz and Li have proposed a technique called DoubleFlip as a delimiter for motion gestures (Ruiz & Li, 2011). To start a motion gesture, users can rotate the smartphone along its long side away and back. Through a user study, they have found that this gesture is reliable for triggering operations because it is unlikely to invoke it accidentally in common use. Hinckley and Song have tried to combine motion inputs with touch inputs to empower interaction (Hinckley & Song, 2011). They have proposed two groups of techniques: touch in motion and motion in touch. Touch in motion refers to motion gestures which can be triggered by a touch input while motion in touch includes touch gestures enhanced by data measured by embedded devices. Tsandilas et al. also explored the combination of motion of mobile devices and touch inputs for interaction (Tsandilas et al., 2014) (Figure 22). TilTouch gestures proposed in this work can be performed by first tilting the device in a compass direction and then dragging the finger also in a compass direction. Therefore, there were a total of 4×4=16 combinations of TilTouch gestures. This set of gestures can serve as quick shortcuts for various operations. Spindler et al. have applied device motions for panning and zooming the view (Spindler et al., 2014). To make some tasks, such as map navigation, easier to perform, this technique enables users to move the device in the plane parallel to the screen for panning and in the perpendicular direction for zooming. This technique outperforms significantly traditional pinch-drag-flick technique.

3.2.11 Around device gestures

Besides touch-based gestures, gestures performed around the device can also be exploited for interaction. For example, Bulter et al. have instrumented a phone with optical sensors to track finger movement at two sides of the device (Butler et al., 2008). The benefit of this technique is that users can not only perform touch-based interaction on the screen but also on the table surface around the device. Similarly, to enlarge the interaction space of the smartwatch whose screen is of small size, magnetic sensors have also been used to track finger movement above the device in Abracadabra (Harrison & Hudson, 2009). Users can rotate an instrumented finger around the smartwatch to browse a marking menu or control a cursor on the small display. For tracking gestures above the device, Kratz and Rohs have proposed a prototype which is called HoverFlow (Kratz & Rohs, 2009). By using six infrared sensors at two sides of an iPhone, it is possible to infer freehand gestures such as horizontal sweep, vertical sweep and rotation by synthesizing all the captured infrared images. Kratz et al. have proposed another prototype which is called PalmSpace for 3D object rotation (Kratz et al., 2012). Hand pronation and supination are mapped to object rotation around two principal axes. A user study has demonstrated that their technique outperforms the usage of virtual trackball technique. Jones et al. developed a similar prototype for multiscale navigation (Jones et al., 2012). This prototype permits to pan and scale the scene simultaneously by using hand movements in air. Chen et al. have explored combining freehand gestures and touch inputs for interaction and they stated that these two modalities are highly complementary (Chen et al., 2014c) (Figure 23). To extract freehand gestures from casual finger movements, they have proposed using touch inputs before or after in-air gestures as delimiters. They have proposed three ways to interweave in-air gestures and touch inputs: Before touch, Between touch and After touch. Before touch allows using a gesture to set the context of touch inputs. Instead of using freehand gestures above the mobile device, song et al. have proposed an algorithm to recognize gestures performed behind the device by using the images captured by the backside camera (Song et al., 2014). The benefit of this technique is that it is unnecessary to use instrumented devices for gesture recognition. They have developed several applications to demonstrate how touch inputs and gestures behind the device can cooperate simultaneously.

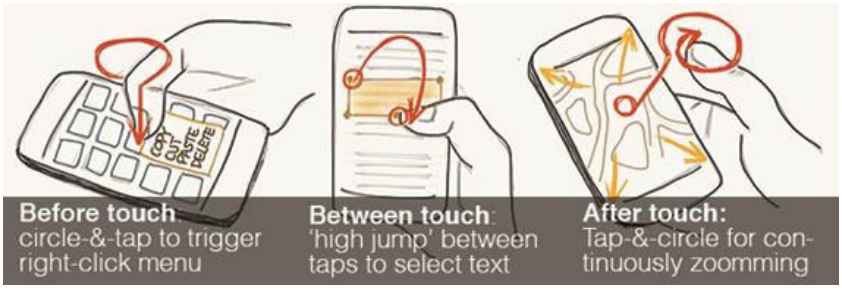


Figure 23: Touch and in-air gestures be interwoven to create fluid and expressive interactions (Chen et al., 2014c).

3.3 Discussion

Through the study of different modalities proposed for mobile usage, we have identified several trends and problems of the development of mobile NUI.

Nowadays it is a standard configuration to use a touchscreen for interaction on a mobile device. To ensure the portability, the size of the mobile device is rather limited compared with that of other equipment designed for indoor usage. On desktop computers, a GUI widget can simply be added to the interface to provide a new function. However, on the screen of mobile devices, there is far less space to display a lot of GUI widgets. A common objective of aforementioned studies of mobile interaction is to find a solution to enrich the interaction possibilities without scarifying the screen space.

In general, different modalities can be grouped into two categories. Techniques in the first category are proposed to augment the interaction power of touchscreens. Besides the number of fingers pressed on the screen and their positions, these techniques have tried to explore the use of other information such as the orientation of the fingers, the size of the contact area and the identity of the pressed finger, etc. With the help of advanced algorithms and additional sensors, the touchscreen cannot only capture the gesture performed on it, but can also infer the state of the hand when the gesture is performed. With the help of this information, a larger interaction vocabulary can be constructed.

In the second category, we have grouped techniques which are proposed to explore the usage of new input channels other than the touchscreen. For example, interaction can be performed by making touch inputs on the backside of the device, by using motions of the device and by gestures in air performed around the device. One benefit of using these additional channels is that they do not occupy any screen space. In addition, these modalities are less influenced by the layout of the interface. For instance, motion gestures of the device can be performed for blind users. Although these modalities may be less precise than touch inputs, they can be used as efficient shortcuts for some functions. Moreover, unlike touch inputs, these modalities permit to avoid the occlusion problem.

From the study of interaction techniques for mobile devices, we have identified two trends. The first trend shows that the body is used more and more as the interface. A lot of research has been done to free the expressive force of the human body, especially the hand. Mobile devices now become more sensitive and reactive to hand motions.

The second trend shows that more and more sensors are embedded into the mobile device. Traditional interaction devices such as the mouse and the keyboard only support one modality. However, with the help of multiple sensors, mobile devices now have multiple channels. Though some work in the literature has installed exterior sensors on mobile devices to build prototypes, we think that in the near future some of these sensors will be embedded into mobile devices. With the help of multiple channels, interaction designers can propose hybrid techniques to improve the experience of an existing scenario, or even explore the new usage of mobile devices for some unknown scenarios.

However, interaction designers have to face some difficulties when designing new natural techniques. On the one hand, using the aforementioned natural modalities can help to save the screen space. On the other hand, these new interaction modalities are less visible. One problem that designers have to solve is how to ensure that these functions can be found out by novice users and how to ensure that these techniques can be learnt within a short learning time. In addition, although a larger design space allows interaction designers to find a solution for some unsolved interaction problems, interaction designers may find it difficult to choose the modalities and use them with an appropriate metaphor. It really requires iterative design work and many user studies to find the most appropriate solutions.

4 Conclusion

In the first chapter, we have presented concepts which are relative to our research topic. After presenting the different definitions of NUIs proposed in the literature, we have given our own definition of NUIs. Besides using learnt skills for interaction, we think that it is also necessary to map them to interaction operations in an appropriate way in order to provide a natural experience. Then, we have presented several potential modalities that can be used to develop NUIs. In the literature, several studies have explored how to use these modalities independently or in collaboration to develop NUIs.

After presenting relative concepts of NUIs, we have presented how interaction possibilities of mobile devices can be enhanced. Because nowadays most of the mobile devices allow using a touchscreen for interaction, we have first presented the concept of 2D gestures. Then we have presented the concepts of 3D gestures and device motion gestures. At last, we have presented previous studies made in the literature to enrich interaction possibilities.

In the next chapter, we will present a state-of-the-art of selection techniques. For both object selection on mobile devices and object selection at a distance, it is difficult to specify a small object from a dense cluster. The difficulty is mainly caused by the fingertip occlusion problem and the lack of physical support respectively for these two scenarios. We will present the different strategies that have been used to improve the selection efficiency and accuracy on mobile devices.

Chapter 2: Selection techniques

Chapter 2: Selection techniques

1 Introduction

For 3D interaction, selection is an essential task. Selection techniques enable users to specify which objects they want to observe and manipulate. Besides 3D interaction, selection is also frequently performed for other interaction tasks. For example, on desktop graphical interfaces, the mouse is used to select files, adjust the cursor position in the text and choose a button to switch the manipulation mode, etc. On mobile devices, touch inputs are made to launch applications and to select phrases from texts. In the interaction community, improving the selection efficiency and precision remains an interesting challenge. A lot of studies have been made to propose new techniques to simplify selection of small targets. In addition, it is more difficult for motor-impaired or old users to perform selection tasks than healthy persons due to their inability to control their body as they wish. Developing new techniques that require less precise control can help them to interact with computers more efficiently. In this chapter, we will present the state of the art of selection techniques which are based on both 2D and 3D inputs.

2 Human models

As Bowman et al. have proposed, a selection task can be decomposed into three subtasks: indication of object, indication to select and feedback (Bowman et al., 2001). For a selection task, the majority of the time is consumed to indicate the target. Because the performance of object indication is highly related to the human behaviors, studies have been made to provide insights into various aspects of selection tasks. Several human pointing models have been proposed as results of these studies. These models are helpful to predict the selection performance and can be used as guidelines to design selection techniques.

2.1 Fitts' Law

Fitts' law is the most famous model which is used frequently to explain selection behavior. This human psychomotor behavioral model first emerged from experimental psychology (Fitts, 1954) and then it was widely adopted in many areas, including human factors, human-computer interaction and ergonomics. The main purpose of Fitts' law is to estimate the time required for performing an aimed movement considering two fundamental properties: the size of the target and the amplitude of movement required to reach it. MacKenzie (MacKenzie, 1992) has proposed a common formulation of Fitts' Law to estimate the time required to reach a target:

$$T = a + blog2\,(\frac{A+W}{W}) \qquad \text{Equation 1}$$

Where T is the selection time, A is the distance of the object and W is the effective width of the object. a and b are two regression coefficients and the logarithmic term is called index of difficulty (ID).

In the literature, Fitts' Law is frequently used to estimate the time required to select a target in GUIs (Bi et al., 2013; Grossman & Balakrishnan, 2005; Su et al., 2014). It can also be applied to estimate the time for selecting 2D and 3D objects in a virtual environment (Kopper et al., 2011). Bi et al. have found that Fitts' Law is still valid to describe selection performance on touchscreens (Bi et al., 2013). Instead of considering the effective width of the target, they use the standard deviation of the touch points and the absolute precision of the input finger to estimate the acquisition time. Wingrave and Bowman have found that Fitts' Law still holds when users perform selection tasks at a distance (Wingrave & Bowman, 2005). They stated that W is related to the visual size of the target, and the calculation of A should consider the rotation angle of the hand.

2.2 Optimized Initial Impulse Model

Although Fitts' Law can be used to estimate the selection time for a specific selection task, it does not provide insight into the selection process to help understand the selection process. For that purpose, the Optimized Initial Impulse Model provided by Meyer et al. reveals more details about human behaviors (Meyer et al., 1988). This model divides an acquisition task into two phases: Ballistic phase and Corrective phase. At the beginning of the task, the selection tool is not close to the object, coarse and rapid motions are favored to reduce the approaching time. It was found that during the Ballistic phase, the selection tool is moved rapidly to approach the target. Before entering the proximity of the target, people care more about velocity than the movement precision. However, when the selection tool is close to the target, users begin to reduce the velocity of the selection tool to adjust its position more carefully. At this moment, the Ballistic phase is switched to the Corrective phase. During this phase several consecutive corrective movements are made until the target is finally reached.

This model also reveals the velocity-accuracy trade-off during the selection task. On the one hand, moving the selection tool at high speed during the Ballistic phase saves time to approach the target. On the other hand, the velocity is increased at the expense of the precision. The higher the velocity is, the larger the deviation of the end point distribution around the target is (Schmidt et al., 1979). This trade-off can be explained by the fact that the bigger the movement is, the larger the muscle groups involved are (Card et al., 1991). Larger muscle groups are less dexterous. Thus, it is more time consuming to use them to adjust the selection toll precisely. Meyer et al. have defined the velocity-accuracy ratio for ballistic movements. A formulation is proposed to estimate the standard deviation of the

end point of movements (Equation 2).

$$S = k\frac{D}{T} \qquad \text{Equation 2}$$

Where S is the standard deviation, D is the distance covered and T is the movement time. k is a proportional ratio.

It is observed that a common strategy to reduce the selection time is to find a balance between ballistic movements and corrective ones. According to the formulation of MacKenzie et al. the speed of ballistic movement is influenced by the distance of the object while the effective width has a crucial impact on the range of corrective movements (MacKenzie, 1992). Both the object distance and the effective width determine the most appropriate time to switch from ballistic movements to corrective ones.

3 2D-based selection techniques

First, we will present the state-of-the-art of selection techniques which are developed based on 2D inputs. 2D-based selection techniques can be coarsely divided into two groups according to the input device: mouse-based selection and touch-based selection. By moving a mouse on the desk, users can move a cursor from its initial position to reach the target located in a specific position. The mapping from the mouse movements to the cursor movements is determined by the Control-Display ratio (CD ratio). The CD ratio is defined as the ratio of input device movements to the selection tool movements. In many applications the CD ratio is not a constant value, but varies according to the mouse velocity. When the mouse moves slowly, the CD ratio is set to a large value to acquire precision. However, it can be decreased when the cursor speeds up to cover a large distance. Although the mouse is an accurate input device, it is still worth exploring new selection techniques. For desktop applications such as Microsoft Office Word that provide numerous functions, many UI widgets are of small size to save the screen space. As a result, users may find it difficult to select the right UI widget from a dense cluster.

For touch-based interaction, instead of dragging a cursor, users can tap one finger directly on the desired target to select it. On the one hand, touch-based interaction turns out to be more natural because moving the finger in air to approach the target is similar to our daily experience. On the other hand, touch-based selection suffers from relative low precision. The common explanation for the inaccuracy of touch is the fat finger problem (Vogel & Baudisch, 2007). According to this model, the softness of the skin causes the touch position to be randomly distributed under the fingertip. In addition, the object is occluded by the fingertip so that users have no visual feedback to help them compensate the offset. However, Holz and Baudisch have proposed the Generalized Perceived Input Point Model to explain the inaccuracy in another way (Holz & Baudisch, 2010). They argue that the offset between the center of the contact area and the target depends on the yaw, roll and

pitch orientation of the fingertip when it is pressed. Besides fingertip orientation, the distribution of the contact region also differs across users. Through a user study, it was found that 67% of the touch inaccuracy can be explained by this model.

In the following section, different selection techniques are grouped into five categories according to the strategy used to simplify the selection task: target expansion in control space, target expansion in visual space, switch of CD ratio, occlusion avoidance and gesture shortcut.

3.1 Target expansion in control space

When clicking a mouse to select a target, the system checks whether pixels of the target are covered by the cursor. The smaller the target is the more concentration is required to position the cursor upon it. On touchscreens, when the target is smaller than the fingertip, it becomes even impossible to judge whether the contact point is generated in the desired location. According to the formulation of Fitts' Law, the smaller the effective width of the target is the longer the selection time is.

One strategy to simplify selection tasks is to expand targets in the control space while their sizes in the visual space remain unchanged. Instead of adjusting the position of the selection tool carefully, it is only required to move the selection tool into the magnified effective region of the target.

One way to enlarge the influence area of the target is to use volumetric selection tools. Instead of casting a line-shaped ray when the mouse is clicked in the workspace, the modeling system proposed by Liang and Green projects a cone-shaped ray to check object intersection (Liang & Green, 1994). With a larger effective volume, the cone-shaped ray has more chance to cover the target. However, if more than one object is located in the region, disambiguation mechanisms are required.

Grossman and Blakrishnan have proposed the Bubble Cursor which can avoid the selection ambiguity (Grossman & Balakrishnan, 2005). The Bubble Cursor is an area cursor which uses a circle area around the cursor to increase the influence area. In fact, a Voronoi diagram is generated to divide the display space into several cells according to the object distribution (Figure 24). Each cell encircles only one object which is the closest one to any point inside the cell. When the cursor enters a cell, the object encircled by the cell becomes the only target covered by the Bubble Cursor. Although the Bubble Cursor significantly improves the selection efficiency by freeing users from positioning the cursor precisely upon the target, its performance is still influenced by the cluster density and degrades almost to Ray-casting performance when the cursor is located in a dense cluster. DynaSpot is similar to the Bubble Cursor, but the activation area is coupled with its speed (Chapuis et al., 2009). When the cursor is static, it just looks like a regular cursor. However, the activation area magnifies proportionally with the speed. Like the Bubble Cursor, only the closest target is covered by the area.

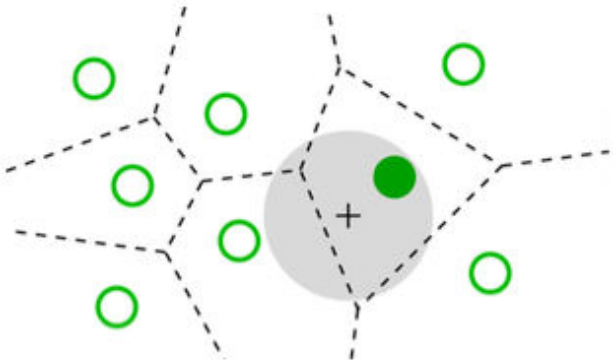


Figure 24: The Bubble Cursor encircles only the closest target (Grossman & Balakrishnan, 2005).

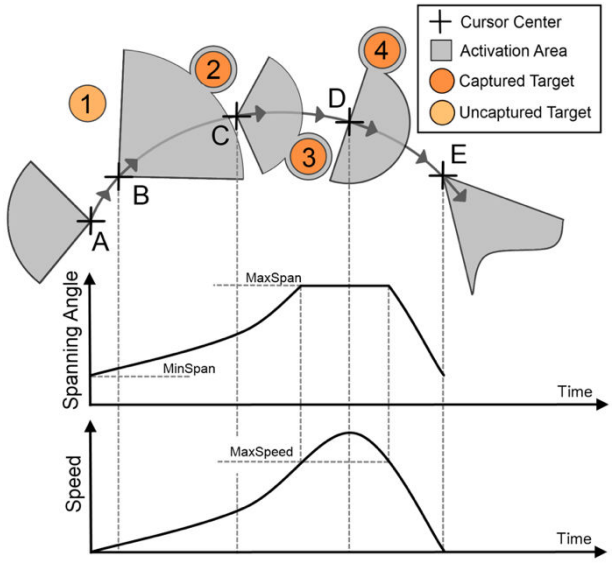


Figure 25: The spanning angle of the IFC is dynamically changing according to the cursor speed, and its orientation is determined by the cursor's moving direction (Su et al., 2014).

Instead of using a fixed circular activation area, Su et al. have designed the Implicit Fan Cursor (IFC) which couples its influence region with both its speed and moving direction (Su et al., 2014). As shown in Figure 25, the spanning angle of the fan-shaped area changes dynamically according to the cursor speed and its orientation is determined by the moving direction. When the cursor is moved, only the closest object located in the moving direction is covered by the fan. A user study has shown that the Implicit Fan Cursor outperforms the Bubble Cursor and Dynaspot, particularly in terms of cursor moving distance.

As mentioned above, the performance of area cursor is highly influenced by the cluster density. Baudisch et al. have proposed a technique, Starburst, to decompose and distribute the space more averagely (Baudisch et al., 2008). After generating the Voronoi diagram, the algorithm looks for cells whose sizes are smaller than a pre-defined threshold. These cells can be expanded by borrowing spaces from adjacent large cells (Figure 26). In a

stylus-based user study, the authors have found that Starburst has smaller error rates than the Bubble Cursor. However, it is necessary to display contours of all the generated cells to provide visual cues for selection. Visual distraction may hinder users from searching objects efficiently.

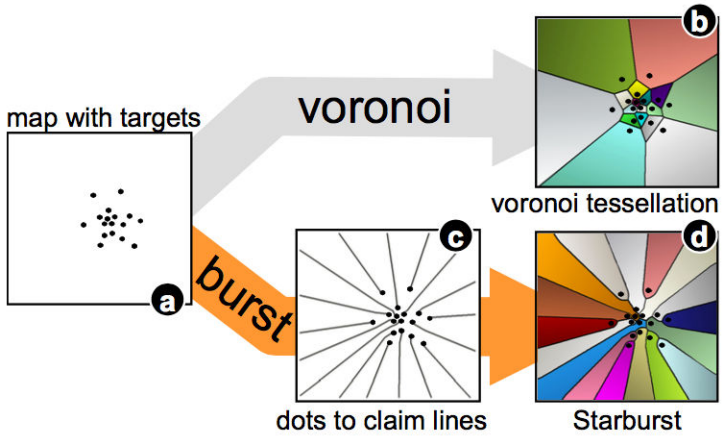


Figure 26: Starburst segments the space more averagely than Voroni tessellation (Baudisch et al., 2008).

Findlater et al. have proposed several enhanced area cursors for motor-impaired users by reducing the need for corrective movements (Findlater et al., 2010). The Click-and-Cross Cursor and the Cross-and-Cross Cursor allow users to first use a circular area cursor to preselect several objects covered by it. After performing a trigger event to for pre-selection, all the objects covered by the circular activation area are listed around a larger circle. To select an object, the cursor should cross its corresponding crossing arc on the large circle. The benefit of this disambiguation mechanism is that the effective widths of small objects are replaced by larger crossing arcs. They have also proposed the Motor Magnifier to slow down the cursor movement after a circular region is specified by the area cursor. Inside the circular region the CD ratio is enlarged by four times to simplify the utilization of the Bubble Cursor.

3.2 Target expansion in visual space

Besides increasing the influence region of objects in motor space, another common approach is to directly magnify objects in the visual space. A magnified object is easier for visual research as well as for selection tool positioning. Benko et al. have proposed the Dual Finger Stretch technique which can be used to magnify a region on the touchscreen to simplify the selection tasks (Benko et al., 2006). If users want to magnify the region around the first pressed finger, a secondary finger can be dropped on the screen to specify a square zooming area centered at the first fingers's location. The secondary finger can be moved to adjust continuously the magnification level while the first finger is used to make a selection. This technique is helpful when selecting small UI widgets docked on the interface.

Pointing Lens proposed by Ramos et al. allows users to magnify an area centered at the

location of the stylus (Ramos et al., 2007). After dwelling the stylus for a short moment or increasing the pressure applied on the stylus, a square magnification lens is displayed. The stylus can be used to select magnified objects inside the lens. In addition, when the stylus is dragged outwards, the lens can be moved to adjust the magnification area.

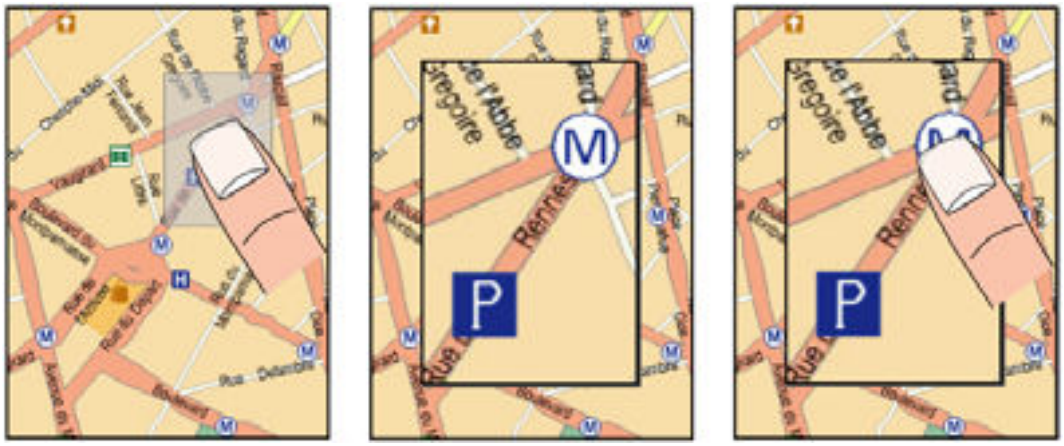


Figure 27: TapTap Design (Roudaut et al., 2008).

Roudaut et al. have developed an easy-to-use technique which is called TapTap to simplify selection of small objects (Roudaut et al., 2008). TapTap enables using the first tap on the screen to zoom in a region and then use the second tap to select an object in the magnified lens (Figure 27). After a target is selected, the magnified view disappears and the initial view is exposed again. Although this technique is easy to perform, it is in conflict with the default use of the tap gesture.

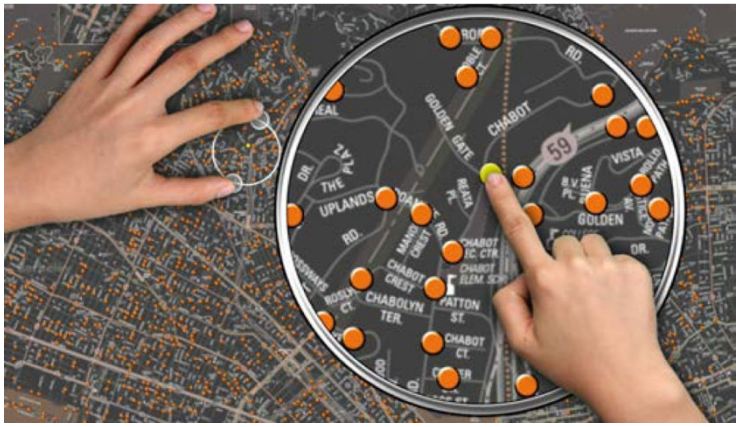


Figure 28: FingerGlass allows using the coarse hand to set a region to magnify and using the fine hand to select objects in the magnified view (Käser et al., 2011).

Käser et al. have proposed a bimanual technique, FingerGlass, to zoom in a specific region on the touchscreen (Käser et al., 2011). After tapping two fingers of the coarse hand on the screen, a circle crossing both touch positions is displayed and the region inside it is magnified and displayed in another circular viewport close to the selected region (Figure

28). The region size evolves dynamically when finger positions of the coarse hand are modified. Users can use the fine hand to select and translate objects in the popup viewport.

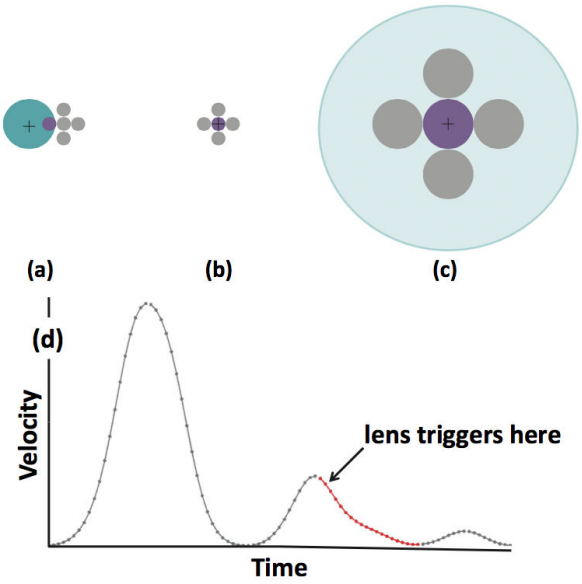


Figure 29: (a) The Bubble Cursor resizes to select the closest target; (b) The Bubble Cursor degenerates to a point cursor in small, dense target situations; (c) Bubble Lens automatically magnifies targets, making them larger in visual and motor space; (d) Magnification is triggered on the downward slope of the first corrective submovement in a smoothed velocity profile. The effect is that as a user corrects his motion near the desired target, the lens triggers automatically, making the target easier to acquire (Mott & Wobbrock, 2014).

Mott and Wobbrock have proposed Bubble Lens to improve the performance of the Bubble Cursor for acquiring small, dense targets (Mott & Wobbrock, 2014). When the selection tool is moved on the screen, its speed profile is measured dynamically to determine when the corrective movements begin. Inspired by the Optimized Initial Impulse Model (Meyer et al., 1990), the authors have decided to prepare Bubble Lens when the second peak of the speed profile appears (Figure 29). Bubble Lens can be only opened when the size of targets close to the selection tool is smaller than a threshold value and when the cluster is dense. Instead of magnifying the region around the selection tool directly, objects inside the region are enlarged while initial distances among objects are kept to avoid losing visual focus in the magnified view. It was found that Bubble Lens helps the Bubble Cursor to reduce both selection time and error rates.

Although using magnification lens help expanding the visual space, some content of the context is more or less lost due to the limited size of the magnified viewport. As a result, users are likely to lose the focus and have to remake the visual search. Instead of providing a magnification lens, another solution is to zoom in the whole scene. Olwal et al. have proposed two families of techniques, rubbing and tapping, that use zooming to make precise selection (Olwal et al., 2008). Users can rub the finger continuously on the screen to zoom in a specific region. The Rub-pointing technique allows users to zoom in a region

continuously by performing the rubbing gesture (Figure 30 (a)). The Zoom-tapping technique permits pressing one finger on the screen to set a region and tapping a secondary finger to zoom in the selected region (Figure 30 (b)). Users can magnify the scene many times by performing consecutive tapings on the screen. Since the scene is zoomed in smoothly, users are able to adjust the magnification level without losing the focus.

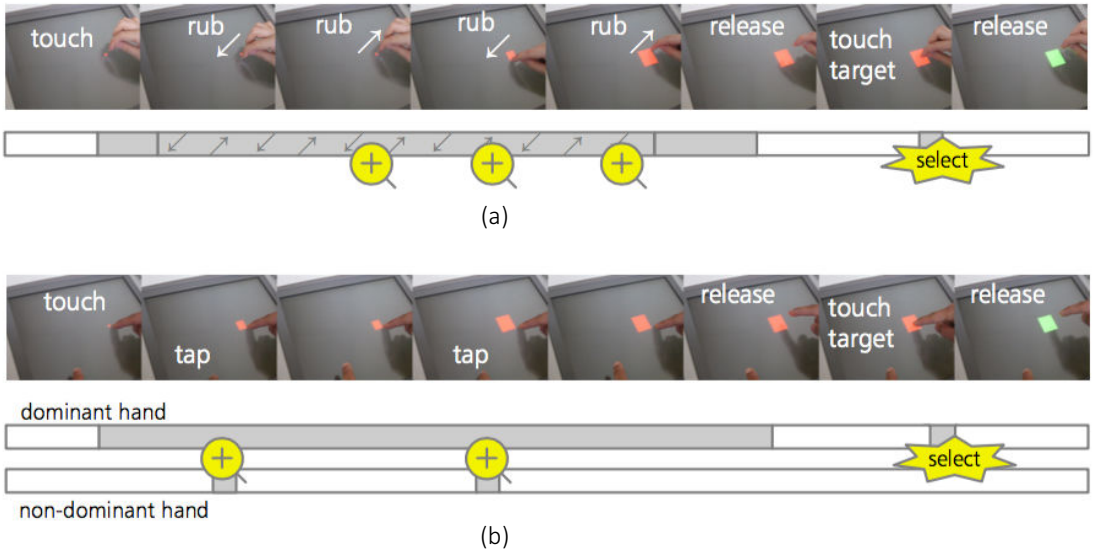


Figure 30: (a) Rub-pointing; (b) Zoom-tapping (Olwal et al., 2008).

Gutwin have designed a technique to zoom in a region on the screen without using a magnification lens (Gutwin, 2002). The solution is to use a fisheye view to zoom in the region around the cursor. One advantage of the fisheye view is that the magnification level evolves smoothly from the region far from the cursor to its proximity. Unlike the magnification lens, the fisheye view does not have an incoherent gap between the initial scene and the magnified viewport. However, the fisheye view also has its own inherent drawbacks. Indeed, magnification makes focus-targeting difficult because objects appear to move as the focus point approaches them. Gutwin have proposed a technique called speed-coupled flattening to compensate this issue.

3.3 Switch of CD ratio

The CD ratio is a factor that plays a very important role in object acquisition tasks. An appropriate designed formulation to map inputs to outputs can significantly improve interaction efficiency and experience. Blanch et al. have designed a mapping formulation of the CD ratio which is called Semantic pointing (Blanch et al., 2004). This formulation adjusts the CD ratio dynamically according to the distance between the cursor and objects. The CD ratio is decreased when objects are far from the cursor and it is increased when objects become closer.

Benko et al. have proposed two techniques which allow adjusting the CD ratio manually on

the touchscreen (Benko et al., 2006). The first one which is called the Dual Finger X-Menu, permits switching the CD ratio in a marking menu (Figure 31). The menu provides several options of cursor speed. Users can choose one option to transfer finger motions to smaller cursor movements, or even freeze the cursor on the screen. Another technique which is called Dual Finger Slider enables users to drag a secondary finger vertically to adjust the CD ratio.

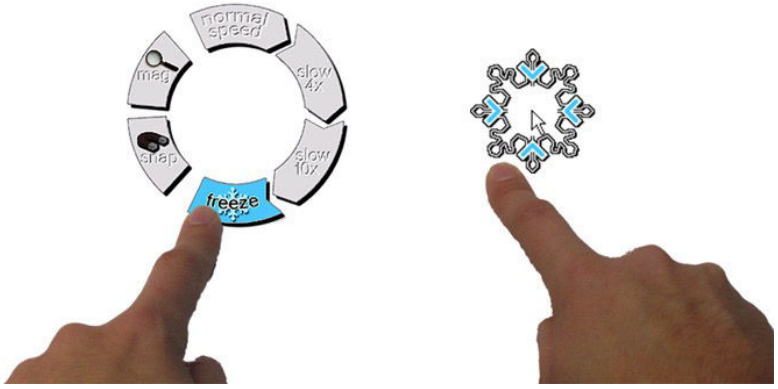


Figure 31: Dual Finger X-Menu (Benko et al., 2006).

According to the definition of the CD ratio, if its value is bigger than 1, the selection tool makes smaller motion than the inputs. Oppositely, if the CD ratio is smaller than 1, the magnitude of selection tool motion is bigger than that of the input device. In some interaction scenarios, users have to control the cursor on a large display indirectly by making indirect inputs on a smaller input device, for example, a mobile device. Some studies have explored using a small CD ratio to transfer input motions to larger cursor movements on the large display.

Figure 32: (a) Cursor is initially at the top right corner; (b) Tapping anywhere with ARC-Pad causes the cursor to instantly jump across the screen; (c) For accurate positioning the user can clutch and slide the finger (McCallum & Irani, 2009).

McCallum and Irani have proposed ARC-Pad which supports using a touch-based smartphone to position the cursor on a large display (McCallum & Irani, 2009). This technique allows users to position the cursor in absolute and relative mode by using the

tapping and dragging gesture respectively (Figure 32). When the finger is tapped and then released without being dragged on the screen, the screen of the smartphone is mapped to the large display and the cursor jumps directly to the corresponding position of the tap point on the large display. To adjust the cursor position more precisely, users can drag a finger on the smartphone screen to make relative movements. ARC-Pad helps to accelerate the cursor translation without sacrificing the accuracy. A user study has shown that it outperforms the relative positioning commonly found on touchpads.

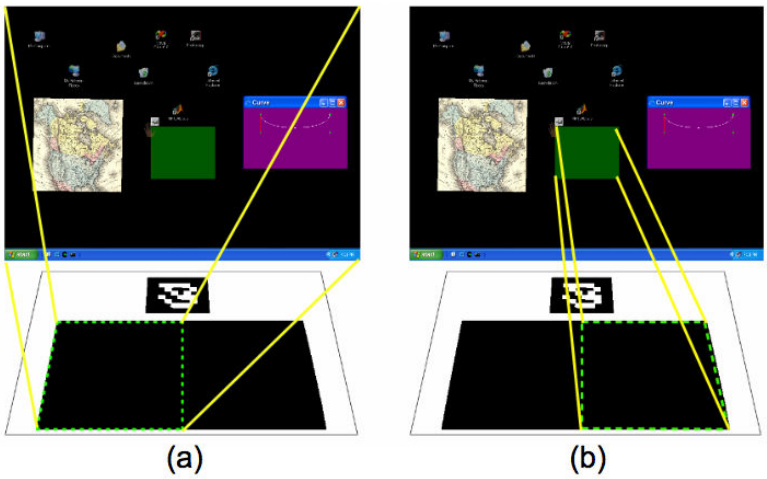


Figure 33: Touchpad mapping for asymmetric interactions for: (a) the left hand is used to translate the green workspace in the whole display; (b) the right hand is used to control the cursor inside the workspace (Malik et al., 2005).

Malik et al. have transformed a desk into a touchpad to enable users controlling a cursor on a large display by touch inputs (Malik et al., 2005). The touchpad is divided into two parts: the left half part is mapped to the whole large display while the right part is mapped to a workspace (Figure 33). The left hand is used to position the workspace coarsely in the large display while the right hand is used to select targets covered by the workspace. The strategy of asymmetric division of labor takes advantage of both hands to interact with the large display in a collaborative way.

3.4 Occlusion avoidance

Because the inaccuracy on the touchscreen is mainly caused by the fingertip occlusion, some techniques have been proposed to overcome this problem. The Offset Cursor technique is one of the notable approaches designed to reduce the influence of the occlusion issue (Potter et al., 1988). Instead of using the contact point under the fingertip, a cursor displayed above the fingertip is used to locate objects. Although the Offset Cursor can be used with more accuracy, a drawback of this technique is that objects at the bottom side of the display cannot be reached. Vogel and Baudisch have proposed an improved technique which is called Shift (Vogel & Baudisch, 2007). Instead of displacing the cursor to the upside of the contact position, the region around the touch input is amplified in a

magnification lens (Figure 34). The occluded region is exposed in the lens while the contact point is still active for checking object intersection. The magnification lens is displayed only when targets in the proximity of the touch point are of small sizes. According to the results of a user study, Shift generates much fewer selection errors than direct pointing and is faster than Offset Cursor for larger targets.

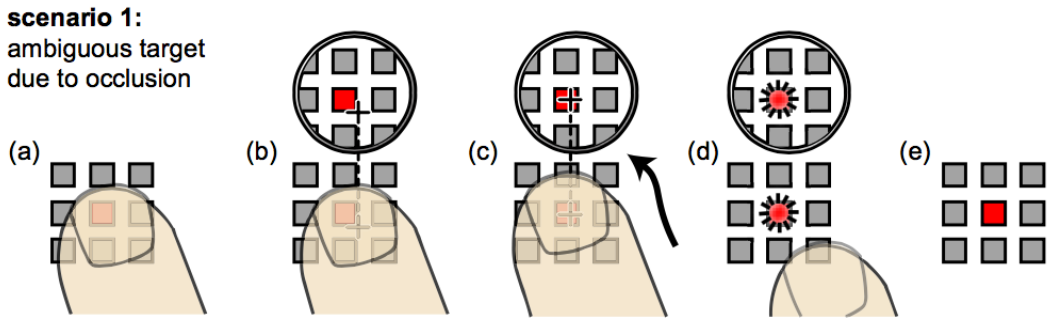


Figure 34: Shift technique walkthrough (Vogel & Baudisch, 2007).

Karlson and Bederson have proposed the ThumbSpace technique which displays a semi-transparent miniature of the whole display above the initial content (Karlson & Bederson, 2007). The contact point of the thumb in the miniature is mapped to a position in the whole display. The thumb can be dragged in the miniature to select a target in the whole display of the mobile device. On the one hand, this technique permits to avoid occluding the target by the thumb. On the other hand, the range of the selection tool movements is bigger than that of the thumb. As a result it requires users to refine the selection carefully.

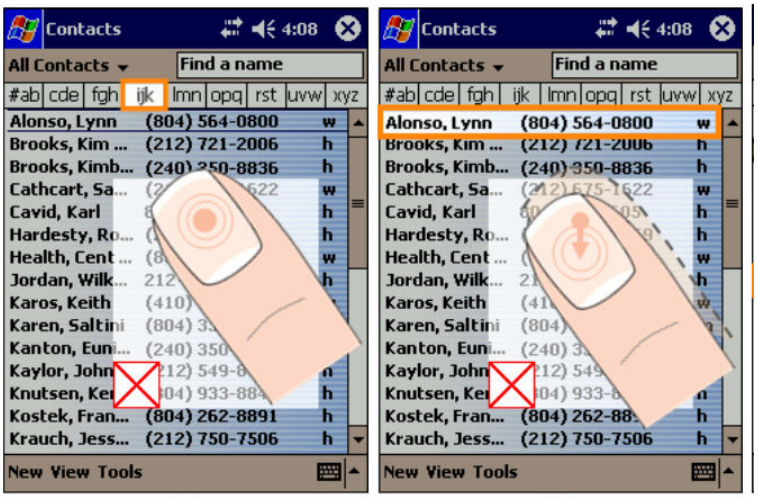


Figure 35: ThumbSpace uses finger movements in the miniature to select a target in the whole display (Karlson & Bederson, 2007).

For large touchscreens, Benko et al. have imagined the Dual Finger Midpoint technique to displace the cursor from the initial contact point (Benko et al., 2006). When a secondary finger is pressed on the screen, the cursor jumps to the midpoint between two touch points.

Users can move either finger or both of them to adjust the cursor. Moreover, to reduce the influence of tremble when fingers are released, the cursor speed is slowed down.

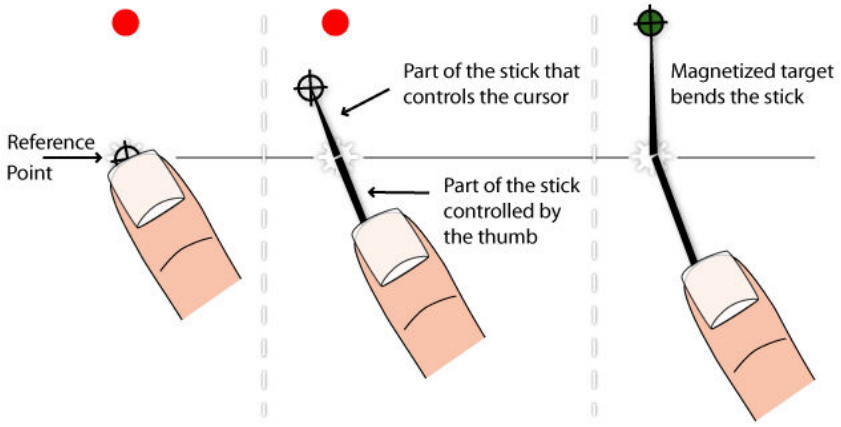


Figure 36: The design of MagStick (Roudaut et al., 2008).

In addition to TapTap, Roudaut et al. have designed another technique which is called MagStick (Roudaut et al., 2008). When the finger is pressed on the screen, a pivot is set at the contact position. When the user starts dragging the finger in one direction, a cursor appears in a position which is symmetrical to the contact point with respect to the pivot (Figure 36). For example, when the finger moves to left down, the cursor goes to the right up corner. To simplify object selection, targets in the display are designed to have a magnetic effect on the stick so that the stick is bent when the cursor is close to a target. The design of magnetized targets helps saving time to adjust the cursor position when the cluster is not of high density. However, the cursor may be pulled to undesired objects when the cluster density increases.

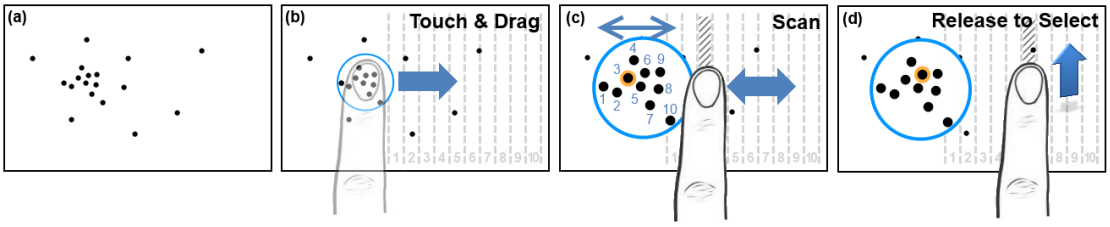


Figure 37: The workflow of LinearDragger (Au et al., 2014).

Au et al. have proposed LinearDragger, which is able to solve the occlusion problem without scarifying the efficiency (Au et al., 2014). To make a selection, users need to first tap the finger in the surrounding region of the target and drag the finger in any direction (Figure 37). If the dragging distance exceeds a threshold value, a magnified view of the tapped region is displayed. After that users can continue dragging in the same direction to scan all the objects covered by the region one by one. When the finger is lifted, the object in focus is selected. The motor space is decoupled from the visual space by using the same effective width for each object in the scanning list, so that the scanning is independent from

the distance between adjacent objects. A user study has demonstrated that LinearDragger outperforms several state-of-the-art techniques such as the Bubble Cursor and Shift.

3.5 Gesture shortcuts

No matter the strategy they are based on, all the techniques mentioned above still require using a selection tool to reach the target. However, few techniques have been proposed to free users from the burden of adjusting the selection tool carefully. This kind of techniques allows making a stroke as a shortcut to select an object or a function. Appert and Zhai have explored using strokes as command shortcuts (Appert & Zhai, 2009). To examine whether gestures can be used to replace keyboard shortcuts, a set of gestures have been designed to map a set of commands. Unlike traditional menus, the menu in their experiment displays gesture shortcuts in substitution to keyboard shortcuts for each command. Users are asked to memorize all the gesture shortcuts and reproduce them to launch different commands. The user study revealed that, with enough practice, both types of shortcuts had the same level of performance. However, gesture shortcuts had substantial cognitive advantages for learning and recall.

Fekete et al. have also explored the use of gestures as shortcuts and they have proposed a set of elliptical motion based gestures (Fekete et al., 2009) (Figure 38). This set of gestures is defined based on an elliptical model by using several parameters such as the movement period and the amplitude on the major axis. Because two mouse trajectories of the same motion but with different frequencies are considered as two different gestures, dynamic gesture motions are necessary to be displayed as visual cues. Compared to the work of Appert and Zhai (Appert & Zhai, 2009), one contribution of this work is that temporal features can be also taken advantage of for gesture design.

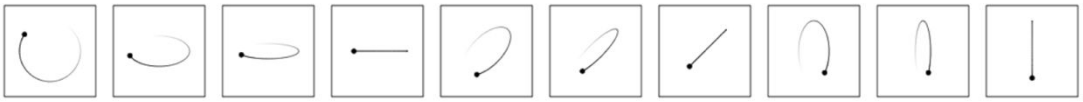


Figure 38: Sample of some of the cyclic elliptical motions used as the basis for the motion-pointing method (Fekete et al., 2009).

In some studies, gesture shortcuts can be used to simplify object selection. Yatani et al. have proposed the Escape technique which allows users to use gestures to make selection refinement (Yatani et al., 2008). For objects in a dense cluster, a direction peak-shaped vector is displayed upon each object. When the user taps the finger close to the desired object, the finger can be dragged in the direction indicated by the peak of the target to make a quick selection. As shown in Figure 39, if the vector direction of the target is Right, the finger should be moved to the right to get access to the target. The authors decided to use only eight compasses to distinguish between adjacent objects. Using more vectors complicates the differentiation between numerous directions, and also increases the difficulty of dragging the finger in the desired direction. As a result, one limitation of Escape is that it cannot be used to make a selection inside a dense cluster.

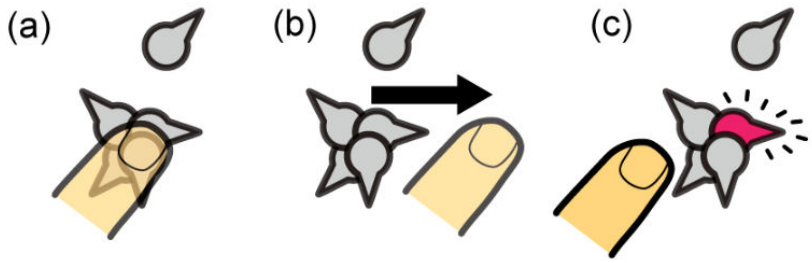


Figure 39: Escape allows users to select a target by dragging the thumb in the direction indicated by the target (Yatani et al., 2008).

Bragdon and Ko have applied a similar strategy to select objects in a large display (Bragdon & Ko, 2011). Instead of moving a cursor to make a selection, users can first drag the stylus in the direction of the desired object. A fan-shaped region is projected in the moving direction to preselect objects covered by it. After setting the selection volume, a unique stoke is displayed above each target which is covered by the selection volume. Users can then perform the corresponding gesture to select the object. Gestures are ordered according to the difficulty of reproduction and simple gestures are distributed to objects closer to users (Figure 40 (a)). However, no gestures are assigned for objects which are located in the motor space of the user. To make a gesture reusable, one or more pigtails can be added to it to modify its shape slightly (Figure 40 (b)). Users can benefit from this technique to select objects which are far from them without moving frequently in front of the large display.

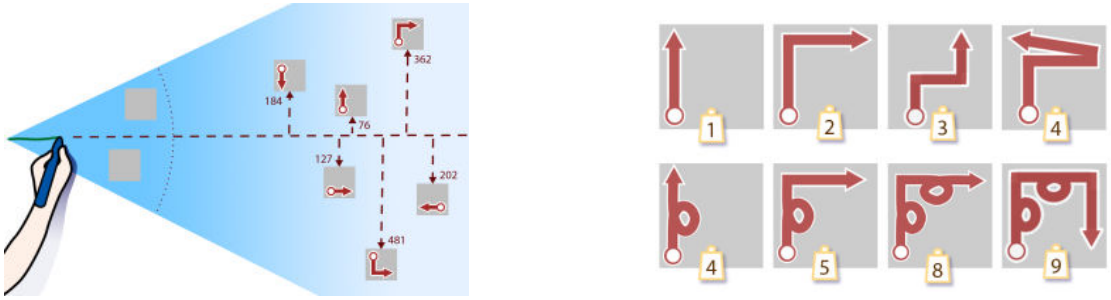


Figure 40: (a) Simpler marks are assigned to targets closest to the center of the region of interest. No gestures are assigned for targets which are very close; (b) Difficulty weights of a range of continuation marks (Bragdon & Ko, 2011).

3.6 Discussion

We group all the aforementioned 2D based selection techniques in Table 3 for comparison. These techniques are characterized according to different aspects. As we have mentioned above, there are several strategies such as magnifying the visual space and switch of CD

ratio that can be used to facilitate object selection. Some techniques use only one strategy while others combine multiple ones synthetically. Although many techniques reduce the selection difficulty, they still require users to adjust the cursor position carefully to locate the target. For instance, the four techniques proposed by Benko et al. (Benko et al., 2006), Shift (Vogel & Baudisch, 2007), ThumbSpace (Karlson & Bederson, 2007) and MagStick (Roudaut et al., 2008), etc. increase the accuracy while the efficiency is not improved significantly. For some other techniques such as Bubble Cursor (Grossman & Balakrishnan, 2005), DynaSpot (Chapuis et al., 2009) and Implicit Fan Cursor (Su et al., 2014), etc., their performances are highly influenced by the density of the cluster. Small objects can be selected easily in a sparse environment but it becomes significantly more difficult when the density increases. Many techniques such as TapTap (Roudaut et al., 2008) and LinearDragger (Au et al., 2014) provide a magnified view to enlarge the targets so that the selection tasks become easier. However, the initial view is replaced or occluded by the magnified view so that the global information is lost. Displaying the larger area in the proximity can be helpful for some applications such as map navigation. Pressure-Activated Lens (Ramos et al., 2007) and FingerGlass (Käser et al., 2011) can display the magnified area beside while the initial view is still visible. However, these techniques are implemented in a large display. There is not enough space to display two different views on a mobile device. Techniques such as Escape (Yatani et al., 2008), Elliptical motion gestures (Fekete et al., 2009), Stroke commands (Appert & Zhai, 2009) and Gestures Select (Bragdon & Ko, 2011) have shown the benefit of using gesture shortcuts for selection. These techniques liberate users from adjusting the selection tool carefully to locate the target. However, if many gestures coexist, more cognitive efforts are required to recognize the corresponding gesture of the target and to reproduce it correctly.

After identifying the limitations of the state-of-the-art techniques, we decided to design a new technique to facilitate the object selection on mobile devices. There are three issues we wanted to overcome. First, we wanted to propose a technique which is efficient and accurate for small object selection in a dense cluster. Users should be able to specify an object quickly without making great efforts. Second, the technique should be less sensitive to the density of the cluster. Third, the objects could be selected easily while the initial view is still displayed. Therefore, in a real scenario, users can make interaction decision with the help of context information. We will present our techniques in Chapter 4.

Table 3: 2D selection techniques.

Technique	Input	Magnified visual space	Magnified motor space	Switch of CD ratio	Reduce occlusion influence	Selection refinement	Disambiguation mechanism	Number of hands	Appropriate for fixed or mobile usage
Offset Cursor (Potter et al., 1988)	Mouse	No	No	No	Yes	No	No	1	Fixed and mobile
Cone-ray (Liang & Green, 1994)	Mouse	No	Yes	No	No	No	No	1	Fixed and mobile
Fisheye (Gutwin, 2002)	Mouse	Yes	No	No	No	No	No	1	Fixed and mobile
Semantic pointing (Blanch et al., 2004)	Mouse	No	Yes	Yes	No	No	No	1	Fixed and mobile
Bubble Cursor (Grossman & Balakrishnan, 2005)	Mouse	No	Yes	No	No	No	Yes	1	Fixed and mobile
Touchpad mapping (Malik et al., 2005)	Touch inputs	No	No	Yes	No	No	No	2	Fixed
Dual Finger Midpoint (Benko et al., 2006)	Touch inputs	No	No	Yes	Yes	No	No	2	Fixed
Dual Finger Slider (Benko et al., 2006)	Touch inputs	No	No	Yes	Yes	No	No	2	Fixed
Dual Finger Stretch (Benko et al., 2006)	Touch inputs	Yes	No	Yes	Yes	No	No	2	Fixed
Dual Finger X-Menu (Benko et al., 2006)	Touch inputs	No	No	Yes	Yes	No	No	2	Fixed
Pressure-Activated Lens (Ramos et al., 2007)	Pen	Yes	No	Yes	Yes	No	No	1	Fixed and mobile
Shift (Vogel & Baudisch, 2007)	Touch inputs	Yes	No	Yes	Yes	No	No	1	Fixed and mobile
ThumbSpace (Karlson & Bederson, 2007)	Touch inputs	No	No	Yes	Yes	No	No	1	Fixed and mobile
Escape (Yatani et al., 2008)	Touch inputs	No	No	No	Yes	Yes	Yes	1	Fixed and mobile
MagStick (Roudaut et al., 2008)	Touch inputs	No	Yes	No	Yes	No	Yes	1	Fixed and mobile
Rub-pointing (Olwal et al., 2008)	Touch inputs	Yes	No	No	Yes	No	No	1	Fixed and mobile
Zoom-tapping (Olwal et al., 2008)	Touch inputs	Yes	No	No	Yes	No	No	2	Fixed
Starburst (Baudisch t al., 2008)	Pen	No	Yes	No	Yes	No	Yes	1	Fixed and mobile

TapTap (Roudaut et al., 2008)	Touch inputs	Yes	No	No	Yes	No	No	1	Fxied and mobile
ARC-Pad (McCallum & Irani, 2009)	Touch inputs	No	No	Yes	No	No	No	1	Fixed
Elliptical motion gestures (Fekete et al., 2009)	Mouse	No	No	No	No	Yes	Yes	1	Fixed and mobile
DynaSpot (Chapuis et al., 2009)	Mouse	No	Yes	No	No	No	Yes	1	Fixed
Stroke Commands (Appert & Zhai, 2009)	Mouse	No	No	No	No	Yes	Yes	1	Fixed and mobile
Click-and-Cross Cursor (Findlater et al., 2010)	Mouse	No	Yes	No	No	Yes	Yes	1	Fixed
Cross-and-Cross Cursor (Findlater et al., 2010)	Mouse	No	Yes	No	No	Yes	Yes	1	Fixed
Motor-Magnifier (Findlater et al., 2010)	Mouse	No	Yes	No	No	Yes	Yes	1	Fixed
Visual-Motor-Magnifier (Findlater et al., 2010)	Mouse	Yes	Yes	No	No	Yes	Yes	1	Fixed
FingerGlass (Käser et al., 2011)	Touch inputs	Yes	Yes	No	Yes	No	No	2	Fixed
Gesture select (Bragdon & Ko, 2011)	Pen	No	No	No	No	Yes	Yes	1	Fixed
Implicit Fan Cursor (Su et al., 2014)	Mouse	No	Yes	No	No	No	Yes	1	Fixed
Bubble Lens (Mott & Wobbrock, 2014)	Mouse	Yes	Yes	No	No	Yes	Yes	1	Fixed
LinearDragger (Au et al., 2014)	Touch inputs	Yes	Yes	No	Yes	Yes	Yes	1	Fixed and mobile

4 3D-based selection techniques

In some interaction scenarios, it is inconvenient to hold a 2D input device in hands. Thus, it is necessary to use other modalities to select virtual objects or contents. For example, in an immersive VR environment, if users use freehand movements to interact inside a CAVE, it is more appropriate to propose a selection technique which takes use of bare hand movements.

In the literature, various metaphors have been proposed for object selection in virtual environments. The concept of using a virtual hand to grab objects directly as in the real world was first proposed by Sturman (Sturman et al., 1989). Using familiar hand actions allows users to adapt to the virtual setup rapidly without making important learning efforts. Poupyrev et al. have designed the Go-Go technique to offer the possibility to acquire objects outside the motor space (Poupyrev et al., 1996). A non-linear mapping from input movements to virtual hand movements allows reaching objects far from the user. Although these techniques are intuitive and easy to use, their performance is restricted by the arm length. Bolt as well as Jacoby et al. have respectively proposed casting a ray in a desired direction for object selection (Bolt, 1980; Jacoby et al., 1994). Because Ray-casting requires the control of only two parameters at any distance, it is one of the most frequently used techniques. However, this technique has also some limitations. The performance of Ray-casting is highly influenced by the visual size of the target. In fact, when trying to select small objects, users have to make extra efforts to stabilize the ray. To obtain a larger region for checking intersection, some occlusion techniques have been proposed (Pierce et al., 1997; Forsberg et al., 1996).

Although the aforementioned early techniques are natural to use, it is difficult to select a small object in the environment with the help of these techniques. Like 2D-based selection techniques, several strategies have been applied to propose reliable and efficient selection techniques. In this section, we will present the strategies found in the literature and introduce their corresponding techniques.

4.1 Switching of CD ratio

One way to simplify 3D selection and make the selection tasks less demanding is to provide the possibility of switching the CD ratio. When the selection tool requires to be moved rapidly in the ballistic phase, a small CD ratio can help users cover larger distance. Because the main purpose of the ballistic movement is to approach the target, the control precision is less important. After the selection tool is close to the target, it is better to switch to a bigger CD ratio to perform precise corrective movements. When inputs are made by bare hands or a device without physical support, a bigger CD ratio can also help reducing the influence of tremble. According to König et al. (König et al., 2009), CD ratio-based techniques can be classified into three groups: Manual Switching, Target Oriented and Velocity Oriented techniques.

4.1.1 Manual Switching techniques

Manual Switching techniques allow users to adjust the CD ratio manually. Vogel and Balakrishnan have developed a technique which is called Hybrid RayToRelative to enable users to use bare hand movements to interact with a large display (Vogel & Balakrishnan, 2005) (Figure 41). Users can use one hand, which is instrumented by several markers for tracking, to control a cursor for object selection. Instead of using RayCasting which maps hand movements to cursor motions by a fixed CD ratio, this technique allows users to perform a clutch gesture to switch the CD ratio. Two manipulation modes are provided: absolute mode and relative mode. When users perform a pointing gesture on the tracked hand, instead of displaying a cursor, a large circle whose center is the intersection point is displayed and follows the hand movement. In the absolute mode, the circle can be manipulated to specify an approximate area. To perform precise control, users can open the hand by stretching other curved fingers. After transforming the circle into a cursor, it can be manipulated relatively with respect to the circle center. The CD ratio in the relative mode is smaller than that in the absolute mode so that the cursor can be adjusted carefully to the desired target. A user study has shown that although this technique requires longer selection time, users made significantly fewer errors than using RayCasting. Most of the participants also stated that they preferred RayToRelative and commented that it is easier to perform.

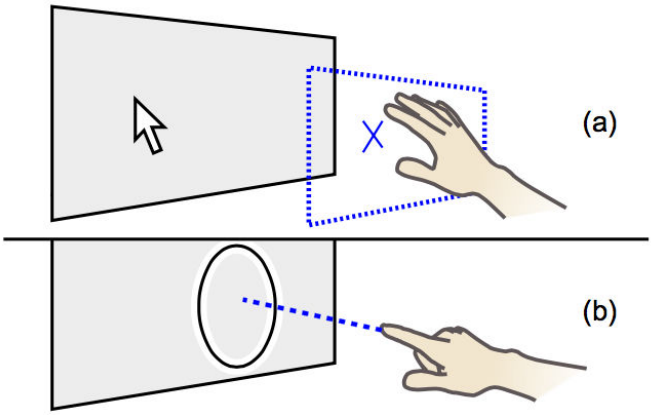


Figure 41: Hybrid RayToRelative Pointing. (a) The open hand is used for relative cursor control, and (b) recalibrating (or clutching) is performed with an absolute ray cast pointing gesture (Vogel and Balakrishnan, 2005).

Nancel et al. have also proposed several hybrid selection techniques which provide both absolute and relative modes (Nancel et al., 2011a). Instead of using bare hand movements, a handheld device is required to control the selection tool. Both Laser+Position and Laser+Gyro use a mouse as the input device. When the user moves the mouse in space without clicking a button, the cursor in the display is controlled by the RayCasting technique. However, when the right button is clicked, Laser+Position allows users to control the cursor in relative mode by translating the mouse in space (Figure 42) while Laser+Gyro maps relative rotational movements to precise cursor motions. They have also

proposed Laser+Track which uses a smartphone. After setting a coarse position of the cursor on the display, users can refine the cursor position by making touch inputs on the screen of the smartphone. A user study has shown that all these three techniques outperform RayCasting regarding both pointing time and error rates.

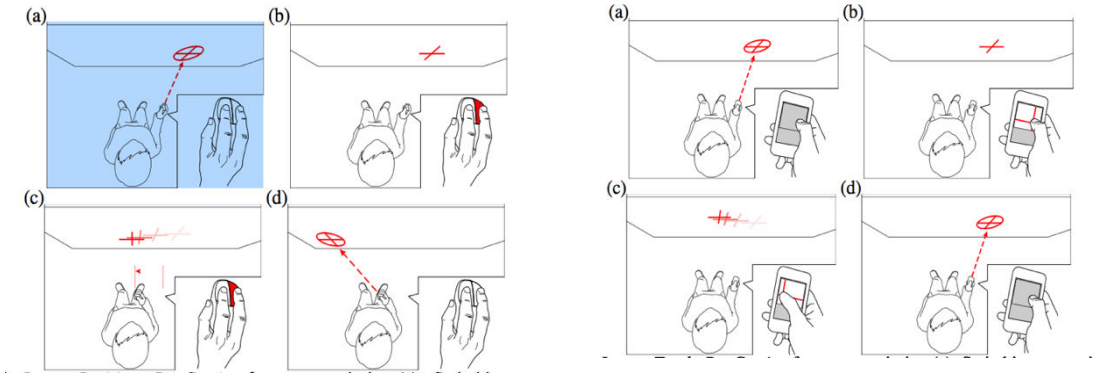


Figure 42: (Left)Laser+Position; (Right) Laser+Track (Nancel et al., 2011a).

Nancel et al. have proposed a hybrid technique for object selection on a large display (Nancel et al., 2013). The authors have tried using the head orientation to control the cursor in Coarse mode. An indirect mapping from the head orientation to the cursor location is applied for two reasons. First, being perspective-based, ray-casting would have caused targets of the same absolute size to have noticeably different motor sizes depending on their location with respect to the user's physical position. This mapping allows any area of the display to be reached with equal motor precision. The second reason is to optimize the input operating range, within the limits of comfortable neck positions. They have also proposed using touch inputs of two joint fingers on the mobile device to control the cursor in Coarse mode. When the cursor is close to the target, users can drag one finger on the mobile device to control the cursor precisely (Figure 43). To study whether the cursor movement can be further accelerated, cursor movements can be made in two different ways in Coarse mode. When Continuous configuration is activated, the cursor moves continuously in the display. However, Discrete configuration divides the display into several isotonic rectangles and the cursor can jump directly between adjacent rectangles. The main objective of the Discrete configuration is to avoid the ballistic movement. A controlled experiment demonstrated that using the head orientation can be as efficient as the use of a mobile device. In addition, participants commented that using the head movement to control the cursor was less tiring. Another finding was that Continuous configuration outperforms Discrete configuration. Although Discrete configuration helped to limit cursor crossing distance, it was more difficult to follow the cursor visually.

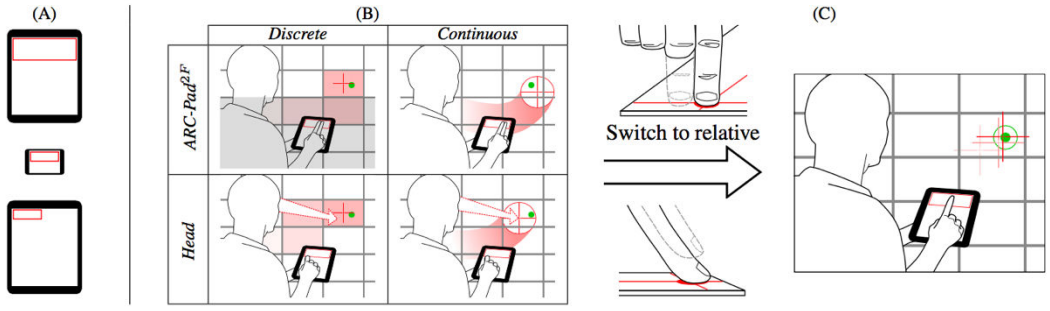


Figure 43: The experiment of high-precision pointing on a large display (Nancel et al., 2013).

4.1.2 Target oriented techniques

The approach of target oriented techniques is to reduce the CD ratio automatically when the selection tool is in the proximity of an object. Andujar and Argelaguet have proposed a target oriented technique for remote selection of 2D GUIs in a virtual environment (Andujar & Argelaguet, 2007). To simplify interaction with 2D GUIs, this technique can adjust the CD ratio automatically according to the width and height of the GUI. After an initial hand orientation is specified, when the orientation difference between the current hand orientation and the initial one is smaller than a threshold, the CD ratio is modified automatically according to the proposed formulation. Because the cursor does not follow the hand direction, the projected ray is bended to connect the hand and the cursor (Figure 44). When the orientation deviation from the initial one is larger than 45º, the projected ray becomes back to a straight line and the CD ratio is reset to a constant value.

Figure 44: Several rays corresponding to different orientations of the input device are shown (Andujar & Argelaguet, 2007).

Ouramdane et al. have proposed the Follow-Me approach to guide object selection and manipulation in a virtual environment (Ouramdane et al., 2006). In this approach, the virtual environment is divided into three parts: Free Manipulation Zone, Scaled Manipulation Zone and Precise Manipulation Zone (Figure 45). When the selection tool is

not in the proximity of any object, it is in the Free Manipulation Zone. For this zone, the authors chose the mapping equation between device inputs and selection tool outputs associated to the Go-Go technique (Poupyrev et al., 1996). If the selection tool gets closer to an object, it enters the Scaled Manipulation Zone centered on the object. In this zone, the CD ratio increases so that the joystick movement is mapped to slower selection tool movement. The CD ratio is stable and does not evolve with the distance to the object. When the selection tool gets even closer to the object, it enters the Precise Manipulation Zone. In this zone, the direction of the selection tool is restricted so that it can move forward or backward only in the direction of the object. This mechanism ensures that the object can be acquired with less demand for precision.

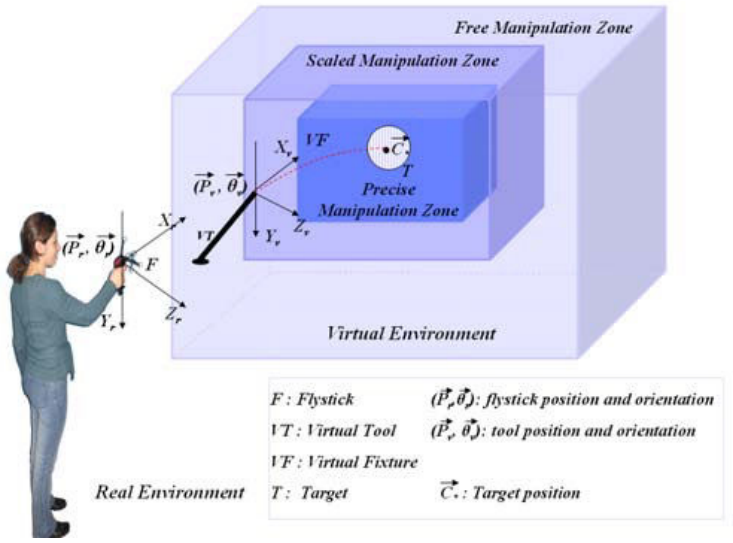


Figure 45: Follow-Me segments the space around a target into three different zones (Ouramdane et al., 2006).

4.1.3 Velocity oriented techniques

Velocity oriented techniques dynamically adjust the CD ratio according to the input device speed. This is similar to the CD ratio formulation of the mouse applied in graphic interfaces. According to the Optimized Initial Impulse Model (Meyer et al., 1988), the cursor accelerates when it requires to cross a large distance to approach the target while it decreases the speed to adjust carefully its position. Based on this model, velocity oriented techniques decreases the CD ratio to allow panning the cursor with a higher speed during the Ballistic phase and increases the ratio to slow down the cursor for the Corrective phase.

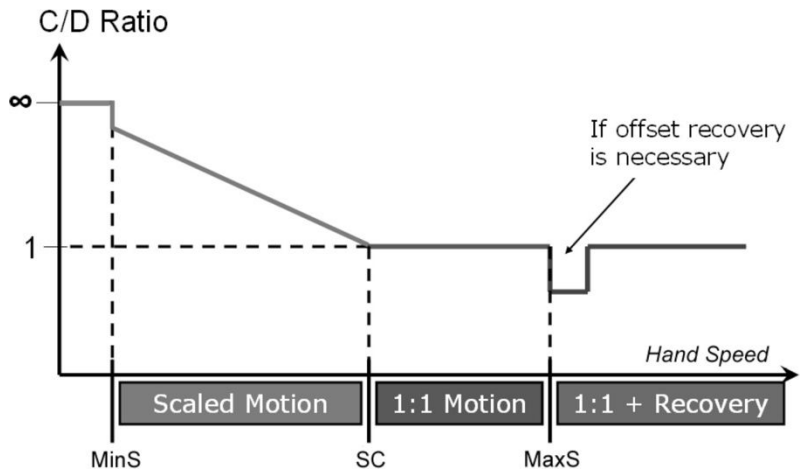


Figure 46: Simplified interface diagram showing how PRISM uses Hand Speed to adjust the CD ratio (Frees et al., 2007).

Frees et al. have proposed a formulation to switch the CD ratio according to the input velocity in 3D space (Frees et al., 2007). As shown in Figure 46, when the input velocity is smaller than MinS, it is considered as a noise. Thus, the selection tool remains static. When the input velocity is bigger than MinS and is smaller than SC, the CD ratio decreases proportionally with the hand speed. When the hand speed reaches SC and continues to increase, the CD ratio remains at its minimum value. One problem caused by the changes of the CD ratio is that changes of the CD ratio introduce a spatial offset between the input device and the hand. The mismatch between the position of the input device and that of the output device becomes obvious after a sequence of interaction. In order to overcome this issue, when the hand speed reaches MaxS, the CD ratio is switched to a value smaller than 1. The CD ratio does not return back to 1 until the offset is recovered.

König et al. have proposed another velocity oriented technique which applies a similar formulation of transition (König et al., 2009). Instead of considering device movements of low speed as input noise, a minimum CD gain is applied when the hand speed is smaller than v(min) (Figure 47). When the speed value is between v(min) and v(max), the CD gain increases proportionally with the hand speed and it reaches the maximum value when the hand speed reaches v(max). It is to be noted that g(max) is greater than 1, so that the offset can be recovered more smoothly to avoid high discrepancies. Similar to PRISM, Adaptive Pointing help to save time for crossing a long distance at high speed and improve pointing accuracy during the corrective phase.

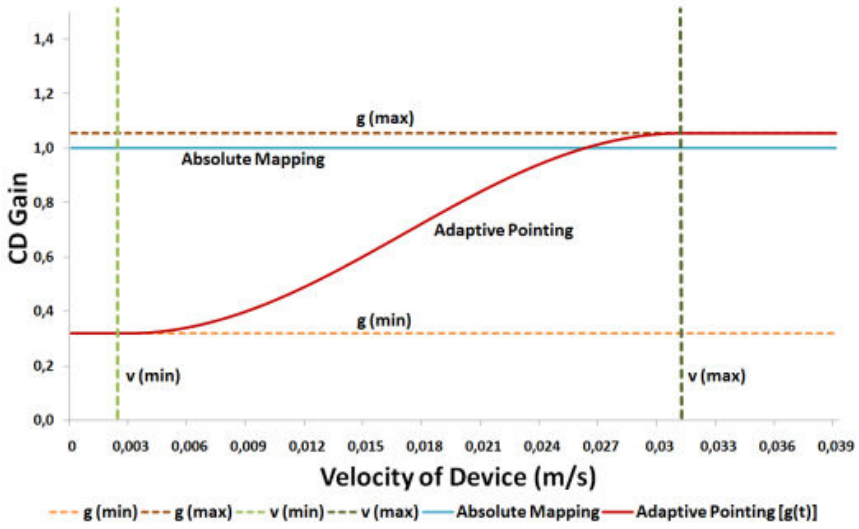


Figure 47: Smooth transition between relative and absolute CD gain of Adaptive Pointing (König et al., 2009).

4.2 Target expansion

One way to simplify 3D selection is to expand targets in the visual space or motor space. Target expansion increases the effective width of the target by substituting its initial size by a larger region of interest. This approach enables users to locate the target with less precision.

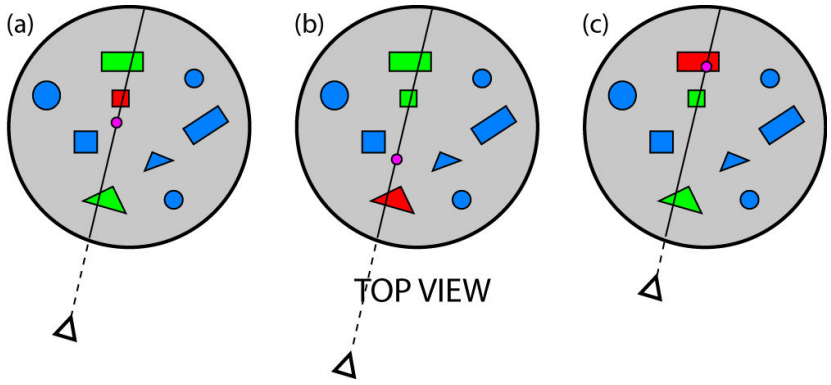


Figure 48: The Depth Ray technique (Grossman & Balakrishnan, 2006).

Grossman and Balakrishnan have compared several selection techniques in a 3D volumetric environment (Grossman & Balakrishnan, 2006). In their work, they have proposed Depth Ray which enables users to select an object from all those passed by the ray line. When the user adjusts the ray position and orientation, a pink depth marker can be moved forward or backward along the ray simultaneously (Figure 48). The system examines the distance between the depth marker and all the targets intersected by the ray. The target with the smallest distance to the depth marker is highlighted. If the hand is

moved forward or backward, the depth marker can be used to scan all the intersected targets. Compared to positioning the depth marker precisely to one target, it is much easier to perform a selection by dropping the marker in the corresponding Voronoi region of the target.

Vanacken et al. have extended the concept of Bubble Cursor to simplify object selection in a 3D virtual environment (Vanacken et al., 2007). The selection tool is a sphere-shaped transparent cursor whose radius evolves dynamically to cover the closest target. The space is divided into numerous regions by applying the Voronoi diagram. A controlled experiment has been conducted to compare RayCasting, 3D Bubble Cursor and Depth Ray. The results show that both Bubble Cursor and Depth Ray outperformed RayCasting and Depth Ray had the best performance regarding selection time.

Steinicke et al. have proposed the Sticky-Ray technique to simplify target selection (Steinicke et al., 2006). Similar to RayCasting, a ray is first projected to the scene following the finger pointing direction to check object intersection. If no object is crossed by the ray, a cone is used in substitution to enlarge the checking region. When there is still no object covered by the cone, the zone is enlarged until an object is found or the whole scene has been examined (Figure 49). When there is more than one object intersected by the cone, the closest object to the initial ray is selected as the active target. If the target is not crossed by the straight ray in the pointing direction, the ray is bent to reach the target.

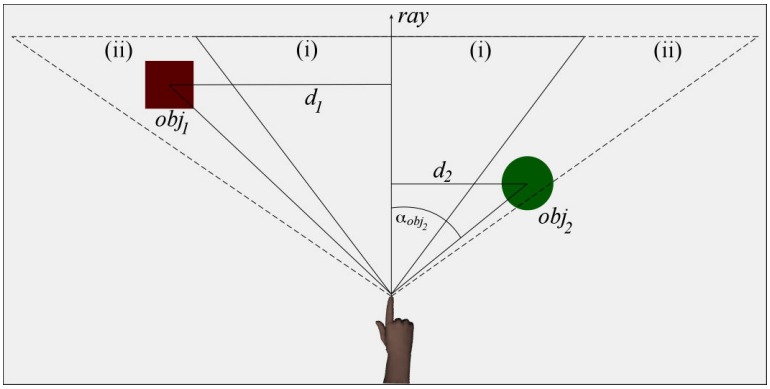


Figure 49: Sticky-Ray first increases the size of the selection volume from (i) to (ii) to check object intersection (Steinicke et al., 2006).

Delamare et al. have designed two two-step techniques to simplify pointing of objects in an augmented reality environment (Delamare et al., 2013). This first technique is called P2Roll and the second one is called P2Slide. In the first step, both P2Roll and P2Slide permit orienting a smartphone in space to set a cone-shape selection volume. When a finger is tapped on the screen of the smartphone, the selection volume is fixed and all objects covered by the volume are pre-selected. To make disambiguation, among all preselected objects, P2Roll enables users to roll the smartphone in the motion range [-80º, 50º] to scan all the targets while P2Slide allows dragging the finger horizontally on the screen to make disambiguation (Figure 50). In the disambiguation process, P2Roll divides the rolling

range into several equal sub-ranges. Each sub-range represents one preselected object and its angular size is used as the effective width of the object. Similarly, P2Slide uses sub-ranges of the sliding range as substitution of preselected objects. Because visual feedback can be augmented in the physical world, users can interact with the smartphone blindly. According to a user study, P2Roll and P2Slide improve object selection because they help users keep their focus in the physical world.

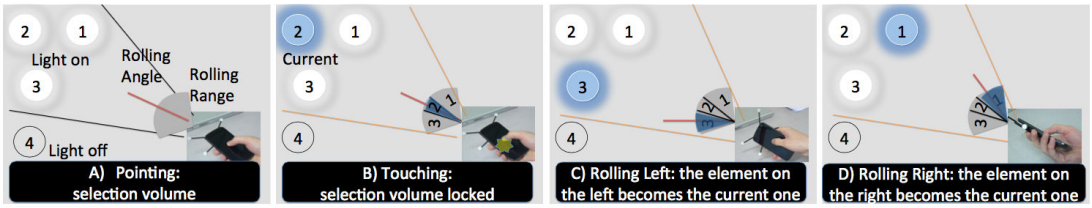


Figure 50: The workflow of P2Roll (Delamare et al., 2013).

4.3 Marking menu

To make object selection less affected by the target distance and effective width, several research works have been done to explore using marking menus to select objects. Kopper et al. have proposed the Sphere-casting refined by QUAD-menu (SQUAD) (Kopper et al., 2011). After specifying a selection volume by casting a sphere in the virtual environment, a QUAD-menu is displayed to separate objects covered by the volume into four quadrants (Figure 51).

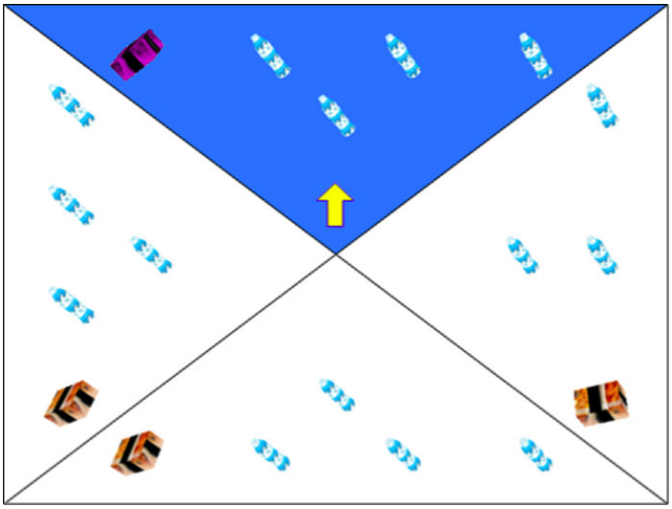


Figure 51: QUAD-menu is used to divide objects into four quadrants (Kopper et al., 2011).

After selecting the quadrant in which the target is located, all the objects inside the selected quadrant are further separated into four new quadrants. Users can repeat this progressive refinement until each quadrant contains no more than one object. In the last step, the target can be selected by choosing its quadrant. An experiment has been conducted to explore the trade-offs between progressive refinement and immediate selection techniques. It was

shown that SQUAD is much more accurate than RayCasting, and faster than RayCasting with small targets and less clustered environments.

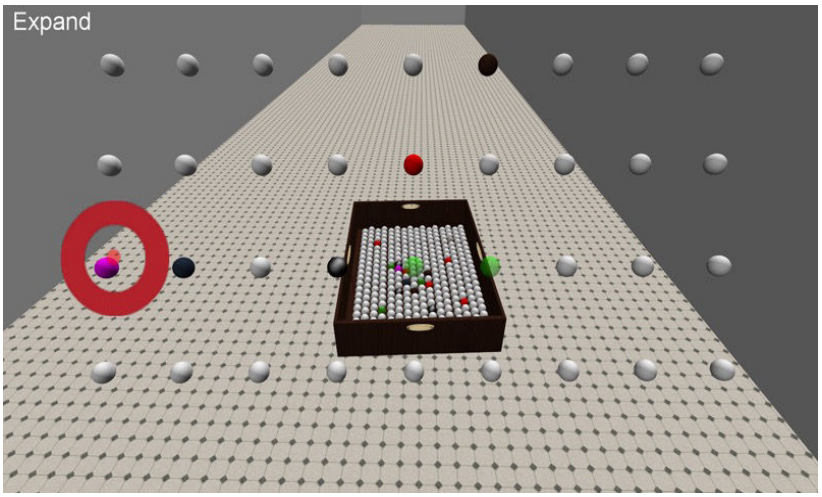


Figure 52: Expand displays objects in a new configuration of grid (Cashion et al., 2012).

Cashion et al. have made an experiment to study the performance of four selection techniques in game-based virtual environments in different conditions (Cashion et al., 2012). The Zoom technique and the Expand technique which are proposed by the authors are compared to RayCasting and SQUAD. The Zoom technique allows users to zoom up the region around the cursor to magnify objects. Similar to SQUAD, Expand displays objects in a new configuration of grid to decrease the influence of the cluster density (Figure 52). However, to avoid losing original context, Expand still keeps the original view. To have a comprehensive understanding of these techniques, five selection scenarios of different object densities and object motions have been tested. Through the experiment, it was found that both RayCasting and SQUAD have weaknesses in terms of speed and accuracy in dense and dynamic environments. The authors argued that RayCasting remains a good technique under normal conditions and SQUAD remains accurate and fast as long as the object density remains relatively low. However, Expand outperforms RayCasting and SQUAD under difficult conditions.

Grossman and Balakrishnan have proposed Flower Ray to make selection refinement (Grossman & Balakrishnan, 2006). Users can first project a ray into the 3D scene to intersect objects. After activating the hand held device, all intersected targets are flowered out into a marking menu. Then a cursor can be moved to pick up the target in the making menu. This technique simplifies the selection task by reducing the cluster density. However, the initial context is lost. Thus, more time is needed to perform visual search for the target in the marking menu. For this reason, Flower ray requires less time during selection phase than Depth ray, but consumes more time during disambiguation phase. In sum, Flower ray is less efficient than Depth ray.

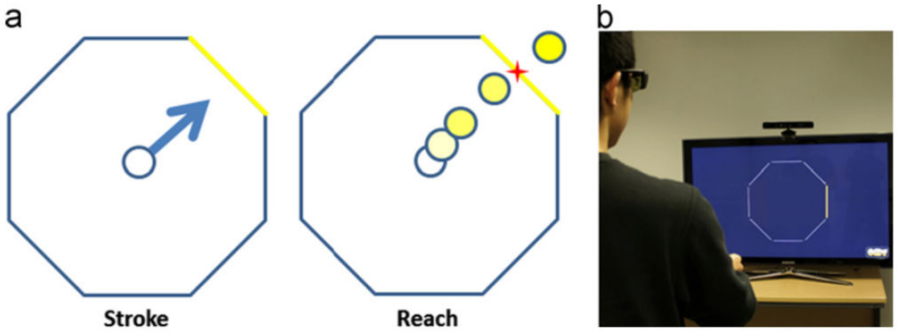


Figure 53: Stroke and Reach selection techniques (Ren & O'Neill, 2013).

Ren and O'Neill have explored using freehand movements in air to select objects displayed at a distance (Ren & O'Neill, 2013). They have proposed the Stroke and the Reach techniques to perform disambiguation in a marking menu (Figure 53). After several potential objects have been specified by a cone-shaped ray, an octagon menu whose edges are used to represent different options is displayed. Stroke permits drawing a stroke in the direction of one edge to select the corresponding option, while Reach requires moving the cursor to reach one edge to trigger selection. The experiment has shown that Stroke is faster. However, the Reach technique generates fewer errors. To further explore the use of the marking menu, the authors have extended the 2D marking menu into a 3D marking menu with 16 options. Through an experiment, it was demonstrated that the performance of marking menu is significantly affected by the direction of options. This performance discrepancy is determined by the anatomic structure of the hand and the wrist.

Figure 54: Finger-Count menu (Kulshreshth & LaViola Jr, 2014).

Because people are familiar with using finger-count gestures to express numbers, Kulshreshth and LaViola Jr have explored using these gestures as command shortcuts

(Kulshreshth & LaViola Jr, 2014). In Finger-Count menu, all the menu items are numbered and the user has to extend a corresponding number of fingers to select a given item (Figure 54). Finger-Count menu supports using both hands simultaneously so that the menu can display at most 10 items. In case there are more items, the authors have proposed labeling the last item as "Next". Users can select the last item to replace original items by new ones. A user study has been made to compare the performance of Finger-Count menu with 3D Marking menu. The results indicate that Finger-Count menus and 3D Marking menus have similar selection accuracy, but Finger-Count menus are almost twice faster than 3D Marking menus.

4.4 Discussion

Similar to 2D based selection techniques; we also group 3D based selection techniques according to a group of different aspects on Table 4. As Table 4 shows, most of the techniques can be divided into two groups according to the input device. In the first group, the techniques allow using a hand held device to control the selection tool while the other techniques depend on the use of freehand gestures. The benefit of using a hand held device or a smartphone is that physical and virtual buttons can serve as effective forms of inputs to trigger an event or switch the operation mode. However, hand held devices are used frequently in indoor environments because precise tracking devices are required to locate them. In addition, it is inconvenient for users to carry hand held devices in a mobile environment. Therefore, freehand interaction is more suitable for mobile usage.

Freehand interaction techniques in the literature use similar strategies to hand held device based techniques to facilitate object selection. For instance, like Laser+Gyro and Laser+Track (Nancel et al., 2011a), Hybrid RayToRelative allows users to switch the CD ratio (Vogel & Balakrishnan, 2005) by switching the freehand gesture. Similar to Flower Ray (Grossman & Balakrishnan, 2006), Stroke and Reach allow using a marking menu to refine the selection (Ren & O'Neill, 2013). As the P2Slide reduces the selection difficulty by allowing users to control only one parameter to refine the selection (Delamare et al., 2013), we also wanted to explore the effect of this strategy for freehand selection. In addition, to switch the operation mode, many freehand techniques require users to perform a gesture explicitly. However, we thought the experience can become more natural if the interface can infer the intention of the user and switch the operation mode automatically when necessary. The automatic switch of operation can also reduce the influence of hand trembling which is introduced when the gesture is performed. Although many studies have been conducted to propose new selection techniques, these studies did not pay much attention to how visual guides can help simplifying the selection. Thus we decided to study how visual guides can influence the performance of object selection as well. We will present our 3D selection technique in chapter 5.

Table 4: 3D selection techniques.

Technique	Input	Selection tool	Mode switch	Magnified motor space	Switch of CD ratio	Disambiguation mechanism	Selection refinement
Virtual hand (Sturman et al., 1989)	Hand held device	Hand avatar	--	No	No	No	No
Ray-Casting (Jacoby et al., 1994)	Hand held device	Ray	--	No	No	No	No
Aperture based selection (Forsberg et al., 1996)	Hand held device	Adjustable cone	--	Yes	No	Yes	No
Go-Go (Poupyrev et al., 1996)	Gestures in air	Hand avatar	--	No	Yes	No	No
Image plane interaction (Pierce et al., 1997)	Gestures in air	Ray	--	Yes	No	No	No
Hybrid RayToRelative (Vogel & Balakrishnan, 2005)	Gestures in air	Ray	Explicit gesture	No	Yes	Yes	No
Follow-me (Ouramdane et al., 2006)	Hand held device	3D Cursor	Automatic	Yes	Yes	Yes	No
Sticky-Ray (Steinicke et al., 2006)	Hand held device	Cone	--	Yes	No	Yes	No
Depth Ray (Grossman & Balakrishnan, 2006)	Hand held device	Ray & 3D cursor	Click the button of the device	Yes	No	Yes	Yes
Flower Ray (Grossman & Balakrishnan, 2006)	Hand held device	Ray	Click the button of the device	No	No	Yes	Yes
Anisomorphic ray-casting (Andujar & Argelaguet, 2007)	Hand held device	Ray	--	Yes	No	Yes	No
PRISM (Frees et al., 2007)	Hand held device	Ray	--	No	Yes	No	No
3D Bubble Cursor (Vanacken et al., 2007)	Hand held device	Adjustable sphere	--	Yes	No	Yes	No
Adaptive Pointing (König et al., 2009)	Hand held device	Ray	--	No	Yes	No	No
SQUAD (Kopper et al., 2011)	Hand held device	Ray & Sphere cursor	Click the button of the device	No	No	Yes	Yes
Laser+Position (Nancel et al., 2011a)	3D mouse	Ray	Click the button of the device	No	Yes	No	Yes
Laser+Gyro (Nancel et al., 2011a)	3D mouse	Ray	Click the button of the device	No	Yes	No	Yes
Laser+Track (Nancel et al., 2011a)	Smartphone	Ray	Touch the screen	No	Yes	No	Yes

Expand (Cashion et al., 2012)	Hand held device	Ray & Circular cursor	Click the button of the device	No	No	Yes	Yes
Head orientation (Nancel et al., 2013)	Head orientation + Smartphone	Ray	Touch the screen	No	Yes	No	Yes
P2Roll (Delamare et al., 2013)	Smartphone	Ray	Touch the screen	No	No	Yes	Yes
P2Slide (Delamare et al., 2013)	Smartphone	Ray	Touch the screen	No	No	Yes	Yes
Stroke & Reach (Ren & O'Neill, 2013)	Gestures in air	Cone	Explicit gesture	No	No	Yes	Yes
Finger-Count menu (Kulshreshth & La Viola Jr, 2014)	Gestures in air	Finger-count gestures	--	No	No	Yes	Yes

5 Conclusion

In this chapter, we have presented the state-of-the-art techniques which are proposed for object selection. Selection techniques are divided into two groups according to the input dimensions: 2D and 3D selection techniques. For 2D selection, we have mainly presented mouse-based and touch-based selection techniques. Based on different strategies, many techniques have been designed to simplify the selection of small objects and overcome the occlusion issue. For 3D selection, we have mainly presented techniques which rely on the use of handheld devices or freehand movements to select objects at a distance. Similar to 2D selection techniques, 3D selection techniques were also divided into different groups according to the selection strategy. Although a lot of 2D and 3D selection techniques have been proposed to simplify object selection, the previous literature review suggests that it is still possible to improve the selection experience both regarding completion time and accuracy.

In the next chapter, we will present a state-of-the-art of manipulation techniques. Although standard transformation widgets are frequently used for object manipulation on desktop 3D applications, it is difficult to manipulate 3D virtual objects efficiently by using touch inputs and freehand gestures in air. The difficulty is mainly caused by the lack of precise input channels. We will present how virtual objects can be manipulated by using the different techniques found in the literature.

Chapter 3: Manipulation techniques

Chapter 3: Manipulation techniques

1 Introduction

Manipulation is one the most basic tasks of 3D interaction. It mainly aims to specify the position, orientation and scale of one or multiple selected objects. Objects manipulations are frequently performed, especially for constructing scenes of video games and sketching 3D models. Because it occurs frequently, it is important to provide efficient and precise techniques for object manipulation. In this chapter, we will present the state of the art of manipulation techniques. This will mostly cover two categories of techniques: 2D input-based techniques and 3D input-based techniques. Hybrid techniques that use both 2D and 3D inputs will also be mentioned and discussed.

2 2D input-based

2D input-based manipulation refers to techniques that rely on input devices which can only control two parameters at a time. Although 2D input devices such as the mouse and the touchscreen are widely used to interact with desktop computers or mobile devices, they have one inherent problem when used for manipulating 3D objects. In fact, to manipulate an object in the 3D space, it is necessary to control 9 DOF Degrees Of Freedom (DOFs): 3 translations, 3 rotations and 3 scalings. However, 2D input devices can only support the control of two parameters at a time. This raises the issue on how to map 2D inputs to achieve 3D transformations. Some techniques permit switching the manipulation mode while others combine a set of gestures to cover all manipulation tasks. To facilitate the comparison of the different techniques, we propose a classification according to their intrinsic characteristics. Similar to the classification made in (Argelaguet & Andujar, 2013), we classify manipulation techniques considering both the underlying device and how it is used by the users. We will describe the factors used for classification.

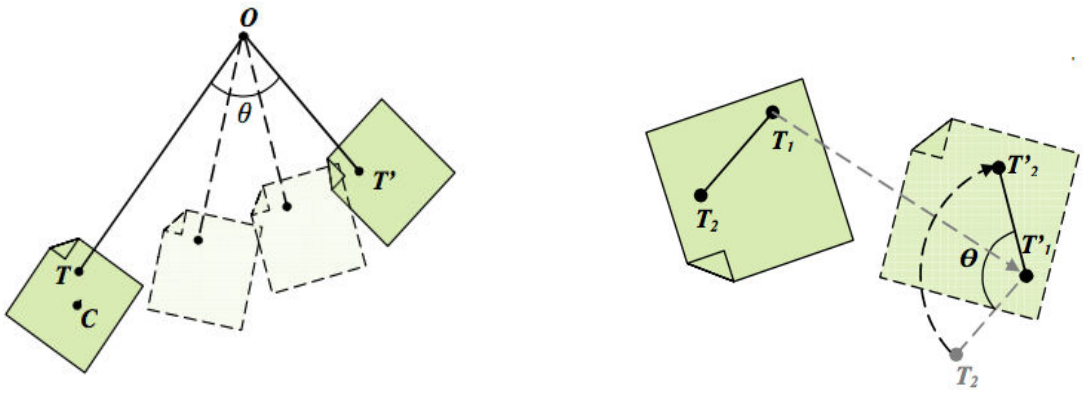
2.1 Degrees Of Freedom (DOF)

The first factor that can be used to classify the different techniques found in the literature is the DOFs of manipulation. The number of DOFs provided by one technique can be different from another. It varies according to the interaction scenario and the task requirements.

2.1.1 Planar manipulation

Several state-of-the-art techniques only provide the possibilities to manipulate objects in the plane parallel to the screen. In the work of Wu and Balakrishnan, multiple users can

share a tabletop display for collaborative interaction (Wu & Balakrishnan, 2003). For a furniture docking application, object translation can be made in the screen plane while only rotation around the axis perpendicular to the screen can be performed. Hancock et al. have also developed a collaborative touch-based manipulation technique to simplify data sharing among multiple users (Hancock et al., 2006). In this study, five translation and rotation techniques have been investigated to determine the suitable scenario for each technique. Users have at most 3 DOF: x translation, y translation and z rotation. Some techniques such as Independent Translation & Rotation allow users to change the object position and orientation independently while Automatic Orientation and Integral Rotation & Translation combine translation with rotation and these techniques can be performed using only one touch input (Figure 55).



(a) (b)

Figure 55: (a) Continuous automatic rotation. (b)Two-point rotation (Hancock et al., 2006).

2.1.2 Spatial manipulation

In some 3D environments, it is necessary to provide more DOF for object manipulation. Martinet et al. have proposed the Z-technique for 3D positioning (Martinet et al., 2010a). 3D translation is decomposed into a 1D and a 2D component. When one finger is dropped on an object, it can be translated freely in the screen plane. After tapping the first finger, dragging the second finger up or down in the empty space of the screen can push or pull the object in the third direction (Figure 56).

Besides 3D translation, 3D rotation can also be performed on touchscreens. Kin et al. have developed an application, which is named Eden, for constructing virtual organic environments (Kin et al., 2011). This application provides a set of gestures for various manipulation operations: x-y translation, z translation, arcball rotation, local z rotation, world z rotation, uniform scale, one-dimensional scale and throw-and-catch. All 9 DOF of manipulation are supported by this application. Moreover, three kinds of rotation

techniques are included so that users can choose the most appropriate one according to their needs. The redundancy in rotation allows uses to change the object orientation more freely.

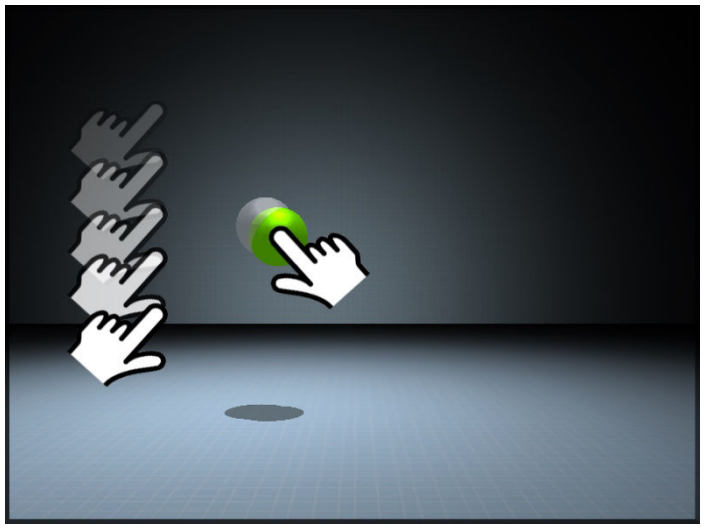


Figure 56: The Z-technique (Martinet et al., 2010).

Although it is easier for humans to separate an arbitrary translation to 3 orthogonal components, it is difficult to decompose intuitively an arbitrary rotation. Scheurich and Stuerzlinger have proposed a one-handed method which is capable to perform automatic rotation to align one object to another (Scheurich & Stuerzlinger, 2013). To make one object snap at another one, the user needs to simply tap these two objects in a consecutive order. After that, the first object is translated and rotated so that the pressed points of two objects overlap each other and the pose of the first pressed face is adjusted to match with the second one. This technique allows controlling 6 DOF, but users cannot modify the object position and orientation manually.

2.2 Input devices

The second factor that can be used for the classification of manipulation techniques is the input device. Indeed, the design of one manipulation technique is highly influenced by the characteristics of the chosen input device and can be very different from techniques which use other input devices. In the literature, the mouse and the touchscreen are two input devices that are widely used for object manipulation. Each of these two devices has its own advantages and disadvantages. In the following, we will classify the techniques according to these two devices.

2.2.1 Mouse-based

Nowadays, most of the commercial 3D modelling and CAD applications such as Maya (Autodesk) and Sketchup (Google) depend on the mouse-based paradigm. On the one hand,

the mouse is an easy-to-use and accurate tool for 2D user interfaces. On the other hand, its limited number of inputs makes it difficult to take full control of a 3D object. Because, only two parameters can be controlled at the same time by the mouse, one common solution is to provide a set of tools, dialog boxes, menu items, or other kinds of 2D UI elements for switching manipulation modes (Jankowski & Hachet, 2013) (Figure 57). A manipulation mode determines the command-mapping from 2D inputs to 3D outputs. Users can choose the desired manipulation mode by clicking the corresponding button and typing the object position or orientation in dialog boxes. However, a more intuitive way is to use a transformation manipulator.

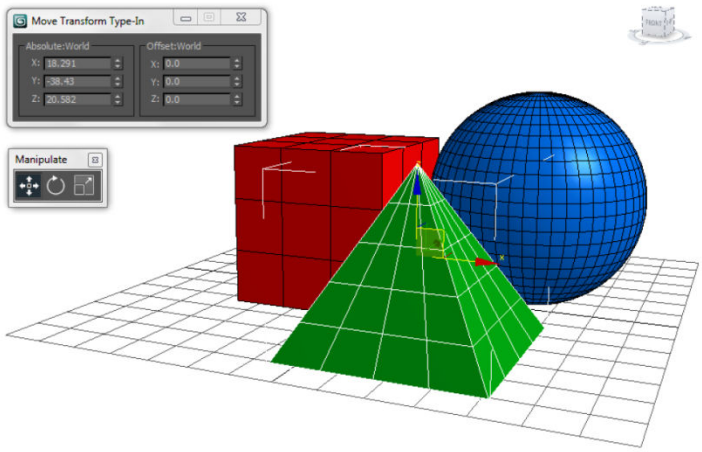


Figure 57: The same functions presented with tools and with a dialog box (left), and with a manipulator (here controlling the green pyramid) (Jankowski & Hachet, 2013).

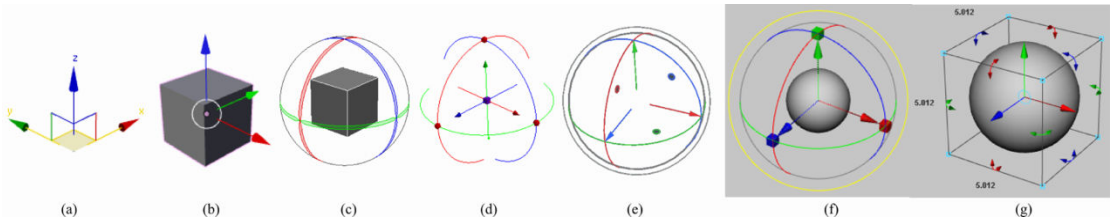


Figure 58: 3D transformation widgets used in different systems. Translation widgets from 3DS Max (a) and Blender (b), Rotation widget from XSI (c), and combo-widgets from Houdini (d), Modo (e), and Maya (f,g). While the visual design varies, functionality is largely identical (Schmidt et al., 2008).

Straus et al. have defined manipulators, which are displayed upon the selected object, as visible graphic representations of an operation on, or state of that object (Strauss et al., 2002) (Figure 58). A manipulator usually consists of several interactive widgets which can be clicked and dragged to control the selected object in the desired way. Most of the manipulators provide three handle bars or cone-shape widgets for translation (Strauss et al., 2002). Each of the translation components is parallel to a principal axis of the reference. This technique originates from the work done in (Nielson & Olsen Jr, 1987) and (Bier, 1987). In addition to axis-constrained translation widgets, some manipulators also display a small rectangle widget between each two principal axes for plane-constrained translation

(Figure 58 (a)). Clicking and dragging a rectangle can translate the object in the corresponding plane. A common approach to perform rotation is to use a virtual ball with several bands encircling the selected object. Dragging one band rotates the object around its perpendicular axis. This design is based on the work of Nelson and Olsen (Nielson & Olsen Jr, 1987) and of Bier (Bier, 1987), and is similar to camera navigation techniques proposed in (Chen et al., 1988) and (Shoemake, 1992). Instead of bands, some techniques provide several arrow-shaped widgets for rotation (Döllner & Hinrichs, 1997) (Figure 58 (g)). Similar to translation, it is a common design to provide three scaling widgets which can be dragged in the principal direction to modify the scale of the selected object (Strauss & Carey, 1992; Conner et al., 1992). For some manipulators, only one manipulation mode can be active at a time. Thus, it is necessary to switch the manipulation mode by clicking buttons or making keyboard shortcuts. Some other manipulators, such as that of Maya, display all the rotation, scaling and translation (RST) widgets together (Figure 58 (f) (g)). In Straus et al.'s notes from SIGGRAPH 2002 course on "*The Design and Implementation of Direct Manipulation in 3D*", there are also useful recommendations for manipulator behavior and look.

Some of the advantages of manipulators are listed in the work of (Strauss et al., 2002):

(1) Manipulators are located in the scene with the objects they control. When users edit these objects, their focus of attention stays with the object, not off to the side with the tools. This reduces the amount of mouse traffic, and reduces mode errors.

(2) The users have a number of different controls available at the same time. Thus, they can perform any one of several related operations at any time without an extra click to change tools. This cuts the number of clicks significantly and reduces memory load, since all the possible controls are displayed where they are needed.

(3) This solution allows the users to separate the desired 3D movement into simpler 1D or 2D components.

(4) Manipulators can graphically show what they are operating on and how they will work. They also show what operations are possible, in a given context, and can give the users additional feedback about intermediate states.

(5) Manipulators invite experimentation. They make using the software more enjoyable.

2.2.2 Touch-based

Manipulators for desktop 3D interfaces benefit from several good features of the mouse such as accurate pointing and an unobstructed view of the screen (Jankowski & Hachet, 2013). Using manipulators designed for desktop interfaces on touchscreens can be heavy because touch-based interaction suffers from the "fat finger" problem and a relatively low accuracy. Due to the size of the fingertip, it is more difficult to locate precisely the manipulator using touch inputs. The occlusion issue also makes it hard to observe the

transformation, especially when the target is of small size. Although providing magnified manipulators can simplify the manipulation task, magnified manipulators occupy more screen space, and are also aesthetic. In addition, because there is no counterpart of keyboard shortcuts on touchscreens, some existing commercial touch-based manipulation techniques simply provide a set of buttons to switch transformation modes. The drawback of this design is that manipulation efficiency is degraded because the manipulation task is frequently interrupted by switching the transformation modes. Moreover, relying on buttons is more or less conflicting with the tool-free philosophy of the tactile paradigm (Au et al., 2012). For these reasons, it is necessary to redesign manipulation techniques for touch-based interaction.

There are some good features of touchscreens that can be used to design efficient and natural manipulation techniques. First, touchscreens permit making multiple touch inputs at the same time. Because one touch input can be used to control two parameters, pressing and dragging multiple fingers simultaneously on the surface permits controlling more DOFs. Besides the position of each touch input, relative position and orientation among multiple fingers can also be used as additional DOFs. For example, dragging two conjoined fingers can be considered as a different gesture than dragging two separated fingers. Moreover, some touchscreens are able to detect some additional information such as the finger orientation and the contact region size. It is possible to take advantage of these several patterns of touch inputs to specify the manipulation mode and range in a more implicit and seamless way.

2.3 Manipulation mechanisms

The third factor used for the classification of manipulation techniques is the manipulation mechanism. In the literature, numerous mechanisms have been explored to propose new manipulation techniques for the touch-based paradigm. We have classified these techniques according to four mechanisms: manipulator-based manipulation, freestyle gesture manipulation, constraint gesture manipulation and hybrid manipulation.

2.3.1 Manipulator-based manipulation

As discussed above, manipulators have several good features such as decomposing a desired 3D movement into simpler components; it is worth exploring how to reshape classic manipulators to adapt to the touch-based paradigm. To provide a better manipulation experience, the new manipulator should have several characteristics. First, the manipulator should be more tolerant to touch inputs which are less precise than mouse inputs. Second, the use of the manipulator should be less influenced by the target orientation and size. Third, a more seamless way to switch manipulation modes should be provided to avoid depending on a group of buttons. Several new manipulators have been designed to support buttonless, imprecise touch-based input without sacrificing usability.

Schmidt et al. have proposed a set of sketching and composing widgets for 3D

manipulation (Schmidt et al., 2008). This technique allows users to draw a line stroke to summon a manipulation widget upon the selected object. The system checks the collinearity between the stroke and several candidate axes to set the widget orientation. Objects can be translated and scaled along, and rotated around the chosen axis (Figure 59). Besides the line stroke, a set of other strokes have been designed to switch the transformation mode of the widget. Instead of moving away the hand to click operation buttons, users can trigger a desired operation merely by drawing a stroke upon or close to the object. Hence, the manipulation task becomes more fluent.

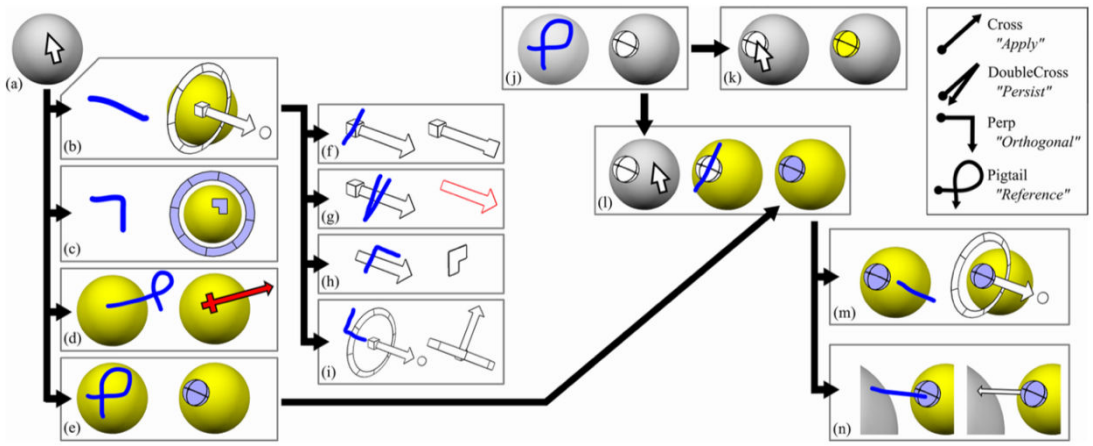


Figure 59: Manipulation techniques of the work of (Schmidt et al., 2008). A stroke can be drawn to summon a specific transformation widget.

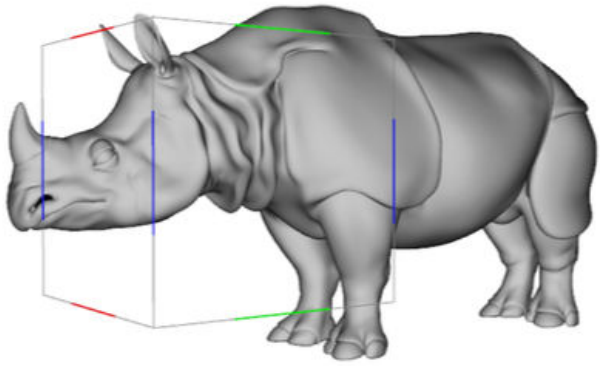


Figure 60: tBox for control of 9 DOF (Cohé et al., 2011).

Cohé et al. have proposed a transformation widget which is called tBox to make full manipulation of 9 DOF (Cohé et al., 2011). They used a box-shaped widget which covers the selected object for manipulation (Figure 60). The widget serves as a proxy of the object and all its edges and faces becomes interactive. To translate the selected object along one principal axis, users should first press a finger on one edge of the box and drag it along one direction of the edge. For rotation, if a finger is dragged across an edge or in a direction

which is perpendicular to an edge, a rotation axis with the same direction of the edge is determined and the object center is crossed by the axis. The rotation direction and range is calculated according to the finger motion. To scale the object, two fingers should be dragged from two parallel edges of one face in the opposite direction. As tBox has a larger interaction region than standard transformation widgets, it is easier to trigger manipulation without being affected by the occlusion issue. Furthermore, the box provides redundant edges for interaction so that users can select the most appropriate one given a specific perspective view. A user study has demonstrated that, for both expert and novice users, tBox is easy to learn and to use.

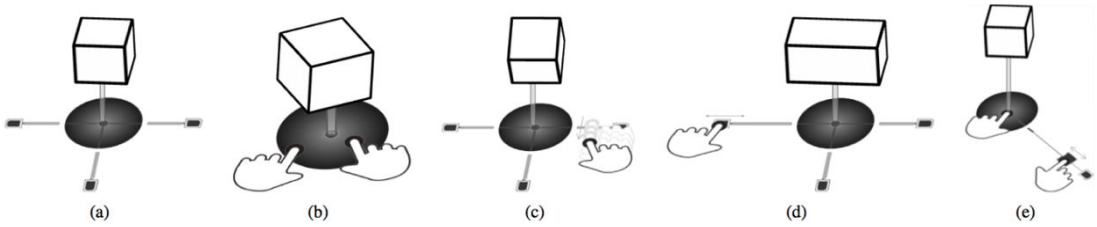


Figure 61: Manipulation gestures of Toucheo (Hachet et al., 2011).

Another manipulator-based technique is proposed by Hachet et al (Hachet et al., 2011). They have designed a setup which can provide stereoscopic display upon a touchscreen. Similar to tBox, the Toucheo technique permits controlling object motions using a proxy widget. After an object is selected, it is lifted from the surface and a circular widget is displayed below it. As shown in Figure 61, a set of gestures can be performed on the widget to perform RST operations. By using a widget below the floating object, the occlusion problem is well solved and manipulation is no longer influenced by the orientation and the size of the object. Moreover, the widget is reasonably designed so that all manipulation tasks can be triggered easily without switching the manipulation mode explicitly.

2.3.2 Constraint gesture manipulation

Besides reshaping the manipulator for the touch-based paradigm, some studies have explored how multi-touch gestures can be used to manipulate virtual 3D objects. According to whether manipulation can be performed with constraints, gesture-based manipulation techniques can be classified into two categories: constraint gesture manipulation and freestyle gesture manipulation. Constraint gesture manipulation techniques allow users to specify the transformation mode (translation, rotation or scaling) and the constraint (axis-constraint or plane-constraint) to manipulate objects accurately. For example, The Z-technique (Martinet et al., 2010a) and Eden (Kin et al., 2011) mentioned previously can be considered as constraint gesture manipulation techniques. Constraint gesture manipulation techniques are always developed for applications which require precise control such as scenes construction and models sketching.

Martinet et al. have proposed the DS3 technique by combining the Z-technique (Martinet

et al., 2010a) and the Screen Space technique (Reisman et al., 2009) to perform both 3D translation and 3D rotation (Martinet et al., 2010b). To avoid manipulation ambiguity, translation and rotation are discriminated by the count of touch inputs made directly on the selected object. If only one finger touches the object, the translation mode is active. Dragging the direct touch can translate the object in the plane parallel to the screen. If a secondary touch input is made in empty space while the first finger is on the screen, the object can be translated in the perpendicular direction. The activation of the rotation mode requires pressing at least two fingers directly on the object. Through a user study of docking tasks, it was found that separating translation and rotation tasks provided better performance and was also preferred by participants.

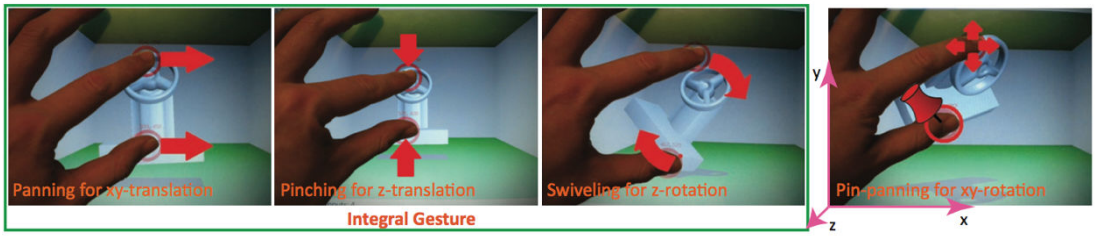


Figure 62: Translation and rotation gestures proposed in the work of (Liu et al., 2012).

Liu et al. have proposed a set of gestures similar to that of DS3 but only one hand is required to perform separated translation and rotation tasks (Liu et al., 2012). Users can use two finger touches in different ways to make manipulation of 6 DOF (3 translations, 3 rotations). As shown in Figure 62, the panning and pinching gestures can be used to perform x-y and z-translation, respectively. Swiveling two fingers can rotate the object around the z axis. For x-y rotation, users should keep pressing the first finger on the object and pan the second finger to set the rotation direction.

Au et al. have proposed another two-finger based manipulation technique (Au et al., 2012). This technique permits determining manipulation mode, selection transformation direction and setting movement range by performing only one gesture. To manipulate the selected object, users should first press two fingers to choose a candidate axis. The axis is chosen by comparing the collinearity between the vector formed by the pressed fingers and the projection of the principal axis on the screen. After setting the candidate axis, the two fingers can be panned along the axis to translate the object in the same direction or can be panned in the direction perpendicular to the axis projection to rotate the object around the axis (Figure 63). Performing a pinch gesture along the axis can scale up or down the selected object. Besides axis-constraint manipulation, plane-constraint translation and scale can also be performed after selecting a plane set by two principal axes. One benefit of this technique is that gestures can be performed in any empty space on the screen. As a result, manipulation is less restricted by the object position, orientation and scale. It also helps reduce the amount of information to retain for users. For example, it is only necessary to learn one gesture to translate the object in all the 3 orthogonal directions.

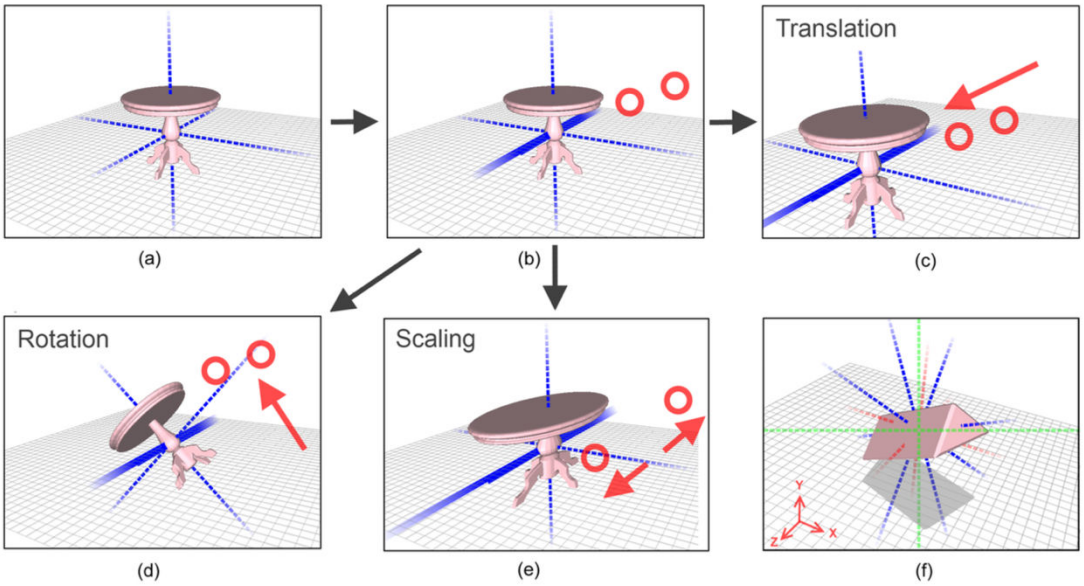


Figure 63: Multitouch gestures for axis-based transformation manipulations (Au et al., 2012).

2.3.3 Freestyle gesture manipulation

In contrast, the freestyle gesture manipulation techniques allow manipulating virtual objects in a freer way. In this case, manipulation is not divided into fine pieces and multiple DOF can be controlled at the same time. This kind of techniques is more suitable for entertainment applications because objects react more freely to inputs.

Hancock et al. have proposed techniques which enable users to manipulate 3D objects with one, two, or three fingers in shallow depth (Hancock et al., 2007). With fewer fingers, the first technique allows users to manipulate objects with less constraint. For example, it permits translating and rotating an object simultaneously with one finger. Moreover, with the help of the technique with two and three fingers, users are capable to perform different separated tasks independently and objects can be controlled in a more predictable way.

Moreover, Hancock et al. have proposed the Sticky Tools technique which benefits from force-based interaction mechanism (Hancock et al., 2009). This technique permits manipulating objects in a physically familiar way. After tapping one or two fingers on an object, the contact points on the object are acquired and they remain beneath the fingers when the object is moved, lifted or spun. Users feel like that objects are driven by the forces applied from fingers. To flip an object around an axis parallel to the screen, two sticky fingers can be tapped on the object to set the rotation axis while a third finger is dragged to set the rotation range.

The Screen Space technique proposed by Reisman et al. does not map gestures strictly to specific manipulation tasks (Reisman et al., 2009). Instead, the object motions are calculated according to the movement of constraint points. After changing positions of

touch inputs on the surface, the object is manipulated to ensure that contact regions on the surfaces of the object remain beneath the constraint touch points (Figure 64). The manipulation mode, direction, and range are determined by the count of touch inputs, contact regions on the object, and movements of constraint points, respectively. The benefit of this technique is that it provides more liberty to specify manipulation parameters and objects react more naturally to touch input movements. On the other hand, object manipulation is less predictable and turns out to be less efficient than constraint manipulation techniques (Martinet et al., 2010b; Liu et al., 2012).

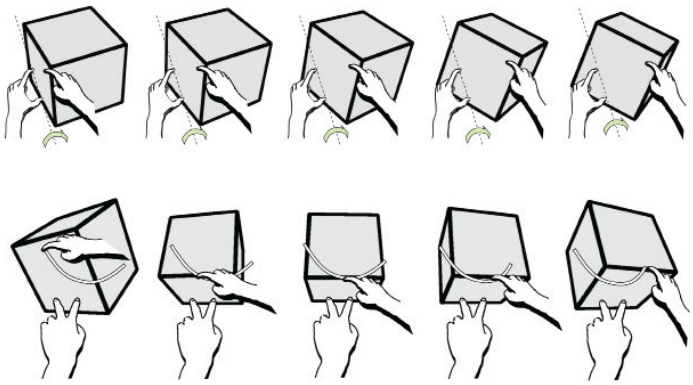


Figure 64: Object rotation of the Screen Space technique (Reisman et al., 2009).

Cao et al. have developed the ShapeTouch technique which can simulate force-based interaction by leveraging contact shapes on the surface (Cao et al., 2008). Instead of providing a set of pre-defined gestures, this technique allows users to manipulate objects in a more realistic and familiar way. Three types of forces can be simulated considering both the contact region on the object and the touch movements: pressing, colliding and friction. In addition, the strength of the force is determined by the area of the contact region. As shown in Figure 65, a 2D object can be manipulated like a piece of paper on the desk by applying different kinds of forces. Figure 65 (a) shows that object translation and rotation can be made at the same time and Figure 65 (b) shows that users can set a pivot point using one hand and control the rotation angle using the other hand. Similarly, Wilson et al. have developed another force-based manipulation technique for the touchscreen (Wilson et al., 2008).

Wilson have proposed another technique to simulate the grasping behavior on the touchscreen (Wilson, 2009). Different from the techniques proposed in (Cao et al., 2008) and (Wilson et al., 2008), in addition to simulating horizontal forces, this technique can also infer the virtual vertical friction applied on the sides of the object. Simulation of vertical friction allows the users to grasp an object by touching two opposite sides of it.

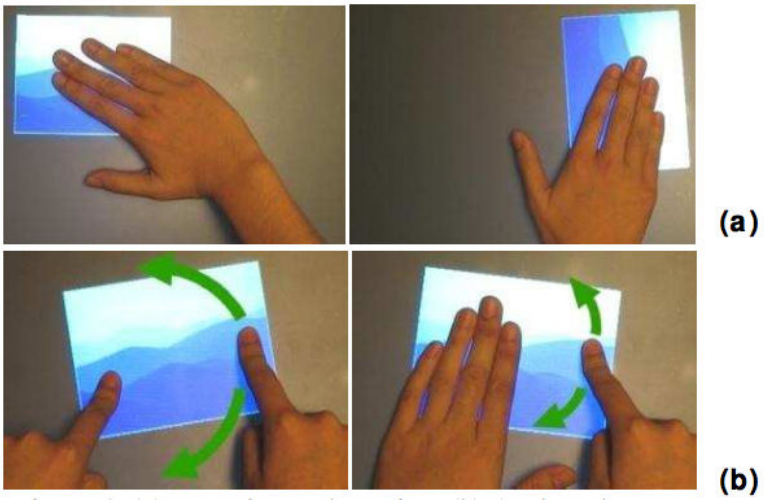
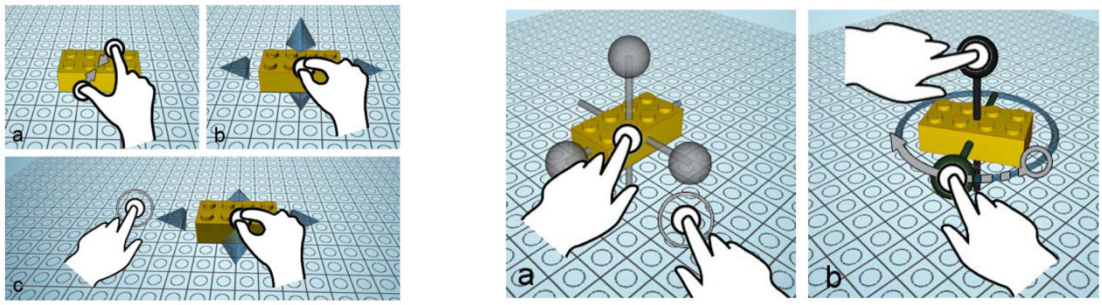


Figure 65: Examples of the ShapeTouch technique (a) Dragging and rotating. (b) Anchored movement (Cao et al., 2008).

2.3.4 Hybrid modalities

Some techniques are based on the combination of more than one modality. One example is the application developed by Mendes et al. for sketching LEGO models (Mendes et al., 2011). In this application, LEGO bricks can be translated using gestures and rotated using a manipulator (Figure 66). When a pinch gesture is made on a brick, it can be translated along the vertical plane. Dropping a secondary finger in the empty space permits to switch the vertical plane to the horizontal one. When a finger taps on a brick, pressing a secondary finger on the screen can display the rotation manipulator. Then, users can tap the sphere of one handle to set the rotation axis and drag another sphere to rotate the brick around the axis.

Because motion capture devices, such as accelerometers and gyroscopes, are currently embedded in almost all mobile devices, some studies have tried to use device motions for object manipulation. In a user study, Liang et al. have asked a group of participants to define interaction techniques for a set of manipulation tasks (Liang et al., 2012). Users can use front-side and back-side touch inputs, as well as device motions. A set of user-defined hybrid techniques has been proposed according to this study. For example, the pan and pinch gestures can be used to translate and scale the object in the xy-plane, respectively. However, for z-translation and z-scaling, users should first rotate the device to make it parallel to the z-axis. And then the pan and pinch gesture can be made to translate and scale the object about the z-axis. Many participants have proposed tilting the device around the horizontal and vertical axis to perform the x- and y-rotation respectively. To rotate the object around the z-axis, a touch input should be moved from the -right corner of the object to the top-left corner.



(a) (b)

Figure 66: Manipulation of LEGO bricks. (a) Translation. (b) Rotation (Mendes et al., 2011).

Mossel et al. have proposed the 3DTouch technique for object translation and rotation on mobile devices (Mossel et al., 2013). 3DTouch enables users to use the orientation of the device to set the manipulation plane (Figure 67). After selecting an object, dragging the finger on the screen permits to perform plane-constraint translation. If the manipulation mode is switched to rotation, dragging the finger in one direction can rotate the object around the axis perpendicular to the dragging direction.

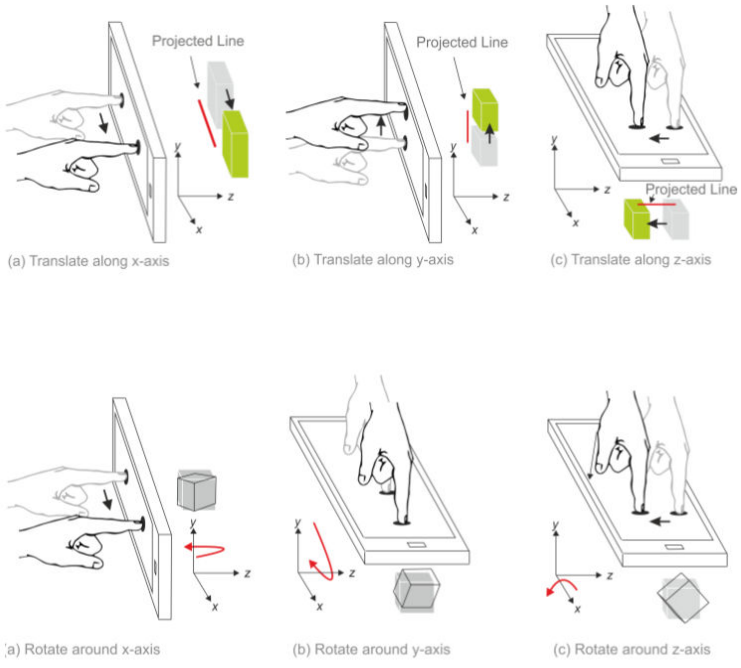


Figure 67: The 3DTouch technique. (a) Object translation; (b) Object rotation (Mossel et al., 2013).

Another example of combing touch inputs and motion gestures is exTouch (Kasahara et al., 2013). This technique provides redundant methods to translate and rotate physical objects using a mobile device. As shown in Figure 68, users can either perform touch-based

gestures or motion-based gestures for object manipulation. Interaction inside an augmented reality environment can benefit from this technique.

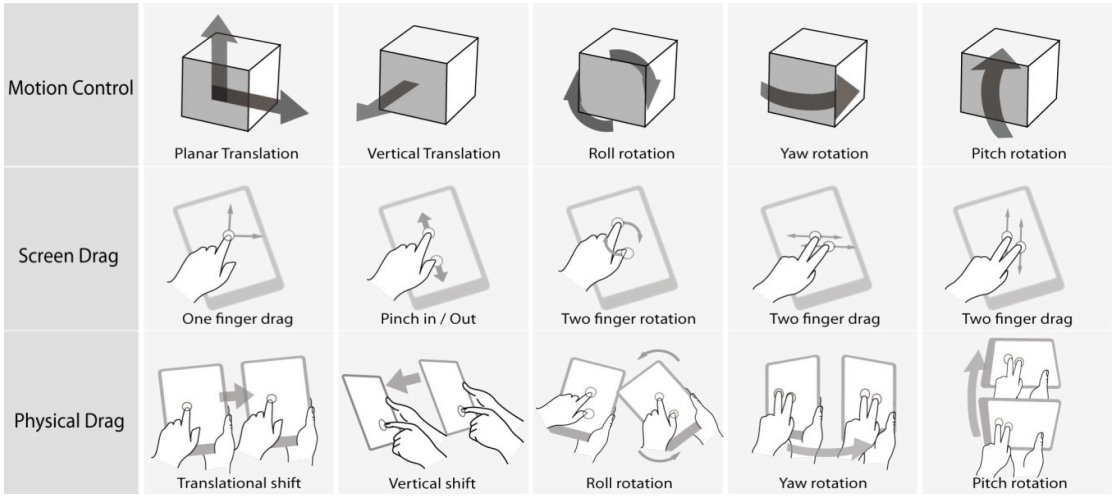


Figure 68: The exTouch technique enables using touch inputs and device motions to translate and rotate physical objects (Kasahara et al., 2013).

2.4 Manipulation reference

In some commercial sketching software, users are able to indicate the manipulation reference. In general, there are three references: Global reference, Local reference and Display reference. In the virtual environment, a point is chosen as the center point of the global coordinate system and the object position is measured with respect to this center point. When the global reference is selected, the orientation of the manipulator attached to the selected object is the same as the coordinate system, and remains the same all the time no matter how the object is rotated. In contrast, the local reference represents the local coordinate system attached to the object. If the local reference is selected, the manipulator follows the object translation and also rotates synchronically with the object. Similar to the local reference, the display reference uses the coordinate system attached to the display. The origin of the reference coincides with one corner of the display. The x-axis is parallel to the width of the display and the y-axis parallel to the height. The z-axis is set perpendicular to the display. When the camera viewpoint is changed, the projection of the three principal axes of both the global and local reference may change. However, the display reference is not influenced by the scene navigation. Although several desktop-based commercial software provide the possibility of manipulating the object in both the global and the local reference, not all the techniques in the literature offers the possibility of switching the manipulation reference.

2.4.1 Display reference

Some techniques allow users to make 2D (Wu & Balakrishnan, 2003; Hancock et al., 2006), or 3D (Hancock et al., 2009; Martinet et al., 2010a; Martinet et al., 2010b) manipulation

with respect to the display reference. Actually, in these applications, the camera is fixed and the display coincides with one principal plane of the global reference, so that it can also be considered that objects are manipulated with respect to the global reference.

2.4.2 Global & local reference

For applications in which the users are allowed to change the camera viewpoint, the screen reference is not commonly used. In this case, it is common to provide manipulation techniques with respect to the global or local reference. For example, in the evaluation of (Knoedel & Hachet, 2011), objects are manipulated with respect to the global reference. Objects can be translated in the plane parallel to the ground and rotated around the axis perpendicular to the ground. Eden displays the scene in a perspective view and objects can be manipulated with respect to the global reference. The xy-plane of the global coordinate system is set to the ground of the scene. In addition to global manipulation, Eden also provides two gestures to perform acrobat and Local Z rotation in the display and local coordinate system, respectively.

Other techniques, such as tBox (Cohé et al., 2011) and exTouch (Kasahara et al., 2013) enable object manipulation with respect to the local reference. The box-shaped manipulator of tBox is attached to the selected object and reproduces the same rotation on itself. The exTouch technique allows users to select one surface on an object as the reference plane. After selecting the object, users can translate the object in the reference plane or along its normal direction. The selected surface serves also as the reference for object rotation.

Some techniques switch the manipulation reference when the operation mode is changed. For example, 3DTouch, proposed in (Mossel et al., 2013), uses the global reference for object translation and the local reference for object rotation and scaling. When the translation mode is active, the dragging vector on the screen is projected into the 3D world and the collinearity is calculated between the projection vector and each principal axis of the world coordinate system to determine the translation direction. However, the rotation axis and the scaling direction are determined in the similar way, but with respect to the local coordinate system of the selected object. The HOME-S technique proposed in the same work allows mapping directly the change of position and orientation of the smartphone in the world coordinate system to translation and rotation of the selected object. However, for object scaling, the system maps the device motions to object scaling with respect to the local reference.

There are also few studies which enable users to switch between the global, local and display reference (Au et al., 2012). In this case, users can manipulate virtual objects in a more flexible way. The technique developed by Schmidt et al. also allows users to specify a principal axis in the global, local or display reference as the candidate axis (Schmidt et al., 2008). The technique of Screen-Space proposed in (Reisman et al., 2009) does not specify one reference explicitly. Because the object is manipulated according to the movement of

constraint points, it is hard to determine whether the manipulation is performed with respect to a given reference.

2.5 Number of hands involved

Manipulation techniques can also be simply classified according the number of hands involved in interaction. Guiard have proposed a kinematic chain model to describe how two hands can handle daily tasks in collaboration (Guiard, 1987). This model reveals that asymmetrical tasks are often performed by two hands respectively to realize a complex work. Although the touchscreen can accept multiple touch inputs, fingers of the same hand cannot be used freely due to anatomic restrictions (Brouet et al., 2013). Using fingers of two hands can reduce the anatomic restriction. For example, the maximum divergence distance of the pinch gesture performed by one hand is significantly smaller than that of two hands. Moreover, the secondary hand can also provide additional input DOF, which augments the design space.

2.5.1 Bimanual manipulation

There are many techniques designed based on bimanual interaction. For example, Z-technique requires the use of a secondary touch on the screen to specify the z-translation range (Martinet et al., 2010a). Sticky Tools (Hancock et al., 2009) and Screen-Space (Reisman et al., 2009) allow using one hand to set the rotation axis and the other hand to set the rotation range. Techniques developed based on physical simulations enable users to use both hands in symmetrical or asymmetrical ways to manipulate objects in a more realistic way (Cao et al., 2008; Wilson, 2009; Wilson et al., 2008). Some gestures of tBox (Cohé et al., 2011) and exTouch (Kasahara et al., 2013) are also designed based on the usage of two hands.

2.5.2 Single-handed manipulation

Although using two hands can enrich the interaction glossary, it is difficult to use two hands in collaboration for mobile devices because there is always one hand holding the device. For this reason, many techniques have been proposed to use only the dominant hand. For example, the technique proposed in (Liu et al., 2012) enables users to perform object translation and rotation along all three principal axes by using two fingers of the same hand. The technique in (Au et al., 2012) also allows users to use only one hand to make a full manipulation of 9 DOF. 3DTouch and HOMER-S are designed to avoid the involvement of the non-dominant hand as well (Mossel et al., 2013).

2.6 Discussion

After presenting different 2D input-based manipulation techniques, we group them in Table 5 for comparison. Note that in this table there are only pen-based and touch-based

techniques, because in the literature we have not identified any new mouse-based manipulation technique. Although some techniques are proposed for usage on a fixed touchscreen, they can be implemented and used easily on mobile devices. Therefore in the last dimension of Table 5, some techniques proposed for fixed usage are considered also appropriate for mobile usage.

Although many techniques have been proposed to adapt to the tactile paradigm, only a few of them can be used for design work. First, designers and architects have to control all 9 DOFs of objects: 3 translations, 3 rotations and 3 scaling. The dimension of DOFs of Table 5 shows that only seven techniques can be used to take full control of 3D objects. Second, for design work which requires high precision, users should be able to make translation, rotation and scaling independently. All these seven techniques meet this requirement. However, because some of them require the coordination of both hands for some manipulation tasks, only four techniques are appropriate for mobile usage (Schmidt et al., 2008; Au et al., 2012; Liang et al., 2012 and Mossel et al., 2013). The user-proposed technique in the work of Liang et al. (Liang et al., 2012) and the 3DTouch in the work of Mossel et al. (Mossel et al., 2013) are more appropriate to explore virtual environments, but less for 3D design work because camera navigation relies on frequent changes of the position and orientation of the device. Although the remaining two techniques allow users to make full control of 3D objects on mobile devices, they have their own limitations. For the technique proposed by Schmidt et al., we think that it is difficult for users to memorize and recall all its functions (Schmidt et al., 2008). In addition, this technique is not fluent enough as well because users have to first draw a stroke to trigger the manipulator, and then control the manipulator for transformation. On the one hand, the technique of (Au et al., 2012) is more fluent and it is not limited by the object position. On the other hand, this technique is limited by the camera perspective and the object orientation. When two principal axes of the coordinate system overlap each other, it is difficult to specify the desired one.

Through this discussion, we find that neither of existing 2D input based manipulation techniques in the literature is appropriate for mobile design work regarding both functionality and usability. This finding motivates us to design a new technique to allow designers to make design work in outdoor environments. In Chapter 6, we will present our touch-based techniques which are designed to improve the efficiency and fluency of manipulation experience on TabletPCs.

Table 5: 2D input-based manipulation techniques.

Technique	DOFs	Input	Manipulation mechanism	Manipulation reference	Number of hands	Trigger on object	Work under different camera viewpoint	Appropriate for fixed or mobile usage
Multi-Finger and Whole Hand Gestural Interaction (Wu & Balakrishnan, 2003)	Tx, Ty, Rz	Touch-based	Constraint gesture	Display/Global	1 or 2	Yes	No	Fixed
Rotation and translation mechanism (Hancock et al., 2006)	Tx, Ty, Rz	Touch-based	Freestyle/Constraint gesture	Display/Global	1 or 2	Yes	No	Fixed
Shallow-depth (Hancock et al. 2007)	Tx, Ty, Rx, Ry, Rz	Touch-based	Freestyle/Constraint gesture	Display/ Global	1 or 2	Yes	No	Fixed
ShapeTouch (Cao et al., 2008)	Tx, Ty, Rz	Touch-based	Freestyle	Display/ Global	1 or 2	Yes	No	Fixed
Sketching and composing widgets (Schmidt et al., 2008)	Tx, Ty, Tz, Rx, Ry, Rz, Sx, Sy, Sz	Pen-based or touch-based	Hybrid	Global/ Local	1	Yes	Yes	Fixed and mobile
Physics simulations (Wilson et al., 2008)	Tx, Ty, Rz	Touch-based	Freestyle	Display/Global	1 or 2	Yes	No	Fixed
Sticky tools (Hancock et al., 2009)	Tx, Ty, Tz, Rx, Ry, Rz	Touch-based	Freestyle	Display/Global	1 or 2	Yes	Yes	Fixed
Screen Space formulation (Reisman et al., 2009)	Tx, Ty, Tz, Rx, Ry, Rz	Touch-based	Freestyle	Not specified	1 or 2	Yes	Yes	Fixed
Grasping (Wilson, 2009)	Tx, Ty, Rz	Touch-based	Freestyle	Display/Global	1 or 2	Yes	No	Fixed
Z-technique (Martinet et al., 2010a)	Tx, Ty, Tz	Touch-based	Constraint gesture	Display/Global	1 or 2	Yes	No	Fixed
DS3 (Martinet et al., 2010b)	Tx, Ty, Tz, Rx, Ry, Rz	Touch-based	Constraint gesture	Display/Global	1 or 2	Yes	No	Fixed
tBox (Cohé et al., 2011)	Tx, Ty, Tz, Rx, Ry, Rz, Sx, Sy, Sz	Touch-based	Manipulator	Local	1 or 2	Yes	Yes	Fixed
Toucheo (Hachet et al., 2011)	Tx, Ty, Tz, Rx, Ry, Rz, Sx, Sy, Sz	Touch-based	Manipulator	Display/Global	1 or 2	Trigger on the projection of the object	No, but stereoscopic display	Fixed
Eden (Kin et al., 2011)	Tx, Ty, Tz, Rx, Ry, Rz, Sx, Sy, Sz	Touch-based	Constraint gesture	Global	1 or 2	Yes	Yes	Fixed
Multi-touch RST (Knoedel & Hachet , 2011)	Tx, Ty Rz Sxyz	Touch-based	Freestyle	Global	1 or 2	Yes	Yes	Fixed

LTouchIt (Mendes et al., 2011)	Tx, Ty, Tz Rx, Ry, Rz	Touch-based	Hybrid	Global	1 or 2	Yes	Yes	Fixed
Multitouch constrained manipulation (Au et al., 2012)	Tx, Ty, Tz Rx, Ry, Rz Sx, Sy, Sz	Touch-based	Constraint gesture	Global	1	No	Yes	Fixed and mobile
Surface+motion (Liang et al., 2012)	Tx, Ty, Tz Rx, Ry, Rz Sx, Sy, Sz	Touch-based + device motions	Constraint gesture	Local	1	Yes	Yes	Mobile
Two-Finger Gestures (Liu et al., 2012)	Tx, Ty, Tz Rx, Ry, Rz	Touch-based	Constraint gesture	Display/Global	1	Yes	No	Fixed and mobile
exTouch (Kasahara et al., 2013)	Tx, Ty, Tz Rx, Ry, Rz	Touch-based + device motions	Constraint gesture	Local	1	Yes	Yes	Mobile
3DTouch (Mossel et al., 2013)	Tx, Ty, Tz Rx, Ry, Rz Sx, Sy, Sz	Touch-based + device motions	Constraint gesture	Global: translation Local: rotation and scaling	1	Yes	Yes	Mobile

3 3D input-based

In the interaction community, instead of designing manipulation techniques based on the use of 2D input devices, numerous studies have been conducted to explore using 3D inputs to manipulate virtual 3D objects. One benefit of 3D inputs is that users are able to control more than 2 DOFs at a time. In this case, it is no longer necessary to decompose a manipulation task into several subtasks of lower dimensions. Moreover, direct mapping 3D inputs to 3D object movements can also be more intuitive because people are familiar with controlling 3 dimensions simultaneously. It enables to reuse learnt skills in the virtual 3D environment. A similar classification strategy is used to describe the different state-of-the-art 3D input-based manipulation techniques.

3.1 Input tools

Because the hands are the most dexterous parts of our body, in the literature, most of 3D inputs are made using the hands. In general, there are two main methods to make 3D inputs: 3D input devices and freehand gestures.

3.1.1 3D input devices

In contrast with the regular 2D mice, 3D input devices allow users to control more than 2 DOFs simultaneously. For manipulation, one approach is to use a 3D input device as the proxy of the selected virtual object and the device movements are reproduced on the object. It is also possible to use a 3D input device as an interaction medium to map its movements in some other ways to the movements of virtual objects.

Because people are familiar with the regular mouse, a natural step is to augment it with some additional DOFs to enrich interaction possibilities. Balakrishnan et al. have designed the Rockin'Mouse which can provide 4 DOFs (Balakrishnan et al., 1997). Having a similar shape as the regular mouse, the Rockin'sMouse can also be panned on the desk to make 2D inputs. In addition, with a round bottom, this device can be tilted around two axis parallel to the desk plane. Hinckley et al. have designed the VideoMouse with 6 DOFs (Hinckley et al., 1999). In addition to the planar translation and rotation around two planar axes, rotation of the VideoMouse around the vertical axis can be detected and it is also possible to sense limited changes of its vertical position. The device is instrumented with a camera which is used to detect the change of a grid pattern to infer the device position and orientation. The SpaceMouse is a 6 DOF commercial device. It is designed as a complementary device for the regular mouse to allow users to make 3D inputs using the non-dominant hand. Perelman et al. have developed the Roly-Poly Mouse which also supports 6 DOF inputs (Perelman et al., 2015). It is a sphere-shaped device that can be tilted around the planar axis and rotated around the vertical axis (Figure 69). Because this device allows users to make large range rotations, if buttons of standard mice are used, users have to perform clutching movements frequently to get access to the buttons. To avoid the rotation adjustment, a ring button encircling all around the device is designed. Table 6 compares different input mice

regarding their DOFs. In the table, 'p' and 'r' are used as abbreviations of position control and rate control. Position control means that position and orientation changes of the device are mapped to output movements. On the other hand, rate control maps device movements to velocities of output movements.

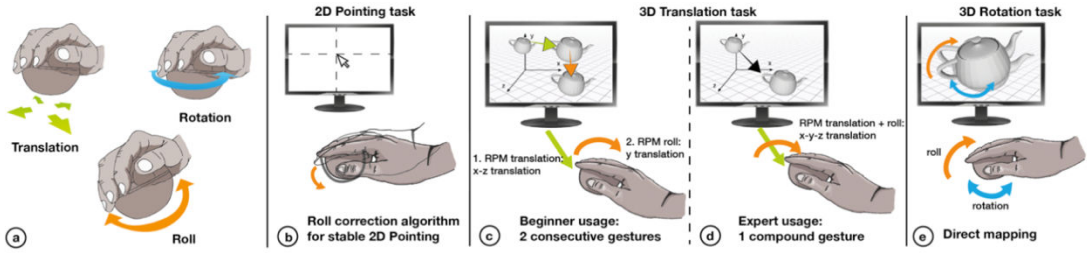


Figure 69: The Roly-Poly Mouse (Perelman et al., 2015).

Besides mouse shaped devices, some other devices have been used to make 3D inputs. Hachet et al. have presented the prototype of the CAT (Control Action Table) to simplify 3D interaction (Hachet et al., 2003). The CAT is a 6 DOF freestanding input device and is primarily designed for interaction in large immersive display environments. Chao et al. have explored using two handheld devices in both 2D and 3D modes (Cho et al., 2013). The position and orientation of each sphere-shaped device can be utilized. Currently, Smartphones and TabletPCs are instrumented with accelerometers, gyroscopes, GPS and magnetometer, etc. With the help of these embedded sensors, it is also possible to take advantage of motions of mobile devices to make 3D interaction. For example, Du et al. have tried tilting the smartphone to rotate virtual objects and switch the camera viewpoint (Du et al., 2011). Yoon et al. have mapped directly the pose of a smartphone to that of the virtual object (Yoon et al., 2011). A 3D model on a large display is aligned dynamically with respect to the orientation of a mobile device. In some studies, the motion of a mobile device can be used in combination with touch inputs to manipulate 3D objects in a more flexible way (Kasahara et al., 2013; Mossel et al., 2013).

Table 6: Comparison of sensed DOF for several multi-DOF mice (Perelman et al., 2015).

	Regular Mouse		Rockin'Mouse		Video Mouse		Space Mouse		Roly-poly Mouse	
DOF	Sensed	Rate/Pos	Sensed	Rate/Pos	Sensed	Rate/Pos	Sensed	Rate/Pos	Sensed	Rate/Pos
Tx	√	p	√	p	√	p	√	r	√	p
Ty	√	p	√	p	√	p	√	r	√	p
Tz	x		x		√	p	√	r	√	p
Rx	x		√	r,p	√	r,p	√	r	√	r,p
Ry	x		√	r,p	√	r,p	√	r	√	r,p
Rz	x		x		√	r	√	r	√	r,p
T+R	x		x		x		√	r	√	r,p

3.1.2 Freehand gestures

There are several benefits of using freehand gestures for manipulation. First, it is unnecessary to hold a device in hand so that the expressiveness of the hands is freed. Second, there are some gestures that people use frequently to control physical objects. Thanks to their semantic meanings, users may find it intuitive to reproduce these gestures for 3D interaction. Using familiar gestures can also reduce the learning cost of new techniques. Third, freehand gestures are also suitable for mobile interaction since users are not forced to carry inputs devices with them.

To capture freehand gestures, there are several solutions. Some studies have proposed instrumenting the hands with data gloves for gesture detection. Data gloves can provide information such as hand position, orientation and angles of joints. This abundant information can be used to record hand movement trajectory and identify pre-defined gestures (Kumar et al., 2012; Camastra & De Felice, 2013). Using data gloves can provide reliable hand tracking which is less influenced by the occlusion problem. Although most of data gloves have embedded sensors or are augmented with tracking markers, there are other low-cost solutions. For example, Wang et al. have proposed using a monocular camera to track hands wearing color gloves (Wang & Popović, 2009). A custom color pattern is used to provide stable tracking results.

To avoid using data gloves, Grossman et al. have instrumented hands directly using several markers. These markers are tracked by four cameras allowing five hand postures and gestures can be identified (Grossman et al., 2004). To acquire reliable finger tracking, De Araújo et al. have attached Gametrak devices on the thumb and index finger (De Araújo et al., 2013).

Although instrumenting the hands with a pair of gloves can provide a reliable and an easy way for hand tracking, it is more or less intrusive. Besides using gloves, some studies have explored using computer vision technologies for hand tracking. Wilson has proposed an algorithm to detect pinch gestures using an RGB camera (Wilson, 2006). This algorithm looks for the close path formed by the thumb and the index finger and then fit the connected component by an ellipse. The center, max axis orientation and size of the ellipse are used to represent the hand planar position, orientation and height, respectively. Although this work uses an RGB camera for hand tracking, this algorithm can be used with a depth camera (Bally et al., 2012; Hilliges et al., 2009). Benko and Wilson have proposed a simple way to detect hand movements using a depth camera (Benko & Wilson, 2009). The hand is extracted by removing the background image and the torso image from the initial image. Due to the limited camera resolution, it is difficult to infer the hand poses precisely so that only the open and fist gesture can be identified.

After Microsoft has launched the Kinect, it was quickly and widely used for hand tracking because of the competitive precision of its depth camera. Klompmaker et al. have proposed a framework called dSensingNI, to support multitouch and tangible interaction with arbitrary objects (Klompmaker et al., 2012). A Kinect is installed above a tabletop to detect

touch events on the tabletop and physical objects. Furthermore, the hand position and orientation can be detected when the hand is held in the air. The hand pose can also be identified by analyzing the shape of the hand contour. Similar to dSensingNI, another toolkit called KinectArms can also track arm and hand for remote communication (Genest et al., 2013).

Hilliges et al. have explored enabling users to manipulate virtual objects in the same way as physical ones (Hilliges et al., 2009). Because using the pinch gesture is not an ideal model of the grasping behavior, they have developed a prototype which is able to reconstruct 3D hand meshes and extract finger positions. Users can grasp and manipulate virtual objects like physical ones. By analyzing the motions of fingers used to hold the object, in addition to translation in 3 dimensions, roll and raw rotation can also be accomplished by natural hand movements. In another work, a Kinect has been used to reconstruct polygon meshes for hands and physical objects (Hilliges et al., 2012). A physic engine is used to transform meshes into a set of particles. These particles are used to simulate physical effects between virtual objects and reconstructed meshes. Kim et al. have proposed a wrist-worn sensor to reconstruct accurately the hand model (Kim et al., 2012). The sensor uses a laser light projector and four IR LEDs to emit light and uses an IR camera to receive light reflected by the hand. The hand model is reconstructed based on a kinematics model proposed in this work.

3.2 Manipulation mechanisms

Here we propose a classification of 3D input based techniques according to the manipulation mechanism used for interaction.

3.2.1 Natural movements

Some studies have reused learnt skills that people are familiar with for object manipulation. The objective is to reduce the cognitive cost for learning these new techniques and also to make the manipulation experience closer to our daily experience.

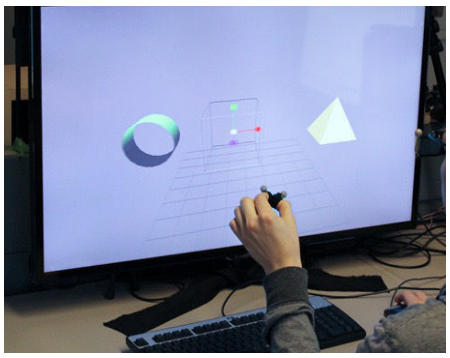


Figure 70: Move one hand in air to translate the virtual 3D object in a stereoscopic environment (Bogdan et al., 2014).

Many techniques allow users to map 3D movements of hands or devices directly to movements of virtual objects. In many applications, after performing a gesture to pick up a virtual object, users can translate it directly by moving one or both hands freely in air (Hilliges et al., 2009; De Araújo et al., 2013; Marquardt et al., 2011a; Wang et al., 2011). Borden et al. have conducted an experiment to study the benefit of 3D translation in a stereoscopic display (Bogdan et al., 2014). They have compared three translation techniques: 2D, 3D and hybrid. The 2D technique is the standard mouse-based translation technique. The 3D technique allows users to move their hands in air to control a 3D cursor for selection and translation (Figure 70). The hybrid technique is a combination of the previous two techniques. The mouse is used for selection and the hand movement in air is used for translation. The results have shown that the hybrid technique is the most efficient one. Although using the 3D cursor turns out to be more time consuming, using hand movements help reducing the translation time. Wang et al. have proposed a throw and catch gesture to teleport an object (Wang et al., 2011). For example, when the left hand holds an object, users can perform the pinch gesture on the right hand and then release the pinch gesture on the left hand. After that the object can be passed from the left hand to the right hand.

In addition to hand movements, changes of device position can also be mapped to object movements. Because the VideoMouse can provide 6DOFs inputs, it enables user to control it in space to translate a selected object (Hinckley et al., 1999). For a regular mouse, the cursor stays stable when the mouse is lifted. To support 3D translation, if the left button is clicked when the VideoMouse is lifted, its motions can still be used to adjust the object position. This mechanism permits using the height of the mouse to move the object in the third dimension. The Roly-Poly mouse allows users to roll the device forward or backward to adjust the height of the selected object (Perelman et al., 2015). The CAT proposed in (Hachet et al., 2003) uses a tabletop embedded with force sensors. It permits applying forces in the desired direction on the tabletop to translate an object in rate control mode. Cho et al. have proposed using a pair of tracked balls to make 3D inputs (Cho et al., 2013).

With the help of embedded devices, it is also possible to leverage the position of mobile devices for translation. The HOMER-S technique proposed in (Mossel et al., 2013) allows users to move a smartphone in space to adjust the object position in an augmented reality environment. Similarly, exTouch permits dragging a mobile device in its screen plane or in the vertical direction to control the position of a robot or a drone in the real world.

Natural movements can also be used to rotate virtual objects. A natural step is to use the wrist rotation directly. In the work of (Hilliges et al., 2009), a prototype has been proposed to allow users to grab a virtual object and perform yaw and roll rotation by wrist movements. The pitch rotation is not supported due to technical limitations of the prototype. Because Marquardt et al. have chosen to use data gloves to track hand movements, wrist rotation can be detected precisely and can be used to rotated virtual object freely in air (Marquardt et al., 2011a). Similarly, since the orientation of the tracked ball used in the work of (Cho et al., 2013) can be acquired, users are able to make object rotation by wrist movements. In the user study of Piumsomboon et al., some participants

also have proposed using wrist movements to rotate virtual objects (Piumsomboon et al., 2013) (Figure 71). With the help of a Kinect, HoloDesk allows users to manipulate virtual objects more freely in an augmented reality application (Hilliges et al., 2012). Because hands and physical objects can be reconstructed and physical forces can be simulated, any gestures can be used for manipulation. As shown in Figure 72, besides grasping virtual objects, users can also scoop a ball by two palms, drop it from one hand to another, and even shake it in a real bowl. Unlike other techniques, the manipulation experience provided by HoloDesk is very lifelike and of great liberty.

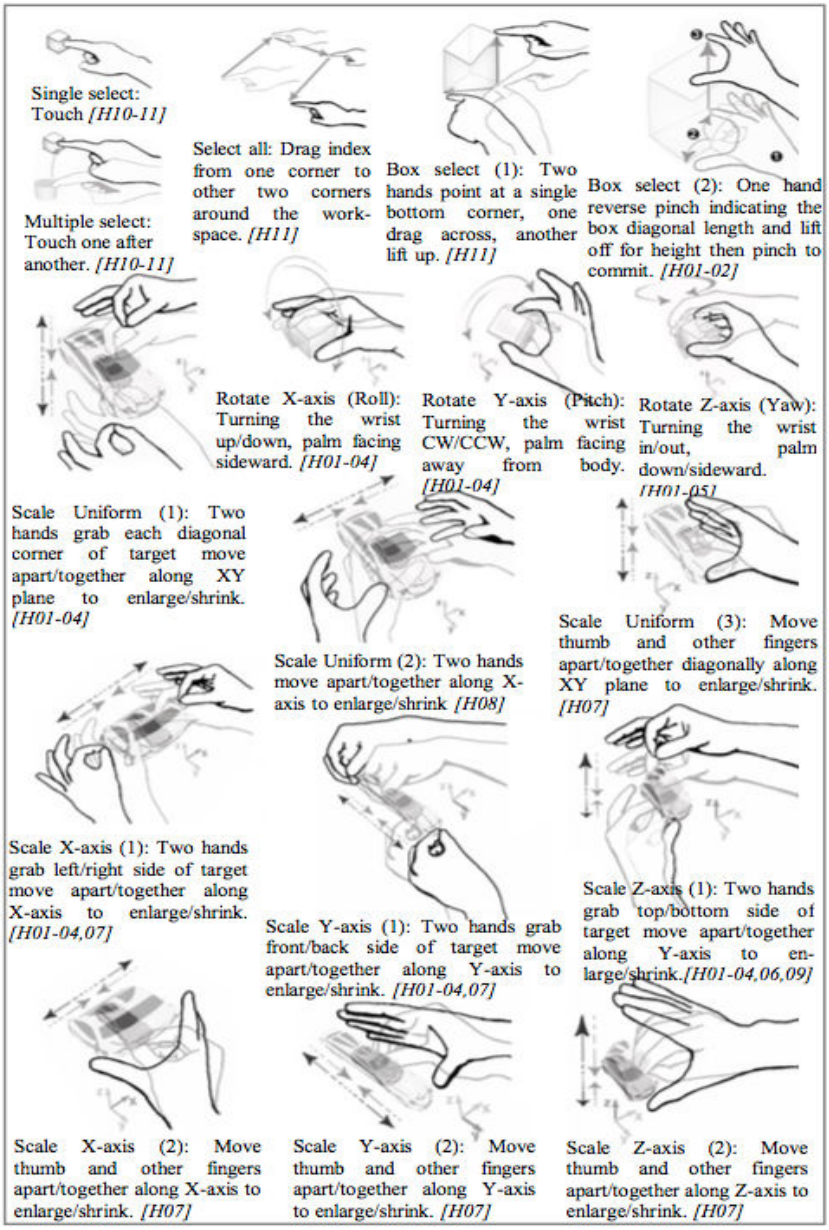


Figure 71: User-defined manipulation gestures (Piumsomboon et al., 2013).

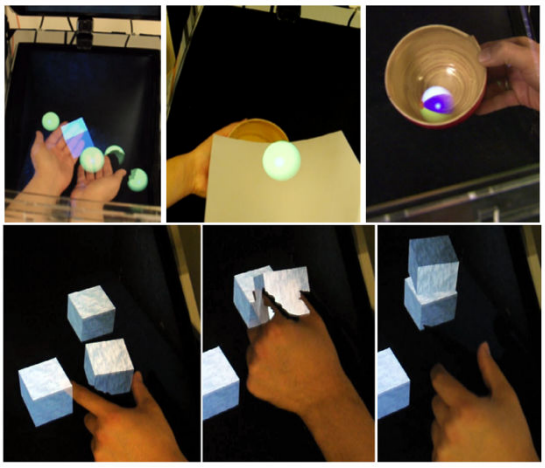


Figure 72: Interaction with virtual objects using natural gestures in the HoloDesk (Hilliges et al., 2012).

Another natural method to rotate virtual objects is designed based on the use of two hands. Wang et al. have proposed mimicking the physical actions of rotating an imaginary piece of paper with two hands (Wang et al., 2011). Rotating the sheet around the y or z axes involves moving the hands in opposite directions along the xz or xy planes, respectively (Figure 73). To rotate the paper around the x axis, one lifts or lowers the hands while bending the wrists. This technique is implemented in the Mockup Builder (De Araújo et al., 2013). Inspired by the work of (Hancock et al., 2006), Mendes et al. have proposed fixing one hand in air and rotating the other around it to rotate objects (Mendes et al., 2014). Bally et al. have proposed the triangle gesture which can be used to rotate virtual objects (Bally et al., 2012). Triangle gestures are a set of two-armed poses formed by creating a triangle with the arms and torso. Users can also adjust the triangle shape continually to make inputs. In one application, this gesture was used to rotate an object on the screen of the smartphone around the axis perpendicular to the screen.

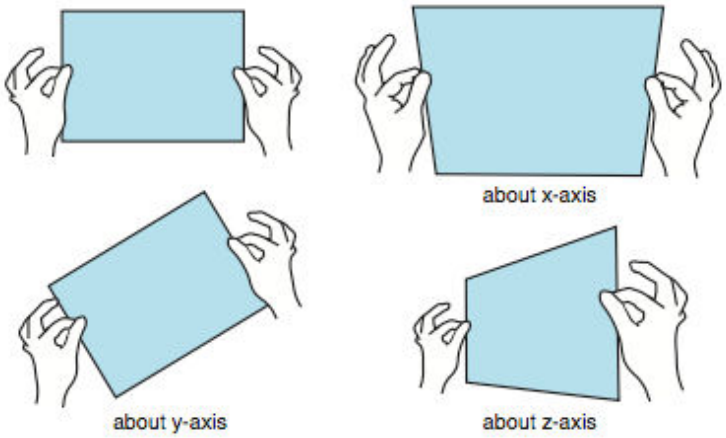


Figure 73: Bimanual rotation gestures proposed by Wang et al (Wang et al., 2011).

Besides freehand gestures, some techniques permit using the movement of input devices to

rotate objects in a natural way. For example, the VideoMouse allows users to tilt it around the axis parallel to the desk to perform roll and pitch rotation. Yaw rotation can be controlled by rotating the device about the axis perpendicular to the desk. With a sphere shape and more rounded bottom, the Roly-Poly Mouse can be rotated more intuitively with greater amplitude to adjust the orientation of virtual objects. The CAT prototype mentioned above permits orientating the tabletop by rotating itself or its frame to reproduce rotation of the same magnitude on the object (Hachet et al., 2003) (Figure 74). With the help of embedded sensors, such as gyroscopes and accelerometers, rotation of smartphones and tablet-PCs can also be leveraged to rotate virtual objects (Du et al., 2011; Yoon et al., 2011; Kasahara et al., 2013; Mossel et al., 2013).

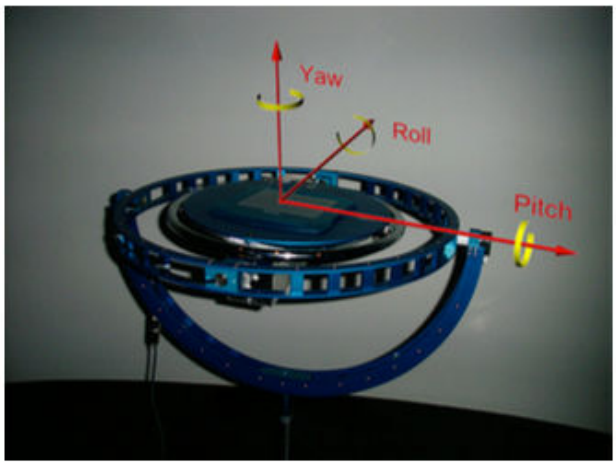


Figure 74: Orientation of virtual objects can be controlled by rotating the CAT prototype (Hachet et al., 2003).

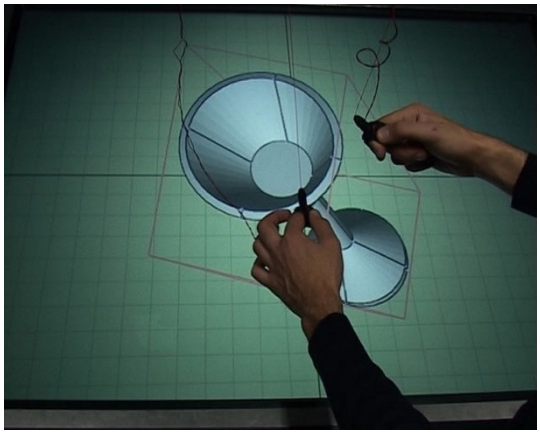


Figure 75: Performing the bimanual pinch gesture in the Mockup builder to scale the object (De Araújo et al., 2013).

In many studies, object scaling can simply be made by performing the pinching gesture. Mockup builder allows using the distance between two hands to change the scale of the selected object uniformly (De Araújo et al., 2013). If both the relative distance and orientation between the two hands change, the selected object can be rotated and scaled

simultaneously (Figure 75). In the comparison made by Mendes et al., all the techniques permit using the relative distance between two hands to perform uniform scaling (Mendes et al., 2014). In the user-defined manipulation glossary proposed in the work of (Piumsomboon et al., 2013), as shown in Figure 71, three uniform scaling gesture are proposed. The first gesture requires users to drag each diagonal corner of the target and move two hands in the xy-plane to enlarge or shrink it. The second one requires moving two hands apart or together along the x axis to change the scale. In contrast with these two gestures, the third one allows using the distance between the thumb and other fingers of the same hand for object scaling. Although users have found these gestures intuitive to perform, some of them have raised some potential problems. For example, the second and the third gestures require a mechanism to trigger or clutch the manipulation. In addition to uniform scaling, some gestures have also been proposed to change the scale along a principle axis. As shown in Figure 71, axis constraint scaling can be performed by using either two hands or a single hand.

The HOME-S technique permits scaling virtual objects by moving a smartphone (Moussel et al., 2013). When an object is selected and the scaling mode is activated, the smartphone can be dragged along one axis to perform axis-constraint scaling. If the smartphone is moved in the positive direction of one axis, the size of the object increases in the chosen dimension. Moving the smartphone in the opposite direction shrinks the object.

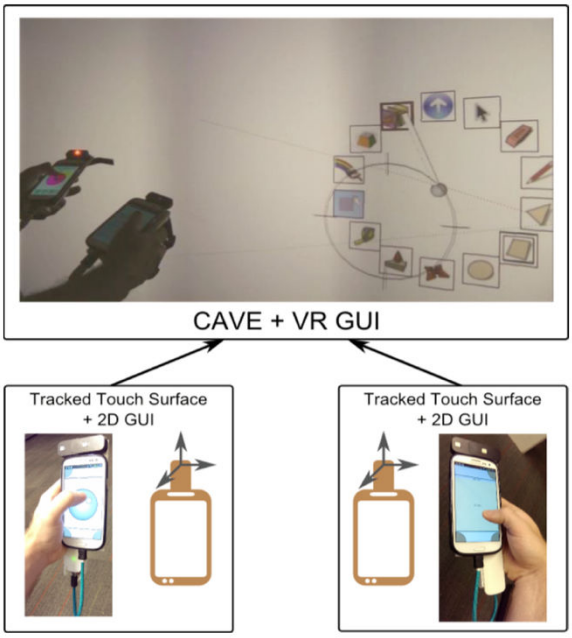


Figure 76: Sketchup VR interface (Mine et al., 2014).

Some techniques provide the possibility of setting a manipulation constraint for fine-grained manipulation. Mine et al. have built a 3D modeling interface in an immersive virtual environment based on the use of two tracked touch surfaces (Mine et al., 2014) (Figure 76). This interface supports three ways for object manipulation. First, users can

grab a virtual object within their arm's reach, and then position and rotate it in a manner similar to real world manipulations. If it is necessary to add some constraints, such as translation direction or rotation axis, such operations can be done on the interface of the touch surface. Users can also choose to map hand movements within the field of view to screen space interaction. They can also manipulate objects via a touchpad widget on the touch screen to simulate mouse interaction. This interface enables users to also make interaction at a distance to control objects outside their motor space. The Action Plane Widget (APW) is proposed to allow users to define the coordinate space in which the selected object is to be transited and rotated. The movements of both hands can be used to translate and rotate the APW in space. After setting the plane constraint, manipulation modalities and transformation ranges can be set by using two touchable surfaces.

Grossman et al. have augmented interaction techniques by freehand gestures for 3D volumetric displays (Grossman et al., 2004). In addition to freeform manipulation, users can also set a constraint axis to perform constrained manipulation. A constraint axis can be set by pointing the index finger of the dominant hand on the screen in the desired direction. Subsequently, object translation and scaling can only be performed along the constraint axis and object rotation is restricted around the same axis as well.

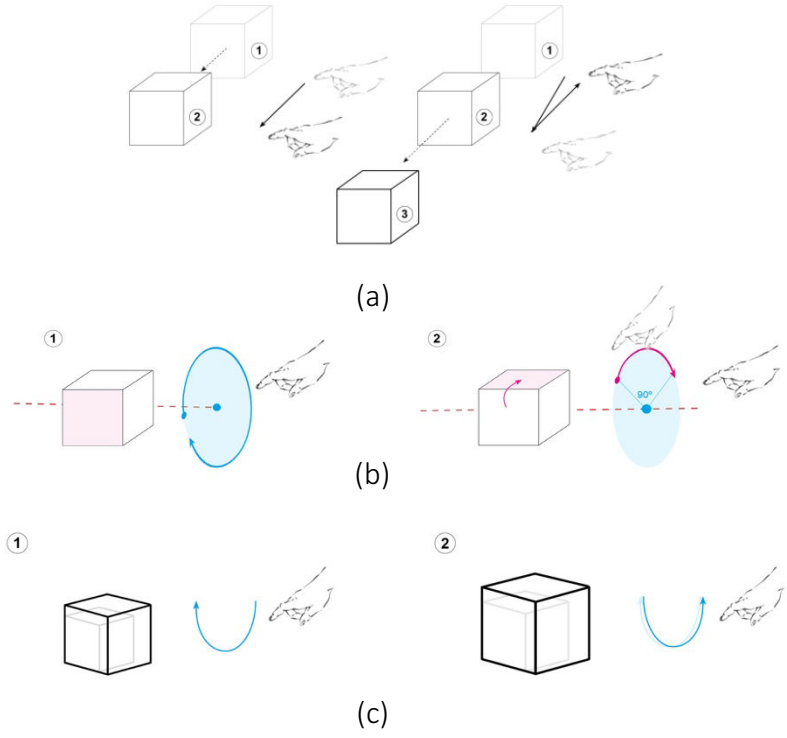


Figure 77: Gestures of the FingerOscillation

Wu et al. have proposed using periodical oscillatory finger movements in air to make constrained manipulation (Wu et al., 2014). Object translation and rotation can be

performed by oscillating the finger in the linear and circular way respectively (Figure 77 (a) (b)). For object translation, the first stroke is used to determine the translation direction while the other strokes are used to control the translation distance and velocity. For object rotation, the rotation axis is determined by the normal of the circle drawn by the user. To zoom the object, users should repeat drawing the upper (or the lower) part of a circle in the plane parallel to the display to magnify (or shrink) the object (Figure 77 (c)). The benefit of this technique is that continuous control of the transformation can be made without using clutching gestures. Moreover, hand and arm movements of large range are avoided and fewer muscles are involved in.

3.2.2 Tool-based mechanism

Besides mimicking the actions that are used to control objects directly, some other studies have proposed some tool-based techniques for object manipulation. Kitamura et al. have proposed using virtual chopsticks for object manipulation (Kitamura et al., 1999). In this work, hand tracking is accomplished by using a hand gesture input device. A virtual hand is reconstructed in a virtual reality environment and a pair of chopsticks is held by the virtual hand. Users can move the fingers and the wrist to manipulate the virtual chopsticks as real ones. Virtual objects can be picked up by virtual chopsticks and then can be positioned and rotated freely. As a lot of persons are familiar with chopsticks, almost no cognitive cost is necessary to learn this technique. A user study has shown that this technique can be used to accomplish precise object alignment tasks.

Song et al. have also leveraged the Kinect to develop mid-air manipulation techniques by using a handle bar metaphor (Song et al., 2012). When both hands perform the pointing gesture, users can control a handle bar by controlling two ends of it by both hands. When a virtual object is crossed by the handle, users can close two hands to select it for manipulation. Object translation, rotation and scaling can be performed intuitively through the handle bar movements (Figure 78). However, the performance of rotating the handle bar around the x axis is degraded by the occlusion problem. Thus, the authors have proposed using the cycling motion of two hands to rotate the object around the x axis. The handle bar can also be used to select multiple objects and perform alignment by tilting it. In addition, object manipulation gestures can also be used to control the camera. In the study of Mendes et al., this technique outperformed other techniques regarding efficiency and had equivalent performance as the 6-DOF technique (Mendes et al., 2014).

Fröhlich et al. have proposed a set of physically-based manipulation techniques for virtual environments using hand held devices (Fröhlich et al., 2000). When a hand held device touches a virtual object, a virtual spring is attached to the device. Users can drag and orientate the device freely in space to control the virtual object as a physical one. This system also supports gravity simulation and collision detection, which makes the virtual environment manipulation more realistic. The string model applied on the object depends on the number and the position of the contact points and also the physical constraint.

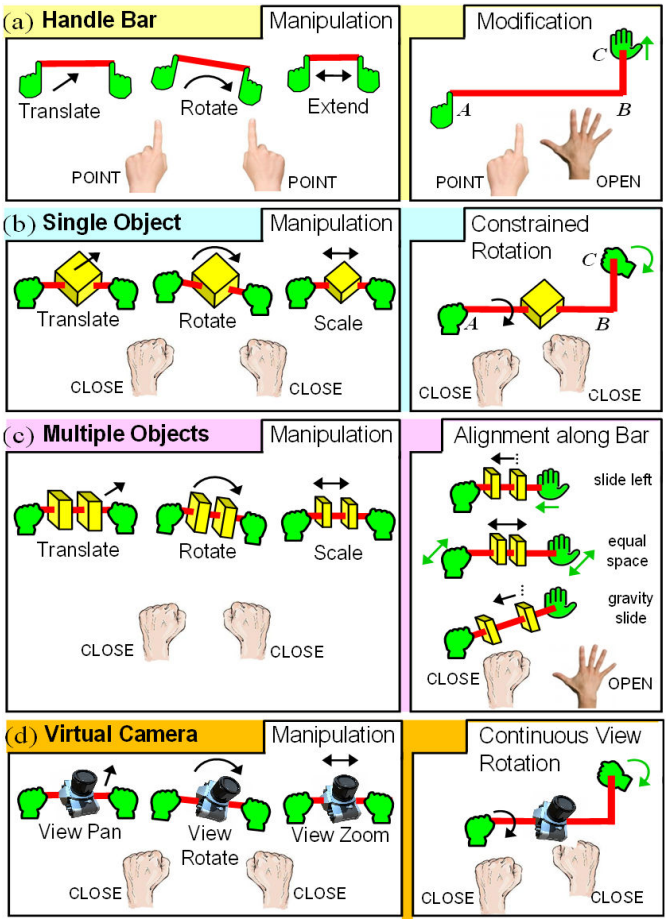


Figure 78: Manipulation gestures of the Handle-Bar technique (Song et al., 2012).

3.3 Discussion

We have also grouped 3D input-based manipulation techniques in Table 7 for comparison. As can be seen on Table 7, many research works in the recent years have been conducted to explore how hand movements in air can be used for 3D manipulation. Inputs can be made with the help of hand held devices or using directly bare hand movements and gestures. Most of the techniques support 6DOFs manipulations while a few of them support 9DOFs manipulations. Although these techniques are developed for different interaction scenarios, many of them use the same or very similar gestures for translation, rotation and scaling. In our opinion, the reason is that users use frequently these gestures to manipulate physical objects. However, we have identified two potential research directions. The first is that there is a lack of techniques which can be used for constrained manipulation. Although many techniques in the literature allow performing translation, rotation and scaling separately, they do not allow users to set a directional constraint. For example, users can rotate an object freely, however it is not possible to specify a rotation axis and rotate the object around this axis. On the one hand, it is better to allow users manipulate virtual objects freely when it is desired to simulate the physical world. On the other hand, for design work, it is more important to control the object precisely. Inspired by the work of

(Cohé et al., 2011) and that of (Hachet et al., 2011), one solution for this issue is to redefine manipulators for gestures in air. The manipulators should adapt to the characteristics of freehand gestures input. For instance, it should be tolerant to the relative low precision of freehand gestures. To reduce the fatigue, the proposed technique should involve fewer and smaller muscles of the hand and the arm. Nowadays, because of the emergence of some advanced HMDs such as Oculus Rift and HoloLens, the virtual reality may become popular for the public. It is interesting to explore how freehand gestures can be used in such environments to help designers improving their productivity.

Table 7: 3D input-based manipulation techniques

Technique	DOFs	Input	Constrained manipulation	Mechanism	Number of hands	Appropriate for fixed or mobile usage
Rockin'Mouse (Balakrishnan et al., 1997)	Tx, Ty Rx, Ry	Rockin'Mouse	Yes	Natural movements	1	Fixed
VideoMouse (Hinckley et al., 1999)	Tx, Ty, Tz Rx, Ry, Rz	VideoMouse	Yes	Natural movements	1	Fixed
Virtual chopsticks (Kitamura et al., 1999)	Tx, Ty, Tz Rx, Ry, Rz	Gestures in air	No	Tool-based	1	Fixed and mobile
Physically based manipulation (Fröhlich et al., 2000)	Tx, Ty, Tz Rx, Ry, Rz	Hand held devices	No	Tool-based	1 or 2	Fixed
CAT (Hachet et al., 2003)	Tx, Ty, Tz Rx, Ry, Rz	CAT (Control Action Table)	Yes	Natural movements	2	Fixed
Multi-Finger gesture interaction (Grossman et al., 2004)	Tx, Ty, Tz Rx, Ry, Rz Sx, Sy, Sz	Gestures in air + touch inputs	Yes	Natural movements	1 or 2	Fixed
Interaction in the air (Hiliges et al., 2009)	Tx, Ty, Tz Ry, Rz	Gestures in air	No	Natural movements	1	Fixed and mobile
Tilt & Touch (Du et al., 2011)	Tx, Ty, Tz Rx, Ry, Rz	Smartphone	No	Natural movements	1	Fixed
Continuous interaction space (Marquardt et al., 2011a)	Tx, Ty, Tz Rx, Ry, Rz	Gestures in air	No	Natural movements	1	Fixed and mobile
6D hands (Wang et al., 2011)	Tx, Ty, Tz Rx, Ry, Rz	Gestures in air	Yes	Natural movements	1	Fixed
Mobiature (Yoon et al., 2011)	Rx, Ry, Rz	Smartphone	No	Natural movements	1	Fixed
ShoeSense (Bally et al., 2012)	Tx, Ty, Tz Ry	Gestures in air	Yes	Natural movements	1 or 2	Mobile
Holodesk (Hiliges et al., 2012)	Tx, Ty, Tz Rx, Ry, Rz	Gestures in air	No	Natural movements	1 or 2	Fixed
Handle-bar metaphor (Song et al., 2012)	Tx, Ty, Tz Rx, Ry, Rz Sxyz	Gestures in air	No	Tool-based	2	Fixed
HyFinBall (Cho et al., 2013)	Tx, Ty, Tz Rx, Ry, Rz	Track balls	No	Natural movements	1 or 2	Fixed
Mockup Builder (De Araújo et al., 2013)	Tx, Ty, Tz Rx, Ry, Rz Sxyz	Gestures in air	No	Natural movements	1 or 2	Fixed
exTouch (Kasahara et al., 2013)	Tx, Ty, Tz Rx, Ry, Rz	TabletPc	Yes	Natural movements	1	Mobile

HOMER-S (Moussel et al., 2013)	Tx, Ty, Tz Rx, Ry, Rz	Smartphone	Yes	Natural movements	1	Mobile
User-defined gestures for AR (Piumsomboon et al., 2013)	Tx, Ty, Tz Rx, Ry, Rz Sx, Sy, Sz	Gestures in air	Yes	Natural movements	1 or 2	Fixed
HybridSpace (Bogdan et al., 2014)	Tx, Ty, Tz	Gestures in air	Yes	Natural movements	1	Fixed and mobile
Air TRS (Mendes et al., 2014)	Tx, Ty, Tz Rx, Ry, Rz Sx, Sy, Sz	Gestures in air	No	Tool-based	1 or 2	Fixed
Making VR work (Mine et al., 2014)	Tx, Ty, Tz Rx, Ry, Rz Sx, Sy, Sz	Smartphones	Yes	Natural movements	1 or 2	Fixed
FingerOscillation (Wu et al., 2014)	Tx, Ty, Tz Rx, Ry, Rz Sxyz	Gestures in air	Yes	Natural movements	1	Fixed and mobile
Roly-Poly mouse (Perelman et al., 2015)	Tx, Ty, Tz Rx, Ry, Rz	Roly-Poly Mouse	Yes	Natural movements	1	Fixed

4 Conclusion

In this chapter, we have presented the state-of-the art of 2D and 3D based manipulation techniques. Manipulation techniques are classified according to intrinsic parameters such as DOFs and manipulation mechanism etc. Because 2D input devices allow only controlling two dimensions, different techniques have been proposed to leverage gestures, new defined manipulators, or even device motions to control objects in all three dimensions. Compared to 2D manipulation techniques, 3D manipulation techniques permit mapping 3D freehand gestures and hand held devices motions directly to virtual object movements. On the one hand, this permits to leverage natural metaphors to provide a more intuitive manipulation experience. On the other hand, due to the lack of physical support, it is more difficult to perform fine-grained manipulation using 3D inputs.

In the following part, we will first make a conclusion of the first part, and then present our contributions in the next three chapter.

Conclusion of PART I

Conclusion of PART I

In chapter 1, we have presented concepts which are relative to our research topic. After presenting several definitions of NUIs proposed in the literature, we have given our own definition of NUIs. Besides using learnt skills for interaction, we think that it is also necessary to map them to interaction operations in an appropriate way to provide a natural experience. Then we have presented several potential modalities that can be used to develop NUIs. In the literature review we have presented, several studies have explored how to use these modalities independently or in collaboration to develop NUIs.

In the second part of this chapter, we have presented how interaction possibilities of mobile devices can be enhanced. Because nowadays most of the mobile devices allow using a touchscreen for interaction, we have first presented the concept of 2D gestures. Then we have presented the concepts of 3D gestures and device motion gestures. At last we have presented previous studies made in the literature to enrich interaction possibilities.

In chapter 2, we have presented the state-of-the-art techniques proposed for improving object selection. Selection techniques are divided into two groups according to the input dimensions: 2D and 3D selection techniques. For 2D selection, we have mainly presented mouse-based and touch-based selection techniques. Based on different strategies, many techniques have been designed to simplify the selection of small objects and avoid the occlusion problem. For 3D selection, we have mainly presented techniques which rely on the use of handheld devices or freehand movements to select objects at a distance. Similar to 2D selection techniques, 3D selection techniques were also divided into different groups according to the selection strategy.

Although a lot of 2D and 3D selection techniques have been proposed to simplify object selection, we think that the selection experience can still be further enhanced. For 2D selection, many techniques allow users to magnify the area close to the target. It is inevitable to lose the global view after magnifying a small area. In some applications such as map navigation, the global view is helpful to visually locate the target. In addition, some techniques allow moving the cursor in the relative mode to approach the target precisely. However, more time is required to adjust the cursor carefully, especially when the density of the cluster increases. To overcome these issues, we propose two novel touch-based selection techniques. Our techniques allow users to make precise selection without losing the global view. Users need to only make two successive simple gestures to specify the target. These techniques will be presented in chapter 4.

For 3D selection, many progressive techniques allow users to use physical buttons or freehand gestures for switching the operation mode explicitly. However, a better user experience can be provided if the technique can anticipate the users' intentions and switch the mode seamlessly. In addition, although a lot of techniques based on the usage of handheld devices are proposed, only a few freehand selection techniques have been designed. In fact, it is more difficult to design freehand techniques than handheld device

based techniques. First, there is a lack of physical buttons. Thus, users can only use gestures to switch the operation mode and trigger functions. Second, it is difficult to extract meaningful gestures from arbitrary motions because there is no explicit delimiter. Therefore, we propose two novel freehand selection techniques to simplify object selection at a distance. Our techniques can switch the operation mode automatically when the selection difficulty increases. Our two techniques allow dragging the hand horizontally and rotate the wrist respectively to refine the selection. To accelerate the selection, additional visual guides are also proposed. These techniques will be presented in chapter 5.

In chapter 3, we have presented the state-of-the-art of 2D and 3D based manipulation techniques. Manipulation techniques are classified according to intrinsic parameters such as DOFs and manipulation mechanism. Because 2D input devices only allow controlling two dimensions, different techniques have been proposed to leverage gestures, new defined manipulators or even device motions to control objects in all three dimensions. Compared to 2D manipulation techniques, 3D manipulation techniques permit mapping 3D freehand gestures and hand held device motions directly to virtual object movements. On the one hand, this permits to leverage natural metaphors to provide a more intuitive manipulation experience. On the other hand, due to the lack of physical support, it is more difficult to perform fine-grained manipulation using 3D inputs.

Although many 2D manipulation techniques have been proposed, there is still a lack of techniques which can be used for constrained manipulation on TabletPCs. First, because some existing touch-based techniques are designed for object manipulation on touchscreens of large size, they cannot be directly applied on mobile devices. Indeed, as the non-dominant hand is used to hold the device, it is difficult to perform some bimanual manipulation gestures. Second, designers and architects change the camera viewpoint frequently to observe and edit 3D models. However, the performance of many of the existing techniques is restricted by the camera viewpoint. To overcome these issues, we first propose a bimanual manipulation technique for mobile usage. Based on the asymmetrical model (Guiard, 1987), this technique allows using the non-dominant hand to set the constraint and using the dominant hand to control the transformation. Operation mode can be switched seamlessly and object manipulation can be performed in any perspective. In addition, we propose two other single-handed manipulation techniques for TabletPCs. These two techniques map the index, middle and ring finger to the x, y and z axes. After pressing an axis finger, a specific gesture can be performed to translate, rotate and zoom the object. Our manipulation techniques will be presented in chapter 6.

PART II: CONTRIBUTION

Chapter 4: 2D selection techniques - LayerStroke & LayerMenu

Chapter 4: 2D selection techniques - LayerStroke & LayerMenu

1 Introduction

For 3D interaction, selection is one of the fundamental tasks (Bowman et al., 2004). Object selection allows users to specify one or more objects that they want to manipulate in a 3D virtual environment. Performance of manipulation tasks depends directly on that of the selection tasks. The whole interaction experience would be significantly degraded if the selection technique is not well designed.

The previous literature review shows that touch inputs still suffer from the occlusion issue. Hence, it is still a challenge to select precisely a small object displayed on the screen. To solve this problem, several techniques such as Shift (Vogel & Baudisch, 2007), MagStick (Roudaut et al., 2008), Bubble Cursor (Grossman & Balakrishnan, 2005), Area Cursor (Worden et al., 1997) or Starburst (Baudisch et al., 2008) have been proposed. However, these techniques have some limitations such as long cursor adjusting time, influence of high cluster density or visual distraction.

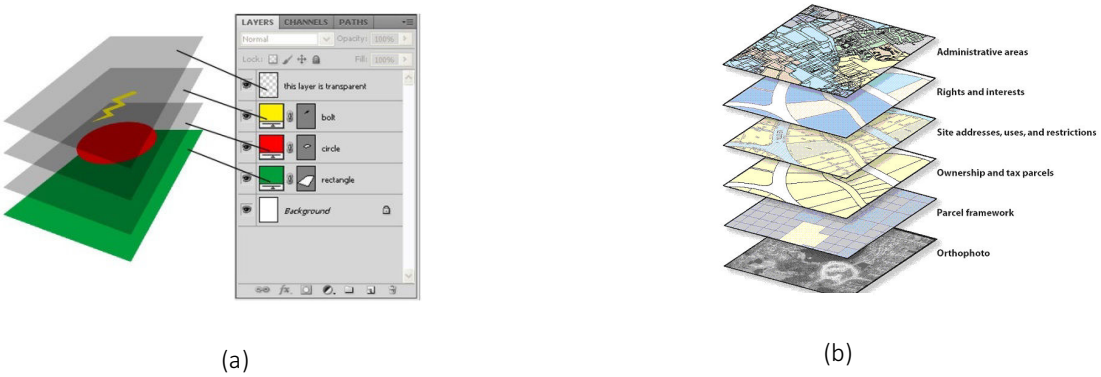
In addition to the occlusion issue, another challenge of selection on touchscreens is to reduce the finger motion precision requirement. Compared with controlling the cursor by a mouse, it is more difficult to tap the finger very accurately at one position on the screen. As a result, users may fail to select a small target by a single tap. Several approaches such as Escape (Yatani et al., 2008) and LinearDragger (Au et al., 2014) have been proposed to reduce the influence of the target size. Similarly, these techniques also suffer from some issues such as the high cluster density and the limited size of the screen.

In this chapter, we introduce two dual-step target selection techniques which are named LayerStroke and LayerMenu (Wu et al., 2015b)[1]. These techniques aim to improve the selection accuracy by dividing the targets space in the display into several layers in order to reduce the cluster density. In the following sections, we discuss the design and implementation of our techniques and present two preliminary studies conducted to refine the design of the techniques. Finally, we present an experimental evaluation to compare our techniques with one state-of-the-art selection technique.

[1] A video demo can be found in this link https://www.youtube.com/watch?v=RydibycQLd4

2 Design of selection techniques

Our techniques can be considered as mutations of Bubble Cursor. As discussed above, the performance of Bubble Cursor is influenced by the cluster distribution and selection becomes more difficult when the density increases. In our design, the objective was to reduce the influence of high distribution density. In some GIS (Geographic Information System) and design software, such as Adobe Photoshop, layers are used to organize different contents (Figure 79). A specific task, such as data manipulation or drawing can only be made in the selected layer. Inspired by this concept, we propose two techniques which can group objects in the display into different layers. Users can specify a layer and then select an object belonging to this layer. Because only parts of the initial distribution are grouped into one layer, the cluster density is significantly diluted. Therefore, using Bubble Cursor to specify an object in a layer is much easier.



(a) (b)

Figure 79: Usage of layers in Photoshop (a) and GIS (b).

When some selectable targets are displayed on the screen, the system first separates them into different layers (Figure 80 (a)). We propose a grouping algorithm to separate targets into several layers. Targets inside the same layer are displayed in the same color. For example, in Figure 80, targets of three different layers are displayed in blue, red and yellow respectively. In addition, similar to Escape (Yatani et al., 2008), a directional layer indicator is also displayed upon each target. Targets having the same indicator are in the same layer. In the example of Figure 80, the direction of blue targets is Up while that of the yellow targets is Right.

Instead of selecting an object directly, users have to first select the layer in which the target is located. Inspired by Escape (Yatani et al., 2008), we propose LayerStroke which permits drawing a stroke on the screen to select a layer. After recognizing the direction of the desired layer, users can draw a stroke using one finger in the same direction to select the layer (Figure 80 (b)). The drag gesture can be started from an arbitrary position on the screen and the stroke can be of any length. After the layer is selected, all the targets located in the other layers become semi-transparent (Figure 80 (c)). Furthermore, targets of the selected layer have the highest display order so that none of them are occluded by

semi-transparent targets. The display order can be changed by modifying the z-buffer value of each target. Finally, tessellation tiles of this layer are generated and visualized. After selecting a layer, the user can highlight a target by tapping one finger in the tile encircling it. Then, the finger can be released directly to select the target or can be dragged to switch the target in focus.

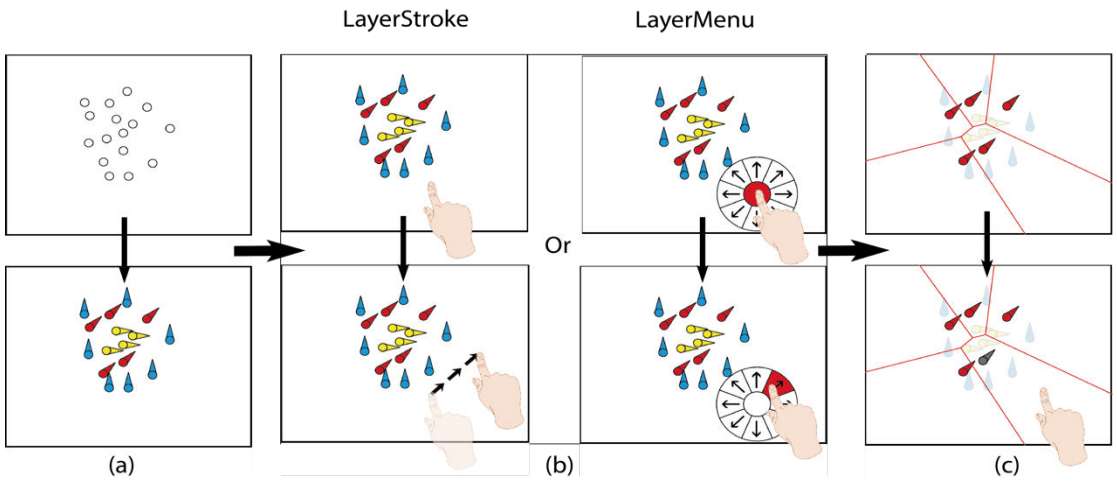


Figure 80: The selection procedures of LayerStroke and LayerMenu. (a) Divide targets into several layers; (b) Draw a stroke to select a layer; (c) Tap the finger in the tile of a target to select it.

Different from LayerStroke, LayerMenu enables layer selection in a different way. When tapping the finger on the screen, a circular menu with eight portions appears (Figure 80 (b)). Each layer is represented by one portion which is displayed in the same direction as the layer indicator. After setting the position of the menu, the finger can be moved to one portion to highlight the layer represented by it. As a result, targets outside this layer become semi-transparent and tessellation tiles of this layer are generated and visualized. Dragging the finger to another portion permits to change the highlighted layer. If the finger returns to the menu center or goes outside the menu, all targets become opaque and tessellation tiles are hidden. If the finger is released from one portion, the layer it represents can be selected.

To cancel the selection of a layer, a two-finger pinch enlarging gesture can be made on the screen. After that, tessellation tiles of the selected layer disappear and all the semi-transparent points turn back to opaque.

In the initial design iteration, we have only designed LayerStroke. However, through an informal test, we have found that the performance of LayerStroke is sometimes degraded if the layer indicator of the target is occluded by other objects. In this case, users cannot determine whether the desired layer is selected correctly only until a stroke is drawn. If the selected layer is not the desired one, it is necessary to cancel the layer selection and draw a stroke in another direction. To solve this issue, we have designed LayerMenu, which supports previewing layers by switching the portion in focus in the menu. When the user is not sure about which layer to be specified, he/she can check each layer and releases the

finger when the desired one is found. Although through an informal test, we did not found a significant performance difference between LayerStroke and LayerMenu, we thought that LayerMenu is more helpful in some conditions. For instance, when occlusion occurs, it is difficult to decide the belonging layer of the target using LayerStroke.

LayerStroke and LayerMenu have several advantages:

(1) **Combination of two simple commands** - They require only one drag gestures to select a layer and one tap gesture to select a target.

(2) **No spatial limit for layer selection -** The drag gesture of LayerStroke can be started from an arbitrary point on the screen. The pivot point of the circular menu of LayerMenu can also be set anywhere.

(3) **Reduced distribution density** - By separating targets into different layers, the distribution density is highly decreased. Moreover, to reduce the visual distraction of unwanted layers, their targets become semi-transparent and are displayed below those of the selected layer.

(4) **No fingertip occlusion** - Because tessellation tiles for each target are large enough, finger tapping can be made in an empty area of the tile to avoid occlusion.

2.1 The grouping algorithm

Our grouping algorithm is based on the following steps to generate layers:

(1) **Make Voronoi Tessellation for all the targets**. As shown in Figure 81 (b), given a dense targets cluster, the coordinates of targets are first normalized and then a Voronoi diagram is generated to make screen partitioning for all the displayed targets.

(2) **Grouping targets according to their tiles size**. After making Voronoi tessellation, we calculate the size of each tile. All the targets whose tile is large enough for direct tap are added to group 1 and all the remaining points are put in group 2. After several tests, we have decided to set the threshold value for the tile area to 0.005.

(3) **Check the tile width**. Besides the tile area, we need also to check the width of the tile. If the tile is too narrow, it is still difficult to tap it even if its area is large enough. To check the width of the tile, we note the coordinates of all its corners and use PCA (Principal Component Analysis) to calculate the eigenvectors and eigenvalues. If one of the two largest eigenvalues is smaller than a threshold value, it means that the tile is too narrow along the direction of the eigenvector corresponding to the small eigenvalue. In this case, we remove the target of this tile from group 1 and put it in group 2. The threshold value for the eigenvalue was set to 0.05 after different pilot tests.

(4) **Generate a new layer**. After extracting targets whose tiles are too narrow to tap from group 1, we generate the first layer and put all group 1 targets inside it (Figure 81 (c)).

(5) **Repeat the same procedure for the remaining targets**. We repeat procedures from step 1 to step 4 for the remaining targets (Figure 81 (d)) until all the targets are added to a new generated layer. Figure 81 (e, f, g, h) show the generation of other layers for the remaining points.

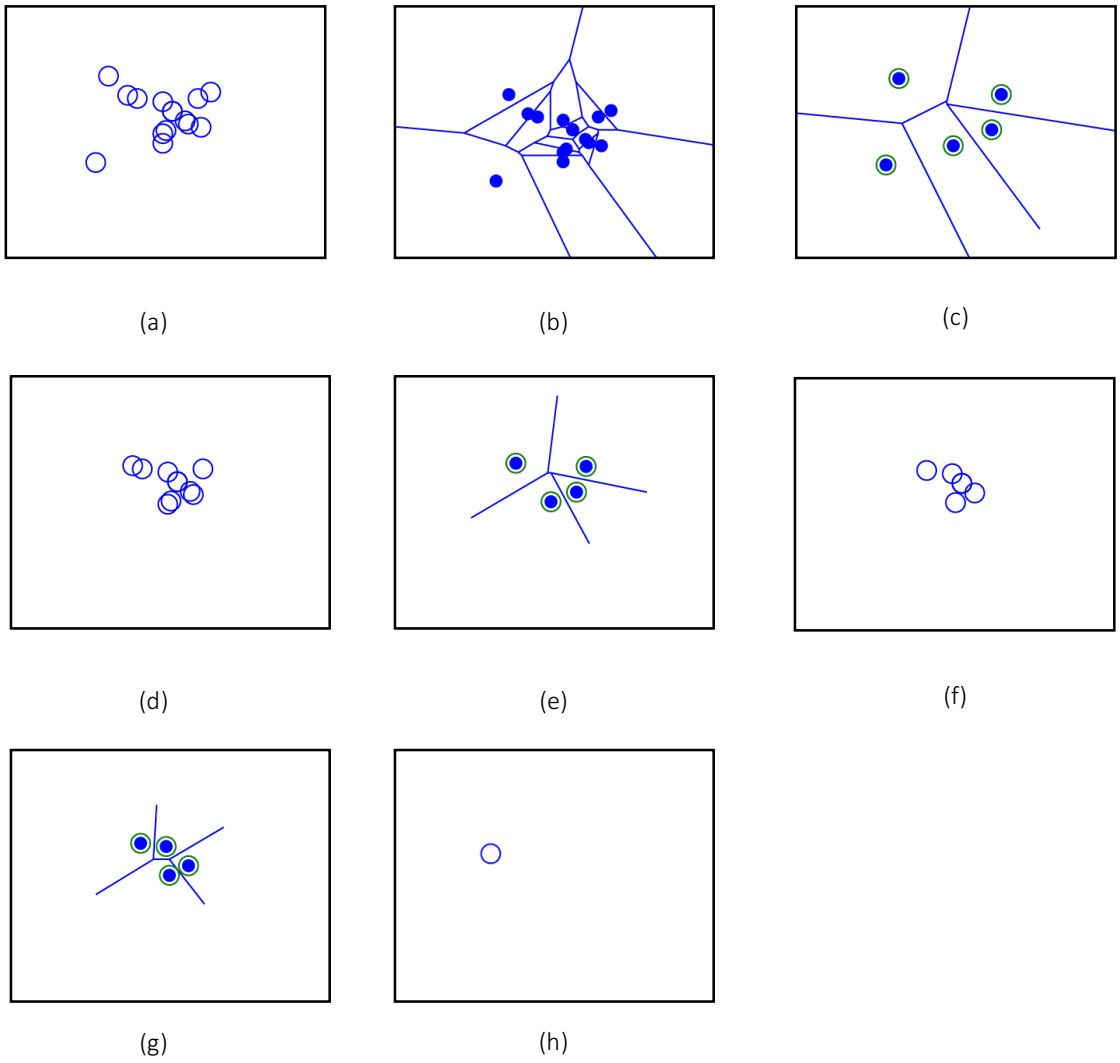


Figure 81: Generation of layers. (a)Initial targets cluster; (b) Voronoi Tessellation of initial distribution; (c) The first layer; (d) The remaining targets after generating the first layer; (e) The second layer; (f) The remaining targets after generating the second layer; (g) The third layer; (h) The fourth layer.

To investigate some design choices of our selection techniques, we have conducted two preliminary studies.

2.2 Display of tessellation tiles

After a layer is selected, visualization of tiles can help the users determine more easily the effective width of targets. However, displaying additional contents may also cause some visual distraction. Thus, we have conducted a preliminary study to determine the effect of displaying tessellation tiles on selection time and errors rate. We hypothesized that displaying tiles after the layer selection is helpful to accelerate the selection tasks.

2.2.1 Apparatus

The experiment was conducted on an Apple iPhone 5s, with a 4.065″ touchscreen. The screen is of 58.6×123.8 mm and has a resolution of 640×1136 pixels. Content was displayed in the Portrait mode. The experimental environment was developed using Unity3D with C#.

2.2.2 Participants

Eight unpaid subjects (7 males and 1 female, 23 to 28 years old) participated in this study. They were all students from a university. Six of them use smartphones frequently in everyday life and all of them are right handed.

2.2.3 Design and procedure

In this experiment, the participants were asked to use LayerStroke to select a goal target from a cluster. Before starting the selection task, a button was displayed in the center of the display. Once the button was tapped, 12 circular targets of 12 pixels in diameter were distributed randomly within a circular region of 100 pixels in diameter. Twenty distracting targets of the same size were generated randomly in the remaining empty space of the screen. In this preliminary study, no target overlapped each other. All the targets were displayed in white color except the goal target, which was displayed in blue. All the targets have already been added to a layer and the layer indicator was displayed upon each target. To evaluate whether displaying tessellation tiles has a significant effect on the selection time and errors rate, each participant was asked to perform two groups of selection trials. In the first group, tessellation tiles were displayed after a layer was selected while in the second group, tessellation tiles only appeared when a finger was tapped on the screen (Figure 82). For each group, 30 selection trials had to be completed.

The study was a within-subjects design. The within-subjects factor was **Group** with two levels (group1 and group 2). To counterbalance the order of selection tasks, four participants started with the first group while the other four participants started with the second group. Before the preliminary study started, we explained to the subjects the objective of the experiment and how LayerStroke works. After that, the participants had 5

to 10 minutes of free practice to become familiar with the selection technique. A total of 8×2×30=480 selection trials were performed in this experiment, with each participant completing 60 trials.

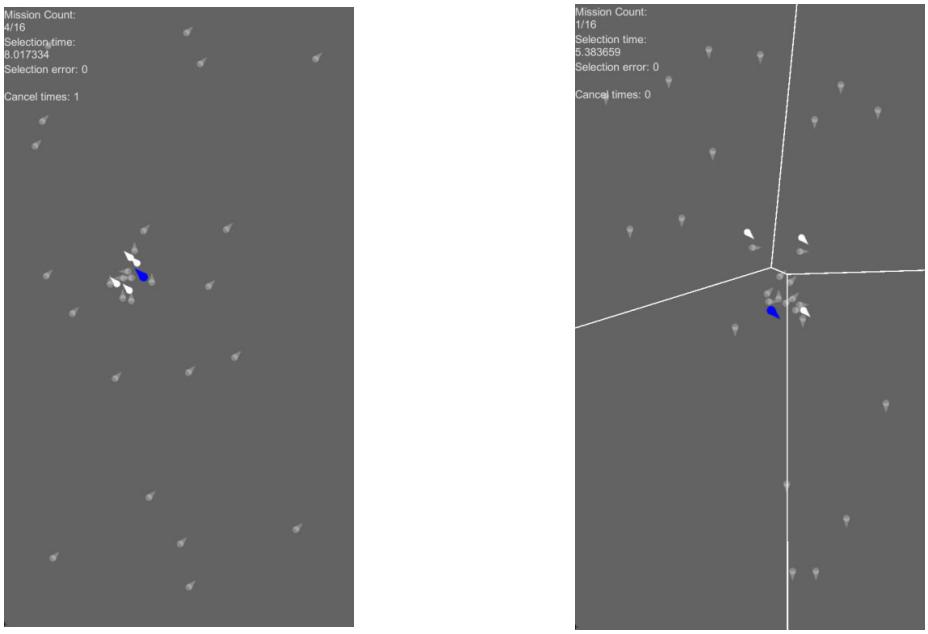


Figure 82: Two groups in preliminary study 1. (a) Voronoi tessellation tiles are not displayed before the screen is touched; (b) Tiles are displayed once a layer is selected.

2.2.4 Measurements and data analysis

For each task, we recorded the selection time and errors rate. The timing started after the start button was tapped and stopped when the goal target was selected. When a selection error was made, the timing did not stop and continued until the right target was selected. An error was recorded when one or multiple targets other than the goal target were selected. The errors rate was calculated by dividing the count of trials with errors by the total count of trials. All statistical analyses were performed using SPSS v.22.0 (IBM Corp., Armonk, NY, USA).

2.2.5 Results and design implications

The mean selection time was 1936 ms when tessellation tiles were displayed all the time and 3022 ms when tessellation tiles were displayed only when the finger taps the screen. The paired-samples t-test show that a significant difference exists between the two groups regarding the selection time ($t_{(7)} = 18.66$, $p < 0.0001$). The errors rate was 0.8% and 5.8% respectively for group 1 and group 2. No significant difference was found between them.

Results indicate that displaying tessellation tiles improves the selection time and does not influence the errors rate. This partially validates our hypothesis. When the targets tiles were visualized, it helped the user deciding where to tap on the screen. Participants were more decisive and almost no dragging motion was necessary. However, when tiles were displayed only after tapping a finger on the screen, users hesitated for a moment before deciding for the tapping position. When the tapping position was not located in the tile of the goal target, participants had to adjust the contact position on the screen to switch the highlighted target. We have also found that when adjusting the finger position on the screen, the desired target was more likely to be occluded by the moving finger. As a result, users had to tilt the smartphone to expose the target. In general, the decrease in selection speed is mainly caused by hesitation before tapping and adjustment of finger position after tapping. After this experiment, participants were asked to choose which group they preferred. All of them have chosen group 1 and they have all commented that displaying tessellation tiles before touching the screen was helpful. This led to the design decision to display tessellation tiles once a layer is selected and this setting is kept for the next experiments.

2.3 Layer colors

In our first prototypes, targets were displayed in the same color. In this case, before selecting a layer, the desired target could be occluded by other targets so that the direction of the layer indicator is hard to recognize. The experiment done in (Yatani et al., 2008) has shown that selection performance can be degraded when the peak of the indicator is occluded. To give users more cues for recognizing the peak direction, we have decided to display targets using various colors. When the indicator of the desired target is occluded, users can still determine the peak direction by observing targets of the same color as long as the color of the desired target can be recognized. Thus, a second preliminary study has been conducted to examine whether displaying various colors for different layers can help reducing the selection time and errors rate. We hypothesized that displaying various colors for different layers can accelerate the object selection. In this study, tessellation tiles are displayed once a layer is selected, as suggested by the previous study.

2.3.1 Apparatus, participants and measurements

The same apparatus as those in the preliminary study 1 were used in this experiment. Eight subjects participated in this study. They were the same participants who have been recruited in preliminary study 1.

2.3.2 Procedure and design

Similar to preliminary study 1, participants were asked to select a goal target from a dense targets cluster. However, some modifications have been made in this experiment. In fact, the goal target was generated in the center of the cluster for all the selection trials and targets could overlap each other. The target size was the same as in preliminary study 1.

Sixteen targets were generated in the circular region in this experiment to make the cluster denser. The study was a 2×2 within-subjects design with two within-subjects factors: **Display order** and **Display color**. **Display order** had two levels: Top and Bottom. Top means that the goal target is displayed on the top of the cluster so that there is no occlusion on the goal target. Bottom means that the goal target is displayed in the bottom of the cluster so that other targets can occlude it. There were also two levels for **Display color**: MonoColor and MultiColor. In MonoColor condition, all but the goal target were displayed in white color while the goal target was displayed in blue (Figure 83 (a)). In MultiColor condition, targets inside the same layer had the same color and their color was different from targets in other layers (Figure 83 (b)). To highlight the goal target, a black contour was displayed around it. Participants were asked to use LayerStroke to accomplish four groups of selection trials. The configuration of **Display order** and **Display color** of each group was different. For each group, 30 selection trials should be performed. In total there were 8×2×2×30 = 960 selection trials. Both selection time and errors rate were recorded.

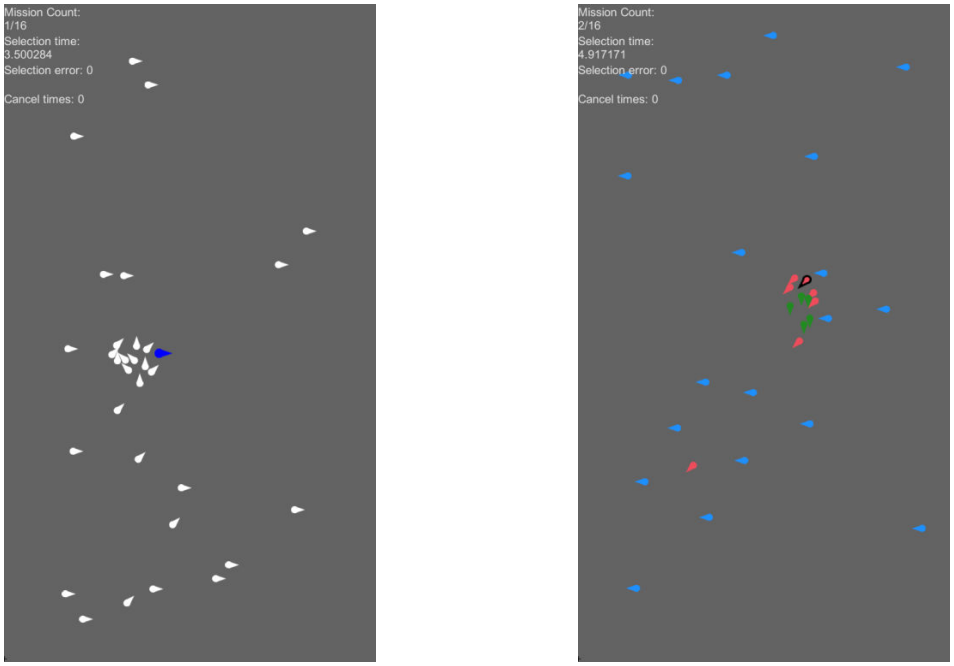


Figure 83: Two ways to display colors of objects. (a) All the objects except the target are displayed in white; (b) Each layer is represented by one specific color.

2.3.3 Results and design implications

A 2-way repeated measure ANOVA was conducted and a significant main effect was found for Display order ($F_{(1,7)} = 12.69$, $p = 0.009$), and for Display color ($F_{(1,7)} = 5.91$, $p = 0.045$).

A significant interaction effect was also found for **Display order**×**Display color** for $F_{(1,7)} = 5.76$, $p = 0.047$). The mean selection times were 2222 ms and 2794 ms for MultiColor and MonoColor respectively. No significant effects were found for errors rate.

It is reasonable to find out that occlusion of the goal target slowed down the selection time. It took more time for participants to recognize the direction of the layer indicator. The results also show that when occlusion occurs, displaying various colors can help improving the selection efficiency. Thus, our hypothesis is validated. All the participants thought that MultiColor was more helpful and they could find the direction of the goal target by observing other targets of the same color. On the other hand, they could only guess a direction when all the targets were displayed in white color. In this case, there was a lack of cues to infer the indicator direction. This led to the design decision of using MultiColor for our technique.

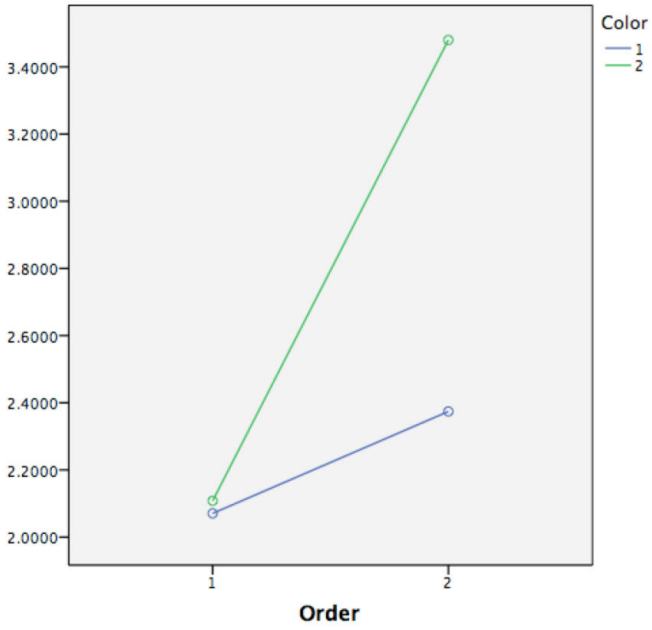


Figure 84: Interaction of Display order and Display color. Order 1 means Top and Order 2 means Bottom. Color 1 means MonoColor and Color 2 means MultiColor.

3 Main experiment: performance comparison

The aim of this experiment was to evaluate the performance of LayerStroke and LayerMenu on mobile devices. The research question is whether our techniques can provide a better selection performance than other state-of-the-art techniques within a dense targets cluster. In this experiment, our acquisition techniques were compared with LinearDragger proposed in (Au et al., 2014). LinearDragger was chosen for comparison because it is the most recently proposed touch-based acquisition technique. According to the evaluation done in (Au et al., 2014), it outperforms some other state-of-the-art methods such as Bubble Cursor (Grossman & Balakrishnanm 2005), Shift (Vogel &

Baudisch, 2007) and Escape (Yatani et al., 2008). The other argument for selecting LinearDragger is that it is also a method proposed for mobile devices. Users can use only one finger to complete precise selection on the screen. Before conducting the experiment, we have made three hypotheses:

H1: Selection time would decrease when using LayerStroke and LayerMenu as compared with LinearDragger.

H2: The errors rate would decrease when using LayerStroke and LayerMenu as compared with LinearDragger.

H3: When using LayerStroke and LayerMenu, changing the direction of the layer indicator of the goal target does not influence the selection performance.

3.1 Apparatus

Same as that used in preliminary studies 1 and 2.

3.2 Participants

Twelve participants (8 males and 4 females, 21 to 28 years old) were recruited. All of them are right handed and have experience in using smartphones with touchscreens. They are all students from a university.

3.3 Experimental design and procedure

As in the preliminary studies, participants were asked to select a goal target from a dense target cluster displayed on the screen of an iPhone 5s. Similar to the main experiment done in (Au et al., 2014), the clustered targets were generated inside a circular region of 100 pixels in diameter. No targets inside the region overlapped each other. The center of the circular region was generated in the central area of the display. Twenty distraction targets were placed in the empty space outside the circular region. Their size was the same as that of the clustered targets. Content in the display was shown in Portrait mode.

The study was a 3 ∗ 3 ∗ 3 within-subjects design experiment, with three within-subjects factors:

(1) Technique (**Tech**) - Three acquisition techniques were compared: LinearDragger, LayerMenu and LayerStroke.

(2) **Size** - Three target sizes were compared: 6, 12 and 18 pixels in diameter.

(3) **Count** -Three target counts inside the cluster were compared: 8, 12 and 16.

To eliminate the influence of trial order, **Tech** order was counterbalanced. Twelve

participants were divided into 6 groups and the permutation order was the same in each group. For each **Tech**, participants had to perform 3×3 = 9 subgroups of selection trials to evaluate the influence of **Size** and **Count**. Thus, there are in total 3×9 = 27 subgroups for each participant to complete. The order of subgroups was randomized. Furthermore, to investigate whether the direction of the layer indicator had a significant influence, the goal target was added twice to each layer and in total participants had to perform 16 selection trials for each subgroup. A total of 12×3×3×3×16 = 5184 selection trials were performed in this experiment. Before the experimental sessions, instructions were given to participants on how to use each selection technique, and they had 5 to 10 minutes of free practice to get familiar with each technique. After the experiment, participants were also asked to fill in a questionnaire to evaluate each technique. They were asked to select the fastest technique, the most precise technique and to choose their preferred techniques.

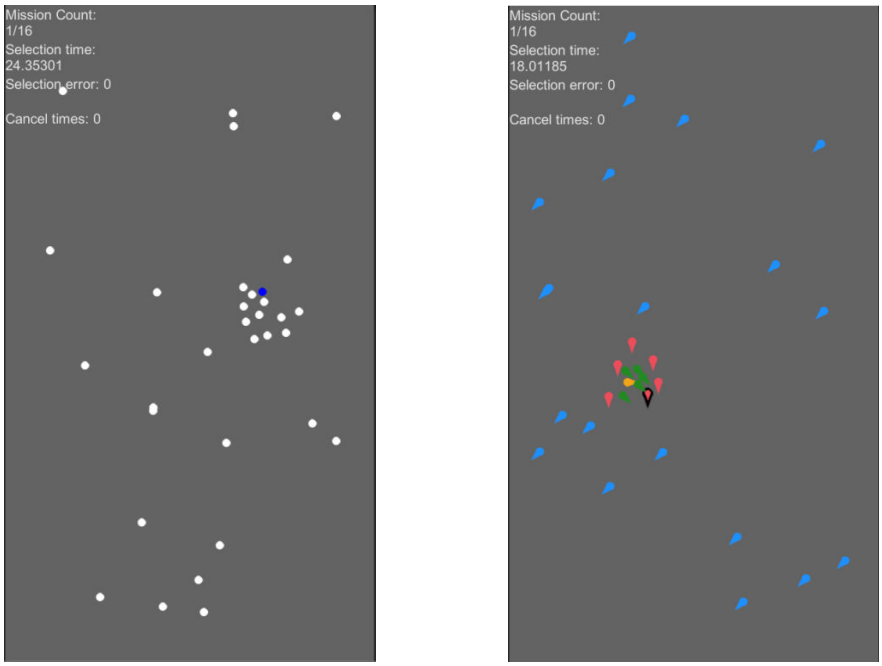


Figure 85: Cluster placement for different techniques. (a) LinearDragger; (b) LayerStroke and LayerMenu.

In our experimental application, the dragging threshold distance of LinearDragger was set to 60 pixels. After tapping the fingertip on the screen, a magnified view around this tapping position could be displayed if the finger motion was longer than the threshold distance. A ROI around the start tapping position was used to preselect close targets for selection refinement. The radius of the ROI is 40 pixels. The effective width of LinearDragger is set to 15 pixels. We have tried using a longer effective width to switch targets inside the ROI. However, the screen of iPhone 5s is much smaller than that of a tablet computer. If a bigger effective width was chosen, the screen space would become

too small for dragging the finger to switch the focus inside the ROI to the goal target.

When using LinearDragger, all the targets other than the goal target were displayed in white color while the goal target was in blue (Figure 85 (a)). For LayerStroke and LayerMenu, targets were displayed in different colors to represent layers (Figure 85 (b)). A black contour was displayed on the goal target. Although circular targets did not overlay each other, the layer indicator displayed upon them could occlude targets. Since we found that displaying various colors for targets of different layers can help the user recognize the indicator direction, we decided to keep this configuration.

3.4 Result

3.4.1 Selection time

A 3-way repeated-measure ANOVA was used to compare the performance of the three techniques. Results show a significant main effect for **Tech** ($F_{(2,22)} = 96.654$, $p < 0.0001$), **Size** ($F_{(2,22)} = 10.708$, $p = 0.003$). No significant main effect was fount for **Count**. The mean selection times were 3239 ms for LinearDragger and 1909 ms for LayerStroke, and 2042 ms for LayerMenu. No significant interactions were observed. The post-hoc tests with Bonferroni correction shown that LinearDragger is significantly slower than the two other techniques at the 0.05 level of significance (Table 8 and Table 9). No other significant differences were observed among the two other techniques. This validates our first hypothesis.

Table 8: The post-hoc tests with Bonferroni correction on mean selection time among Tech by Size. LD: LinearDragger, LS: LayerStroke, LM: LayerMenu. The mean selection time and the standard deviation is shown in each cell of the first three columns.

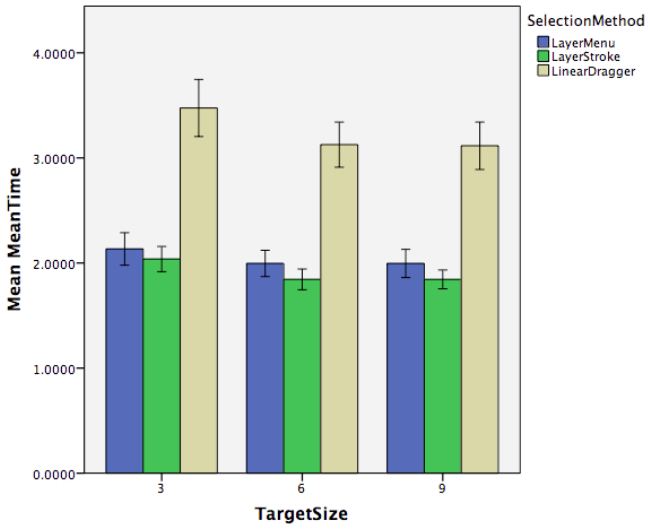
Bonferroni test (alpha = 0.05)						
Size	LD	LS	LM	LD/LS	LD/LM	LM/LS
3	3.475s (0.803s)	2.038s (0.354s)	2.134s (0.457)	p<0.0001 *	p<0.0001 *	p=1.000
6	3.126s (0.634s)	1.845s (0.291s)	1.996s (0.369)	p<0.0001 *	p<0.0001 *	p=0.486
9	3.116s (0.667s)	1.844s (0.265s)	1.996s (0.397)	p<0.0001 *	p<0.0001 *	p=0.532
*Significant effect						

Table 9: The post-hoc tests with Bonferroni correction on mean selection time among Tech by Count. LD: LinearDragger, LS: LayerStroke, LM: LayerMenu.

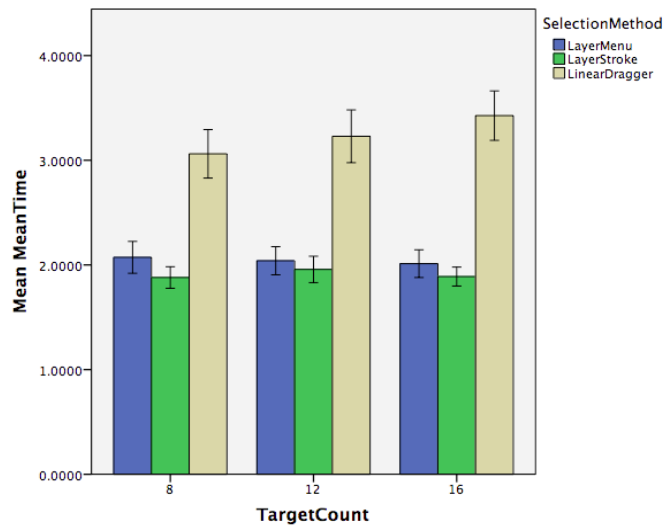
Bonferroni test (alpha = 0.05)						
Count	LD	LS	LM	LD/LS	LD/LM	LM/LS
8	3.062s (0.684s)	1.880s (0.304s)	2.073 (0.450s)	p<0.0001 *	p<0.0001 *	p=0.325
12	3.229s (0.745s)	1.958s (0.372s)	2.041s (0.398s)	p<0.0001 *	p<0.0001 *	p=1.000
16	3.427s (0.698s)	1.889s (0.268s)	2.013s(0.392s)	p<0.0001 *	p<0.0001 *	p=0.532

*Significant effect

Figure 86 shows the mean selection times of different techniques grouped by Size and Count. In all the cases, LaerStroke takes the shortest mean selection time while LayerMenu had a little longer mean selection time. Compared to LinearDragger, LayerStroke reduced the mean selection time by 41% while LayerMenu reduced it by 37%.



(a)



(b)

Figure 86: The selection times of different techniques grouped by Size (a) and Count (b). The error bars are at the 95% confidential level.

3.4.2 Errors rate

The errors rates of the three techniques have also been compared. The 3-way repeated measure ANOVA shows a significant main effect for **Tech** ($F_{(2,22)} = 45.358$, $p < 0.0001$) and a significant interaction **Tech**×**Size** ($F_{(4,44)} = 5.421$, $p = 0.021$). No other significant main effects or interactions were found. The mean errors rate was 8.85% for LinearDragger, 1.56% for LayerStroke and 1.56% for LayerMenu, indicating that LinearDragger had significantly higher errors rate. So our second hypothesis is validated.

3.4.3 Direction of layer indicator

We have also investigated the influence of the layer indicator direction. After conducting a one-way ANOVA, no significant effect was found for directions of layer indicator. The errors rates for different directions were also checked and no significant difference was found. This suggested that the direction of the layer to which the goal target belongs to had no significant influence of the selection speed and precision. So our third hypothesis is validated.

3.4.4 Subjective evaluation

Eleven participants chose LayerStroke as the fastest technique while one chose LayerMenu. Five participants chose LayerStroke as the most precise one while six chose LayerMenu. One participant thought LayerStroke and LayerMenu were both the most precise. Ten participants chose LayerStroke as their preferred technique while the other two chose LayerMenu. Participants commented that LayerStroke was faster than LayerMenu because drawing a stroke had less constraint than using a circular menu. When using the circular menu, participants had to avoid dragging the finger outside the menu. One interesting observation is that some participants preferred examining the results of layer selection before releasing the finger. This may explain why LayerMenu required a little more time than LayerStroke. Regarding the selection precision, half of the participants commented that LayerMenu was more precise. When the goal target was occluded, it was possible to rotate the finger on the menu to find out the layer of the goal target. Using LayerStroke does not allow previewing the layer before selection.

3.5 Discussion

The results indicated that there was a performance gap between our techniques and LinearDragger. This could be explained by several observations. First, LayerStroke and LayerMenu were less influenced by the position of the goal target. To select the layer of the goal target, users could make a stroke or arouse the layer menu in an arbitrary position on the screen. On the other hand, LinearDragger required a little amount of time

to select a target, although the ROI could help to cover the desired target. When the smallest target size was applied, it required more concentration to locate the goal target. Second, almost no precise control of finger motion was necessary for LayerStroke or LayerMenu. The use of tessellation tiles simplified the target selection. However, LinearDragger required switching the focuses one by one to reach the desired target. Participants had to avoid overshooting finger motion. Although the target in focus was highlighted in red (yellow for the goal target), it lacked the hint to tell when the focus could switch from one target to the other. As a result, some cautious participants had to drag the finger slowly. To overcome this issue, the target color can be altered dynamically to give some cues to help control the finger motion instead of changing the target color completely. Third, for LinearDragger, before the magnified view was displayed, it was hard to know which dragging direction corresponded to the shortest dragging distance in order to reach the desired target. In several trials participants chose a direction and found that they had to switch the focus many times to reach the goal target. Moreover, some participants found the screen not large enough for dragging. Thus, they had to perform the cancellation gesture and reset the ROI. Most of the participants dragged the finger down on the screen to open the magnified view of LinearDragger though the finger can be dragged in an arbitrary direction. After the experiment, some participants commented that dragging down the finger seemed more natural and more comfortable. This is mainly determined by the size of the smartphone and the posture used to hold it.

LayerStroke and LayerMenu were more precise because tessellation tiles were large enough for simple tapping. The participants rarely had to drag the finger to adjust the contact position. However, when using LinearDragger, participants had to drag the finger more carefully to switch the focus. Furthermore, if the goal target was not inside the ROI, participants had to perform a cancellation gesture to close the magnified view. However, some participants commented that they sometimes forgot to perform the cancellation gesture and released the fingertip directly. Because releasing the finger meant selecting a target, this could explain some of the errors that occurred with LinearDragger. Moreover, when releasing the fingertip to select the goal target, instead of lifting the finger vertically, some horizontal motion of the fingertip was generated in several times. As a result, the focus switched to another target and selection of the goal target failed.

4 Influence of distribution and location

Through the main experiment we have found that LayerStroke and LayerMenu significantly improve selection of small targets on touchscreens. In our prototype, we have set the maximum layers count to 8. However, if a target is surrounded by too many other targets, eight directions are not enough to encode each target by a unique vector. Thus, we have to check to which degree of density our algorithm can separate targets successfully into no more than 8 layers.

We have decided to apply our algorithm to different target distributions and observe how the layers count evolves when the distribution density increases. We have generated clusters of normal distribution whose density was controlled by the target count and the

standard deviation. We have first examined our algorithm with a target cluster generated in the center of the screen. For target count, we have chosen 8 settings: 5, 10, 15, 20, 25, 30, 35, 40, 45, and 50. For standard deviation, we have chosen 6 settings: 6.4 pixels, 12.8 pixels, 19.2 pixels, 25.6 pixels, 32 pixels, 38.4 pixels, 44.8 pixels, and 51.2 pixels. Therefore, in total, we had 6×8 = 48 configurations of target count and standard deviation. For each configuration, we have generated 100 clusters and applied our algorithm to each of them. The count of layers for each cluster was recorded to calculate the mean value for each combination.

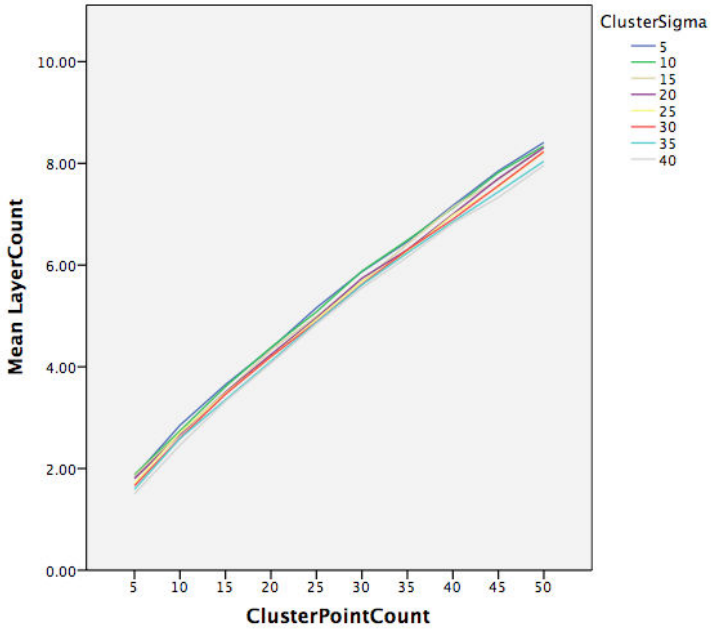


Figure 87: Count of layers of clusters in the center of the display.

Figure 87 shows how the layers count increases with the change of the target count and the standard deviation. First, we have observed that for all 8 settings of standard deviation, the layer count passes 8 when the target count is between 40 and 50. Second, we have found that our algorithm is insensitive to the change of standard deviation. Profiles of different deviations evolve at almost the same speed. No significant difference exists between profiles. This means that our algorithm can be used for a wide range of target distribution.

However, similar to Starburst in (Baudisch et al., 2008), LayerStroke and LayerMenu are influenced by the cluster center position. If the cluster is generated close to one edge or even a corner of the display, it becomes more difficult to give enough space to each target. Thus, we wanted to check how the cluster position influences the performance of our algorithm. Instead of generating the cluster in the center of the display, we have decided to generate two groups of clusters. In the first group, the cluster center was located close to the right edge of the display. The horizontal distance to the right edge was 32 pixels. The vertical coordinate of the cluster center was half the height of the display. In the second group, the cluster center was located close to the right bottom corner of the display. Both

the horizontal distance to the right edge and the vertical distance to the bottom edge were 32 pixels. For each group, we have conducted the same experiment to study the influence of target count and standard deviation.

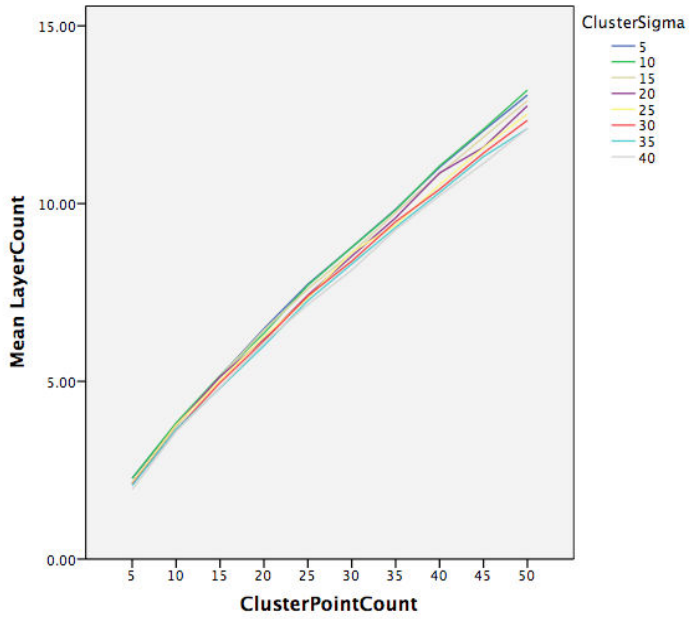


Figure 88: Count of layers of clusters close to right edge of the display.

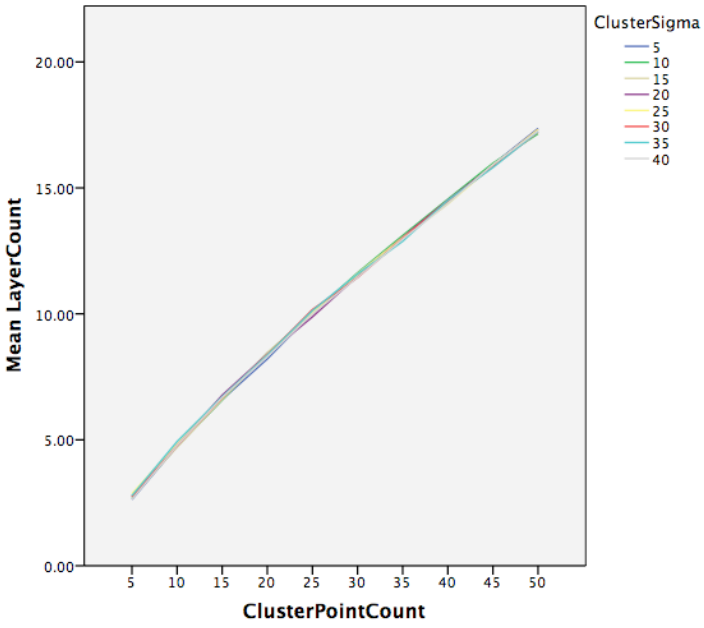


Figure 89: Count of layers of clusters close to the right bottom corner of the display.

As expected, we have observed that profiles increase at a higher velocity (Figure 88 and Figure 89). For the first group the layer count passes 8 when the target count is between 30

and 35. For the other group, 8 layers can separate about only 20 targets. This indicates that the performance of our algorithm degrades when the cluster is located close to the border of the display. The performance of our techniques is limited by the cluster location and may fail to divide targets into no more than 8 layers.

5 Limitations

The results of the main experiment show that LayerStroke and LayerMenu facilitate the selection of small targets from dense clusters efficiently. Our techniques can be helpful in touch-based applications, such as map navigation and manipulation of biomolecules. However, three limitations should be considered when implementing these techniques for mobile applications. First, the gestures used in our techniques may be already assigned to other functions. For example, in the Goggle Maps application, dragging one finger on the map is used to translate the map. There are several alternative solutions for this problem. First, instead of using one finger, two joint fingers can be used to draw strokes for LayerStroke or can be pressed on the screen to trigger layer selection for LayerMenu. Second, in the literature, some techniques have been proposed to identify fingers pressed on the screen and each finger can be mapped to a different function (Sugiura & Koseki, 1998; Goguey et al., 2014a; Goguey et al, 2014b). Although mobile devices in the market do not support finger identification, we think this can be achieved in the near future. To avoid the mapping conflict, we can map the thumb of the dominant hand to the existing function and use the thumb of the non-dominant hand to trigger LayerStroke or LayerMenu. Moreover, we can also use a circular button displayed in the corner to trigger the selection mode as shown in Figure 90.

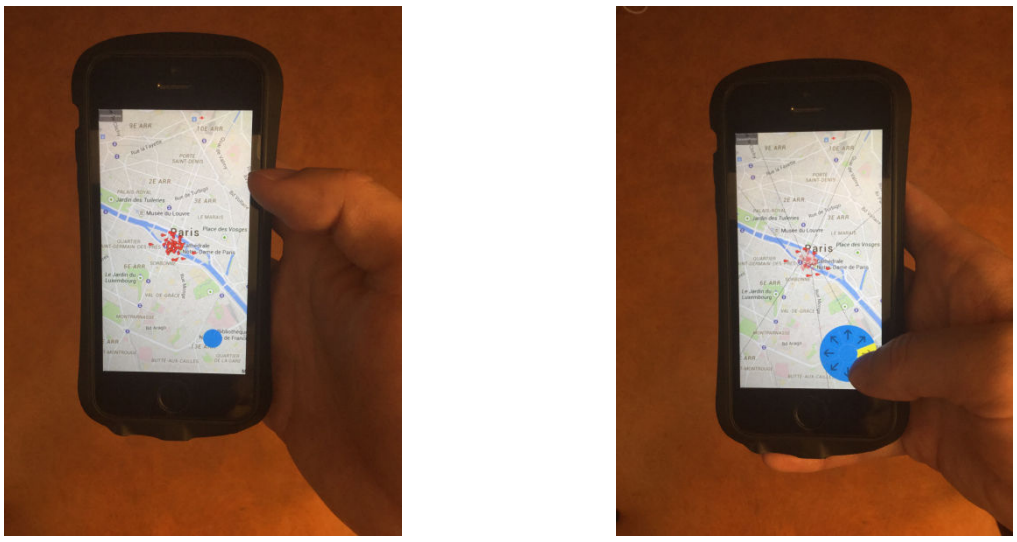


Figure 90: Implementation of LayerStroke in Google Map. (a)The layer menu before selection.;(b) Select a layer on the menu.

After tapping the button located in the right bottom of the display, a layer marking menu appears and layer selection can be made by dragging the finger in the corresponding

direction. The second limitation is that directional layer indicators and different colors are displayed to indicate layers. However, displaying vectors on some content, such as atoms of molecular models can be weird and may be visually distracting. Thus, displaying visual cues without distraction remains an interesting issue to investigate in the future. Third, in some applications, the camera view can be moved so that the target position may change in the display. As a result, two targets that were inside the same layer before moving the camera view may be separated in two different layers. Further investigations should be made to study whether these changes have an influence on the selection performance. In the future, we are planning to implement our technique in several other interaction scenarios and evaluate their selection performance. Besides 2D targets, we also plan to use these techniques for target selection in a 3D virtual environment.

6 Conclusion

In this chapter, two touch-based target acquisition techniques LayerStroke and LayerMenu were presented. By using our proposed algorithm, these two techniques can divide targets on the screen into several overlapping layers. Users can select a layer and make a tap gesture to select a target inside it. Our algorithm ensures that tessellation tiles are large enough for simple tapping without fingertip occlusion. Our techniques were compared with the state-of-the-art LinearDragger technique and results show that LayerStroke and LayerMenu outperform LinearDragger regarding both the selection time and selection accuracy. The advantage of our techniques is that accurate selection of small targets can be done without controlling very precisely the finger movement. They are also less limited by the small screen size of the smartphone. Although the main experiment show no significant performance gap between LayerStroke and LayerMenu, we think that LayerMenu is more convenient for real applications. Indeed, when the density of the cluster is very high and objects can overlap each other, LayerMenu permits to preview all the layers before selecting one of them. On the other hand, only one layer can be checked each time when using LayerStroke.

Besides 2D selection techniques, we propose also our own 3D selection techniques. In the next chapter, we will present HorizontalDragger, a freehand selection technique which is designed to select objects at a distance. Our hybrid technique can switch the operation mode automatically according to the selection difficulty. When it becomes difficult to select an object, our technique sets a region of interest and magnify it. After setting the region of interest, users can drag the hand horizontally to refine the selection.

Chapter 5: 3D selection technique - HorizontalDragger

Chapter 5: 3D selection technique - HorizontalDragger

1 Introduction

Nowadays, people are used to interact with computers using a mouse and a keyboard. However, in some situations, the use of conventional techniques to interact with the computer is inappropriate. For instance, high-resolution displays are used more and more frequently for scientific visualization, urban navigation and military operations planning thanks to their ability to display large scale information in details. In these scenarios, instead of being seated in front of the display, users are more likely to walk around to observe the content on different parts of the display. With these requirements for mobility, the use of the mouse and keyboard for interaction becomes inadequate. Besides interaction in front of a large display, Virtual Reality (VR) is also in need of interaction techniques which support mobility. In fact, to get a realistic immersive experience, virtual 3D environments are often projected in a CAVE or displayed directly using a head mounted display (HMD). As the virtual environment reacts to movements of users in real time, it is more suitable to design interaction techniques which are not limited by the position, the orientation and the pose of the user's body.

The requirement for mobility provides an opportunity but also a challenge for the design of interaction techniques by supporting spatial hand movements. On the positive side, using 3D hand movements offers new possibilities to explore the expressiveness and richness of the interaction. Users are not only able to use dexterous hand movements to take control of more degrees of freedoms (DOFs), but also can use familiar natural hand actions to interact directly with content of the virtual world. On the negative side, using 3D hand movements is more demanding for both users and input devices. Indeed, unlike controlling a mouse on the desk, there is a lack of physical support to stabilize the hand when holding it in air. As a result, the interaction is more likely to be influenced by the hand shaking movements and the fatigue. In addition, when using a mouse, the arm is almost static and only small muscles of the dominant hand are involved. However, moving the hand in air requires the participation of both the upper arm and the forearm. Larger and slower muscles are involved (Card et al., 1991; König et al., 2009). Finally, when interacting in the real world, haptics, in addition to vision, helps people better understand their environment. However, for most of the virtual environments, haptic feedback is not provided.

Numerous immediate and progressive selection techniques proposed in the literature rely on the use of hand held devices. Some techniques permit using freehand movements to make interaction, but users still have to wear gloves or attach fiducial markers for hand tracking. Recently, some tracking devices based on stereoscopic computer vision techniques became available in the market at a low cost. Devices equipped with a depth camera such as Microsoft Kinect and Asus Xtion can identify the user and reconstruct the

body skeleton. Leap Motion controller is a detector which focuses on real-time hand tracking with a precision of 0.1 mm. Unlike some other devices which require calibration before using, these tracking devices are more walk-up-and-use without special configuration. Without holding a device in the hand, more interaction techniques can be imagined and designed to take advantage of hand dexterity. However, before using freehand gestures directly for interaction, some problems should be solved. First, without the help of a hand held device mounted with physical buttons, it is necessary to propose a set of gestures for freehand interaction. It is still a challenge to design a set of gestures which are efficient for interaction, and also can be learned and memorized easily without long training periods. Second, it is necessary to solve the delimiter problem (Benko, 2009). Indeed, when using a hand held device, pressing a button can serve as a trigger. A touch-based gesture begins when the first finger is pressed on the screen and ends when all the fingers are released. However, it is difficult to determine the beginning and the end of a freehand gesture and extract meaningful hand movements from casual hand movements.

In this chapter, we present a new freehand selection technique which is named HorizontalDragger (Wu et al., 2014b, Wu et al., 2015a)[2]. Our first objective is to design a technique to simplify selection of small objects from a dense cluster which are displayed at a distance. Although some progressive selection techniques help improving the selection accuracy (Kopper et al., 2010), the accuracy is obtained at the cost of efficiency. Thus, we aim to find a better balance between the efficiency and the accuracy. Our second objective is to explore the use of freehand movements for interaction. Because there is a lack of physical buttons to serve as gesture delimiters, we design a technique which can switch the operation mode automatically when necessary. As a result, less explicit interaction is required to select an object.

In the following, we present the design and implementation of HorizontalDragger. During the design phase, we have conducted three preliminary experiments to fine tune some parameters for the technique. After that, we present a main experiment that was conducted to evaluate our technique by comparing it with two state-of-the-art selection techniques. Finally, we discuss the results of this experiment and give some recommendations for future developments.

2 HorizontalDragger

In this section, we first present the design of HorizontalDragger and then explain the implementation details.

2.1 Design

HorizontalDragger is designed to perform remote selection through freehand gestures. As

[2] A video demo can be found in this link https://www.youtube.com/watch?v=9kBRIwMqCyc

mentioned above, our objective is to propose a selection technique which ensures both selection efficiency and accuracy in different environments. Our technique is designed to adapt to the change of the density and switch automatically to the appropriate mode. When the selection tool is in a sparse cluster, the selection difficulty is low. Thus, the technique should make it possible to select an object rapidly. When the target is located in a cluttered environment, the technique should provide some guidance to ensure that the target can be specified correctly without making great efforts.

For rapid selection in a dense cluster, we have chosen an immediate selection technique from the literature. After making a comparison, we have chosen the Bubble Cursor technique (Grossman & Balakrishnan, 2006) from all the aforementioned immediate selection techniques. In fact, because Bubble Cursor allows users to select a target without casting a ray upon it, the selection is less affected by the hand trembling and the selection time is significantly decreased. Although techniques such as PRISM (Frees et al., 2007) and Adaptive Pointing (König et al., 2009) help to increase accuracy, they require users to locate the target using a selection device. Moreover, adjusting the CD ratio causes a mismatch between the hand pointing direction and the output direction. Bubble Cursor permits to avoid those issues.

As shown in Figure 91 (a), after stretching the index finger of the active hand in front of the display, a cursor is displayed on the screen and follows the movement of the index finger. A circle is displayed around the cursor and its radius evolves continuously to cover the closest target. If the target covered by the circle is large enough and is not located in a dense cluster, users can perform a thumb click gesture to select it directly.

However, when the target inside the circle is small and located in a dense cluster, our technique can switch to another mode automatically. Thus, selection refinement can be made in a progressive way. If the cursor slows down its movement close to a dense cluster in which an immediate selection technique does not work well, a region of interest (ROI) is set and magnified automatically in the display (Figure 91 (b)). After the ROI is set, selection refinement can be made among all the objects inside it. As shown in Figure 91 (b), these objects are magnified and copies of them are rearranged in a horizontal layout besides the ROI. Users can drag the index finger in the direction parallel to the display to scan objects inside the ROI one by one (Figure 91 (c)). Dragging the finger to the right can scan objects in the horizontal layout from the left to the right. The scanning direction can be inversed by dragging the finger to the left. An object can be highlighted if the finger is located inside its corresponding dragging interval. In Figure 91 (c) and Figure 91 (d), dragging intervals of the same width are separated by dash lines. Note that they are not visible for users and are only used for explanation. If the finger is dragged into the dragging interval of one object, its color changes continuously according to the finger location inside the interval. All objects whose intervals have already been passed by the finger are highlighted completely (Figure 91 (c) and Figure 91 (d)). The colors of objects and their copies change simultaneously in the same way. This configuration is designed to help users predict the moment when the focus will change and make the visual feedback coherent with the hand movements. Once the desired target is highlighted, the user should stop the

finger movement and keep it static for 0.5 second to trigger the selection.

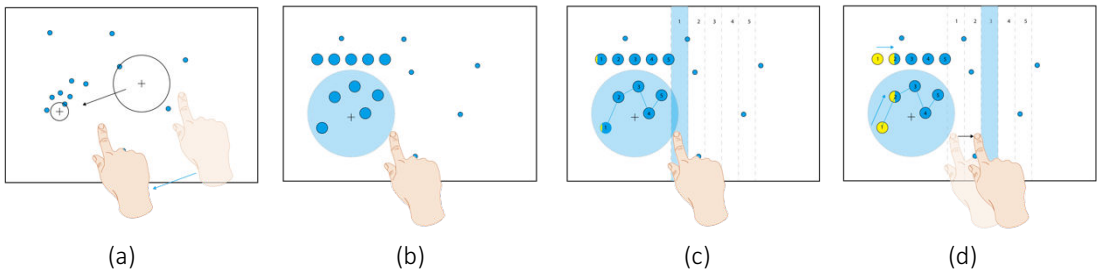


Figure 91: The selection procedures of the HorizontalDragger. (a) Move the cursor by index finger movement; (b) Set the region of interest and open the magnified view automatically; (c)(d) HorizontalDragger: Move the index finger horizontally to scan targets inside the ROI.

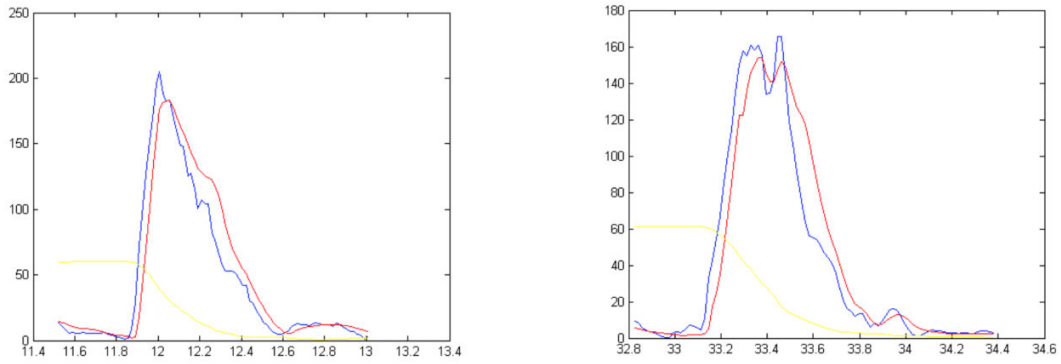
2.2 Setting the ROI

In our first design iteration, HorizontalDragger required performing a thumb click gesture to set manually the ROI. However, through an informal study, we have identified some drawbacks for this design. First, we have observed that many users did not perform the thumb gesture until the cursor was very close to the target. They commented that the gesture was performed so late because they wanted to ensure that the target could be covered by the ROI. As adjusting the cursor position and stabilizing the finger is time consuming, the selection efficiency is affected. Second, for some users with large hands, when the thumb click gesture was performed, the index finger jumped up involuntarily all the time. As a result, the ROI was set in an incorrect position and sometimes the target was missed. Subsequently, we have conducted a new design iteration that aimed to allow the users to specify the ROI without the need for a triggering gesture.

Our proposition was to design a technique similar to Bubble Lens proposed in (Mott & Wobbrock, 2014). Bubble Lens is a mouse-based technique which can magnify automatically the area around the cursor when the algorithm finds it difficult to make object selection using the Bubble Cursor. The design of Bubble Lens is inspired by Optimized initial impulse model proposed in (Meyer et al., 1990). Optimized initial impulse model divides the approaching movements into two phases: Ballistic phase and Corrective phase. In the Ballistic phase, the selection tool is moved rapidly to approach the target while in the Corrective phase, a set of consecutive corrective movements are made to locate the target precisely. The main idea of Bubble Lens is to anticipate the transition between the Ballistic phase and the Corrective phase and provide a magnified view when Corrective movements start. To determine the right moment to open the magnified view, the velocity profile of the cursor is checked in each frame. The authors found that the second downslope of the profile is a dependable pattern when corrective movements start.

While in Bubble Lens the cursor was controlled by a mouse (Mott & Wobbrock, 2014), in our interaction scenario, the cursor is controlled by freehand movements in air. To check whether the model is still valid for freehand interaction, we first made a pointing test to collect data of cursor movements. At the beginning of this pointing test, a circle button was

displayed in the center of the display. After keeping the cursor on the button for 0.5 second, the button disappeared and a target of 20 pixels in diameter appeared in the display in a random position. The user had to move the cursor to the target. After the cursor was kept upon the target for 0.5 second, the trial ended and the circle button appeared again to be used to start the next trial. Three volunteers were asked to repeat the pointing task 30 times, and we have recorded the trajectory of the cursor and also its velocity profile. After smoothing the speed profile using a Gaussian kernel filter, we have checked how the cursor speed changed during the pointing task. In Figure 92, velocity profiles of two pointing tasks are displayed. We found that after applying the filter, the velocity profile had a similar pattern as that of the cursor controlled by the mouse (Figure 92 (a)). However, in some profiles the ballistic phase contained two peaks of similar values and they were close to each other (Figure 92 (b)). Compared with the mouse-based paradigm, more muscles of the arm and the hand are involved for freehand interaction. The cursor movement is influenced by more human factors and more jitter is likely to be generated. We have tried applying a Gaussian kernel filter with a larger kernel to further smooth the velocity profile, but this generated more latency and smoothed the peak of the corrective movement. Therefore, we have decided to add one condition to check the value of the second peak of the profile. In fact, because in the corrective phase, the cursor velocity is much slower than that of the ballistic phase, we have to check the value of the second peak. If it is slower than a threshold velocity, then it is considered as the downslope of the second corrective movement. Otherwise, it is considered as noise and is ignored. After several pilot tests, we have set the threshold to 20 mm/s. We have also added a condition to identify the first peak of the profile. Because the hand cannot be held completely static, we did not want the system to consider some low peaks of the profile as the starting of the ballistic movement. Then, we have set the value to 60 mm/s.



(a) (b)

Figure 92: velocity profiles of two pointing tests. Blue one is the raw velocity profile, red one is the filtered profile and the yellow one is the distance front the cursor to the target.

In general, to know whether it is necessary to prepare the magnified view, the velocity profile is checked to find the downslope of the second peak. Two velocity conditions are checked to remove noise. Because we wanted to provide the magnified view only when the

target is located in a dense cluster, after the second downslope of the profile is found, the algorithm checks the edge distance between the cursor and the nearest object, and also the effective size of the closest object. The edge distance is the distance between the cursor center and the nearest edge of the object. To measure the size of the target and the density in its vicinity, we use Equation 3 to calculate the effective edge:

$$Effective\ Size = D_A + \frac{\overline{AB} - R_A - R_B}{2} \qquad \text{Equation 3}$$

where D_A is the diameter of object A covered by the circle, $\overline{AB}$ is the distance between object A and its closest neighbor, object B. R_A and R_B are the radius of object A and object B, respectively. We have set the threshold value of the edge distance to 20 pixels and the threshold value of the effective size to 15 pixels. If all the conditions are satisfied, the magnified view can be displayed.

2.3 Drag start direction

The time required to highlight the target in the scanning list depends on the target count inside the ROI and the order of the target in the list. Our first version of HorizontalDragger allowed scanning objects inside the ROI only from left to the right. Thus, if the desired target was at the right extremity of the scanning list, the user had to switch the focus one by one to reach the end of the list. Selecting a target with a higher order took longer time but also required longer finger crossing distance.

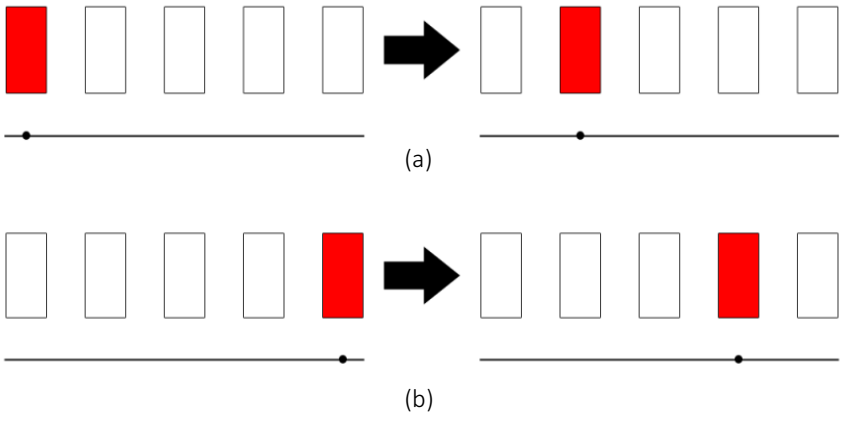


Figure 93: (a) Start dragging the finger from left to right; (b) Start dragging the finger from right to left.

In a circular menu different options are organized in a circular layout where the first option and the last are adjacent. Inspired by the circular menu layout, we have decided to connect the beginning of the scanning list to the end in the control space. In the second version, once the magnified view is displayed, users can still start dragging the finger to the right to

switch the focus from the left extremity to the right extremity (Figure 93 (a)). Furthermore, when the object at the right extremity of the list is reached but the finger continues moving to the right, the focus will jump back to the left extremity. Moreover, after magnifying the ROI, dragging the finger to the left can switch directly the focus to the rightmost object in the list (Figure 93 (b)). Continuing dragging to the left switches the focus to an object of lower order in the list. During a non-formal test, we have found that this mechanism could help reducing the dragging distance. For example, if the target is located near the right edge of the display, the user drags first the finger to the right to set the ROI. If the desired target is at the right extremity of the scanning list in the ROI, instead of continuing dragging the finger to the right, the user can drag back the finger to switch directly the focus to the end of the list. In addition to reducing the dragging time, this also permits to avoid dragging the finger too far away from the body. The active hand can be kept in a relative smaller space to perform object acquisition tasks. Therefore, freehand interaction can be made in a more comfortable motor space.

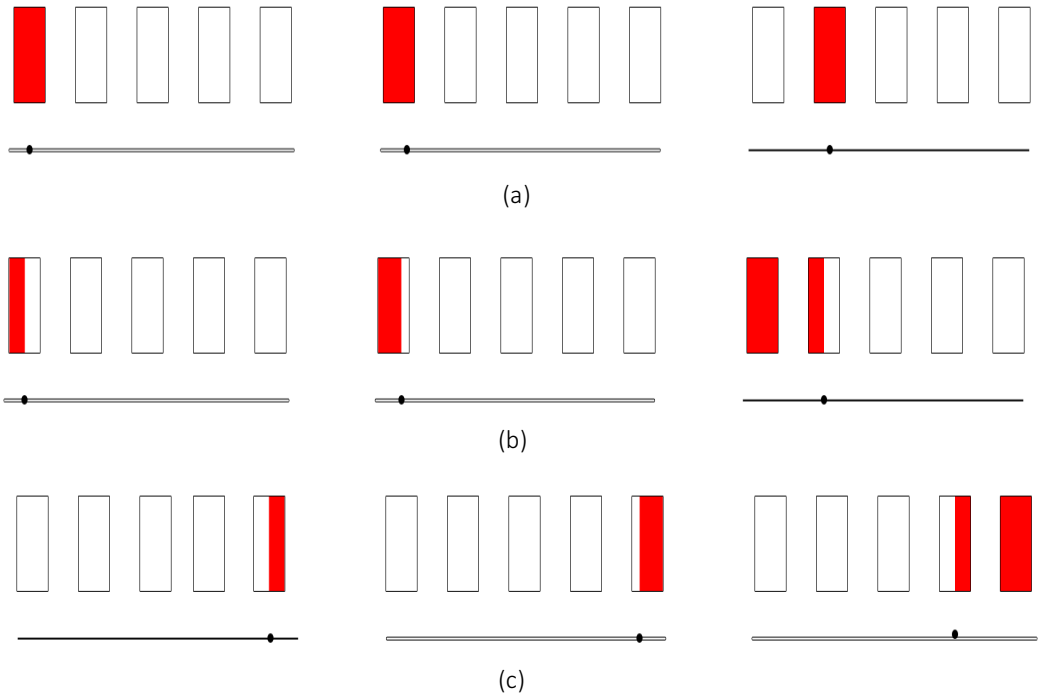


Figure 94: (a) Static visual feedback; (b) Dynamic visual feedback, begin dragging from left to right; (c) Dynamic visual feedback, begin dragging from right to left.

2.4 Visual feedback

Our first version of HorizontalDragger provided only a static visual feedback as shown in Figure 94 (a). The black point in the bottom is used to represent the finger position. Static visual feedback highlights the target when the finger is located in its corresponding interval. Although it is possible to determine which target can be selected with the help of this

feedback, there is a lack of visual cues to inform the user when the finger will move to the adjacent interval. As a result, it is difficult to adjust the finger velocity. If the speed is too slow, more time is required to reach the desired target. Oppositely, dragging the finger too fast is likely to overshoot the desired target and move the finger towards an undesired interval. To overcome this issue, we have decided to provide more visual cues to help the users anticipate when the focus will change.

Thus, to indicate the finger position, we have designed a dynamic visual feedback similar to a progress bar. As shown in Figure 94 (b), when the finger starts moving from the left to the right and is located in the left side of the first interval, only a small part of the first target is filled in red. Then, when the finger moves to the right, the area filled in red becomes bigger. Finally, once the finger passes the threshold of the first interval and enters the second interval, the first target is completely filled in red and the second target starts to be filled in red progressively. Because the color changes progressively according to the finger position, it is easier for the user to adjust the finger dragging speed. In an informal test, we have asked some volunteers to try both the static and the dynamic visual feedback. Our observations have shown that, with the help of dynamic visual feedback, finger movements were more fluent and users were able to control the movement speed more easily. In preliminary study 2, we have compared different visual feedback to evaluate their effects on the performance.

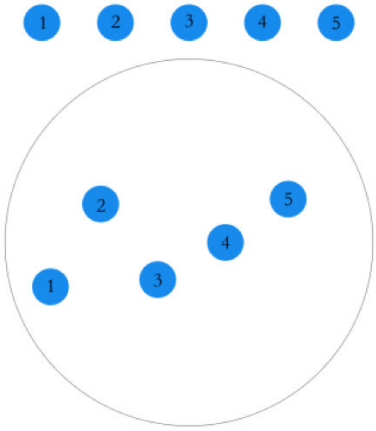


Figure 95: Horizontal layout of copies of pre-selected targets.

2.5 Horizontal layout of target proxies

Because targets inside the ROI are not horizontally distributed, inputs of horizontal finger movements and outputs of target scanning are not visually coherent. To help the user understand the focus switch order, we have first considered drawing segments to connect adjacent targets in the scanning list. However, the dragging order was still a little confusing, more particularly when the variance of the horizontal coordinates was much smaller than that of the vertical coordinates. Indeed, users were hesitating when trying to figure out the dragging direction and this hesitation decreased the selection speed. To provide a better

mapping between the target distribution and the finger dragging movement, we have decided to display duplications of targets in a horizontal layout beside the ROI (Figure 95), while the initial distribution remains in the magnified view. One potential problem is that targets of similar appearances are hard to distinguish. Displaying distinctive patterns for each target can solve this problem. For example, we can use numbers to encode all the pre-selected targets.

2.6 Implementation

The mapping function of HorizontalDragger uses a constant effective width to switch the focus in the ROI. The mapping function is shown in Equation 4.

$$k = \begin{cases} \left(\frac{P_x - P_{x0}}{EW}\right) mod N + 1, & P_x > P_{x0} \\ n - \left(\frac{P_{x0} - P_x}{EW}\right) mod N, & P_x < P_{x0} \end{cases} \qquad \text{Equation 4}$$

Where k is the index of the focused target, P_x is the horizontal coordinate of the current position of the index finger and P_{x0} is the initial horizontal coordinate of the index finger when the ROI is set, EW is the effective width of the dragging interval and N is the count of potential targets. When HorizontalDragger is triggered, if the finger position is at the right side of the starting position, the first condition is satisfied. Thus, the list can be scanned from the left to the right. In contrast, when the finger position is at the left side of the starting position, the second function is used to calculate the index of the highlighted target. The index is calculated using the remainder of the division between $\frac{P_x - P_{x0}}{EW}$ and N in order to connect the two extremities of the scanning list. EW is used to control the sensibility of the dragging movement. Its appropriate value was determined based on a preliminary test.

3 Preliminary Study 1: dragging sensibility

Before comparing HorizontalDragger with other remote target acquisition techniques, we have first conducted a preliminary study to determine the appropriate value of the dragging interval used for selection refinement. In fact, if the interval is too small, selection refinement would be too sensitive to the finger movement. In contrast, if the interval is too big, the efficiency would be sacrificed.

3.1 Apparatus

The experiment was conducted on a Macbook Pro computer, with a 15" screen. Hand movements and gestures were captured using a Leap Motion controller which was placed in front of the computer (Figure 96). The experimental application was implemented using Unity3D with C#.

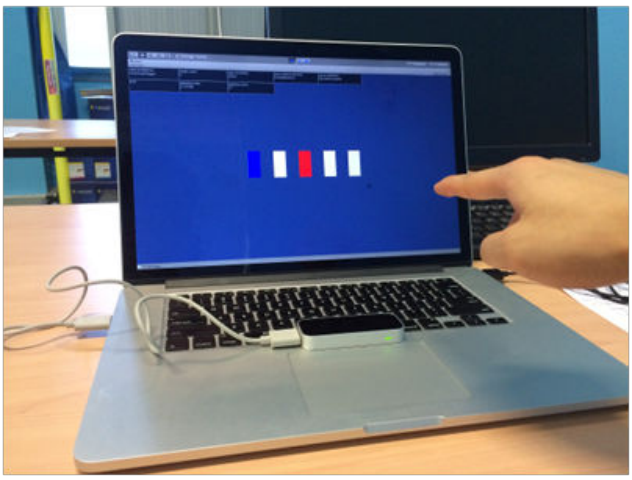


Figure 96: Experimental setup.

3.2 Participants

Eight participants (3 males and 5 females) between the ages of 25 to 28 were recruited. All of them have experience with computers, but none of them have used the Leap Motion controller before. All the participants were right handed and they were recruited from a university.

3.3 Procedure and Design

In this experiment, participants were asked to perform a selection task using the selection refinement method of HorizontalDragger. In each trial, they had to select a target from 5 rectangles displayed on the screen (Figure 96). Before starting each trial, 5 rectangles were displayed in a horizontal layout and all of them were colored in white. After performing a thumb click gesture, one of the rectangles was chosen randomly as the target and it was colored in blue. After starting the trial, the object in the leftmost position was set by default as focus and was highlighted in red. To switch the focus to another object, the user had to drag his/her finger horizontally to its corresponding dragging interval. If one object was in focus for more than 0.5 second, it was selected. Once the target was selected, the trial was completed and all the objects turned back to white. Participants had to move the hand back to the center and perform the thumb click gesture to start the next trial. They were asked to select the goal target as accurate and fast as possible.

This study was a one-factor within-subject design. To compare the performance of different dragging sensibilities, we have chosen five configurations for the dragging interval: 10, 20, 30, 40 and 50 mm. In this preliminary study, the leftmost object and the rightmost object in the layout were not connected to each other in the motor space. Therefore, the scanning could only be started by dragging the finger to the right. In addition, only Static visual feedback was provided to guide the scanning. For each configuration of the dragging interval, each participant was asked to perform 20 trials.

The configurations of the dragging interval were chosen experimentally. In an informal study, we found it very difficult to control the precision when the drag sensibility was smaller than 10 mm. On the other hand, if the value was bigger than 50 mm, the hand moved sometimes outside the field of view of the Leap Motion controller and the hand tracking was lost. Subsequently, we set 50 mm as the maximum value. Before the experiment, we explained the study procedures to the subjects and gave each of them 5 to 10 minutes to get familiar with the selection method and also with each configuration. Each participant spent about 20 minutes in total. A total of 8×5×20 = 800 selection trials were performed in this experiment, with each participant completing 100 trials.

3.4 Measurement and data analysis

For each trial, we have recorded the selection time and errors rate. The timing started after the thumb click gesture was performed and ended when the goal target was selected. When a selection error was made, the timing did not stop and continued until the target was selected. An error was recorded when one or multiple objects other than the target were selected. The errors rate was calculated by dividing the count of trials with errors by the total count of trials. In the other experiments presented in the chapter, the selection time and errors rate were measured in the same way. For data analysis, SPSS (version 22) was used.

3.5 Results and discussion

When the dragging interval was set to 10, 20, 30, 40 and 50 mm, the mean selection times were 1599 ms, 1703 ms, 1879 ms, 1996 ms and 2690 ms, respectively. The 1-way repeated measure ANOVA shows a significant effect for the dragging interval ($F_{(4,28)} = 13.693$, $p < 0.0001$). The post-hoc tests with Bonferroni correction show a significant difference in selection time between 10 mm and 50 mm and also between 20 mm and 50 mm. No other significant differences were found for other pairs of configurations. The corresponding error rates were 2.5%, 4.5%, 0.6%, 1.9% and 2.5%. No significant effect on the error rate was found.

The results show that only the mean selection time of 50 mm is greater than 2000 ms. Although it is rational to find that the configuration of 50 mm requires the longest time, the performance gap between 40 mm and 50 mm is higher than the other pairs. This can be explained by the limited sensing region of the Leap Motion controller. When the hand tracking is lost, the participant has to move the hand back inside the sensing region and stretch the index finger another time to continue selection. This can explain why only the configuration of 50 mm had a significant performance difference compared with the configuration of 10 mm and 20 mm. Although the configuration of 10 mm had the shortest selection time, all the participants commented that the configuration of 20 mm is the most appropriate one. Indeed, they thought that the dragging interval of 10 mm was too sensitive to control while the configurations of 30 mm and 40 mm were less efficient. The configuration of 20 mm achieved the best tradeoff between the sensibility and the

efficiency. Therefore, we have set the drag sensibility to 20 mm and kept it for the following experiment.

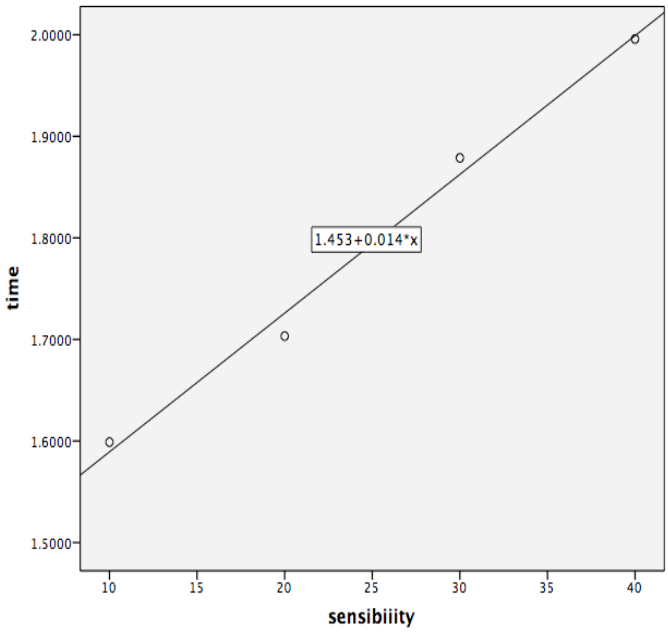


Figure 97: The linear model used to predict the selection time according to the dragging interval.

In order to describe the relation between the selection time and the dragging sensibility, we performed a linear regression analysis. Note that we have excluded the configuration of 50 mm for the linear regression because its mean selection time was affected by the Leap Motion controller. We have found that the linear model can be described by equation 5 and the linear model is shown in Figure 97.

$$T = 0.025 * dragSensibility + 1.453 \qquad \text{Equation 5}$$

The value of the dragging sensibility significantly predicted the performance time ($b=0.014$, $t=14.564$, $p < 0.005$). The overall model with the drag sensibility can predicted 99% of the performance time (adjustedR=0.991, $F=212.099$, $p=0.005$). With a tracking device of a larger sensing region, given a dragging sensibility, we can use this function to predict the selection time after the ROI is set.

4 Preliminary study 2: starting direction & visual feedback for dragging

The objective of the second preliminary study was to evaluate how the dragging starting direction and visual feedback affect the performance of HorizontalDragger. In addition, we wanted to examine whether object scanning can be made in a more predictable way with

the help of Dynamic visual feedback. Our hypothesis was that that both Double start direction and Dynamic visual feedback would help to reduce the selection time.

4.1 Apparatus

Same as those in preliminary study 1.

4.2 Participants

Eight participants (6 males and 2 females) between the ages of 24 to 27 were recruited. None of them have used the Leap Motion controller before and all the participants were right handed.

4.3 Procedure and Design

The participants were asked to perform the same selection task as in preliminary study 1. They had to select a target from a horizontal layout of 5 rectangular objects. This experiment was a 2 × 2 within-subjects design with the following factors:

(1) **DIRECTION**: Single start direction and Double start direction.

(2) **FEEDBACK**: Static visual feedback and Dynamic visual feedback.

Thus, each participant had to perform the selection task with 4 combinations of **DIRECTION** and **FEEDBACK** respectively. For each combination, each participant had to perform 20 trials. A total of 8 × 20 × 4 = 640 selection trials were performed in this experiment, with each participant completing 80 trials.

4.4 Results and discussion

The mean selection times were 2156 ms, 2048 ms, 1808 ms and 1702 ms for {Single, Static}, {Double, Static}, {Single, Dynamic} and {Double, Dynamic}, respectively. The mean selection times, grouped by **FEEDBACK**, were 2102 ms and 1755 ms for Static visual feedback and Dynamic visual feedback, respectively. The two-way repeated measure ANOVA shows a significant main effect for **FEEDBACK** ($F_{(1,7)} = 9.882$, $p = 0.016$) on the selection time. The mean selection times grouped by **DIRECTION** were 1982 ms and 1875 ms for Single start direction and Double start direction, respectively. No significant main effect for **DIRECTION** was found and no significant interaction was observed. The mean error rates were 11.9%, 18.6%, 5.7% and 5% for {Single, Static}, {Double, Static}, {Single, Dynamic} and {Double, Dynamic}, respectively. The mean error rates grouped by **DIRECTION** were 8.8% and 11.8% for Single and Double, respectively. The mean error rates grouped by **FEEDBACK** were 15.3% and 5.3% for Static and Dynamic, respectively. No significant main effect for either **DIRECTION** or **FEEDBACK** was found.

The results show that Dynamic visual feedback helped to significantly reduce the completion time. Participants also found it helpful because it was easier to control the movement speed and predict the change of focus when scanning. However, Double start direction was not very helpful regarding both the selection time and the error rate. Thus, only the hypothesis about the Dynamic visual feedback is validated. This can be explained by the fact that only 5 objects were displayed in the list. The difference of scanning in two opposite directions was not so obvious. If there were more targets in the scanning list, the benefit of Double start direction could be more significant.

The results indicate that Dynamic visual feedback improved the performance of Double start direction. Some participants also commented that in the combination of {Double, Static}, it was difficult to locate the focus visually when the color jumped between two extremities of the list. As a result, participants slowed down the movement to stabilize the focus. This may explain why this combination had the highest error rate. However, with the help of Dynamic visual feedback, the color changed progressively, so that participants could locate the focus position more easily.

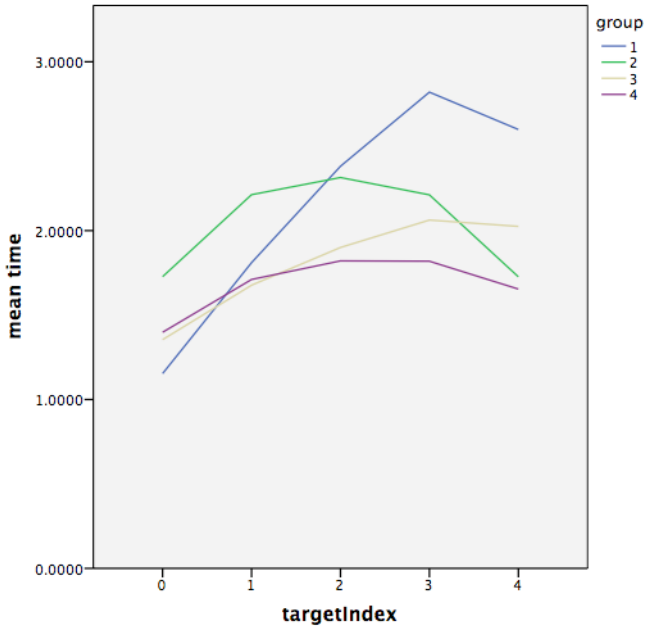


Figure 98: The mean selection time used to select targets of different orders in the layout. Values on x-axis indicate the index of the target. Group 1, 2, 3, and 4 represent the combination of {Single, Static}, {Double, Static}, {Single, Dynamic} and {Double, Dynamic}.

To examine more deeply how each combination influences the selection, we have also plotted the relationship between the mean selection time and the order of targets in the layout for each combination (Figure 98). For group 1 and 3, in which Single start direction was applied, except for the object in the rightmost, the mean selection time increases when the index of the target augments. This trend matches our expectations because more dragging time was necessary to select a target in the right than one in the left. However, it

took less time to select the object of index 4 than that of index 3. In both group 1 and 3, when the object in the rightmost is highlighted, continuing dragging the finger to the right did not result in any change. For this reason, the finger can be dragged with less corrective movements to approach the object in the rightmost. For group 2 and 4, in which Double start direction was applied, the profile of the mean selection time was more flat. The mean selection time increases from the index 0 to index 2, and then decreases from index 2 to index 4. This indicates that Double start direction helped reducing the selection time for objects located in the right.

Because the performance of the combination of Double start direction and Dynamic visual feedback was the best regarding both the selection time and error rate, we have chosen this configuration for HorizontalDragger.

5 Preliminary study 3: target count

The objective of preliminary study 3 was to evaluate the effect of the target count in the scanning list on the selection performance. In fact, having more objects in the scanning list increases the mean dragging distance, but may also cause more visual distractions.

5.1 Apparatus

Same as those in preliminary study 1.

5.2 Participants

Eight participants (5 males and 3 females) between the ages of 21 to 27 were recruited. None of them have used the Leap Motion controller before and all the participants were right handed.

5.3 Procedure and Design

We have followed the same general procedure as in preliminary study 1. The dragging sensibility was set to 20 mm, the dragging starting direction was set to Double start direction and the visual feedback was set to Dynamic visual feedback. To compare the performance of different target counts, 7 configurations were used in this within-subjects study: 4, 5, 6, 7, 8, 9 and 10. Each participant was asked to perform 20 selection trials for each configuration. We have set the maximum target count to 10 considering the limited sensing region of the Leap Motion controller. A total of $8 \times 7 \times 20 = 1120$ selection trials were performed in this experiment, with each participant completing 140 trials.

5.4 Results and discussion

When the target count was set to 4, 5, 6, 7, 8, 9 and 10, the mean selection times were 1579 ms, 1684 ms, 1765 ms, 1675 ms, 1740 ms, 1867 ms and 1931 ms, respectively. A 1-way

ANOVA shows a significant main effect of targets count on the selection time ($F_{(6, 42)}$ = 3.434, p = 0.007). The post-hoc tests with Bonferroni correction show a significant difference in selection times only between 4 and 10 and between 7 and 10. The corresponding mean error rates were 3.1%, 1.25%, 0%, 0%, 0.62%, 0.62% and 0.63%, respectively. No significant main effect on the error rates was found.

The results show that there is a positive correlation between the number of targets and the selection time. However, the increase of the selection time is placid and only significant differences were found between the target count of 4 and 10, and between the target count of 7 and 10. Similar to preliminary study 1, we have also performed a linear regression analysis to determine the linear model that best describes the relationship between the two variables. This model can be described by Equation 6:

$$T = 0.05 * targetCount + 1.390 \quad \text{Equation 6}$$

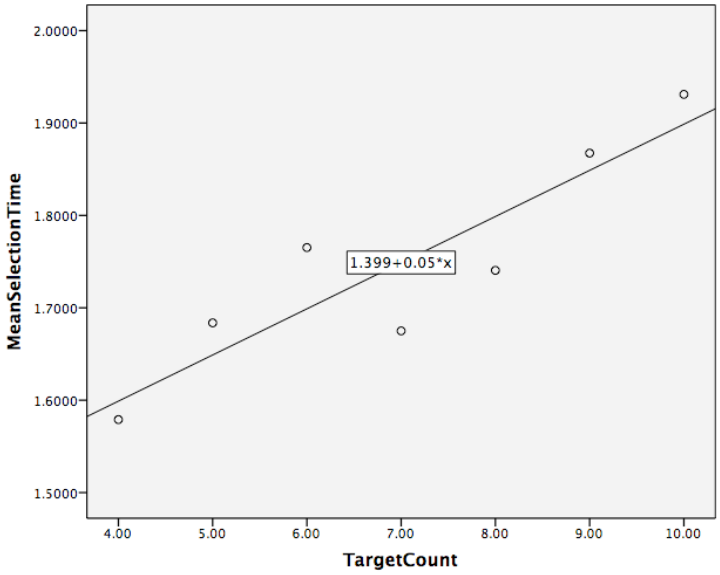


Figure 99:The linear model used to predict the selection time according to the target count in the layout.

The value of the target count significantly predicted the performance time (b=0.05, t=4.637, p < 0.0001). The overall model with the targets count can predicted 77.4 % of the performance time (adjustedR=0.774, F(1,54) =21.502, p=0.006) (Figure 99). This indicate that this model, based on the targets count, does not describe completely the selection time and that other factors may also have an effect. This may be explained by the fact that participants could start dragging the finger in both directions. As shown in Figure 100, for each profile, it took less time to select objects close to the two extremities of the layout than those located in the center. Thus, the selection is not only affected by the length of the scanning list, but also by the order of the target inside the layout.

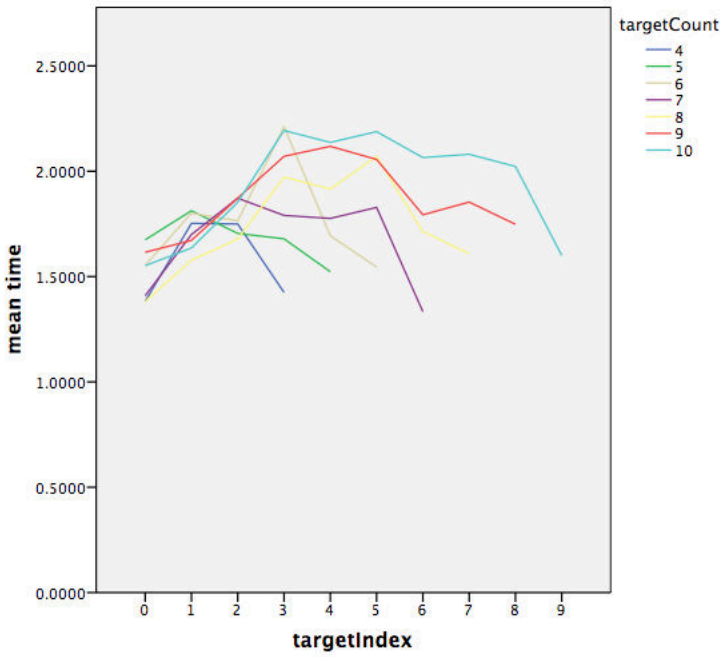


Figure 100: The mean selection time used to select targets of different orders for different configuration of target count.

6 Main experiment: performance comparison

After conducting three preliminary studies to determine some parameters of HorizontalDragger, we conducted a main experiment to evaluate the performance of HorizontalDragger. We have decided to compare it with some state-of-the-art selection techniques. In this experiment, Stroke (Ren & O'Neill, 2013) and FingerCount (Kulshreshth & LaViola Jr, 2014) were chosen for comparison. These two techniques were selected because they were two recently proposed freehand selection techniques which outperform some other state-of-the-art techniques. In the work of (Ren & O'Neill, 2013), Stroke outperformed Depth Ray proposed in (Grossman & Balakrishnan, 2006). In the work of (Kulshreshth & LaViola Jr, 2014), Finger-Count Menu outperformed Hand-n-Hand Menu and Thumbs-Up Menu for option selection. Note that Thumbs-Up Menu is similar to HorizontalDragger because it also allows dragging the hand horizontally to specify the target. Because we have proposed the configuration of Double start direction and Dynamic visual feedback to accelerate selection, we wanted to investigate whether our technique can outperform FingerCount.

Thus, we have conducted an experimental study in which we have compared these selection techniques regarding the completion time and accuracy.

Our hypotheses were:

H1: HorizontalDragger would have the shortest selection time.

H2: HorizontalDragger would have the lowest errors rate.

6.1 Apparatus

Same as those in preliminary studies 1, 2 and 3.

6.2 Participants

24 participants (13 males and 11 females) of age 20 to 27 were recruited. All of them were right-handed. They were all recruited from a university.

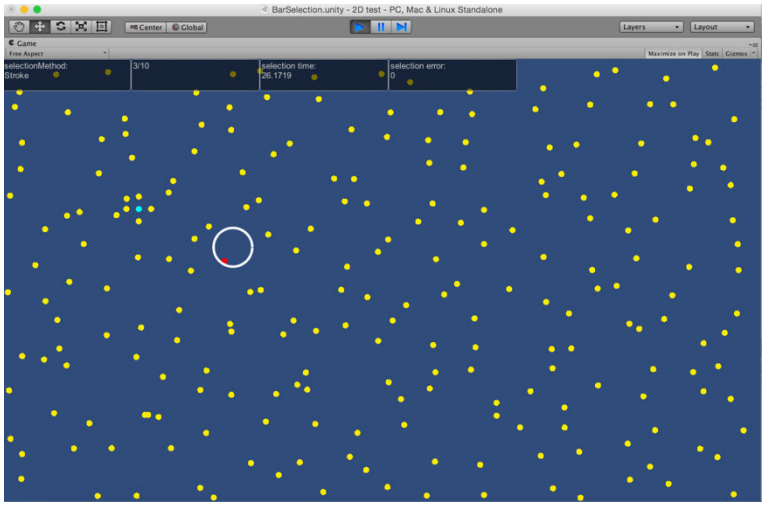


Figure 101: Target selection task of the main experiment.

6.3 Procedure and Design

In this study, participants were asked to perform a selection task. In each trial, the participants were asked to select a target displayed in a random position on the screen. To limit the degree of accessibility, four distractors of the same size were placed above, below, and to either side of the target. The distance between these distractors and the target was the same and was measured from their closest edge. Similar to the work of (Vanacken et al., 2009), we have called this distance variable the Density Spacing (DS). Besides these 5 objects, 225 other objects were placed randomly on the screen for visual distraction and no target overlapped each other. As shown in Figure 101, the target was colored in blue while all the other objects were colored in yellow.

This experiment was a 3×2×2 within-subjects design with the following factors:

(1) Three techniques **TECH** for comparison: HorizontalDragger, FingerCount and Stroke.

(2) Two target size **SIZE**: 5.6 pixels and 2.8 pixels in diameter.

(3) Two density spacing **DS**: 5.6 pixels and 2.24 pixels.

The values of target size and density spacing were chosen to ensure that the target could not simply be selected by using immediate selection techniques. This configuration permitted to evaluate the speed and the precision of the different selection techniques. It was also possible to check whether these techniques were sensitive to target size and density spacing.

Because there were 3 **TECHs** to compare, there were 6 possible orders for the techniques in the experiment. To eliminate the influence of the technique order, Tech order was counterbalanced. Each participant accomplished this experiment in a distinctive order. For each technique, the order of combinations of **SIZE** and **DS** was the same: {5.6 pixels, 5.6 pixels}, {5.6 pixels, 2.24 pixels}, {2.8 pixels, 5.6 pixels}, {2.8 pixels, 2.24 pixels}. For each combination, each participant completed 10 selection trials. A total of 24×3×2×2×10 = 2880 selection trials were performed. Before using each technique, participants were instructed and had about 5 minutes to become familiar with the technique.

In this experiment, Bubble Lens was implemented for all the techniques and selection could be made only after the ROI is magnified. When using FingerCount, only five numbers can be represented by one hand. Although both hands can be used together to perform up to ten gestures, taking use of the other hand would introduce more factors. In order to ensure that all the techniques were used in almost the same manner, we have decided for all the techniques, to display only the 5 targets which were the closest to the center of the ROI in the magnified view. Setting the number to 5 did not affect significantly the performance of HorizontalDragger because the result of preliminary study 3 show that no significant difference existed when the target count changed from 4 to 9. For Stroke, because it could be used to distinguish 8 targets, setting the target count to 5 would not slow down the performance.

After providing the magnified view of the ROI, when using HorizontalDragger, proxies of pre-selected targets were displayed in a horizontal layout beside the ROI. For FingerCount, the number corresponding to each target was displayed on it. The number was ordered according to the distance to the center of ROI. The closer the object was to the center, the smaller its corresponding number was. Similarly, Stroke displayed a vector on the target to indicate the stroke direction. The work in (Ren & O'Neill, 2013) has compared the performance of strokes in different directions, and has shown that strokes in several directions outperformed strokes in other directions. It was found that cross-body movements, such as left-down, are less comfortable. According to this work, five stroke directions were chosen: Right, DownRight, Down, DownLeft, Up. These strokes are distributed to targets according to their distances to the ROI center in ascending order.

To confirm the selection, HorizontalDragger requires the participant to highlight the desired target and hold the hand for 0.5 second. In (Kulshreshth & LaViola Jr, 2014), FingerCount requires participants to make only a count gesture to validate the selection of the desired target. However, in our setup, when the ROI was magnified, the index finger

was already stretched. The participants had to observe first the number displayed on the desired target and then figure out how to stretch fingers to make the correct gesture. During this mental process, only the index finger was stretched. If we adapted the setting in (Kulshreshth & LaViola Jr, 2014), the system would misunderstand the user's intension and trigger an incorrect selection. To solve this issue, the users were asked to perform a fist gesture after making the counting gesture. When the fist gesture was performed, the target corresponding to the last counting gesture before confirmation was selected.

6.4 Results and discussion

6.4.1 Selection time

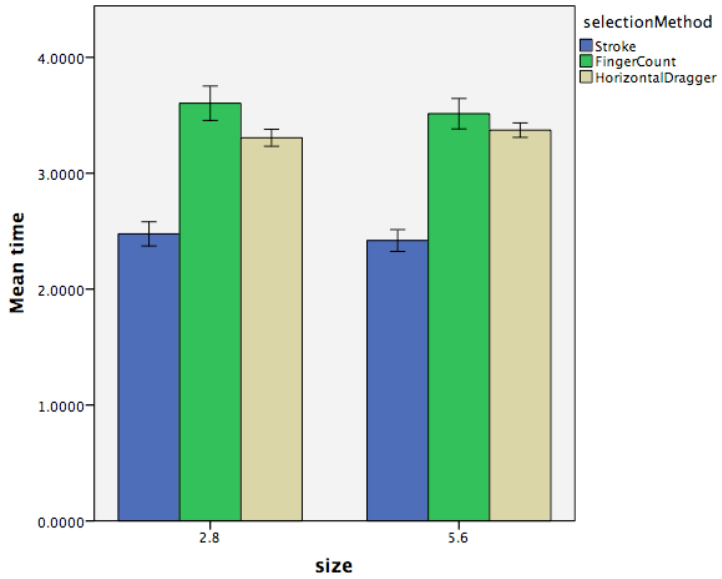
A 3-way repeated measure ANOVA shows a significant main effect for **TECH** ($F_{(3,69)}$ = 157.97, $p < 0.0001$). No significant main effect is found for **SIZE** and **DS**. The mean selection time was 3271 ms for HorizontalDragger, 3412 ms for FingerCount and 2335 ms for Stroke. One significant interaction is also observed between: **TECH** × **SIZE** ($F_{(3,69)}$ = 5.973, $p = 0.004$). Figure 102 shows the mean selection times of the different techniques grouped by **SIZE** and **DS**. Stroke had the shortest mean selection time in all the cases. So our first hypothesis is not validated. The post-hoc tests with Bonferroni correction show that Stroke was significantly faster than the two other techniques at the 0.05 level of significance. No significant difference was observed between the two other techniques. The results of the post-hoc tests are summarized in Table 10 and Table 11.

Table 10: The results of the post-hoc tests with Bonferroni correction on mean selection times among TECH by Size. HD: HorizontalDragger, FC: FingerCount, ST: Stroke.

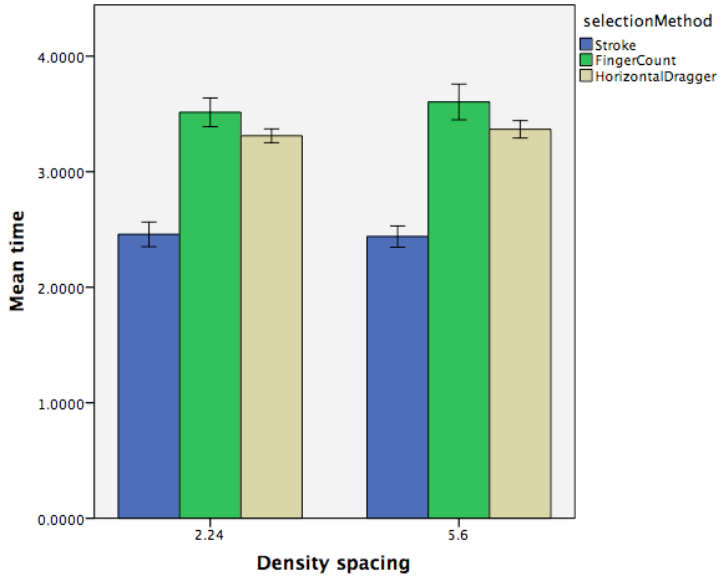
The post-hoc tests with Bonferroni correction (α=0.05)						
Size	HD (SD)	FC (SD)	ST (SD)	HD/FC	HD/ST	FC/ST
Big (5.6)	3.304s (0.215)	3.398s (0.375s)	2.307s(0.275s)	p=0.699	p<0.0001 *	p<0.0001 *
Small (2.8)	3.237s (0.200s)	3.426s (0.428s)	2.363s(0.258s)	p=0.011 *	p<0.0001 *	p<0.0001 *

Table 11: The results of the post-hoc tests with Bonferroni correction on mean selection times among TECH by DS. HD: HorizontalDragger, FC: FingerCount, ST: Stroke.

The post-hoc tests with Bonferroni correction (α=0.05)						
DS	HD (SD)	FC (SD)	ST (SD)	HD/FC	HD/ST	FC/ST
Far (5.6)	3.273s (0.207s)	3.427s (0.448s)	2.351s (0.249s)	p=0.081	p<0.0001 *	p<0.0001 *
Close (2.24)	3.268s (0.214s)	3.397s (0.351s)	2.319s (0.286s)	p=0.169	p<0.0001 *	p<0.0001 *



(a)



(b)

Figure 102: The selection times of different techniques grouped by Size (a) and Density spacing (b). The error bars are at the 95% confidential level.

6.4.2 Error rate

A 3-way repeated measures ANOVA shows that the error rate was significantly affected by TECH ($F = 18.45$, $p < 0.0001$). The mean error rates were 0.3% for HorizontalDragger, 4.3% for FingerCount and 4.5% for Stroke. Post-hoc tests show that HorizontalDragger outperformed FingerCount and Stroke regarding errors rates. No significant effect was

observed between FingerCount and Stroke. So our second hypothesis is validated.

6.5 Discussion

The results show that Stroke was significant faster than the other three techniques. Stroke was 40% and 46% faster than HorizontalDragger and FingerCount respectively. This can be explained by two reasons. First, only the stroke corresponding to the desired target should be made to perform a selection. This permitted to reduce the selection time because it was unnecessary to scan the objects like HorizontalDragger. Second, no confirmation gesture was necessary for Stroke. When using HorizontalDragger participants had to hold the hand for 0.5 second while FingerCount required performing a confirmation gesture. Using another confirmation gesture can help to reduce the performance gap by about 21%. However, according to the mean selection times of three techniques, Stroke may still be the fastest technique even with a different confirmation gesture.

Although FingerCount used distinctive gestures for selection, the observations indicate that it is more mental demanding because it requires users to determine how to stretch fingers to perform the gesture. In addition, more knuckles were involved to make a finger count gesture increasing the selection time as compared with Stroke. Moreover, we have observed that the hand model reconstructed by the Leap Motion controller was not correct all the time. For example, in one trial a participant stretched three fingers, while only two fingers of the hand model were stretched. In this case, the participant had to stretch all the five fingers to correct the hand model and repeat the desired gesture.

Although a significant difference was found among these techniques, all of them were insensitive to the target size and target density. For HorizontalDragger, the effective width and effective angle for each pre-selected target were the same and did not change based on target size and target density. Therefore, once participants got familiar with the technique, their selection performance was less influenced by these two parameters. For FingerCount and Stroke, participants only had to perform the gesture corresponding to the desired target. This selection mechanism made the performance stable when the configuration of target size and target density changed.

Regarding the errors rate, the results indicate that with HorizontalDragger users made significantly fewer errors than in the two other techniques. When using HorizontalDragger, Dynamic visual feedback helped users to know whether the desired target was in focus. However, when using Stroke, once a stroke was performed, the corresponding object would be selected. Participants could not know whether the right stroke was performed until one object was selected. The initial hand position also influenced the selection precision. Holding the hand in a bad posture increased the chances to perform a stroke in the wrong direction. To increase the precision of Stroke, a confirmation gesture can be added, as done for FingerCount. However, this will inevitably increase the selection time. Finally, the precision of FingerCount is affected by the precision of the Leap Motion controller. Most of the errors were made when the hand model was not reconstructed correctly. In the work of (Kulshreshth & LaViola Jr, 2014), the FingerCount gesture is

identified from images captured by a depth camera using simple computer vision techniques. However, when the experiment was conducted, the Leap Motion controller could only provide the information of the skeleton, we were not able to modify the algorithm to improve the gesture identification.

After the main experiment, participants were asked to select the fastest technique and the most precise technique. Six, and eighteen participants respectively, thought that HorizontalDragger, and Stroke was the fastest method for object selection. Nine, four, and 11 participants respectively, chose HorizontalDragger, FingerCount and Stroke as the most accurate technique. Participants were also asked to order their preference in a descending order. There were 8, 3, and 13 participants who chose HorizontalDragger, FingerCount and Stroke as their most preferred method.

7 Design implication

Through the evaluation study, three design directions for 3D interface design can be suggested.

(1) First, we think that it is better to use dexterous body parts such as fingers and wrists to make interaction. Participations of less agile parts such as upper arms accelerate the fatigue of users. As our preliminary study 1 has shown, users were capable of scanning the target without difficulties when the effective width was only 20 mm. Using dexterous body parts for interface design can be a feasible direction to reduce the arm fatigue.

(2) Second, to overcome the lack of physical feedback, user guides, such as visual and audio feedback should be provided. With the help of dynamic visual and audio feedback, it is possible to construct a predictable mapping between hand inputs and interface outputs. In our case, finger dragging distance is more perceivable with dynamic visual feedback.

(3) Third, extending some 2D interfaces can be a reliable solution for 3D freehand interface design. HorizontalDragger extended directly the idea of LinearDragger. In 3D space, hands can be moved forward and backward. Nevertheless, works in (Grossman and Balakrishnan 2004) commented that hand movements in the third dimension is slower and error-prone than the other two dimensions. One way to use hand movements in the third dimension is as a trigger to invoke certain commands. Functions of a mouse click or a finger tap in a 2D interface can be triggered by moving the hand forward in a 3D interface. Moreover, the space in front of the display can be divided into several parts. Moving the hand from one subspace to another can change the interface configuration. For example, moving the hand in the third dimension can be used to switch the CD ratio or adjust the magnification degree of the view.

8 Conclusion

In this chapter, we have presented HorizontalDragger, a freehand 3D selection technique which is designed to select objects at a distance. When the cluster density in the proximity of the cursor is low, our technique can be used as the Bubble Cursor to acquire the target rapidly. However, if the density is higher than a threshold value and objects around the cursor are of small size, our techniques can automatically set a region of interest and allow dragging the hand horizontally to specify the target from all the objects covered by the region. Because the effective width of each potential target is set to a constant value, the user can switch the focus in the region of interest in a predictable way. We have conducted a main experiment to compare our technique with two state-of-the-art techniques. Although our technique did not provide the shortest completion time, it generated significantly fewer errors. This demonstrates that our technique permits to improve the selection accuracy. However, another design iteration will be necessary in order to improve the completion time as compared with existing techniques.

In the next chapter, we will present our touch-based constrained manipulation techniques designed for TabletPCs. First we will present a bimanual manipulation technique, which is called Constraint Menu. This technique allows using the non-dominant hand to specify the constraint without changing the holding posture. After specifying the constraint, the dominant hand can be used to control the transformation. After that we will present Classic3D and Single3D, two single-handed manipulation techniques. Based on identified touch inputs, these techniques map the index, middle and ring fingers to the x, y and z axes. Users can press an axis finger of the dominant hand to specify the constraint, and then perform a specific gesture to trigger transformation.

Chapter 6: Manipulation technique - Constraint Menu, Classic3D & Single3D

Chapter 6: Manipulation technique - Constraint Menu, Classic3D & Single3D

1 Introduction

As mentioned in Chapter 3, in many commercial desktop 3D modeling applications, standard 3D transformation widgets are commonly used to make fine-grained manipulation of objects. Transformation widgets are well designed to take advantages of the high accuracy of mouse inputs. In addition to mouse inputs, keyboard shortcuts are commonly used to allow the users changing the manipulation modes more efficiently. Because of the prosperous development of Smartphones and TabletPCs, new requirements of performing 3D sketches and manipulating 3D models on mobile devices have emerged in the recent years. Thanks to the portability and the mobility of mobile devices, the working space can be extended to an outdoor or mobile environment. This can have some benefits. For example, designers and customers can exchange ideas and opinions about the design of products in a cafe or a restaurant. Moreover, when inspiration comes, designers can also make quick sketches before going back to the office where their computers are.

Although mobile devices permit to free users from sitting in front of their computers to accomplish simple 3D modeling work, there is a lack of efficient manipulation techniques to accomplish fine-grained manipulation on touch-based mobile devices. Through observing users interacting with standard 3D transformation widgets on touchscreens, Cohé et al. have found that the performance of standard transformation widgets on touchscreen suffers from a relatively low accuracy of touch inputs and the fingertip occlusion problem (Cohé et al., 2011). Thus, it is difficult to specify the desired DOF using a standard manipulator on touchscreens. It was also observed in this work, that users are disturbed by ergonomic issues during operations. Au et al. have also pointed out that although traditional 3D transformation widgets can be directly integrated into touch-based interfaces, their design is based on a tool-switching metaphor, which conflicts with the tool-free philosophy of the tactile paradigm (Au et al., 2012). Further, small or crowded widgets in standard interfaces are difficult to operate due to the fingertip occlusion problem.

To facilitate touch-based manipulation, 123D Design developed by Autodesk, provides a set of redefined transformation widgets (Figure 103). Compared to standard widgets, these new widgets are easier to select because they have larger interaction areas. Thus, less concentration is required to specify the desired transformation direction. However, this kind of solutions still has several issues. First, object manipulation can be affected by the camera viewpoint. In some viewpoints, the projections of two transformation widgets

coincide with each other. This increases the difficulty to specify the correct one. Second, because there is a lack of keyboard shortcuts, a toolbox is displayed on the left side of the screen. To switch the transformation mode, users have to click the buttons in the toolbox frequently to trigger the translation, rotation or scaling. As a result, both the efficiency and consistency of the design work are degraded.

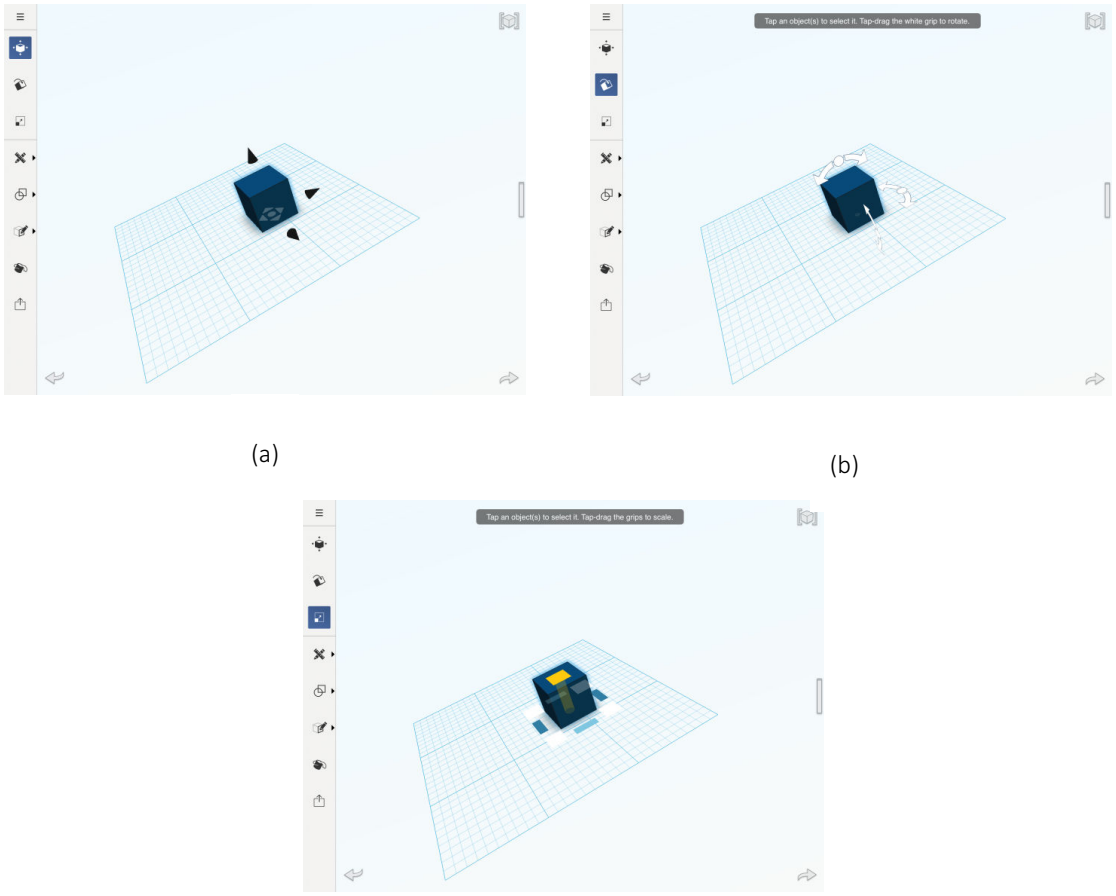


Figure 103: Manipulation widgets of 123D Design: (a) Translation widgets; (b) Rotation widgets; (c) Scaling widgets.

Manipulation of 3D objects consists of three main tasks: rotation, scaling and translation (RST). To allow users building accurately 3D objects and scenes, it is usually desired to perform RST manipulations separately with an axis- or plane-constraint. Previous studies such as Eden (Kin et al., 2011), tBox (Cohé et al., 2011) and Toucheo (Hachet et al., 2011) provide different solutions to perform RST in all three dimensions. One limitation of these techniques is that manipulation gestures need to be be performed upon the target object or a proxy widget. Performance of these techniques may be affected by the object cluster

density. In addition, some interaction gestures proposed by these techniques are difficult to perform on TabletPCs because they require using both hands. As the non-dominant hand (NDH) is often used to hold the device, its motor space is very limited.

In this chapter, we first propose a new manipulation technique which is called the Constraint Menu technique (Wu et al., 2015c)[3] for touch-based paradigm based on the bimanual asymmetrical model proposed by Guiard (Guiard, 1987). This technique allows users to specify the manipulation reference using the NDH while the dominant hand (DH) is used to determine the manipulation mode and control the transformation. One advantage of this technique is that axis- and plane-constraints can be specified simply by using the NDH without changing the holding posture. Moreover, object manipulation can be made without touching the target. Therefore, the performance is less affected by the object position, orientation and scale. Besides proposing a bimanual manipulation technique for TabletPCs, we have also designed a GUI which provides a group of functions to edit the shape of 3D objects. For example, users can transform or extrude a face to modify the shape of an object. For these functions which are commonly implemented in desktop sketching applications, we have redefined their interaction methods to adapt to the touch-based paradigm. To examine the usability of Constraint Menu, we have conducted a user study to compare it with two other state-of-the-art techniques. The results of 3D objects docking tasks have shown that our technique has better performance regarding both efficiency and fluency.

Besides Constraint Menu, we have also designed two other manipulation techniques which are named Classic3D and Single3D. These two techniques are designed based on the usage of identified touch inputs. In the literature, some solutions have been proposed to identify the sources of touch inputs (Sugiura & Koseki, 1998; Colley & Häkkilä, 2014; Marquardt et al., 2011). Some work have been conducted to explore how identified touch inputs can be used to enrich interaction possibilities (Goguey et al., 2014a; Goguey et al., 2014b). Inspired by these works, we were interested in whether object manipulation can benefit from the usage of identified touch inputs. On the one hand, because each finger can be mapped to a different function, we can have a large design space to propose a set of manipulation gestures using only the DH. On the other hand, learning these gestures could be more difficult and time consuming. To examine the learnability of Classic3D and Single3D, we have conducted an experiment to study whether they can be used without difficulties after a short period of training.

In the following part, we first present the design of the Constraint Menu technique and how the design interface can be used to make quick 3D sketches. Then, we present the user study of the docking task and discuss its results. After that, we introduce the design of Classic3D and Single3D. Finally, we present the learnability study and discuss its results.

[3] A video demo can be found in this link https://www.youtube.com/watch?v=WVQU7h7XAXM

2 Constraint Menu & TouchSketch

2.1 Design rationale

Our main objective is to redefine 3D sketching interfaces on TabletPCs for designers familiar with traditional desktop software. Our objective is not to replace traditional desktop sketching software. Instead, we aim to design a new interface that can become complementary with existing software when designers are not in front of their desktop computers. This can permit to free designers from the constraints of the traditional work space and allow them to explore their creativity anywhere when they want.

Following an iterative design process, we have first conducted a task and needs analysis through discussion with 3 expert users of 3D modeling software. Through this discussion, we aimed to determine what design flows do they usually follow when constructing 3D models. We have also tried to understand how they manipulate 3D models and camera perspective. Finally, we have presented to them some state-of-the-art touch-based manipulation techniques. After that we have recorded their evaluation about the advantages and inconvenience of existing techniques. This analysis permitted to identify several requirements that should be met by a new touch-based interface:

(1) Users should be able to manipulate an object without keeping fingers pressed upon it. In fact, it is suited to avoid misoperations caused by the occlusion issues, more particularly when multiple objects are close to each other.

(2) 3D objects should be controlled in all 3 dimensions independently. It is also necessary to permit switching between the world coordinate system and the object local system. Therefore, users can specify the appropriate coordinate system according to manipulation tasks.

(3) Due to the lack of keyboard shortcuts, a substitution mechanism should be provided to switch between transformation modes seamlessly. This can help improving the fluency of the 3D objects design process.

(4) Because the camera is frequently manipulated during design and sketching tasks, the proposed technique should work properly under any camera viewpoint.

Among all touch-based manipulation techniques discussed in Chapter 3, the work of (Au et al., 2012) satisfies the first three requirements. However, the performance of this technique is sometimes degraded by the camera viewpoint. For this reason, we propose a new technique that aims to meet all four requirements.

When manipulating 3D objects in desktop modeling applications, intermediates and experts use the DH to control the transformation widget while the NDH is used to switch operation modes by pressing keyboard shortcuts. The coordination of the two hands can be described by the bimanual asymmetrical model proposed by Guiard (Guiard, 1987).

Guiard regards both hands as abstract motors, defining a motor as any natural or artificial

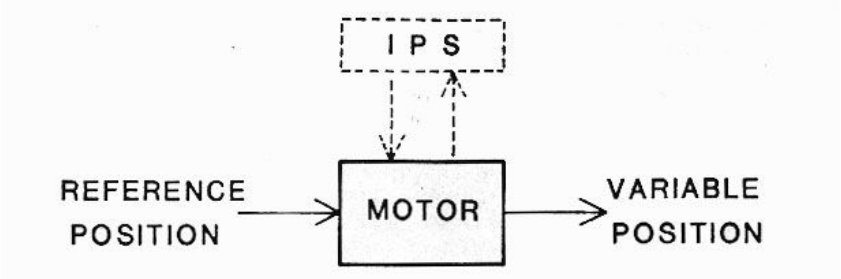


Figure 104: Minimal representation of a motor simply defined as a natural or artificial device serving the function of producing motion. More specifically, a motor acts on the difference between a variable position, which it controls, and a reference position, which it receives as an input. This macroscopic representation ignores the internal complexity of the motor considered as a black box. The suggested approach favors analysis at the level of mechanics, on the grounds that control of the motor by an information processing system (IPS) must conform to pre-existent mechanical constraints (Guiard, 1987).

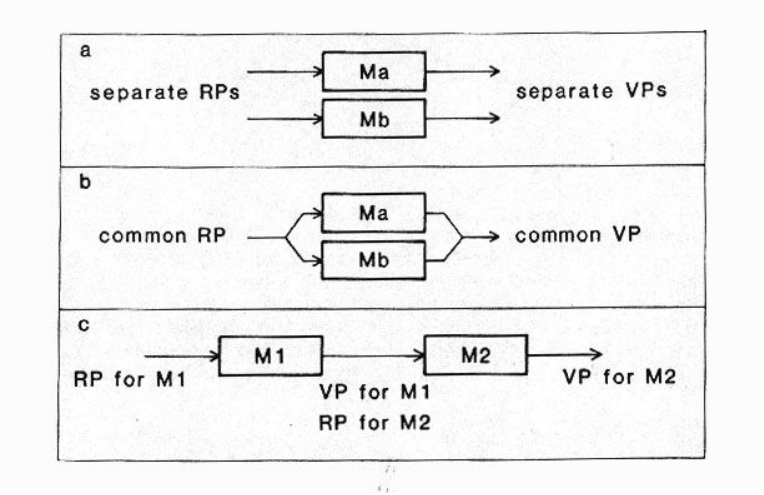


Figure 105: Three ways of assembling two motors to form a cooperative structure: (a) orthogonal assembly, (b) parallel assembly, and (c) serial assembly. Ma, Mb, M1, M2 = Motors a, b, 1, 2. RP = reference position. VP = variable position (Guiard, 1987).

device serving the function of creating motion. The word 'abstract' refers to the fact that the internal mechanism of the motor is not considered and it is represented as a black box as shown in Figure 104. Because both hands have been likened to two abstract motors, they can be assembled to form three kinds of cooperative structures: orthogonally, in parallel and in series (Figure 105). Guiard argued that the property of asymmetrical division of labor of the serial assembly of two motors makes it potentially suitable for modeling the way the two hands typically cooperate with one another. When two motors are assembled in series, they form a kinematic chain. The kinematic chain model reveals the fact that the variable position of the NDH is used to issue coarse movements or set the spatial frame of reference for the DH. After the variable position of the NDH, which is also the reference position of the DH, is produced, the DH is used to perform fine movements to yield the

final output of bimanual cooperation. Based on this model, some touch-based techniques, such as BiTouch and BiPad (Wagner et al., 2012), have been proposed and are proved to be intuitive and efficient to use. Subsequently, we have decided to explore how this model can be adapted for use on mobile devices to facilitate object manipulations. We thought that this model can help reducing the burden of the DH and leverage more effectively the NDH.

However, because the NDH is usually used to hold the device and only the thumb can be used to make inputs, bimanual interaction techniques can be difficult to use on mobile devices. To increase the expressiveness of the thumb, some previous solutions such as MicroRolls (Roudaut et al., 2009) and ThumbRock (Bonnet et al., 2013) have proposed to map different thumb movements on the screen to various operations. Boring et al. have used the size of the contact area to switch functions (Boring et al., 2012). Wagner et al. have designed bimanual interaction on tablets in another way (Wagner et al., 2012). They have first made a user study to identify all the common postures that are used to hold tablets. Then, based on different holding postures, they have designed a set of interactive widgets which are located inside the motor space of fingers of the NDH (Figure 106). These interactive widgets permit leveraging the NDH to augment the interaction without switching the holding posture. The usage of the NDH enriches the interaction possibilities and can make the interaction more fluent. Some interaction tasks which should be performed previously using the DH, can now be achieved using the NDH. Moreover, holding fingers of the NDH on interactive widgets can modify the function of the DH. For example, if a finger of the NDH holds on a zooming button, pressing the DH on the screen can scale up the text in the touched area. Their controlled experiment has demonstrated that these interactive widgets could help improving interactions. Our manipulation techniques are inspired by the bimanual asymmetrical model proposed by Guiard (Guiard, 1987) and the concept of bimanual interaction on mobile device proposed by Wagner et al. (Wagner et al., 2012). We have decomposed the manipulation task into two subtasks which are in a serial assembly and assigned each of them to a different hand. Because the NDH is commonly used to set the spatial frame of reference for the DH, we have decided to use it to specify manipulation constraints. After a constraint is specified, the DH can then be used to trigger the operation and control the transformation through only one gesture. We will first present how a constraint can be specified by the NDH, and then explain how RST manipulation can be performed using the DH.

Figure 106: Five support-hand interaction zones (Wagner et al., 2012).

2.2 Constraint Menu

When performing fine-grained manipulations, 3D objects are always controlled with a constraint. Depending on the type of the constraint, manipulation can be divided into four categories:

(1) Free manipulation.

(2) Axis-constrained manipulation.

(3) Plane-constrained manipulation.

(4) Uniform manipulation.

Free manipulation refers to manipulation without restriction. It means that object manipulation in all three dimensions can be performed simultaneously. This is the way that people manipulation physical objects in the real world. However, because touch inputs are of two dimensions, it is difficult to map them directly to 3D movements of virtual objects. Thus, we did not intend to provide the possibilities to perform free manipulation. Axis-constrained manipulation refers to manipulation which is restricted by a selected principal axis. Objects can only be translated and scaled along the selected axis and rotated around it. Object manipulation in the other two directions is deactivated. Plane-constrained manipulation refers to manipulation which is restricted by a selected plane determined by two principal axes. Objects can be translated freely in the selected plane. However, it is not permitted to move the object in the direction perpendicular to the plane. Object rotation can be made around the normal vector of the plane. In fact, plane-constrained rotation equals to axis-constrained rotation if the normal vector of the plane equals to the selected axis. Plane-constrained manipulation also includes uniform scaling parallel to the plane but not in the normal direction. The scaling ranges in two orthogonal directions parallel to the plane are the same. Unlike axis-constrained and plane-constrained manipulation, uniform manipulation only allows uniform scaling in all three dimensions. Object translation and rotation are deactivated when the uniform constrained is specified.

In our kinematic chain model, to manipulate an object, users should first use the NDH to set a constraint (Figure 107). To facilitate the selection of constraints through the use of the NDH, we have designed the Constraint Menu. As shown in Figure 6, the Constraint Menu is a marking menu with seven options which allows users to select one option among three axis-constraints, three plane-constraints and the uniform mode. Three fan-shaped options of larger sizes which are labeled 'X', 'Y' and 'Z' respectively, are used to set axis-constraints, while the other three smaller fan-shaped options labeled "X-Y", "Y-Z" and "Z-X" respectively, are used to set plane-constraints. Finally, the circular button in the center is used to trigger the uniform mode. The x-, y- and z-axis are three principal axes of the active coordinate system. For right handed users, the Constraint Menu is displayed in the left down corner of the display (Figure 109) to make it reachable by the thumb of the NDH. The interface configuration can be inversed for left handed users.

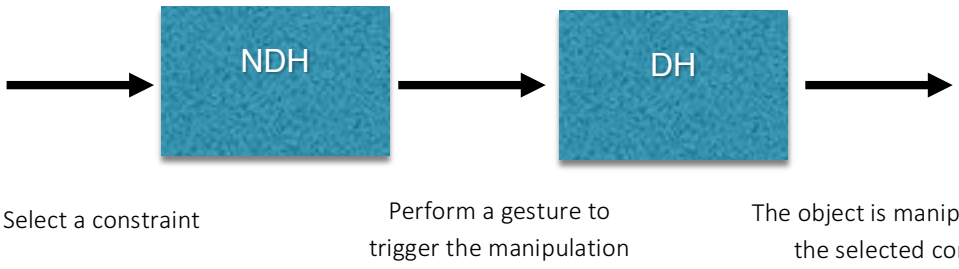


Figure 107: Kinematic chain model of Constraint Menu.

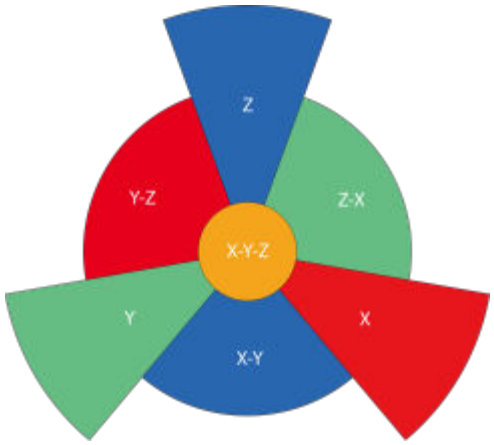


Figure 108: The Constraint Menu which is used to specify manipulation constraints.

For the first version of the Constraint Menu, the circular menu is divided into six equal parts and sectors of three axis-constraints and of three plane-constraints are of the same size. However, in an informal study, we have found that users cannot find out rapidly the desired constraint. They had to check the label of each option to find out the desired one. Thus, we have decided to modify the shape of the menu to make axis-constraint and plane-constraint easier to distinguish. Because axis-constrained manipulation is performed more frequently than plane-constrained manipulation, we have decided to display sectors of axis-constraints with longer radius than those of plane-constraints to highlight them (Figure 108). The shape difference between two kinds of constraint sectors makes them more distinguishable. Note that the corresponding angle of axis-constraint is smaller than that of plane-constraint. Indeed, the larger angle serves as a visual hint and suggests that sectors with larger angle correspond to plane-constraints. Because X-axis, Y-axis and Z-axis are perpendicular to YZ-plane, ZX-plane and XY-plane respectively, each pair of axis and plane is displayed in the diagonal configuration and they also share the same color. It is important to note that objects manipulation can only be performed when the thumb is pressing the corresponding constraint on the Constraint Menu.

2.2.1 Translation, Rotation & Scaling

Once an axis-constraint is specified, the selected object can be translated, rotated or scaled according to the selected axis. As shown in Figure 109 (b), when the X-axis constraint is selected, only the red axis is displayed. The appearance change informs the user that an axis-constraint is specified and the DH can be used to manipulate the object. According to the kinematic chain model, this means that the reference position of the DH is ready. Object translation can be performed by dragging one finger of the DH along the selected axis (Figure 109 (c)). Rotating the object also requires dragging one finger, but in the perpendicular direction of the selected axis (Figure 109 (d)). Both the amount of translation and the amount of rotation are calculated according to the average distance between the current contact point and the initial contact location. Our translation and rotation techniques are similar to those of (Au et al., 2012), but can be performed by only one finger. During a pilot study, we have found that it is difficult to rotate the object when the axis is almost perpendicular to the screen. In this case, the projection of the axis on the screen is too short to determine in which direction the finger should be panned to trigger rotation. To overcome this issue, we have designed another rotation method based on the rotation algorithm applied by tBox (Cohé et al., 2011). When the constraint axis is perpendicular to the screen, two interactive rotation arrows are displayed (Figure 109 (e)). Users can tap one arrow and rotate it around the object to control the rotation. When the drag direction can be easily determined, only one arrow is displayed to provide visual hint but it is not interactive. Unlike object translation and rotation, object scaling can be performed by using a pinch gesture. Moving two fingers apart (or towards each other) can scale up (or down) the object (Figure 109 (f)). The pinch gesture can be performed in any direction. In other words, it is not necessary to perform the pinch gesture in the direction of the axis.

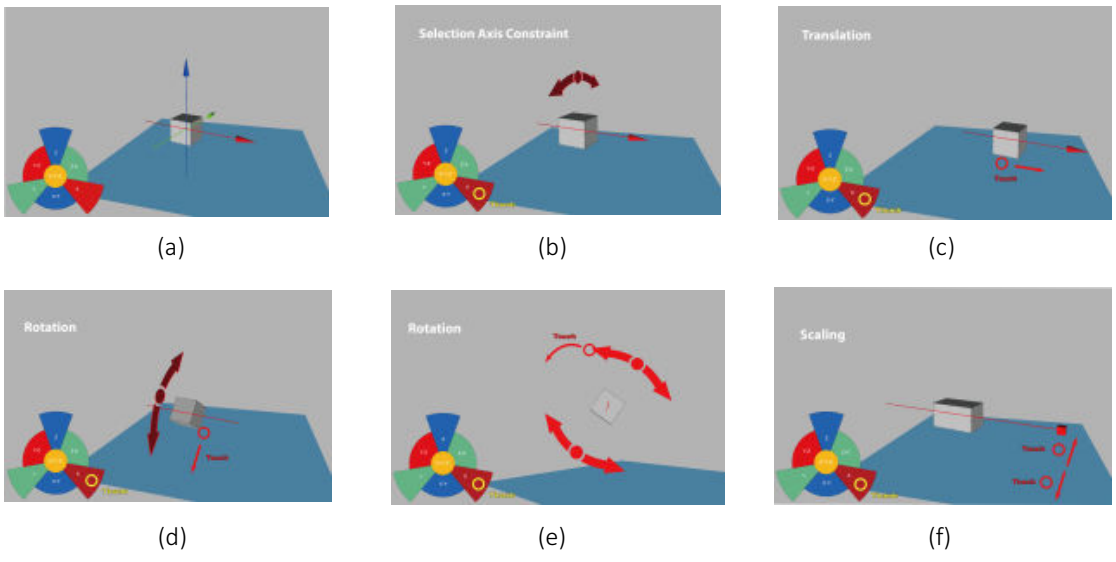
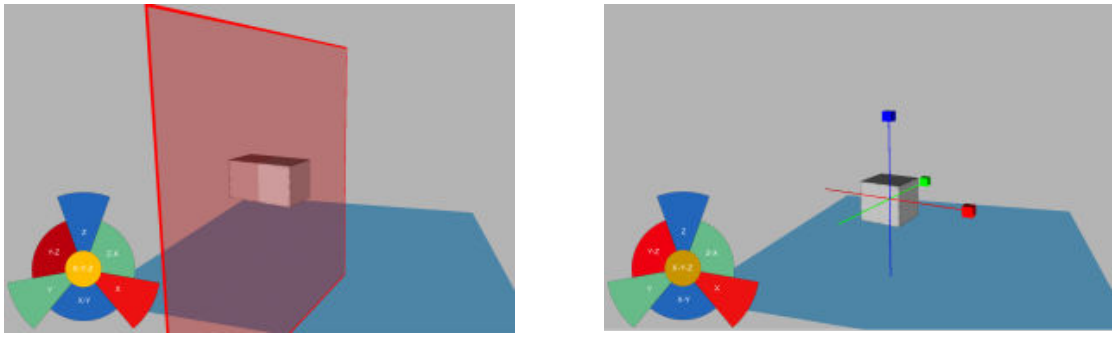


Figure 109: Gestures for axis-based transformation manipulations. (a) The initial state of an object before manipulation; (b) After specifying the x-axis constraint, only the red axis is displayed; (c-f) After an axis-constraint is selected, users can translate, rotate and scale the object by using the DH.

When a plane-constraint is specified, users can manipulate the object in a similar way. However, only translation and scaling are supported. As shown in Figure 110 (a), a semi-transparent plane is displayed to indicate the selected plane-constraint. Users can drag freely one finger on the screen to translate the selected object along the plane or perform the pinch gesture to perform uniform scaling in the plane (no scaling along plane normal). Because object rotation around the normal axis can be done when the constraint of this axis is specified, our interface deactivates the object rotation when a plane-constraint is selected.

When the circular button is pressed to set the uniform manipulation, performing the pinch gesture can launch uniform scaling (Figure 110 (b)). Because in design and sketching tasks it is rare to translate objects freely in all three directions simultaneously, our technique does not support object translation when the uniform mode is activated.



(a) (b)

Figure 110: (a) When a plane-constraint is selected, the corresponding plane is displayed. Users can translate and rotate the object along the plane; (b) When the uniform mode is activated, the object can be scaled uniformly.

In general, our technique supports the following features:

(1) **Position, orientation and scale independence:** Except object rotation around an axis perpendicular to the screen, manipulation tasks are not affected by the object position, orientation and scale. Manipulation gestures can be performed anywhere on the screen using the DH.

(2) **Coordination of both hands:** The NDH and the DH can be leveraged together to accomplish object manipulation. The NDH can be used to specify the manipulation constraint without changing the holding posture.

(3) **Low memory load:** Classic interaction gestures such as panning and pinching are reused for manipulation. This facilitates the memorization of gestures and thus the learnability of the technique.

(4) **Unaffected by the camera viewpoint:** Our technique allows users to manipulate objects under any camera perspective. Because manipulation constraints can simply

be selected from the Constraint Menu, object manipulation is less affected by the camera position and orientation.

2.3 TouchSketch

Few studies have been conducted to design new interfaces for 3D modeling on touchscreens. Mockup Builder demonstrates how 3D sketching can be done on and above the surface (De Araújo et al., 2013). Users can first draw a 2D primitive on the screen and then extrude it to the third dimension using gestures in air. Sun et al. have proposed a touch-based interface for fast architectural sketching (Sun et al., 2013). A set of natural and intuitive gestures are designed for drawing the building outline. Similarly, Paper3D presents a powerful touch-based interface for paper folding work (Paczkowski et al., 2014). It is possible to manipulate the virtual paper in a very familiar way to construct complex 3D models. Chellali et al. have developed VR4D to support collaborative design work (Chellali et al., 2013). A user can use a sketch-based application to sketch 3D objects, and then another user can use a 3D immersive application to manipulate objects in an immersive environment.

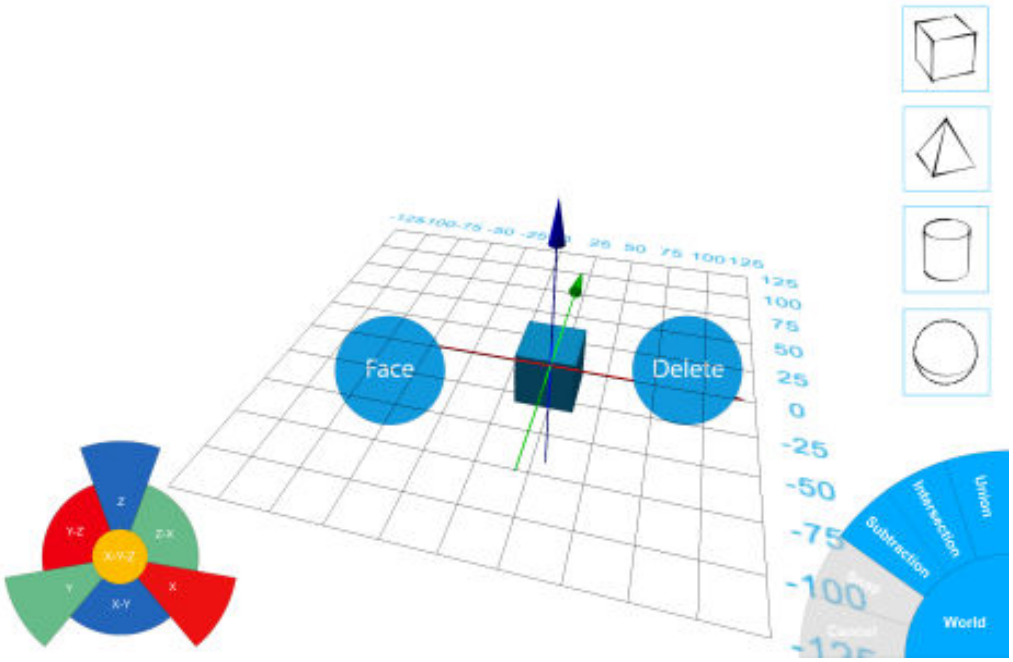


Figure 111: The interface of TouchSketch.

Besides proposing a set of manipulation techniques, we have also developed a user interface which is called TouchSketch to enable objects shape editing on tablets. Through discussion with expert users of desktop sketching software, we have identified a set of functions which are commonly used during design and sketching activities. To adapt our

interface to the touch-based paradigm, we have redefined the interaction methods of these functions to improve the user experience on touch-based mobile devices.

2.3.1 Primitive generation

As shown in Figure 111, to start sketching, users can first press one button in the right side to select a primitive to be generated. Then, they can tap the screen to generate the selected primitive in the touched location. A grid is displayed in the scene as a spatial reference and the bottom of each new generated primitive is attached to it.

2.3.2 Object & Face Selection

An unselected object can be selected by tapping it. Tapping it a second time cancels the selection. It is also possible to select multiple objects. To deselect all the selected objects, the user needs to tap the finger in the empty space. As shown in Figure 111, when the finger is pressed on the object without being released, two circular buttons are displayed. The finger can be dragged to the left button to select the face of the touched object when the finger is dropped. Dragging the finger to the right button deletes the touched object. After selecting an object or a face, the Constraint Menu is displayed.

2.3.3 Object duplication

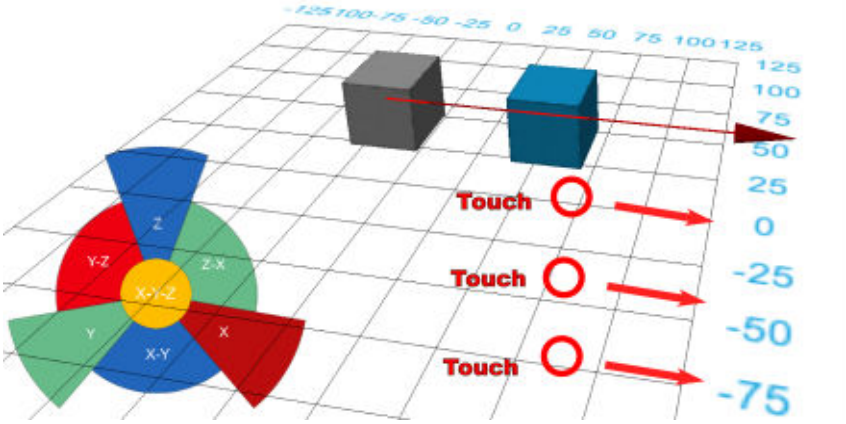


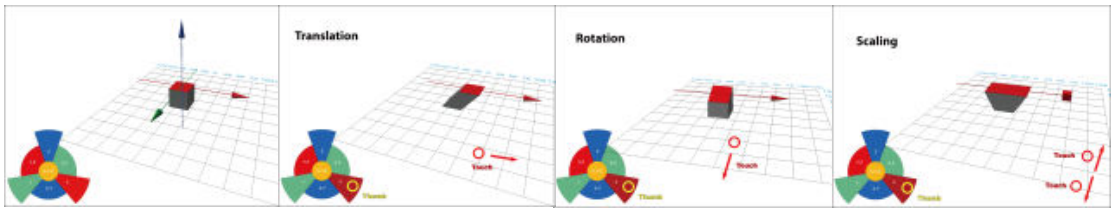
Figure 112: After an axis-constraint is specified, dragging three fingers on the screen and copy the selected object and translate its copy along the specified axis.

After discussing with expert users, we have determined that duplicates of objects are commonly generated and then translated along one principal axis. To meet this requirement, we have designed a copy gesture that performs object duplication and translation of the copy at the same time. To copy a selected object, users specify first one axis-constraint. After that, they drag three fingers of the DH along the selected axis (Figure 112). Once the

dragging distance passes a threshold value, the object is copied and its duplicate can be translated along the axis. If a plane-constraint is selected, the copy of the object can be translated along the selected plane.

2.3.4 Face manipulation

After a face is selected, it can be manipulated similar to an object. As shown in Figure 113, object manipulation gestures can be reproduced to translate, rotate and scale the selected face. It is important to note that face manipulation is made in the local coordinate system of the selected face. In other words, the x-axis and the y-axis of the system are respectively parallel to the tangent and bi-tangent of the selected face. The z-axis is parallel to the normal of the face. The face center is set as the origin of the coordinate system. Because the height of a planar face is set to 0, performing the scaling gesture on the z-axis has no effect on the object shape. Reusing object manipulation techniques to control selected faces is expected to reduce the learning cost and allow users to retain less information.

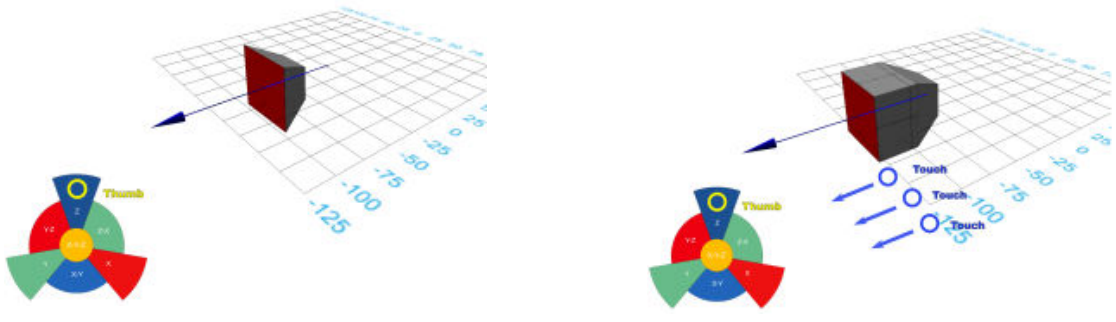


(a) (b) (c) (d)

Figure 113: (a) A face is selected; (b-d) The selected face can be translated, rotated and scaled in the same way as an object.

2.3.5 Face extrusion

Face extrusion is a common function in commercial sketching applications that permits to extrude one face from its initial position to modify the object shape. Because face extrusion is a very practical function when modifying the shape of objects, we propose a method to perform this operation in a fluent way. Face extrusion can be performed in two ways. First, after a face is selected, users can directly tap three fingers of the DH to extrude the face without specifying a constraint. In this case, the original face and the extruded face overlap each other. This operation is equivalent to duplicating the face in its initial position. In addition, users can also extrude the face along an axis-constraint. After an axis-constraint is specified, and similar to object duplication, dragging three fingers can extrude the face along the selected axis (Figure 114). The extrusion range can be controlled by the dragging distance of fingers. Through combing the face duplication and control of extrusion range, the process of face extrusion is simplified.



(a) (b)

Figure 114: (a) Object before extrusion; (b) The shape of the object is modified after one face is extruded.

2.3.6 Camera navigation

The scene of the sketching environment is displayed using a perspective camera. When no manipulation constraint is selected, the camera can be manipulated. Dragging one finger on the screen can orbit the camera on a virtual trackball. Performing the drag and pinch gesture with two fingers can pan and zoom the camera, respectively.

2.3.7 Functions marking menu

When an object or a face is selected, a marking menu is also displayed in the right down corner of the interface (Figure 111). Clicking the center of the menu permits to switch between the world coordinate system and the object local system. The selected object can be manipulated with respect to the active coordinate system. Constrained translation, rotation and uniform scaling can be performed in both the world and local coordinate systems, while axis- and plane-constrained scaling can only be performed in the local coordinate system.

Besides switching the manipulation reference, the marking menu provides additional functions. For instance, when an object is selected, users can choose a boolean operation from the menu and select a target object to calculate the intersection, union or subtraction between them. If the subtraction operation is selected, the intersection of two objects is subtracted from the first selected object. If a face is selected, users can choose the snapping operation and select another face of the target object. Then, the first object is transformed to stick its selected face to the touched face of the target object.

2.4 Main experiment one: evaluation of Constraint Menu

To evaluate the performance of our manipulation technique, we have conducted a

controlled experiment. The objective was to determine whether the Constraint Menu technique outperforms other state-of-the-art techniques regarding its efficiency. For comparison, we have chosen the manipulation technique of 123D Design and the technique proposed by (Au et al., 2012). Different from standard transformation widgets, the manipulation widgets of 123D Design have been redefined for utilization on mobile devices. However, its performance has never been evaluated. The technique of 123D Design is reproduced as shown in Figure 115. Users can press the translation, rotation and scaling buttons in the left side to activate the corresponding transformation widgets. The technique proposed in (Au et al., 2012) allows users to achieve manipulation on 9 DOFs using only the DH. Moreover, it outperforms the standard transformation widgets. Therefore, we have decided to compare the performance of these two techniques to ours. In the following, we use "Widget-Based technique" to refer to the technique of 123D Design and "Duo-Finger Constraint technique" to refer to the technique of (Au et al., 2012).

Although some other techniques, such as Eden (Kin et al., 2011), tBox (Cohé et al., 2011) and LTouchIt (Mendes, 2011), can be used to make fine-grained manipulation, we did not select them for comparison because some of their functions require bimanual gestures. For instance, to scale up an object using tBox, users have to drag two opposite edges on the same face. When the display size of tBox is large, it is uncomfortable to reach two opposite edges using fingers of one hand.

2.4.1 Hypotheses

We have made two hypotheses before conducting the controlled user study:

H1: The Constraint Menu technique is more efficient than the two other techniques, because the coordination of two hands simplifies specifying the manipulation constraint, and reduces the burden of the DH.

H2: The Constraint Menu technique requires less camera navigation than the other two techniques because its performance is less affected by the viewpoint of the camera.

2.4.2 Apparatus

The experiment was conducted on an Apple iPad Air 2, with a 9.7" touchscreen. The screen has a resolution of 2048×1536 pixels. Content was displayed in Portrait mode. The experimental environment was developed using Unity 3D with C#.

2.4.3 Participants

Twelve unpaid subjects (10 males and 2 females, 20 to 27 years old) participated in this study. They were all students from a university. All of them use smartphones frequently in everyday life but only 4 of them use frequently 3D modeling software. Only one

participant is left handed. For the left handed participant, the Constraint Menu and buttons of the Widget-based technique were positioned on the right-hand side of the screen. For other participants, these interactive UI widgets were displayed on the left-hand side.

2.4.4 Procedure and Design

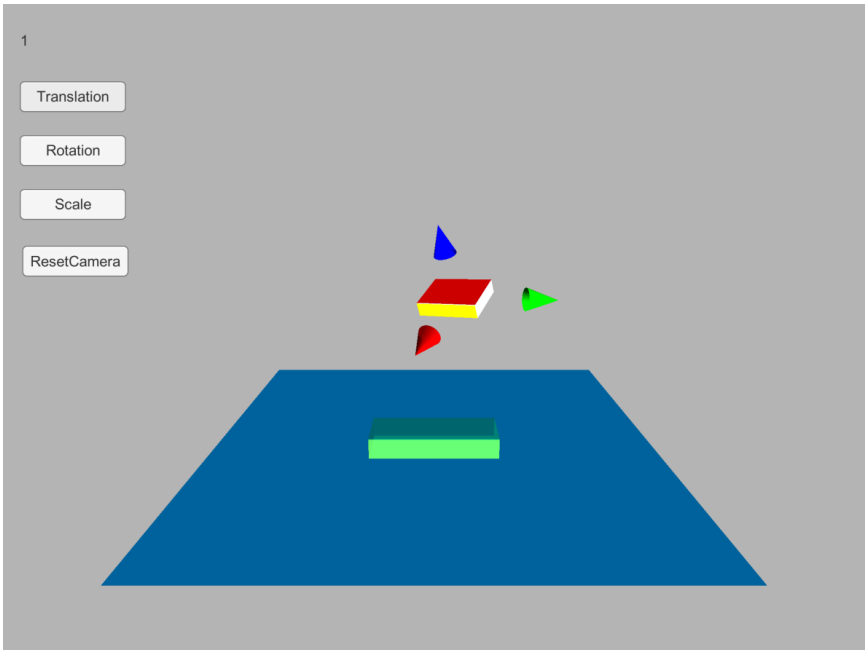


Figure 115: The environment of evaluation.

Similar to (Martinet et al., 2010b), the participants were asked to perform a docking task using the three manipulation techniques. As shown in Figure 115, at the beginning of each trial, a cuboid was displayed in a random position in the scene and another semi-transparent cuboid that serves as a target was fixed on the surface. For each trial, participants had to move the cuboid towards the target. A trial was considered to be completed when the cuboid was moved close enough to the target position, orientation and scale. Indeed, once the difference in position, orientation and scale were under a pre-defined threshold value, the trial was finished and a start button was displayed for launching the next trial. To avoid orientation ambiguity, one face of the cuboid was colored in red and another face was colored in yellow. The red face should coincide with the top of the target and the yellow face should coincide with the front side of the target. The initial position of the cuboid was generated randomly on a semi-sphere whose center was the target. The radius of the sphere was set to a fixed value to ensure that the cuboid was inside the initial viewpoint of the camera at the beginning of the task. The initial difference in orientation between the cuboid and the target was 120º around a random vector. The initial scale difference was generated using the equations Equation 7, Equation 8 and Equation 9. (Sx_c, Sy_c, Sz_c) is the scale of the cuboid in the local coordinate system while (Sx_t, Sy_t, Sz_t) is the scale of the target. (Sx_o, Sy_o, Sz_o) is the scale difference between the cuboid and the target and (Sx_v, Sy_v, Sz_v) is the random scale direction. The scale range is set to 0.5. The

target was generated at a fixed position for each trial. During the experiment, the camera could be manipulated freely to change the viewpoint. However, the starting point of view was the same at the beginning of each trial. Moreover, we have added a button to reset the camera viewpoint to avoid losing the cuboid and the target when the camera is translated or rotated too much. When this button was pressed during one trial, this trial was considered to be invalid and the participant had to start it over.

In this within-subjects design experiment, the only independent variable was the manipulation technique (TECH) with three levels; Widget-Based, Duo-Finger Constraint and Constraint Menu. To eliminate any effect of the trial order, TECH order was counterbalanced. Because 3 techniques were compared, there were 6 possible experimental orders. Twelve participants were divided into 6 groups and the permutation order was the same in each group. For each TECH, participants were asked to perform 10 manipulation trials. A total of 12×3×10=360 manipulation trials were performed in this experiment. Before the experimental sessions for each technique, instructions were given to participants on how to manipulate objects and they had 5 to 15 minutes to get familiar with the technique. To reduce the learning effect, participants were asked to perform at least 5 and at most 10 docking trials for each technique before the formal experiment. After the experiment, participants were asked to fill in a questionnaire to evaluate each technique.

$$(Sx_c, Sy_c, Sz_c) = (Sx_t * Sx_v, Sy_t * Sy_v, Sz_t * Sz_v) \quad \text{Equation 7}$$

$$(Sx_v, Sy_v, Sz_v) = (1 + 0.5 * X_v, 1 + 0.5 * Y_v, 1 + 0.5 * Z_v) \quad \text{Equation 8}$$

$$\sqrt{X_v^2 + Y_v^2 + Z_v^2} = 1 \quad \text{Equation 9}$$

2.5 Results of experiment one

2.5.1 Completion time

A one-way repeated-measures ANOVA was used to compare the performance of the different techniques. Results show a significant main effect for TECH ($F_{(2,10)} = 6.573$, $p = 0.015$). The mean completion time was 90.83s for Widget-based, 86.08s for Duo-Finger Constraint and 75.31s for Constraint Menu (Figure 116). The post-hoc tests with Bonferroni correction show that Widget-based was significantly slower than Constraint Menu ($p = 0.01$). No other significant differences were observed. Therefore, the first hypothesis is only partially validated.

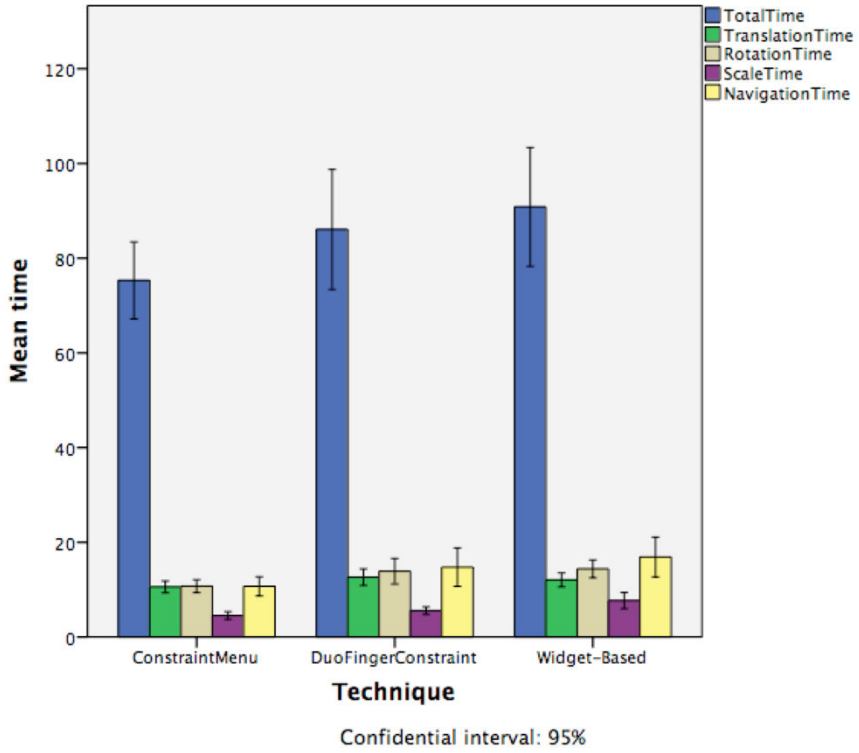


Figure 116: Mean completion time of different techniques. The translation, rotation, scaling and navigation mean time is also displayed.

Besides the total completion time, we have also compared the translation, rotation, scale and camera navigation times among the three techniques (Figure 116). The one-way repeated-measures ANOVA show a significant main effect of TECH for the rotation time ($F_{(2,10)} = 11.525$, $p = 0.003$), scale time ($F_{(2,10)} = 9.507$, $p = 0.005$) and camera navigation time ($F_{(2,10)} = 6.360$, $p = 0.017$). The post-hoc tests with Bonferroni correction show that Widget-based was significantly slower than Constraint Menu for rotation ($p = 0.001$), scale ($p = 0.003$) and camera navigation ($p = 0.02$). Widget-Based was also significantly slower than Duo-Finger Constraint for scaling ($p = 0.022$). Duo-Finger Constraint was significantly slower than Constraint Menu for rotation ($p = 0.02$) and camera navigation ($p = 0.042$). This validates our second hypothesis. No other significant differences were observed.

2.5.2 Translation & Rotation Coordination

In addition, we have calculated the translation and rotation coordination coefficient to compare the translation and rotation efficiency. Similar to (Martinet et al., 2010), the translation coordination coefficient is defined as the ratio of the length of shortest path and the length of actual path. The rotation coordination coefficient is defined as the ratio of the initial rotation mismatch and the amount of actual rotation.

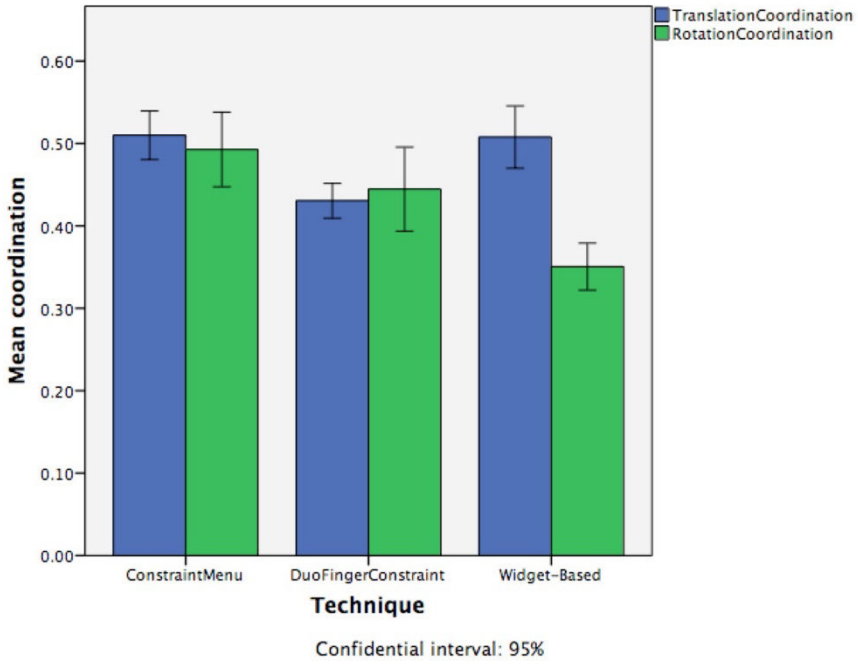


Figure 117: Mean translation and rotation coordination of different techniques.

The one-way repeated-measures ANOVA show a significant main effect of TECH for both translation coordination ($F_{(2,10)} = 17.943$, $p < 0.0001$) and rotation coordination ($F_{(2,10)} = 16.762$, $p = 0.001$) among the different techniques. The mean translation coordination was 0.51 for Widget-Based, 0.43 for Duo-Finger Constraint and 0.51 for Constraint Menu. The mean rotation coordination was 0.35 for Widget-Based, 0.445 for Duo-Finger Constraint and 0.493 for Constraint Menu (Figure 117). The post-hoc tests with Bonferroni correction show that Duo-Finger Constraint had a significantly lower translation coordination value than both Widget-based ($p = 0.002$) and Constraint Menu ($p = 0.001$).

2.5.3 Subjective evaluation

After accomplishing manipulation tasks for each technique, participants were asked to answer a questionnaire to evaluate the technique regarding six different criteria using a seven point Likert scale (1-very bad, 7-very good). We have also asked them to sort all the techniques according to their preference. Seven participants have chosen Constraint Menu as the preferred technique and four participants have selected Duo-Finger Constraint. Table 12 shows the answers for the different criteria. The criterion of intuitiveness is about whether the metaphor of the technique is appropriate and is consistent with the manipulation. The criterion of fluidity is about whether manipulations can be performed without being interrupted by other operations. The Friedman test shows that there was no significant difference among the different techniques for preference of translation, rotation and intuitiveness.

Table 12: The result of subjective evaluation. Significant differences were found for scaling, efficiency and fluidity.

	Widget-Based	Duo-Finger Constraint	Constraint Menu
Translation	5.6 (0.9)	5.5 (1.0)	5.8 (0.7)
Rotation	5.6 (1.3)	4.7 (1.3)	5.5 (1.1)
Scale *	4.1 (1.5)	5.5 (1.1)	5.7 (1.2)
Efficiency *	5.1 (1.2)	4.8 (1.0)	5.7 (1.2)
Intuitiveness	5.3 (1.2)	4.5 (1.1)	5.1 (0.9)
Fluidity *	4.4 (1.5)	4.8 (0.9)	5.4 (0.9)

On the other hand, a significant difference exists among different techniques for scaling ($\chi^2 = 12.905$, $p = 0.002$). The Wilcoxon Signed Rank test shows that participants agreed that Widget-Based was more difficult to use for scaling than Duo-Finger Constraint ($Z = -2.803$, $p = 0.005$) and Constraint Menu ($Z = -2.699$, $p = 0.007$). Regarding manipulation efficiency, a significant difference was found among the different techniques ($\chi^2 = 8.579$, $p = 0.014$). The Wilcoxon Signed Rank test shows that participants strongly preferred Constraint Menu to Duo-Finger Constraint ($Z = -2.810$, $p = 0.005$). A significant difference was also found among the different techniques for fluidity ($\chi^2 = 6.061$, $p = 0.048$). The Wilcoxon Signed Rank test shows that participants strongly agreed that Constraint Menu was more fluent than Widget-based ($Z = -2.308$, $p = 0.021$).

2.6 Discussion of experiment one

We have conducted a controlled experiment to examine whether our technique helps simplify object manipulation on touch-based mobile devices. On the one hand, we wanted to investigate whether our technique outperforms the other two manipulation techniques regarding the completion time and manipulation coordination. On the other hand, we wanted to know how users would subjectively evaluate our technique.

The results indicate that there was a performance gap between Widget-Based and Constraint Menu. The significant difference mainly came from the performance difference in rotation, scale and navigation tasks.

The difference in rotation performance can be partly explained by the fact that different finger movements were used. In fact, the Widget-Based technique requires rotating the finger around the controlled object while the Constraint Menu technique maps linear finger movements to object rotation. Our hypothesis was that making precise control is easier when using linear movements than circular movements. Moreover, when rotating the widget, the object was occasionally occluded by the stretched finger. This affected the

rotation precision and participants had to take more time to adjust the object orientation. Because our technique permits making touch inputs in the empty space, the occlusion problem can be avoided. In addition, when the principal axis is parallel to the screen, the rotation algorithm could not infer the rotation intention correctly due to lack of depth information. In this case, moving the widget may produce undesired rotations. This can also explain why the Widget-Based technique had significantly lower value of rotation coordination than the other two techniques.

For object scaling, the Widget-Based technique was more affected by the point of view. In fact, for some viewpoints, the scaling widgets were hidden. We have observed that some users were changing repetitively the viewpoint in order to modify the scale and then coming back to the original viewpoint to examine the result. This increased the time for both scaling and navigation. Sometimes, when the camera was far from the object and the display size of the object became smaller, undesired scaling widgets were touched. Users had to make touch inputs more carefully to avoid triggering undesired scaling operations. For Duo-Finger Constraint and Constraint Menu, only the specified axis or plane was displayed. Hence, users were aware of whether the desired constraint was selected before performing the pinch gesture to modify the scale. Because both rotation and scaling of Widget-based were influenced by the perspective, users had to perform more camera movements to adjust the point of view. This can explain why the Widget-Based technique required more navigation time than the Constraint Menu technique.

The results also indicate that Duo-Finger Constraint had significantly slower rotation and navigation time than Constraint Menu. We have observed that some participants preferred checking the orientation mismatch when the camera faced directly the side faces of the object. However, one limitation of Duo-Finger Constraint is that the object could not be rotated around the axis that is perpendicular to the screen. As a result, they changed the camera viewpoint to adjust the orientation and then came back to check the result. For Constraint Menu, users could rotate the rotation widget around the object when facing directly the side face. This helped them saving both rotation and navigation time.

One interesting finding is that although no significant difference for translation time exists among the different techniques, Duo-Finger Constraint had significantly less translation coordination than the two other techniques. Because in some camera perspectives, the projection of two axes overlapped each other, it was difficult to select the right one using two fingers. As a result, sometimes, undesired translations were triggered. We think that the longer translation path of Duo-Finger Constraint is duo to translations along the undesired axis.

For subjective evaluation, we have found significant differences for scaling, efficiency and fluidity. Some participants said that it is more likely to trigger undesired scaling actions in some camera perspectives when using the Widget-based technique. They thought that the scaling methods of the other two techniques are easier to use. This comment is consistent with our quantitative results and observations.

However, we did not expect Duo-Finger Constraint to get the lowest preference score regarding efficiency because it required less mean completion time than Widget-Based. Some users pointed out that the performance of Duo-Finger Constraint required switching the wrist angle frequently to select the desired axis and it was uncomfortable to perform touch inputs when the wrist was twisted. Instead of adjusting the wrist angle, some participants rotated frequently the tablet with the NDH to simplify axis-constraint selection. In addition, some participants also commented that they had to make more camera navigation to avoid the overlap of two axes.

Participants strongly agreed that Constraint Menu was more fluent than Widget-based. This can be explained by the fact that in Widget-based, participants had to press function buttons frequently to switch operation modes. Once a constraint was selected, Constraint Menu allowed users to switch operation modes more seamlessly by changing the gesture of the DH.

Some participants have also pointed out some limitations of our technique. First, they said that the rotation gesture is less intuitive to perform than the translation and scaling gestures. In fact, it is less intuitive to determine in which direction the finger should be dragged to rotate an object. They have found that the rotation arrow was misleading when its projection on the screen was parallel to that of the axis. Participants tended to follow the direction of the arrow to trigger object rotation. However, because the finger was dragged in the direction parallel to the axis, object translation was triggered. To overcome this issue, we propose two potential solutions. The first one is to refresh the position of the arrow when the constraint is specified to ensure that the projection of the arrow is perpendicular to the axis. Thus, the visual hint can help users drag the finger in the correct direction. The second solution is to use the two joint fingers to trigger the rotation task. We will examine whether these two potential methods can effectively reduce the ambiguity in a future work.

Another problem we have observed is that the use of the Constraint Menu is limited by the hand size and the holding posture. One participant with large hands commented that he had to tilt the fingertip of the thumb to avoid pressing unwanted constraint options. Two participants have preferred displaying the Constraint Menu higher because they were used to hold the tablet at a higher position. It seems necessary to provide the possibility to adjust the position and size of the menu to adapt to personal preference.

3 Classic3D & Single3D

After proposing the Constraint Menu technique, we wanted to investigate whether experience of object manipulation on mobile devices can be further improved. Because we have already proposed a technique based on the asymmetrical bimanual model, we wanted to step forward to design another technique by leveraging only the DH.

3.1 Identified touch inputs and chording gestures

Actually, one challenge to redesign manipulation and editing techniques for the tactile

paradigm is the limited number of touch input vocabulary. Most of commercial applications on mobile devices only support several classic gestures: drag, rotate and pinch. Because capacitive touchscreens of mobile devices cannot identify the finger pressed on the screen, performing the same gesture using different fingers triggers the same operation. However, in the literature there exist some techniques which have been

proposed to utilize different fingers to expand the design space. Sugiura and Koseki have proposed using different fingerprint patterns to identify the active finger (Sugiura & Koseki, 1998). They thought that instead of providing a group of buttons and mapping each of them to a specific function (Figure 118 (a)), an alternative solution is to map each function to a different finger (Figure 118 (b)). They have also proposed using fingers as storage of information. Users can use a finger to 'save' some frequently-used personal information. Once the finger is pressed on the fingerprint scanner, the saved information can be inputted automatically.

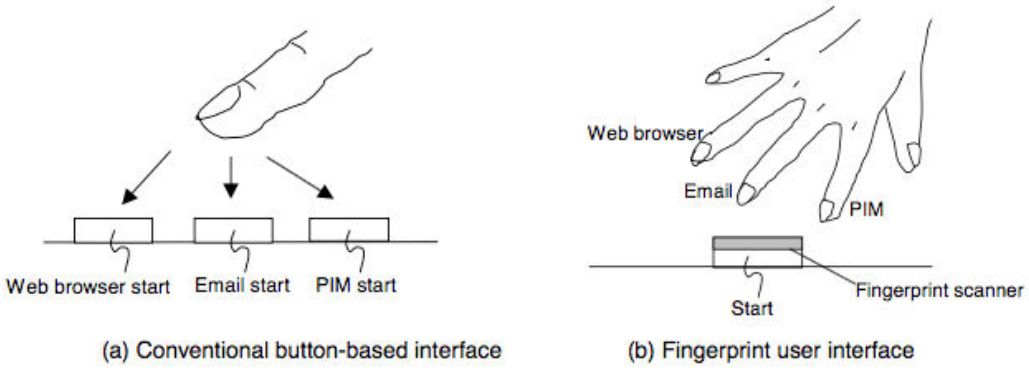


Figure 118: Comparison between two types of user interfaces (Sugiura & Koseki, 1998).

Similarly, Colley and Häkkilä have investigated the concept for mobile phone touch screen input by distinguishing between different fingers (Colley & Häkkilä, 2014). In their first user study, they have found that users perceive the speed and comfort of the tap input differently for most pair of fingers. It was also found that little finger is notably slower and less accurate in tapping targets than the other fingers. Through the other two user studies, they have found that user's perception of the usefulness of the functionalities in the prototype implementation was generally on the positive side.

Several studies have been made to explore how chording gestures can be used to augment touch-based interaction. Chording input involves the simultaneous use of multiple fingers. By mapping each combination of fingers to a specific function, the interaction possibilities can be largely extended. As shown in Table 13 if only one hand is considered, there are in total 31 available chording gestures. If both hands are considered, the count of chording gestures increases significantly to 1023. Lepinski et al. have combined chording inputs with marking menus to increase the number of menu items and eliminate the level of depth (Lepinski et al., 2010). Chording inputs are recognized by analyzing raw images captured by a tracking camera installed below the screen. In their work, 64 functions are averagely

grouped into 8 categories according to their properties. For each category, a unique chording gesture is chosen to represent it. To select a function, users have to first perform a chording gesture to call the corresponding marking menu and then drag the hand in the direction of the desired function to select it. A comparison between multitouch marking menus and traditional marking menus demonstrated that multitouch making menus help saving completion times for both novice and expert users. Although a lot of chording gestures are available to be mapped to different functions, it remains a challenge for users to remember the entire chord-command mapping and recall them when necessary. Through a user study, Wagner et al. have demonstrated that grouping similar chording gestures using categorical mapping reduces the information to retain and effectively helps users remember the chord-command mappings (Wagner et al., 2014).

Table 13: Count of k-combinations of n fingers for different values of n (Goguet et al., 2014a).

k	1	2	3	4	5	6	7	8	9	10	Total
C(10,K)	10	45	120	210	252	210	120	45	10	1	1023
C(5,K)	5	10	10	5	1						31
C(4,K)	4	6	4	1							15
C(3,K)	3	3	1								7
C(2,k)	2	1									3

Goguey et al. have proposed two solutions to identify fingers pressed on touchscreen. For mobile devices, such as smartphones and tablePCs, a camera is installed above to track finger movements and color rings are attached to fingers to simplify visual tracking (Goguey et al., 2014b). For touchscreens of large size, a GameTrak, which is a tracking device, is attached to each finger to acquire stable and accurate finger tracking (Goguey et al., 2014a). To reduce the information for users to retain, they have decided to use both hands in a serial assembly to enlarge the touch vocabulary. In this work, fingers of the NDH are used as selector fingers and fingers of the DH are used as trigger fingers. To interact with the system, users should first make a chording gesture using the NDH to determine the command mapping for trigger fingers of the DH. Then, a trigger finger can be used to launch its corresponding function. As shown in Figure 119, icons of available command mapping are displayed as visual guides. An experiment has shown that once mastered, users can accomplish interaction tasks more effectively with the interface.

Figure 119: (a) Performing a chording gesture of the NDH determines and displays the corresponding command mapping. In this example, 5 drawing tools are displayed. (b) Using the middle finger to trigger the drawing of an eclipse. (c) The movement of the DH controls the size of the eclipse. (d) Another chording gesture of the NDH adds a constraint for the drawing (Goguet et al., 2014a).

Marquardt et al. have also explored how bimanual interaction can be augmented by identifying hand parts (Marquardt et al., 2011). They have developed the TouchID toolkit which allows users to interact through a touchscreen in a more advanced way. With the help of the fiduciary-tagged glove (Figure 120), the toolkit can identify which part of the hand, which side of the hand and which person is actually touching the surface. Besides mapping fingers to different functions, the authors have also explored pressing the knuckle to show the tool assigned to this finger, or pressing the back of hand to show all assigned tools. Moreover, this toolkit permits developers to self-define other hand gestures and postures by using data captured by the glove.

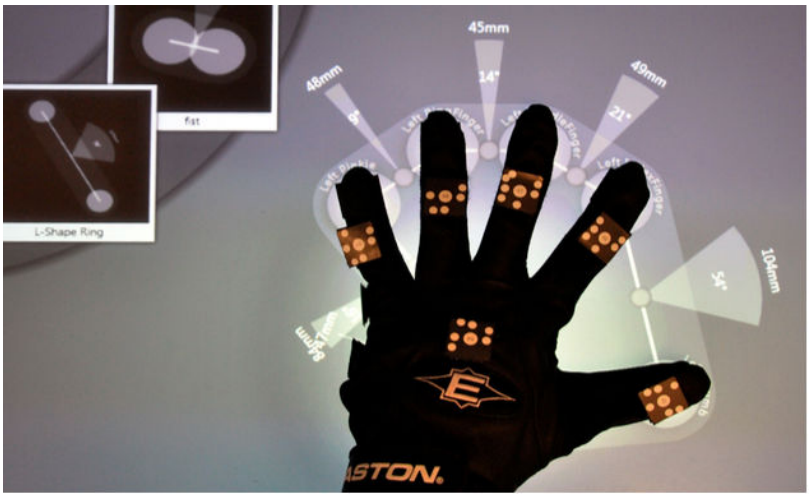


Figure 120: Using the TouchID Posture Configurator to train a new posture with the fiduciary-tagged glove (Marquardt et al., 2011).

All the aforementioned work demonstrates that identified touch inputs and chording gestures open a new path for interaction designers to imagine new interaction methods. However, to our best knowledge, no previous work has been made to propose 3D manipulation techniques using identified touch inputs and chording gestures. Thus, it is worth exploring how to take advantage of this concept to design new touch-based manipulation techniques.

3.2 Design rationale

For the design rationale of the Constraint Menu technique, four requirements that the new techniques should meet were listed. When designing manipulation techniques using identified touch inputs, we still need to satisfy these four requirements. Furthermore, to make the interaction more fluent, this time we aim to design a set of gestures which can be performed using only the DH. Similar to the technique proposed in (Au et al., 2012), we want the selection of manipulation mode, transformation constraint and control of movement to be all embedded in one gesture.

For manipulation tasks, if we consider the operation mode and the transformation

constraint, they can be divided into 16 subtasks (Table 14). For axis-constrained manipulation, it contains 3 axis translations, 3 axis rotations, and 3 axis scaling. For plane-constrained manipulation, it contains 3 plane translations and 3 plane scaling. For uniform manipulation, it only contains 1 uniform scaling. To design single-handed manipulation techniques, the biggest challenge is how to choose 16 separate gestures and map each of them to a different manipulation subtask. For 2D manipulation, objects can be translated, rotated and zoomed using 1-finger drag, 2-finger rotate and 2-finger pinch gestures, respectively. However, this set of classic gestures cannot be used to control all 9 DOF of 3D objects. Although it is possible to design some more 2D gestures and map them to 3D object transformation, it is difficult to find appropriate metaphors to help users learn and memorize a set of 16 gestures.

Table 14: Axis-constraint, plane-constraint and uniform manipulation.

	Axis-constraint manipulation			Plane-constraint manipulation			Uniform manipulation
	X	Y	Z	X-Y	Y-Z	Z-X	X-Y-Z
Translation	*	*	*	*	*	*	
Rotation	*	*	*				
Scaling	*	*	*	*	*	*	*

As discussed above, some techniques have explored the use of identified touch inputs to enrich interaction possibilities (Sugiura & Koseki, 1998; Marquardt et al., 2011; Goguey et al., 2014a). Thus, the design space provided by identified touch inputs can be extended for our purpose. Because Wagner et al. argue that grouping similar chording gestures into the same category is helpful to reduce the information for users to retain (Wagner et al., 2014), we decided to group manipulation subtasks according to the constraint and map the index, middle and ring finger to the constraint of x-, y- and z-axis of the coordinate system, respectively. Users can press one or more axis fingers on the screen to specify the constraint, and then perform specific finger motions to trigger manipulation tasks. These fingers were chosen because they are adjacent fingers on the hand. This is expected to help users to recall easily the conventional order of three principal axes (x/y/z). The thumb and the pinky finger were not used as axis fingers due to their anatomic limitations. Indeed, dragging the thumb and/or the pinky fingers on the screen is less comfortable for the user.

3.3 Classic3D

With the help of identified touch inputs, we have designed our first technique which is called Classic3D[4] by extending classic 2D manipulation gestures. Because dragging a

[4] A video demo can be found in this link https://www.youtube.com/watch?v=XSWpH7-oQgo

finger is a very natural metaphor for object translation, we decided to reuse it for axis-constrained translation. To translate the selected object along one principal axis, users only have to tap the corresponding axis finger on the screen and drag it in the direction parallel to the axis (Figure 121 (a)). After a direction is specified, the selected axis is first projected on the screen space and then the dragging movement is projected to its projection to calculate the translation range for each frame.

Similar to object translation, we follow the same strategy to design gestures for object rotation and scaling. Some techniques allow dragging one or two fingers in the perpendicular direction of the projection of one axis to rotate the object around it (Cohé et al., 2011; Au et al., 2012). This gesture is reused by Classic3D to trigger object rotation. Users can press one axis finger to specify the rotation axis, and then drag the finger in the vertical direction of the rotation axis to trigger the rotation (Figure 121 (b)). To scale the object along one principal axis, a pinch gesture can be performed with the thumb and the corresponding axis finger (Figure 121 (c)). It is unnecessary to make the pinch gesture in the direction of the selected axis.

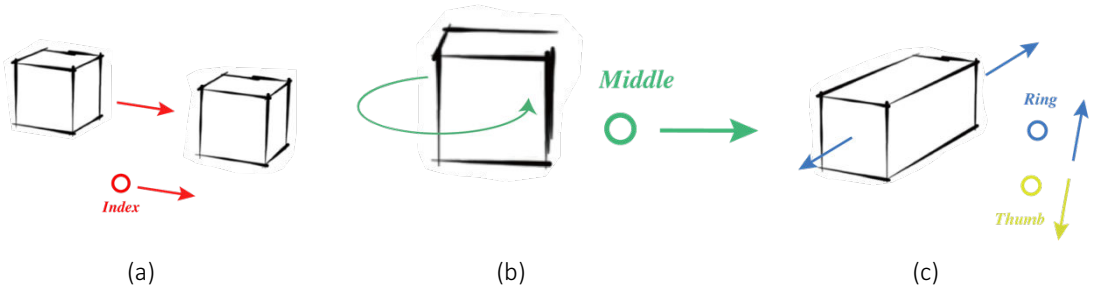


Figure 121: Gestures of Classic3D for axis-constrained manipulation. (a) X translation; (b) Y rotation; (c) Z scaling.

3.4 Single3D

To explore other manipulation metaphors, we propose a second technique called Single3D. The translation gestures of Single3D are the same as those of Classic3D (Figure 122 (a)), however rotation gestures and scaling gestures are different. Besides dragging the finger in the vertical direction of the rotation axis, we thought that circling the finger on the screen can be a more intuitive metaphor for object rotation. Users can press one axis finger to specify the rotation axis, and then draw circles continuously with the finger to control the rotation (Figure 122 (b)).

Although the pinch gesture is a natural metaphor for object scaling, using the middle or ring finger with the thumb to perform the pinch gesture may be less comfortable. Moreover, Ghomi et al. have recommended avoiding the use of chording gestures in which the middle or ring finger is lifted while its neighbors touch the surface (Ghomi et al., 2013). In some desktop modeling applications, objects can be zoomed in one

direction by dragging a handle bar. Objects can be enlarged (shrunk) when the handle bar is dragged in the positive (negative) direction. However, this metaphor cannot be used directly for object scaling because the motion of finger dragging along the axis is already used for object translation. To differentiate object translation and scaling, Single3D allows using the double click to trigger object scaling (Figure 122 (c)). After the axis finger is pressed and lifted, pressing it another time on the screen and dragging it along the positive (negative) direction of the axis can increase (decrease) the size of the object.

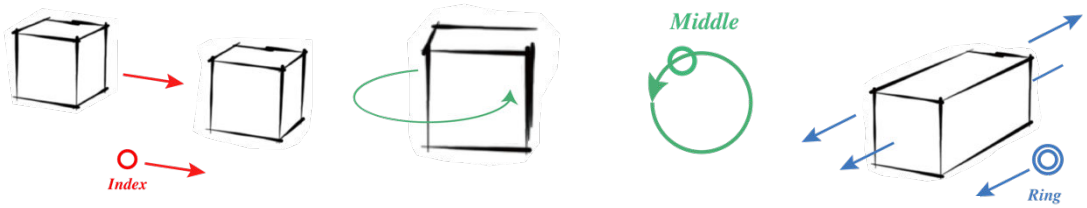


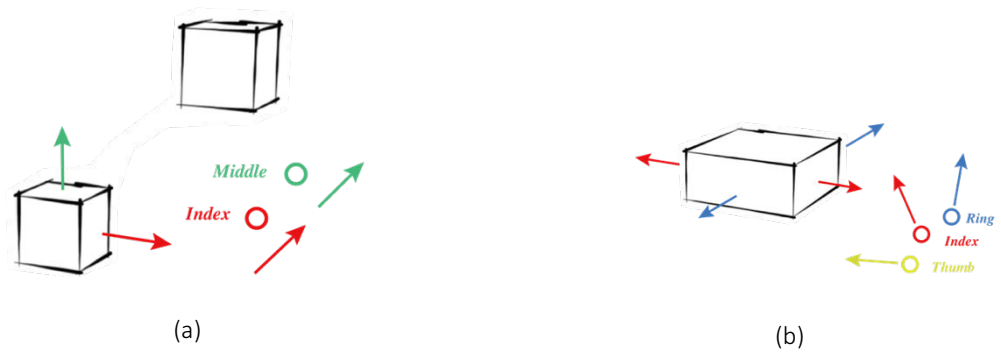
Figure 122: Gestures of Single3D for axis-constrained manipulation. (a) X translation; (b) Y rotation; (c) Z scaling.

3.5 Plane-constrained & uniform manipulation

Besides axis-constraints, both Classic3D and Single3D can also be used to perform plane-constrained and uniform manipulation. For plane-constrained translation, both Classic3D and Single3D allow pressing two axis-fingers to set the plane constraint and then dragging both fingers together to move the object in the selected plane (Figure 123 (a)).

When using Classic3D, users can perform the pinch gesture with the thumb and two axis fingers to zoom the object in the two specified directions (Figure 123 (b)). Similarly, pressing all the fingers except the pinky finger and performing the pinch gesture can zoom the object uniformly.

For Single3D, users can press two or three axis fingers twice on the screen to trigger plane-constrained or uniform scaling (Figure 124). If the pressed fingers are dragged in the positive direction of the sum vector formed by the unit vectors of the selected axes, the object is enlarged. Otherwise, the object is shrunk.



(a) (b)

Figure 123: Gestures of Classic3D for plane-constrained manipulation. (a) XY translation; (b) XZ scaling.

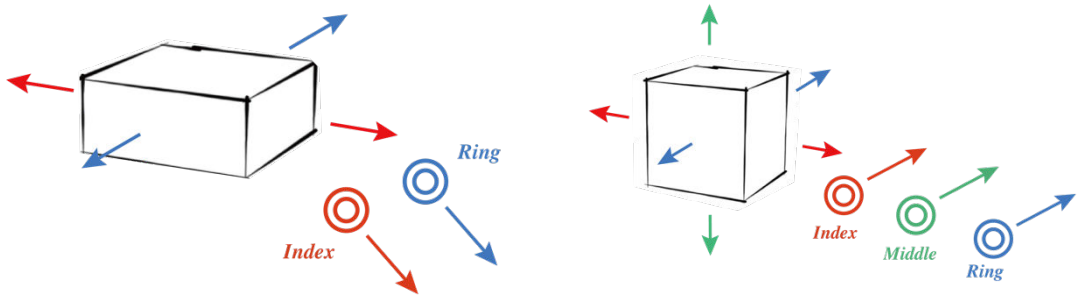


Figure 124: Gestures of Single3D for plane-constrained and uniform scaling. (a) XZ scaling; (b) Uniform scaling.

3.6 Setup of finger identification

To explore the usage of identified touch inputs, it is necessary to detect finger movements above the screen. Although some consumer devices, such as Samsung Galaxy S4, can track hand movements in the air, they are not able to locate each finger on and above the screen. In the literature, some solutions have been proposed to identify fingers using techniques of computer visions (Goguey et al., 2014b; Lepinski et al., 2010), using embedded tracking devices (Goguey et al., 2014a; Marquardt et al., 2011) or scanning fingerprints (Sugiura & Koseki, 1998; Holz & Baudisch, 2010). In this work, we propose a low-cost solution to identify touch inputs.

Similar to (Colley & Häkkilä, 2014), a Leap Motion Controller is used to identify finger inputs. As shown in Figure 125, the Leap Motion Controller is installed upside down above an iPad air 2 and it is fixed on a metal frame which is attached to the iPad. Sensors of the Leap Motion Controller have a field of view of about 150 degrees and the effective range extends from approximately 25 to 600 mm above the device. The frequency of it is 60 Hz. Finger tracking is performed on a PC and extracted finger information is sent to the iPad through wireless network. The program was written in Unity3D with C#. This setup allows us to realize and examine quickly our design ideas. Although we have used an external

device to track finger movement, we are of the view that, in the near future, finger tracking technology can be embedded into mobile devices.

Figure 125: The setup of our prototype.

3.7 Calibration

To make touch source identifiable, we have first developed a calibration application to locate the iPad in the coordinate system of the Leap Motion Controller (Figure 126). A white circle was displayed in one position on the screen in the application. When the index finger tapped the circle, we recorded both the 2D touch position in the screen coordinate system and the 3D finger position in the Leap Motion Controller system in 100 consecutive frames. The circle turned red when it was touched. In total four circles in different positions were used for calibration. Once data recording was finished, given the known screen size and resolution, the location of the screen could be obtained by resolving linear equations.

Once the iPad and the Leap Motion Controller are fixed on the support and the calibration is done, sources of touch inputs can be identified. When a touch input appears, its 3D touch position in the coordinate system of the iPad can be calculated and is compared with positions of all the extended fingers. The finger with the smallest distance to the touch position is identified as the source of the touch (Figure 127 (a)). Besides identifying fingers, our prototype can also distinguish finger touches and knuckle touches. When a finger touches the screen, the palm of the hand faces the screen. When the knuckle of one finger taps the screen, it was the hand back that faces the screen. Using the palm normal orientation provided by the Leap Motion Controller, the system can tell the difference between fingers and knuckles (Figure 127 (b)). This further extends our gestures design space.

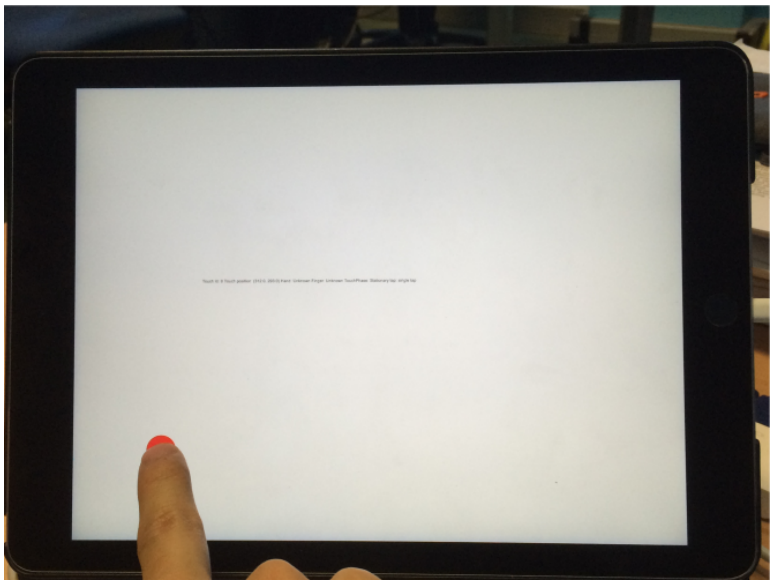


Figure 126: The calibration application.

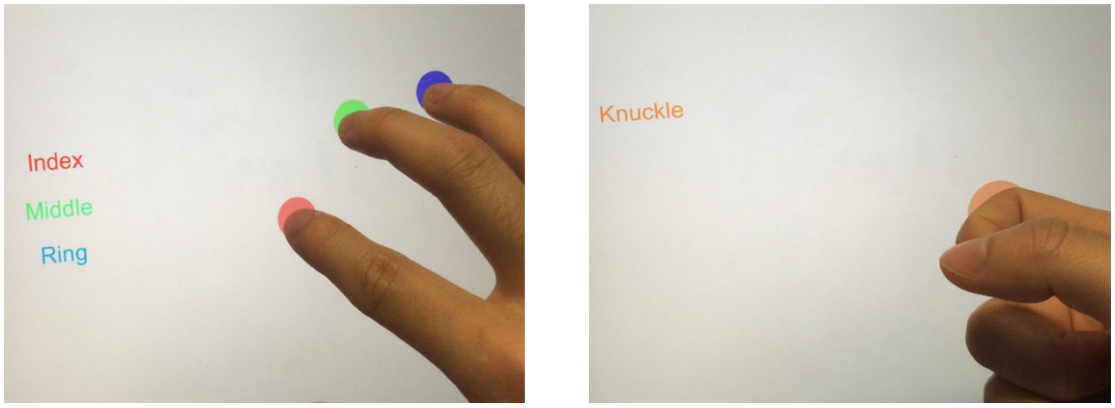


Figure 127: Identifiable touch inputs. (a) Each finger can be identified; (b) Touch inputs made by knuckles can be identified as well.

3.8 Main experiment two: learnability of Classic3D and Single3D

After designing Classic3D and Single3D, we conducted a controlled experiment to evaluate two aspects of these techniques. First, we investigated the learnability of these two techniques. In fact, because using identified touch inputs for interaction is an unfamiliar concept for the public, it is necessary to study whether users can understand this concept rapidly and learn our techniques easily. Second, we investigated the usability

of these two techniques. Research question are: (1) can these techniques be used to manipulate 3D objects effectively? And (2) Do users feel comfortable when using different fingers for interaction? In this experiment, we compared Classic3D and Single3D with Constraint Menu.

3.8.1 Hypotheses

We have made three hypotheses before conducting the controlled user study:

H1: Classic3D and Single3D would be faster to learn than Constraint Menu. Because only the DH is involved, we hypothesize that gestures of our techniques are easier to understand and to memorize.

H2: After training, all the techniques can be used with the same level of precision.

H3: Classic3D and Single3D would outperform Constraint Menu regarding the efficiency. Indeed, manipulation gestures of our techniques can be performed using only the DH and less visual navigation is required.

3.8.2 Participants

Twelve unpaid subjects (9 males and 3 females, 22 to 28 years old) participated in this study. They are all students from one university. All of them use smartphones frequently in everyday life. They are all right handed and they only have novice experience of using 3D desktop modeling software.

3.8.3 Procedure and Design

The experimental design was a within-subject design with one primary factor (TECH) with three levels: Constraint Menu, Classic3D and Single3D. Participants were first instructed to learn each technique and then they were asked to use it to accomplish a set of manipulation tasks. The task set consisted of 3 axis-constrained translations, 3 axis-constrained rotations and 3 axis-constrained scaling (Figure 129). We did not ask participants to perform the docking task used in the first experiment because the efficiency of a complex docking task is not only affected by the usability of the technique, but also influenced by the spatial ability of the participant. Thus, to reduce the effect of participants individual spatial abilities, they were asked to perform simple tasks which can be accomplished by only one gesture. As shown in Figure 128, the experiment was divided into 3 phases.

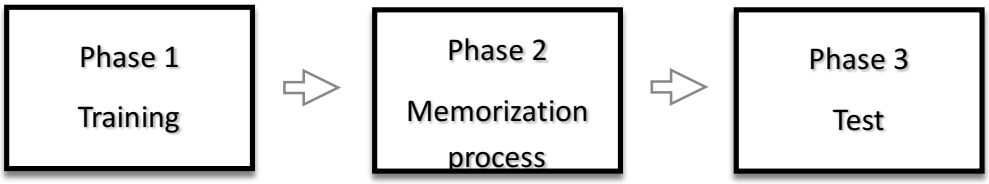


Figure 128: The different phases of the experiment.

In Phase 1, participants were taught the translation, rotation and scaling gestures, successively. For each task, participants performed 5 trials in series to learn the corresponding gesture. At the beginning of each trial, a cube object was displayed in the center of the screen. For translation tasks, a target was displayed in the scene and participants were asked to translate the cube close enough to the target position (Figure 129 (a)). For rotation tasks, the cube was inside a target of a larger size. The participants were asked to rotate the cube so that its colored face was parallel to the colored face of the target (Figure 129 (b)). For scaling tasks, a cuboid larger than the cube along one principal direction was displayed at the same position. The participants were asked to enlarge the cube in the same direction to become as large as the target (Figure 129 (c)). The cube could be controlled only when the correct gesture was performed. For example, if the participant had to translate the cube along the x-axis, performing the gesture to translate it along the y-axis or to rotate it around the x-axis did not change the state of the cube.

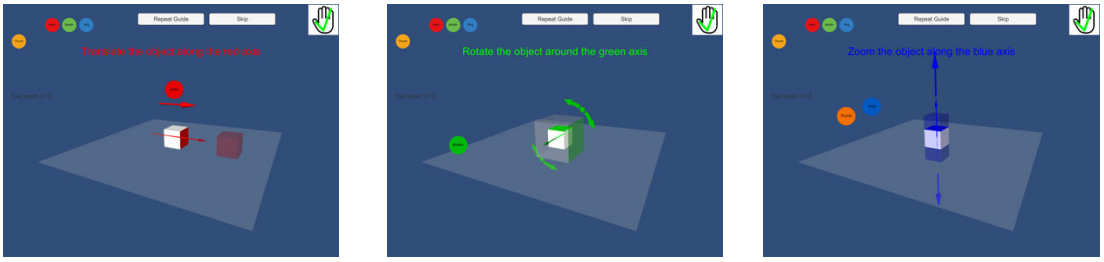


Figure 129: Manipulation tasks of the user study. (a) Translation; (b) Rotation; (c) Scaling.

Before each trial, a start button was displayed in the center of the screen. Participants had to press the button to start the trial. Once the trial was started, a text message describing the objective of the trial was displayed on the top of the screen. To explain how manipulation gestures should be performed, both audio and visual instructions were provided to instruct participants. Similar to (Ghomi et al., 2013), two successive instructions for each task were provided. At the beginning of each trial, the first instruction was provided to explain which finger(s) should be pressed on the screen. After the operation was performed correctly, the second instruction was provided to explain the finger movements. The instructions were displayed once, but participants could click a button to repeat the instructions (Figure 129). For Classic3D and Single3D, a hand icon

was displayed on the top right corner of the screen to indicate whether the right hand was detected by the Leap Motion Controller. Because the hand tracking was affected by the size and orientation of the hand, sometimes the pressed fingers were not recognized correctly. Thus, a correct gesture might be recognized as an incorrect gesture. In this case, participants were allowed to press the skip button to move to the next trial.

After learning the gestures in Phase 1, participants were asked to accomplish blocks of trials to memorize all the manipulation gestures in Phase 2. Each block consisted of 9 trials, each eliciting one manipulation task in a random order. Similar to Phase 1, participants should perform the correct gesture to manipulation the cube as demanded. However, at the beginning of each trial, the instructions were not provided. Participants had to recall the gesture by themselves. If a participant did not remember the gesture, he/she was allowed to use a help button to get the instructions. The memorization process ended when participants reached the objective and subjective end-criterion. The objective end-criterion was reached when participants successfully accomplished two blocks of trials in sequence without making errors or clicking the help button. An error was made when the gesture performed by the participant did not correspond to the active trial. Participants reached the subjective end-criterion when they believed that they have memorized correctly all the gestures. After the memorization process, participants were asked to take a rest. Then, they were asked to accomplish 2 blocks of trials in Phase 3. Similar to Phase 2, each block consisted of 9 different trials. During the test, no instructions were provided to help the users recall the gestures. To eliminate any effect of the trial order in this experiment, TECH was ordered in a balanced Latin-square. Participants were divided into three groups respectively. Once the experiment of one technique was finished, participants were asked to fill in a questionnaire to evaluate the technique.

3.8.4 Data Collection

In this experiment, different measurements were recorded for evaluation. To evaluate the learnability of each technique, the number of required blocks until participants performed correctly all gestures twice in sequence was recorded. The number of required blocks until participants felt trained enough to memorize all the vocabulary was recorded as well. For both Phase 2 and Phase 3, we recorded the number of errors. To analyze the efficiency of each technique, we recorded the reaction time and completion time for each trial. The reaction time is calculated from the moment when the start button was clicked and ended when the first touch input was made after the trial began. The completion time started from the moment when the start button was clicked and ended when the start button was displayed again for the next trial.

3.9 Results of experiment two

3.9.1 Objective end-criterion & subjective end-criterion

For the objective end-criterion, a one-way repeated measure ANOVA show a significant main effect of TECH ($F_{(2,10)} = 4.623$, $p = 0.038$) on the number of required blocks in phase 2. Participants required in average 2.417 ($S\,D = 0.229$), 3.833 ($S\,D = 0.534$) and 3.750 ($S\,D = 0.579$) blocks for Constraint Menu, Classic3D and Single3D, respectively to reach the objective end-criterion. The post-hoc tests with Bonferroni correction show a marginal effect for the TECH between Constraint Menu and Classic3D ($p = 0.056$). No other significant effect was found. Because our techniques do not require fewer blocks of training than Constraint Menu, our first hypothesis is not validated.

For subjective end-criterion, the one-way repeated measure ANOVA shows also a significant main effect of TECH ($F_{(2,10)} = 5.298$, $p = 0.027$). Participants required in average 2.417 ($S\,D = 0.229$), 3.917 ($S\,D = 0.514$) and 3.917 ($S\,D = 0.596$) blocks for Constraint Menu, Classic3D and Single3D, respectively to reach the subjective end-criterion. The post-hoc tests with Bonferroni correction show that Constraint Menu required in average significantly fewer blocks to reach the subjective end-criterion than Classic3D ($p = 0.036$). There were only three participants who have repeated more blocks for subjective end-criterion than for objective end-criterion. One participant repeated one more block to reach the subjective end-criterion for Classic3D while two participants repeated one more block respectively to reach the subjective end-criterion for Single3D.

Table 15:The errors distribution of the different techniques in phase 2. T=Translation, R=Rotation, S=Scale.

	RGWF	WGRF	WGWF
Constraint Menu-T	0	0	0
Constraint Menu-R	0	6	0
Constraint Menu-S	0	1	0
Classic3D-T	2	1	2
Classic3D-R	2	12	1
Classic3D-S	6	4	0
Single3D-T	2	6	0
Single3D-R	0	8	0
Single3D-S	2	7	0

Table 16: The errors distribution of the different techniques in phase 3. T=Translation, R=Rotation, S=Scale.

	RGWF	WGRF	WGWF
Constraint Menu-T	0	0	0
Constraint Menu-R	0	3	0
Constraint Menu-S	0	2	0
Classic3D-T	0	0	0
Classic3D-R	0	3	1
Classic3D-S	0	0	0
Single3D-T	0	0	0
Single3D-R	0	2	0
Single3D-S	0	0	0

3.9.2 Errors rate

The one-way repeated measure ANOVA shows no significant effect of TECH on the errors rate in phase 3. Thus, our second hypothesis is validated. The mean errors rate was 2.3% (*SD* = 0.013), 1.9% (*SD* = 0.010) and 0.9% (*SD* = 0.006) for Constraint Menu, Classic3D and Single3D, respectively. The total number of errors was 5, 4 and 2 for Constraint Menu, Classic3D and Single3D, respectively in phase 3. The total number of errors was 7, 30 and 25 for Constraint Menu, Classic3D and Single3D, respectively in phase 2. The errors are categorized into three types of errors: (1) *right gesture, wrong finger* (RGWF), (2) *wrong gesture, right finger* (WGRF), and (3) *wrong gesture, wrong finger* (WGWF). Table 15 and Table 16 show the distribution of errors for the different techniques in phase 2 and phase 3, respectively.

3.9.3 Completion time

The one-way repeated measure ANOVA shows a significant main effect for TECH $F_{(2,10)} = 9.492$, $p = 0.005$) on the completion time. The mean completion time was 3.729*s* (*SD* = 0.304), 4.306*s* (*SD* = 0.361) and 4.524*s* (*SD* = 0.360) for Constraint Menu, Classic3D and Single3D, respectively. The post-hoc tests with Bonferroni correction show that Constraint Menu was significantly faster than Classic3D ($p = 0.009$) and Single3D ($p = 0.005$). So our third hypothesis is not validated. Moreover, the one-way ANOVA shows a significant main effect for TECH ($F_{(2,10)} = 6.139$, $p = 0.018$) on the reaction time. The mean reaction time was 1.540*s* (*S D* = 0.162), 1.919*s* (*S D* = 0.211) and 1.791*s* (*S D* = 0.149) for Constraint Menu, Classic3D and Single3D, respectively. The post-hoc tests

with Bonferroni correction show that Constraint Menu has a significantly shorter mean reaction time than Classic3D ($p = 0.012$).

For a deeper analysis of the performance of the different techniques, the reaction time and completion time for translation, rotation and scale were also calculated. The one-way repeated measure ANOVA shows no significant effect for translation. The mean translation reaction time was 1.549*s* ($SD = 0.206$), 1.736*s* ($SD = 0.176$) and 1.706*s* ($SD = 0.185$) for Constraint Menu, Classic3D and Single3D, respectively.

On the other hand, a significant main effect for TECH was observed for rotation ($F_{(2,10)} = 5.824$, $p = 0.021$). The mean rotation reaction time was 1.675*s* ($SD = 0.199$), 2.195*s* ($SD = 0.264$) and 2.018*s* ($SD = 0.208$) for Constraint Menu, Classic3D and Single3D, respectively. The post-hoc tests with Bonferroni correction show that Constraint Menu had a significantly shorter mean rotation reaction time than Classic3D ($p = 0.014$).

A significant main effect of TECH was also found for scale ($F(2,10) = 5.628$, $p = 0.023$). The mean scale reaction time was 1.404*s* ($SD = 0.119$), 1.827*s* ($SD = 0.209$) and 1.653*s* ($SD = 0.121$) for Constraint Menu, Classic3D and Single3D respectively. The post-hoc tests with Bonferroni correction show that Constraint Menu had a significantly shorter mean scale reaction time than Classic3D ($p = 0.025$).

Regarding the completion times, the one-way repeated measure ANOVA shows no significant main effect of TECH for translation and rotation. The mean translation completion time was 3.686*s* ($SD = 0.335$), 3.836*s* ($SD = 0.326$) and 4.176*s* ($SD = 0.295$) for Constraint Menu, Classic3D and Single3D respectively. The mean rotation completion time was 4.194*s* ($SD = 0.375$), 4.662*s* ($SD = 0.436$) and 5.488*s* ($SD = 0.558$) for Constraint Menu, Classic3D and Single3D, respectively.

On the other hand, a significant main effect of TECH was found for scale ($F_{(2,10)} = 7.210$, $p = 0.012$). The mean scale completion time was 3.324*s* ($SD = 0.269$), 4.421*s* ($SD = 0.426$) and 3.946*s* ($SD = 0.349$) for Constraint Menu, Classic3D and Single3D, respectively. The post-hoc tests with Bonferroni correction show that Constraint Menu had a significantly shorter mean scale completion time than Classic3D ($p = 0.013$) and Single3D ($p = 0.038$).

3.9.4 Subjective evaluation

After accomplishing the required tasks for each technique, participants were asked to answer a questionnaire to evaluate the technique regarding six different criteria using a seven-point Likert scale (from 1-very bad to 7-very good). In addition, participants were asked to evaluate the comfort level of the different finger motions. Finally, the participants were asked to sort the three techniques according to their preference. Four, five and three participants have chosen Constraint Menu, Classic3D and Single3D, respectively as the preferred technique.

The answers for the different criteria are shown in Table 17. The criterion of intuitiveness is about whether the metaphor of the technique is appropriate and is consistent with the manipulation. The Friedman test shows a significant difference among the different techniques for the difficulty of learning ($\chi^2 = 13.941$, $p = 0.001$). The Wilcoxon Signed Rank test shows that participants agreed that Constraint Menu was easier to learn than Classic3D ($Z = -2.460$, $p = 0.014$) and Single3D ($Z = -2.972$, $p = 0.003$). However, there was no significant difference among the different techniques for the difficulty of use and for preference of translation, rotation, scale and intuitiveness as well.

Table 17: The result of subjective evaluation. Significant difference was only found for difficulty of learning.

	Constraint Menu	Classic3D	Single3D
Learn *	6.7 (0.5)	5.9 (0.7)	5.7 (0.7)
Use	6.2 (0.9)	5.6 (1.5)	5.8 (1.0)
Translation	6.1 (0.8)	6.3 (0.8)	6.3 (0.8)
Rotation	5.6 (1.1)	5.1 (1.0)	5.4 (1.2)
Scale	6.4 (1.0)	5.8 (1.4)	6.3 (0.8)
Intuitiveness	6.3 (0.7)	5.7 (1.1)	5.8 (0.8)

The answers for the degree of comfort of different finger motions are shown in Table 18. The Friedman test shows a significant difference among the index, middle and ring finger for performing the panning motions ($\chi^2 = 15.235$, $p < 0.0001$). The Wilcoxon Signed Rank test shows that participants agreed that using the index finger for panning was more comfortable than the middle finger ($Z = -2.111$, $p = 0.035$) and the ring finger ($Z = -2.687$, $p = 0.007$). They also agreed that using the middle finger for panning was also more comfortable than the ring finger ($Z = -2.565$, $p = 0.010$).

The Friedman test shows a significant difference among the index, middle and ring fingers for performing the circling motions ($\chi^2 = 14.000$, $p = 0.001$). The Wilcoxon Signed Rank test shows that participants agreed that using the ring finger for circling was less comfortable than the index finger ($Z = -2.823$, $p = 0.005$) and the middle finger ($Z = -2.791$, $p = 0.005$).

The Friedman test shows a significant difference among the index, middle and ring finger for performing the pinching motions ($\chi^2 = 18.571$, $p < 0.0001$). The Wilcoxon Signed Rank test shows that participants agreed that using the index finger for pinching was more comfortable than the middle finger ($Z = -2.041$, $p = 0.041$) and the ring finger ($Z = -2.821$, $p = 0.005$). Using the middle finger was also more comfortable than the ring finger ($Z = -2.879$, $p = 0.004$).

Table 18: The result of subjective evaluation for different finger motions. Significant differences were found for panning, circling and pinching.

	Index	Middle	Ring
Panning *	6.8 (0.6)	6.3 (0.6)	5.1 (1.4)
Circling *	6.7 (0.7)	6.3 (1.0)	4.5 (1.2)
Pinching *	6.9 (0.3)	6.2 (1.0)	4.9 (1.4)

3.10 Discussion of experiment two

We have conducted a user study to examine whether Classic3D and Single3D can be learnt by users without a great cognitive effort. We also investigated whether these techniques can be performed correctly for object manipulation and can provide acceptable efficiency. Besides objective performance, we wanted to know how users would subjectively evaluate these techniques and the degree of comfort of the different fingers motions.

The results indicate that Constraint Menu required in average fewer numbers of blocks for training and a significant difference was observed between Constraint Menu and Classic3D. The results of the subjective evaluation indicate that participants agreed that Constraint Menu is significantly easier to learn than the other two techniques.

The difference in learning difficulty can be explained by the fact that people are not familiar with using each finger for a different purpose. More cognitive efforts are necessary to understand and memorize the mapping between the index, middle and ring fingers to the three principal axes of the coordinate system. However, Constraint Menu provides a menu to display explicitly the mapping between the constraint buttons and the principal axes. Moreover, because each constraint button shares the same color with its corresponding axis, it is easier to construct the mapping in the mind. It is to be noted that participants of this study only have novice experience of manipulation 3D virtual objects. We think that it should be easier for expert users to construct the mapping between the axis-fingers and manipulation constraints because of their better understanding of 3D virtual environments. However, further investigations are needed to verify this hypothesis in the future.

Although Constraint Menu is easier to learn, in phase 3 no significant effect was found for TECH regarding the errors rate from either the data of the experiment or the subjective evaluation. Therefore, we argue that Classic3D and Single3D can be used without great difficulties after a short period of training. Some participants appreciated the design of Constraint Menu and they thought that using the NDH for specifying constraints helps reducing the burden of the DH. They also commented that it is easier to find out the correct button for the constraint than to recall the corresponding finger. However, some participants said that more visual navigation is required for Constraint

Menu because they had to visually locate the desired button. For those participants who prefer Classic3D and Single3D, they thought that the mapping between the fingers and the principal axes can be recalled quickly and they thought that using only the DH for manipulation is more convenient.

The results indicate that there was a performance gap regarding the reaction time between Constraint Menu and Classic3D. The significant difference mainly came from the difference of reaction time in rotation and scaling tasks. To rotate an object around a principal axis, both Constraint Menu and Classic3D require dragging the finger of the DH in the direction perpendicular to the axis. As some participants have pointed out, they only had to determine the dragging direction when Constraint Menu was used, while they had to further recall the mapping between fingers and constraints when Classic3D was used. The performance gap can be explained by the additional cognitive load of Classic3D. Although Single3D also requires users to recall which finger should be pressed, some participants commented that the circling motions of the axis finger are a better metaphor for object rotation than the panning motions. Thus, they could recall the rotation gesture of Single3D faster. Although the scaling gesture of Classic3D is similar to that of Constraint Menu, some participants commented that performing the pinching motions with one axis finger and the thumb is more troublesome than simply dragging the axis finger, especially for the ring finger. They had to find a comfortable hand posture to perform the scaling gesture of Classic3D.

The results indicate that both Classic3D and Single3D had significantly longer completion time than Constraint Menu. The performance gap mainly came from the difference in scaling tasks. For Classic3D, first it requires longer reaction time to start scaling. Moreover, some participants were more cautious when using the scaling gesture of Classic3D. After pressing two fingers on the screen, they did not start the pinching motions until they ensured that the correct constraint was specified. For Single3D, we think the performance gap came from the fact that a double click motion was necessary before dragging the finger along the axis. In addition, because the translation and scaling gestures use the same finger motion and the only difference is the registration, it is a little ambiguous to differentiate these two gestures.

Although no significant main effect was found for rotation, Single3D required more rotation time than both Constraint Menu and Classic3D. First, our algorithm can only recognize the circling finger motion after a quarter circle is drawn. Thus users have to wait the start of the rotation until the circular finger movement is recognized. Second, because the circling motion is less easy to control than the linear motion (Nancel et al., 2011b), the CD ratio for object rotation was set to a high value. Some participants complained that the rotation gesture of Single3D is less efficient than the other two techniques. Therefore, decreasing the CD ratio could improve the performance of Single3D for object rotation.

There is no surprise to find out that the degree of comfort for using the ring finger is lower than that of the index and middle finger. Although in average, the degrees of

comfort for all the fingers are higher than 4, some motions have received negative values. One, three and two participants gave a score of 3 for the panning motion, circling motion, and pinching motion of the ring finger, respectively. Regarding the difference of the hand structure among participants, we think that using identified touch inputs has its potential value for interaction and can be an acceptable design strategy for users who do not dislike using the ring finger.

3.11 Design implications

After conducting the user study and analyzing the results, several design implications can be drawn to further improve our techniques.

(1) The main problem for learning our techniques is that there is a lack of hints to construct the mapping between axis-fingers and constraints. To help users recall the mapping more rapidly, one possible solution is to display the shadow of each axis-finger with the color of its corresponding axis on the screen once the hand is detected. These explicit visual guides may be helpful to reduce effectively the cognitive effort and accelerate users' reaction.

(2) All the participants thought that translation gestures of Classic3D and Single3D are very easy to use. Many of them commented that using only the DH for translation is more convenient and more intuitive.

(3) For object rotation, many participants found that the metaphors of Constraint Menu and Classic3D are not intuitive enough. Some of them complained that they had to be careful to avoid triggering undesired translations. In phase 1 of the user study, we observed that after pressing the axis-finger, they tended to drag the finger in a circular way instead of drawing a straight line. Many participants thought that the circling motion is more coherent with the object rotation movement. For this reason we think that the rotation gestures of Single3D are better than that of Classic3D. However, we think that providing some additional visual guidance may be helpful to improve the performance of Constraint Menu and Classic3D. For instance, after the finger is pressed on the screen, two arrows can be displayed to indicate the dragging direction for clockwise and anti-clockwise rotations.

(4) According to the results of the subjective evaluation, most of the participants found that object scaling gestures for Classic3D and Single3D intuitive and easy to use. Only one participant found it uncomfortable to make the pinch gesture with the thumb and ring fingers. However, in general our observation suggests that the gestures of Classic3D are better because they do not introduce ambiguity between translation and scaling.

In conclusion, we think that a hybrid technique which uses gestures of Single3D for rotation and gestures of Classic3D for scaling can provide better experience for users.

4 Conclusion

In this chapter, we have first proposed a bimanual touch-based manipulation technique which is called Constraint Menu for TabletPCs. To manipulate the selected object, users should first use the thumb of the NDH to specify a manipulation constraint, and then use the DH to perform a gesture to specify the manipulation mode and control the transformation. We have also proposed an interface which is called TouchSketch to simplify 3D model sketching on TabletPCs. This interface redefines interaction methods for a set of common-used 3D modeling functions, such as object duplication and face extrusion. We have conducted a user study to evaluate the performance of Constraint Menu by comparing it with two state-of-the-art techniques. The results show that our technique has better performance regarding both efficiency and fluency than the existing techniques.

Besides Constraint Menu, we have also proposed two other manipulation techniques which are called Classic3D and Single3D. Based on the utilization of identified touch inputs, each technique provides a set of manipulation gestures which can be performed using only the DH. Users can first use the index, middle or ring finger to specify the manipulation constraint, and then perform a specific gesture to trigger object manipulation. To investigate the learnability and usability of Classic3D and Single3D, we have conducted a user study with novice users, in order to compare the techniques with Constraint Menu. The results indicate that although Classic3D requires significantly more training time than Constraint Menu, participants could memorize and recall gestures of Classic3D and Single3D without difficulties. Regarding the efficiency, Constraint Menu outperforms Classic3D and Single3D. The performance gap mainly came from object scaling. In the future, we plan to evaluate the techniques with a group of expert users of desktop modeling software. This can show whether expert users can learn our techniques more quickly and accomplish the manipulation tasks with higher efficiency than novice users. In the current study, participants were asked to control only one DOF in each manipulation task. In the future, we will evaluate the performance of our techniques in a more realistic scenario where several DOF are combined. In addition, in some touch-based applications, such as 123D Design, when the fingers are pressed on the screen without touching an object, users can drag the fingers to control the camera. Because gestures of Classic3D and Single3D can be made anywhere on the screen to control the object, we have to propose a novel camera navigation technique. For instance, the NDH or motions of the mobile device could be used for efficient camera navigation. All these improvements can permit to have a more efficient interaction technique for 3D objects manipulation on TabletTCs.

Conclusion and perspectives

Conclusion and perspectives

1 General conclusion

In this thesis, our research work focuses on proposing natural interaction techniques to facilitate object selection and manipulation in virtual environments for mobile devices.

Natural User Interfaces (NUIs) have gained momentum during the last decade and are considered as the next generation of User Interfaces. Indeed, they are expected to provide a more intuitive and seamless interaction experience than traditional UIs by taking advantage of learnt skills of human beings. Nowadays, mobile devices have already become essential tools for both work and entertainment applications. Since the emergence of touchscreens and many other embedded devices, a larger design space now exists and many interaction modalities can be used to enhance mobile interaction. Because the computational power of mobile devices is improving year after year, it is now possible to interact with 3D virtual environments on mobile devices. However, because mobile devices have their own characteristics, traditional interaction techniques, designed for desktop computers, cannot be directly applied for mobile usage. It is then necessary to propose new interaction techniques to support mobile 3D interaction.

For mobile devices such as Smartphones and TabletPCs, interaction mainly relies on the touch-based paradigm. On the one hand, natural interaction experience can be provided by allowing users to select and manipulate the contents directly. On the other hand, both the precision and efficiency can be degraded by the fingertip occlusion problem. Moreover, fine-grained manipulation requires specifying both the constraint and the manipulation mode. However, because the vocabulary of touch-based gestures is limited, there is a lack of mechanisms to switch between the different operations seamlessly.

For mobile devices such as the Oculus Rift, users can enter an immersive virtual environment and interact with the virtual content using freehand gestures in air. However, due to the lack of physical feedback, it is difficult to use freehand gestures to make precise inputs. In addition, compared to using handheld devices, freehand interaction has no access to physical buttons. Thus, there is a need to design a set of gestures to trigger different functions. Compared to touch-based gestures, it is more difficult to extract meaningful freehand gestures from arbitrary hand movements because there are no explicit delimiters.

To simplify selection and manipulation tasks for mobile usage, the main contributions of this thesis can be summarized as follows:

(1) We have designed LayerStroke and LayerMenu, two selection techniques that are designed to make precise selection on Smartphones. To reduce the influence of the cluster density, our techniques divide the targets space into different layers. Users

should first specify the layer in which the target is located, and then select the target from its belonging layer. LayerStroke allows drawing a stroke on the screen to specify a layer while LayerMenu permits calling out a menu for layer selection. In each layer, Voronoi tessellation is generated to decompose the space and an object can be selected by tapping its corresponding tile. Our algorithm ensures that in each layer, the tile of each object is large enough to be simply selected. An experimental study was conducted to compare our techniques with a recent state-of-the-art acquisition technique regarding the selection speed and accuracy. Results show that our techniques decrease both the selection time and errors rate.

(2) We have designed a freehand technique which is called HorizontalDragger for object selection at a distance. When the cluster density in the proximity of the cursor is low, our techniques can be used as the Bubble Cursor technique to select the target rapidly. However, if the density is higher than a threshold value and objects around the cursor are of small size, our technique can automatically set a region of interest and allow dragging the hand horizontally, to specify the target from all the objects covered by the region. Because the effective width of each potential target is set to a constant value, the user can switch the focus in the region of interest in a predictable way. We have conducted an experimental study to compare our technique with two state-of-the-art techniques. Although our technique did not provide the shortest completion time, it significantly decreased the errors rate.

(3) To facilitate constrained manipulation on TabletPCs, we have designed a bimanual technique which is called Constraint Menu. This technique is developed based on the asymmetrical bimanual model (Guiard, 1987). One advantage of this technique is that the non-dominant hand can be used without changing the holding posture. Users can use the non-dominant hand to specify the constraint. Then, they use the dominant hand to perform a specific gesture anywhere on the screen to specify the operation mode and control the transformation. We have made a controlled user study to compare the performance of this technique with two state-of-the-art techniques. The results show that our technique outperforms a Widget-based technique regarding both efficiency and fluency. To further explore the interaction possibilities, we have also designed two single-handed manipulation techniques, Classic3D and Single3D. With the help of identified touch inputs, these two techniques map the index, middle and ring fingers to the x, y and z axes of the coordinate system. Users can first press one or more axis-fingers to specify the constraint, and then use the same hand to translate, rotate or scale the object with respect to the selected constraint. Through a user study, we have found that these two techniques can be learnt easily after a short period of training. Regarding the efficiency, these two techniques have the same level of performance as Constraint Menu for translation and rotation, but require more time for scaling objects.

2 Design implications

This thesis permits to draw some recommendations for the design of natural experience of

selection and manipulation tasks in virtual environments.

For object selection, to design natural selection techniques, designers should consider several aspects:

(1) It is recommended to design selection techniques using gestures which are easy to perform and that can be controlled accurately. For freehand interaction, they should consider designing gestures which take use of small and dexterous muscles.

(2) Try to reduce the selection difficulty by applying different strategies, such as object expansion and adjusting of the CD ratio. The selection technique should be tolerant to imprecise inputs.

(3) Providing appropriate visual guides simplifies significantly the selection task. With the help of visual cues, users can understand the environment more easily, and they can also refine the selection more precisely.

For object manipulation, to design natural manipulation techniques, designers should consider several aspects:

(1) To reduce the cognitive effort required for learning manipulation techniques, the metaphors should be selected carefully. A metaphor that corresponds well to the essence of the manipulation tasks would facilitate the learning and the understanding of the technique.

(2) If the techniques can be developed based on other techniques and metaphors which people are already familiar with, it will provide a more natural experience.

(3) It is recommended to embed the setting of the manipulation mode, the constraint specifying and the control of the transformation into only one command. Therefore, the manipulation experience becomes more seamless and it also helps users to concentrate on their work.

(4) If the camera navigation is supported, it is necessary to consider whether the performance of the manipulation would be affected by some camera perspectives.

3 Limitations and research perspectives

Based on our work in this thesis, we have some long-term perspectives for future developments:

(1) **Propose selection techniques for selection of 3D objects**: In this thesis, we have only proposed techniques for selection of 2D objects. However, selection of 3D objects can be more complicated. First, in a 3D environment, the visual size of the object is not only determined by its real size, but also affected by its distance to the camera. A large object can have a very small visual size when it is located far away

from the camera. Second, in 3D environments it is always permitted to move the camera perspective. Changing the camera viewpoint can also affect the selection performance. For example, after rotating the camera, an object which is occluded in a previous viewpoint can become visually exposed in the next viewpoint. Thus, we plan to propose a new selection technique which will fit better the requirements of 3D environments. We think it worth exploring how to simplify object selection with the help of camera navigation. One possible option to explore is to let the system anticipate the selection intention of the user. Therefore, the camera can be manipulated automatically to provide an appropriate viewpoint for selection. Another possible solution is to provide the possibility of controlling the camera manually. If necessary, users can trigger the camera navigation to adjust the view for selection.

(2) **Study freehand selection in immersive 3D environments:** Although we have proposed HorizontalDragger for object selection at a distance, this technique is implemented in a fixed interaction environment in our work. In the future, we want to improve the design of HorizontalDragger and study its performance in immersive 3D environments. We plan to use a Head Mounted Display to display the immersive environments and use the Leap Motion Controller to track the hands movements. Instead of studying the performance of selection independently, we also plan to investigate object manipulation. Because the performance of manipulation tasks depends on that of the selection tasks, investigating freehand selection and manipulation together will help evaluate the selection technique in a more comprehensive way.

(3) **Use freehand gestures in air to enhance tactile interaction:** In the literature, some studies have explored using freehand gestures performed above the touchscreen to provide more interaction possibilities (Marquardt et al., 2011a; De Araújo et al., 2013). Besides using freehand gestures to make non-constrained manipulation, we want to explore how hand movements in air can be used for constrained manipulation. One possible solution is to design new manipulators for freehand manipulation. Because it is difficult to control the hand movements precisely, the manipulators should have large interactive regions and should be tolerant to hand trembling. Another solution is to use touch inputs made by the non-dominant hand to set the constraint and use freehand movements of the dominant hand to control the transformation. Besides object manipulation, we also plan to explore using freehand gestures for other functions, such as camera navigation. In addition, freehand gestures can also be used as shortcuts for other editing functions. This can help saving the limited screen space of the TabletPCs.

(4) **Combine camera navigation and object manipulation:** Through the evaluation of Constraint Menu, we have found that the performance of object manipulation is affected by the camera viewpoint. Ortega has proposed a technique to switch the viewpoint automatically to simplify 3D object position (Ortega, 2013). Inspired by this work, we have designed a technique which allows transforming the camera to a principal face of the coordinate system and then switching the perspective view to the

orthographic view. After that, the object can be manipulated with respect to the selected plane. Our technique allows users to choose one of the three principal planes determined by the principal axes. We want to examine whether object manipulation can be simplified and accelerated with the help of camera navigation.

(5) **Sketching interface:** Although we have proposed an interface to support sketching work on TabletPCs, our interface only allows users to sketch 3D models using simple 3D primitives. In the future, we plan to provide more advanced functionalities so that users can realize their concepts of design more freely on mobile devices.

References

Reference

Andujar, C., & Argelaguet, F. (2007). Anisomorphic ray-casting manipulation for interacting with 2D GUIs. Computers & Graphics, 31(1), 15-25.

Anthony, L., & Wobbrock, J. O. (2012). $ N-protractor: a fast and accurate multistroke recognizer. Paper presented at the Proceedings of Graphics Interface 2012.

Appert, C., & Zhai, S. (2009). Using strokes as command shortcuts: cognitive benefits and toolkit support. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Argelaguet, F., & Andujar, C. (2013). A survey of 3D object selection techniques for virtual environments. Computers & Graphics, 37(3), 121-136.

Au, O. K. C., Su, X., & Lau, R. W. (2014). LinearDragger: a linear selector for one-finger target acquisition. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Au, O. K. C., & Tai, C. L. (2010). Multitouch finger registration and its applications. Paper presented at the Proceedings of the 22nd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction.

Au, O. K. C., Tai, C. L., & Fu, H. (2012). Multitouch gestures for constrained transformation of 3d objects. Paper presented at the Computer Graphics Forum.

Balakrishnan, R., Baudel, T., Kurtenbach, G., & Fitzmaurice, G. (1997). The Rockin'Mouse: integral 3D manipulation on a plane. Paper presented at the Proceedings of the ACM SIGCHI Conference on Human factors in computing systems.

Bally, G., Müller, J., Rohs, M., Wigdor, D., & Kratz, S. (2012). ShoeSense: A New Perspective on Hand Gestures and Wearable Applications. Paper presented at the Proc. CHI.

Bartindale, T., Harrison, C., Olivier, P., & Hudson, S. E. (2011). SurfaceMouse: supplementing multi-touch interaction with a virtual mouse. Paper presented at the Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction.

Baudel, T., & Beaudouin-Lafon, M. (1993). Charade: remote control of objects using free-hand gestures. Communications of the ACM, 36(7), 28-35.

Baudisch, P., & Chu, G. (2009). Back-of-device interaction allows creating very small touch devices. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Baudisch, P., Zotov, A., Cutrell, E., & Hinckley, K. (2008). Starburst: a target expansion algorithm for non-uniform target distributions. Paper presented at the Proceedings of the working conference on Advanced visual interfaces.

Benko, H. (2009). Beyond flat surface computing: challenges of depth-aware and curved interfaces. Paper presented at the Proceedings of the 17th ACM international conference on Multimedia.

Benko, H., & Wilson, A. (2009). DepthTouch: Using depth-sensing camera to enable freehand interactions on and above the interactive surface. Paper presented at the Proceedings of the IEEE workshop on tabletops and interactive surfaces.

Benko, H., Wilson, A. D., & Baudisch, P. (2006). Precise selection techniques for multi-touch screens. Paper presented at the Proceedings of the SIGCHI conference on Human Factors in computing systems.

Bhalla, M. R., & Bhalla, A. V. (2010). Comparative study of various touchscreen technologies. International Journal of Computer Applications, 6(8), 12-18.

Bi, X., Li, Y., & Zhai, S. (2013). FFitts law: modeling finger touch with fitts' law. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Bier, E. A. (1987). Skitters and jacks: interactive 3D positioning tools. Paper presented at the Proceedings of the 1986 workshop on Interactive 3D graphics.

Blake, A., & Isard, M. (1994). 3D position, attitude and shape input using video tracking of hands and lips. Paper presented at the Proceedings of the 21st annual conference on Computer graphics and interactive techniques.

Blake, J. (2011). Natural User Interfaces in. NET: WPF 4, Surface 2, and Kinect. Manning.

Blanch, R., Guiard, Y., & Beaudouin-Lafon, M. (2004). Semantic pointing: improving target acquisition with control-display ratio adaptation. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.

Bogdan, N., Grossman, T., & Fitzmaurice, G. (2014). Hybridspace: Integrating 3d freehand input and stereo viewing into traditional desktop applications. Paper presented at the 3D User Interfaces (3DUI), 2014 IEEE Symposium on.

Boland, D., & Murray-Smith, R. (2013). Finding my beat: Personalised rhythmic filtering for mobile music interaction. Paper presented at the Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services.

Bolt, R. A. (1980). "Put-that-there": Voice and gesture at the graphics interface (Vol. 14): ACM.

Bondy, M. D., Georganas, N. D., Petriu, E. M., Petriu, D. C., Cordea, M. D., & Whalen, T. E. (2001). Model-based face and lip animation for interactive virtual reality applications. Paper presented at the Proceedings of the ninth ACM international conference on Multimedia.

Bonnet, D., Appert, C., & Beaudouin-Lafon, M. (2013). Extending the vocabulary of touch events with ThumbRock. Paper presented at the Proceedings of Graphics Interface 2013.

Boring, S., Ledo, D., Chen, X. A., Marquardt, N., Tang, A., & Greenberg, S. (2012). The fat thumb: using the thumb's contact size for single-handed mobile interaction. Paper presented at the Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services.

Bowman, D., Johnson, D. B., & Hodges, L. F. (2001). Testbed evaluation of virtual environment interaction techniques. Presence, 10(1), 75-95.

Bowman, D. A., Kruijff, E., LaViola Jr, J. J., & Poupyrev, I. (2004). 3D user interfaces: theory and practice: Addison-Wesley.

Bragdon, A., & Ko, H. S. (2011). Gesture select: acquiring remote targets on large displays without pointing. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Bragdon, A., Nelson, E., Li, Y., & Hinckley, K. (2011). Experimental analysis of touch-screen gesture designs in mobile environments. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Bregler, C., Covell, M., & Slaney, M. (1997). Video rewrite: Driving visual speech with audio. Paper presented at the Proceedings of the 24th annual conference on Computer graphics and interactive techniques.

Brouet, R., Blanch, R., & Cani, M.-P. (2013). Understanding Hand Degrees of Freedom and Natural Gestures for 3D Interaction on Tabletop Human-Computer Interaction. INTERACT 2013 (pp. 297-314): Springer.

Busso, C., & Narayanan, S. S. (2007). Interrelation between speech and facial gestures in emotional utterances: a single subject study. Audio, Speech, and Language Processing, IEEE Transactions on, 15(8), 2331-2347.

Butler, A., Izadi, S., & Hodges, S. (2008). SideSight: multi-touch interaction around small devices. Paper presented at the Proceedings of the 21st annual ACM symposium on User interface software and technology.

Buxton, B. (2010). CES 2010: NUI with Bill Buxton. Microsoft Research.

Camastra, F., & De Felice, D. (2013). LVQ-based hand gesture recognition using a data glove Neural Nets and Surroundings (pp. 159-168): Springer.

Cao, X., Wilson, A. D., Balakrishnan, R., Hinckley, K., & Hudson, S. E. (2008). Shapetouch: Leveraging contact shape on interactive surfaces. Paper presented at the Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on.

Card, S. K., Mackinlay, J. D., & Robertson, G. G. (1991). A morphological analysis of the design space of input devices. ACM Transactions on Information Systems (TOIS), 9(2), 99-122.

Cashion, J., Wingrave, C., & LaViola Jr, J. J. (2012). Dense and dynamic 3D selection for game-based virtual environments. Visualization and Computer Graphics, IEEE Transactions on, 18(4), 634-642.

Casiez, G., Roussel, N., & Vogel, D. (2012). 1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Chang, Y., L'Yi, S., Koh, K., & Seo, J. (2015). Understanding Users' Touch Behavior on Large Mobile Touch-Screens and Assisted Targeting by Tilting Gesture. Paper presented at the Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems.

Chapuis, O., Labrune, J.-B., & Pietriga, E. (2009). DynaSpot: speed-dependent area cursor. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Chellali, A., Jourdan, F., Dumas, C. (2013) VR4D: An Immersive and Collaborative Experience to Improve the Interior Design Process of Limited Spaces. In the proceedings of the 5th Joint Virtual Reality Conference, JVRC 2013, Paris, France, pp. 61-65

Chen, C., Perrault, S. T., Zhao, S., & Ooi, W. T. (2014a). BezelCopy: an efficient cross-application copy-paste technique for touchscreen smartphones. Paper

presented at the Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces.

Chen, M., Mountford, S. J., & Sellen, A. (1988). A study in interactive 3-D rotation using 2-D control devices. Paper presented at the ACM SIGGRAPH Computer Graphics.

Chen, X. A., Grossman, T., Wigdor, D. J., & Fitzmaurice, G. (2014b). Duet: exploring joint interactions on a smart phone and a smart watch. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Chen, X. A., Marquardt, N., Tang, A., Boring, S., & Greenberg, S. (2012). Extending a mobile device's interaction space through body-centric interaction. Paper presented at the Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services.

Chen, X. A., Schwarz, J., Harrison, C., Mankoff, J., & Hudson, S. E. (2014c). Air+ touch: interweaving touch & in-air gestures. Paper presented at the Proceedings of the 27th annual ACM symposium on User interface software and technology.

Cho, I., Wang, X., & Wartell, Z. J. (2013). HyFinBall: a two-handed, hybrid 2D/3D desktop VR interface for multi-dimensional visualization. Paper presented at the IS&T/SPIE Electronic Imaging.

Cohé, A., Dècle, F., & Hachet, M. (2011). tBox: a 3d transformation widget designed for touch-screens. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Cohen, P. R., Johnston, M., McGee, D., Oviatt, S., Pittman, J., Smith, I., Liang, Chen., & Clow, J. (1997). QuickSet: Multimodal interaction for distributed applications. Paper presented at the Proceedings of the fifth ACM international conference on Multimedia.

Colley, A., & Häkkilä, J. (2014). Exploring finger specific touch screen interaction for mobile phone user interfaces. Paper presented at the Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: the Future of Design.

Conner, B. D., Snibbe, S. S., Herndon, K. P., Robbins, D. C., Zeleznik, R. C., & Van Dam, A. (1992). Three-dimensional widgets. Paper presented at the Proceedings of the 1992 symposium on Interactive 3D graphics.

De Araújo, B. R., Casiez, G., Jorge, J. A., & Hachet, M. (2013). Mockup Builder: 3D modeling on and above the surface. Computers & Graphics, 37(3), 165-178.

Delamare, W., Coutrix, C., & Nigay, L. (2013). Mobile pointing task in the physical world: balancing focus and performance while disambiguating. Paper presented at the Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services.

Djelil, F., Otmane, S., & Wu, S. (2013). Apport des NUIs pour les Applications de Réalité Virtuelle et Augmentée: État de l'Art. Les 8ème journées de l'Association Française de Réalité Virtuelle, AFRV 2013, 13-20.

Ding, W., Chen, P., Al-Mubaid, H., & Pomplun, M. (2009). A Gaze-Controlled Interface to Virtual Reality Applications for Motor-and Speech-Impaired Users. HCI International, San Diego, CA.

Döllner, J., & Hinrichs, K. (1997). Object-Oriented 3D Modelling, Animation and Interaction. Journal of Visualization and Computer Animation, 8(1), 33-64.

Du, Y., Ren, H., Pan, G., & Li, S. (2011). Tilt & touch: mobile phone for 3D interaction. Paper presented at the Proceedings of the 13th international conference on Ubiquitous computing.

Fares, R., Fang, S., & Komogortsev, O. (2013, April). Can we beat the mouse with MAGIC?. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 1387-1390). ACM.

Fekete, J. D., Elmqvist, N., & Guiard, Y. (2009). Motion-pointing: target selection using elliptical motions. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.

Findlater, L., Jansen, A., Shinohara, K., Dixon, M., Kamb, P., Rakita, J., & Wobbrock, J. O. (2010). Enhanced area cursors: reducing fine pointing demands for people with motor impairments. Paper presented at the Proceedings of the 23nd annual ACM symposium on User interface software and technology.

Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. Journal of experimental psychology, 47(6), 381.

Forsberg, A., Herndon, K., & Zeleznik, R. (1996). Aperture based selection for immersive virtual environments. Paper presented at the Proceedings of the 9th annual ACM symposium on User interface software and technology.

Frees, S., Kessler, G. D., & Kay, E. (2007). PRISM interaction for enhancing control in immersive virtual environments. ACM Transactions on Computer-Human Interaction (TOCHI), 14(1), 2.

Friedman, D., Leeb, R., Antley, A., Garau, M., Guger, C., Keinrath, C., Steed, Anthony., Pfurtscheller, Gert., & Slater, M. (2004). Navigating virtual reality by thought:

First steps. Paper presented at the Proceedings of the 7th Annual International Workshop on Presence.

Fröhlich, B., Tramberend, H., Beers, A., Agrawala, M., & Baraff, D. (2000). Physically-based manipulation on the responsive workbench. Paper presented at the Virtual Reality, 2000. Proceedings. IEEE.

Genest, A. M., Gutwin, C., Tang, A., Kalyn, M., & Ivkovic, Z. (2013). KinectArms: a toolkit for capturing and displaying arm embodiments in distributed tabletop groupware. Paper presented at the Proceedings of the 2013 conference on Computer supported cooperative work.

Ghomi, E., Bau, O., Mackay, W., & Huot, S. (2010). Conception et apprentissage des interactions tactiles: le cas des postures multidoigts. In FITG'10: French workshop on tactile and gestural interaction.

Ghomi, E., Faure, G., Huot, S., Chapuis, O., & Beaudouin-Lafon, M. (2012). Using rhythmic patterns as an input method. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Ghomi, E., Huot, S., Bau, O., Beaudouin-Lafon, M., & Mackay, W. E. (2013). Arpège: Learning multitouch chord gestures vocabularies. Paper presented at the Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces.

Goel, M., Wobbrock, J., & Patel, S. (2012). GripSense: using built-in sensors to detect hand posture and pressure on commodity mobile phones. Paper presented at the Proceedings of the 25th annual ACM symposium on User interface software and technology.

Goguey, A., Casiez, G., Pietrzak, T., Vogel, D., & Roussel, N. (2014a). Adoiraccourcix: Sélection de Commandes sur Écrans Tactiles Multi-Points par Identification des Doigts. Paper presented at the IHM'14, 26e conférence francophone sur l'Interaction Homme-Machine.

Goguey, A., Casiez, G., Vogel, D., Chevalier, F., Pietrzak, T., & Roussel, N. (2014b). A three-step interaction pattern for improving discoverability in finger identification techniques. Paper presented at the Proceedings of the adjunct publication of the 27th annual ACM symposium on User interface software and technology.

Grossman, T., & Balakrishnan, R. (2005). The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.

Grossman, T., & Balakrishnan, R. (2006). The design and evaluation of selection techniques for 3D volumetric displays. Paper presented at the Proceedings of the 19th annual ACM symposium on User interface software and technology.

Grossman, T., Wigdor, D., & Balakrishnan, R. (2004). Multi-finger gestural interaction with 3d volumetric displays. Paper presented at the Proceedings of the 17th annual ACM symposium on User interface software and technology.

Guiard, Y. (1987). Asymmetric division of labor in human skilled bimanual action: The kinematic chain as a model. Journal of motor behavior, 19(4), 486-517.

Guimbretière, F., & Nguyen, C. (2012). Bimanual marking menu for near surface interactions. Paper presented at the Proceedings of the SIGCHI conference on human factors in computing systems.

Gutwin, C. (2002). Improving focus targeting in interactive fisheye views. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.

Hachet, M., Bossavit, B., Cohé, A., & de la Rivière, J. B. (2011). Toucheo: multitouch and stereo combined in a seamless workspace. Paper presented at the Proceedings of the 24th annual ACM symposium on User interface software and technology.

Hachet, M., Guitton, P., & Reuter, P. (2003). The cat for efficient 2d and 3d interaction as an alternative to mouse adaptations. Paper presented at the Proceedings of the ACM symposium on Virtual reality software and technology.

Hancock, M., Carpendale, S., & Cockburn, A. (2007). Shallow-depth 3d interaction: design and evaluation of one-, two-and three-touch techniques. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.

Hancock, M., Ten Cate, T., & Carpendale, S. (2009). Sticky tools: full 6DOF force-based interaction for multi-touch tables. Paper presented at the Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces.

Hancock, M. S., Vernier, F. D., Wigdor, D., Carpendale, S., & Shen, C. (2006). Rotation and translation mechanisms for tabletop interaction. Paper presented at the Horizontal Interactive Human-Computer Systems, 2006. TableTop 2006. First IEEE International Workshop on.

Harrison, C., Benko, H., & Wilson, A. D. (2011). OmniTouch: wearable multitouch interaction everywhere. Paper presented at the Proceedings of the 24th annual ACM symposium on User interface software and technology.

Harrison, C., & Hudson, S. E. (2009). Abracadabra: wireless, high-precision, and unpowered finger input for very small mobile devices. Paper presented at the Proceedings of the 22nd annual ACM symposium on User interface software and technology.

Harrison, C., Schwarz, J., & Hudson, S. E. (2011). TapSense: enhancing finger interaction on touch surfaces. Paper presented at the Proceedings of the 24th annual ACM symposium on User interface software and technology.

Harrison, C., Xiao, R., Schwarz, J., & Hudson, S. E. (2014). TouchTools: leveraging familiarity and skill with physical tools to augment touch interaction. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Hasenfratz, J. M., Lapierre, M., & Sillion, F. X. (2004). A real-time system for full body interaction with virtual worlds. Paper presented at the Eurographics Symposium on Virtual Environments.

Hayashi, E., Maas, M., & Hong, J. I. (2014). Wave to me: user identification using body lengths and natural gestures. Paper presented at the Proceedings of the 32nd annual ACM conference on Human factors in computing systems.

Heo, S., Gu, J., & Lee, G. (2014). Expanding touch input vocabulary by using consecutive distant taps. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Heo, S., & Lee, G. (2011a). Force gestures: augmenting touch screen gestures with normal and tangential forces. Paper presented at the Proceedings of the 24th annual ACM symposium on User interface software and technology.

Heo, S., & Lee, G. (2011b). Forcetap: extending the input vocabulary of mobile touch screens by adding tap gestures. Paper presented at the Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services.

Hilliges, O., Izadi, S., Wilson, A. D., Hodges, S., Garcia-Mendoza, A., & Butz, A. (2009). Interactions in the air: adding further depth to interactive tabletops. Paper presented at the Proceedings of the 22nd annual ACM symposium on User interface software and technology.

Hilliges, O., Kim, D., Izadi, S., Weiss, M., & Wilson, A. (2012). HoloDesk: direct 3d interactions with a situated see-through display. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Hinckley, K., Sinclair, M., Hanson, E., Szeliski, R., & Conway, M. (1999). The videomouse: a camera-based multi-degree-of-freedom input device. Paper presented at the Proceedings of the 12th annual ACM symposium on User interface software and technology.

Hinckley, K., & Song, H. (2011). Sensor synaesthesia: touch in motion, and motion in touch. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Holz, C., & Baudisch, P. (2010). The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Isenberg, T., & Hancock, M. (2012). Gestures vs. Postures:'Gestural'Touch Interaction in 3D Environments. Paper presented at the Proceedings of the CHI Workshop on"The 3rd Dimension of CHI: Touching and Designing 3D User Interfaces"(3DCHI 2012, May 5, 2012, Austin, TX, USA).

Ishii, H. (2008). Tangible bits: beyond pixels. Paper presented at the Proceedings of the 2nd international conference on Tangible and embedded interaction.

Jacobson, A., Panozzo, D., Glauser, O., Pradalier, C., Hilliges, O., & Sorkine-Hornung, O. (2014). Tangible and modular input device for character articulation. Paper presented at the Proceedings of the adjunct publication of the 27th annual ACM symposium on User interface software and technology.

Jacoby, R. H., Ferneau, M., & Humphries, J. (1994). Gestural interaction in a virtual environment. Paper presented at the IS&T/SPIE 1994 International Symposium on Electronic Imaging: Science and Technology.

Jain, M., & Balakrishnan, R. (2012). User learning and performance with bezel menus. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Jankowski, J., & Hachet, M. (2013). A survey of interaction techniques for interactive 3D environments. Paper presented at the Eurographics 2013-STAR.

Jones, B., Sodhi, R., Forsyth, D., Bailey, B., & Maciocci, G. (2012). Around device interaction for multiscale navigation. Paper presented at the Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services.

Karlson, A. K., & Bederson, B. B. (2007). ThumbSpace: generalized one-handed input for touchscreen-based mobile devices. Human-Computer Interaction–INTERACT 2007 (pp. 324-338): Springer.

Kasahara, S., Niiyama, R., Heun, V., & Ishii, H. (2013). exTouch: spatially-aware embodied manipulation of actuated objects mediated by augmented reality. Paper presented at the Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction.

Käser, D. P., Agrawala, M., & Pauly, M. (2011). FingerGlass: efficient multiscale interaction on multitouch screens. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Kim, D., Hilliges, O., Izadi, S., Butler, A. D., Chen, J., Oikonomidis, I., & Olivier, P. (2012). Digits: freehand 3D interactions anywhere using a wrist-worn gloveless sensor. Paper presented at the Proceedings of the 25th annual ACM symposium on User interface software and technology.

Kin, K., Miller, T., Bollensdorff, B., DeRose, T., Hartmann, B., & Agrawala, M. (2011). Eden: a professional multitouch tool for constructing virtual organic environments. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Kitamura, Y., Higashi, T., Masaki, T., & Kishino, F. (1999). Virtual chopsticks: Object manipulation using multiple exact interactions. Paper presented at the Virtual Reality, 1999. Proceedings., IEEE.

Klompmaker, F., Nebe, K., & Fast, A. (2012). dSensingNI: a framework for advanced tangible interaction using a depth camera. Paper presented at the Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction.

Knoedel, S., & Hachet, M. (2011). Multi-touch rst in 2d and 3d spaces: Studying the impact of directness on user performance. Paper presented at the 3D User Interfaces (3DUI), 2011 IEEE Symposium on.

König, W. A., Gerken, J., Dierdorf, S., & Reiterer, H. (2009). Adaptive Pointing–Design and Evaluation of a Precision Enhancing Technique for Absolute Pointing Devices. Human-Computer Interaction–INTERACT 2009 (pp. 658-671): Springer.

Kopper, R., Bacim, F., & Bowman, D. (2011). Rapid and accurate 3D selection by progressive refinement. Paper presented at the 3D User Interfaces (3DUI), 2011 IEEE Symposium on.

Kopper, R., Bowman, D. A., Silva, M. G., & McMahan, R. P. (2010). A human motor behavior model for distal pointing tasks. International Journal of Human-Computer Studies, 68(10), 603-615.

Kratz, S., Chiu, P., & Back, M. (2013). Pointpose: finger pose estimation for touch input on mobile devices using a depth sensor. Paper presented at the Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces.

Kratz, S., & Rohs, M. (2009). HoverFlow: expanding the design space of around-device interaction. Paper presented at the Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services.

Kratz, S., Rohs, M., Guse, D., Müller, J., Bailly, G., & Nischt, M. (2012). PalmSpace: continuous around-device gestures vs. multitouch for 3D rotation tasks on mobile devices. Paper presented at the Proceedings of the International Working Conference on Advanced Visual Interfaces.

Kulshreshth, A., & LaViola Jr, J. J. (2014). Exploring the usefulness of finger-based 3D gesture menu selection. Paper presented at the Proceedings of the 32nd annual ACM conference on Human factors in computing systems.

Kumar, P., Verma, J., & Prasad, S. (2012). Hand data glove: A wearable real-time device for human-computer interaction. International Journal of Advanced Science and Technology, 43.

Wei, L., & Hu, H. (2011). Towards multimodal human-machine interface for hands-free control: A survey.

Lécuyer, A., Lotte, F., Reilly, R. B., Leeb, R., Hirose, M., & Slater, M. (2008). Brain-computer interfaces, virtual reality, and videogames. Computer(10), 66-72.

Lee, J. Y., Park, H. M., Lee, S. H., Shin, S. H., Kim, T. E., & Choi, J. S. (2014). Design and implementation of an augmented reality system using gaze interaction. Multimedia Tools and Applications, 68(2), 265-280.

Leeb, R., Friedman, D., Slater, M., & Pfurtscheller, G. (2012). A tetraplegic patient controls a wheelchair in virtual reality. Paper presented at the BRAINPLAY 07 Brain-Computer Interfaces and Games Workshop at ACE (Advances in Computer Entertainment) 2007.

Lepinski, G. J., Grossman, T., & Fitzmaurice, G. (2010). The design and evaluation of multitouch marking menus. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Li, Y. (2010). Gesture search: a tool for fast mobile data access. Paper presented at the Proceedings of the 23nd annual ACM symposium on User interface software and technology.

Liang, H. N., Williams, C., Semegen, M., Stuerzlinger, W., & Irani, P. (2012). User-defined surface+ motion gestures for 3d manipulation of objects at a

distance through a mobile device. Paper presented at the Proceedings of the 10th asia pacific conference on Computer human interaction.

Liang, J., & Green, M. (1994). JDCAD: A highly interactive 3D modeling system. Computers & Graphics, 18(4), 499-506.

Lim, C. J., & Kim, D. (2012). Development of gaze tracking interface for controlling 3D contents. Sensors and Actuators A: Physical, 185, 151-159.

Liu, J., Au, O. K. C., Fu, H., & Tai, C. L. (2012). Two-Finger Gestures for 6DOF Manipulation of 3D Objects. Paper presented at the Computer Graphics Forum.

Lotte, F., Lécuyer, A., Renard, Y., Lamarche, F., & Arnaldi, Bruno. (2006). Classification de données cérébrales par système d'inférence flou pour l'utilisation d'interfaces cerveau-ordinateur en réalité virtuelle. Paper presented at the 1ères journées de l'AFRV.

MacKenzie, I. S. (1992). Fitts' law as a research and design tool in human-computer interaction. Human-computer interaction, 7(1), 91-139.

Malik, S., Ranjan, A., & Balakrishnan, R. (2005). Interacting with large displays from a distance with vision-tracked multi-finger gestural input. Paper presented at the Proceedings of the 18th annual ACM symposium on User interface software and technology.

Mann, S. (2001). Intelligent image processing: John Wiley & Sons, Inc.

Marquardt, N., Jota, R., Greenberg, S., & Jorge, J. A. (2011a). The continuous interaction space: interaction techniques unifying touch and gesture on and above a digital surface. Human-Computer Interaction–INTERACT 2011 (pp. 461-476): Springer.

Marquardt, N., Kiemer, J., Ledo, D., Boring, S., & Greenberg, S. (2011b). Designing user-, hand-, and handpart-aware tabletop interactions with the TouchID toolkit. Paper presented at the Proceedings of the ACM international conference on interactive tabletops and surfaces.

Martinet, A., Casiez, G., & Grisoni, L. (2010a). The design and evaluation of 3d positioning techniques for multi-touch displays. Paper presented at the 3D User Interfaces (3DUI), 2010 IEEE Symposium on.

Martinet, A., Casiez, G., & Grisoni, L. (2010b). The effect of dof separation in 3d manipulation tasks with multi-touch displays. Paper presented at the Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology.

McCallum, D. C., & Irani, P. (2009). ARC-Pad: absolute+ relative cursor positioning for large displays with a mobile touchscreen. Paper presented at the Proceedings of the 22nd annual ACM symposium on User interface software and technology.

McGlashan, S., Fraser, N. M., Gilbert, N., Bilange, E., Heisterkamp, P., & Youd, N. J. (1992). Dialogue management for telephone information services. In Proceedings of the International Conference on Applied Language Processing, Trento, Italy.

McGlashan, S. (1995). Speech interfaces to virtual reality. Paper presented at the Proceedings of 2nd International Workshop on Military Applications of Synthetic Environments and Virtual Reality.

Mendes, D. (2011). LTouchIt: LEGO Modeling on Multi-Touch Surfaces. Master's thesis, Instituto Superior Técnico.

Mendes, D., Fonseca, F., Araujo, B., Ferreira, A., & Jorge, J. (2014). Mid-air interactions above stereoscopic interactive tables. Paper presented at the 3D User Interfaces (3DUI), 2014 IEEE Symposium on.

Mendes, D., Lopes, P., & Ferreira, A. (2011). Hands-on interactive tabletop lego application. Paper presented at the Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology.

Meyer, D. E., Abrams, R. A., Kornblum, S., Wright, C. E., & Keith Smith, J. E. (1988). Optimality in human motor performance: ideal control of rapid aimed movements. Psychological review, 95(3), 340.

Meyer, D. E., Keith-Smith, J. E., Kornblum, S., Abrams, R. A., & Wright, C. E. (1990). Speed-accuracy tradeoffs in aimed movements: Toward a theory of rapid voluntary action.

Mine, M., Yoganandan, A., & Coffey, D. (2014). Making VR work: building a real-world immersive modeling application in the virtual world. Paper presented at the Proceedings of the 2nd ACM symposium on Spatial user interaction.

Mossel, A., Venditti, B., & Kaufmann, H. (2013). 3DTouch and HOMER-S: intuitive manipulation techniques for one-handed handheld augmented reality. Paper presented at the Proceedings of the Virtual Reality International Conference: Laval Virtual.

Mott, M. E., & Wobbrock, J. O. (2014). Beating the bubble: using kinematic triggering in the bubble lens for acquiring small, dense targets. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Nancel, M., Chapuis, O., Pietriga, E., Yang, X.-D., Irani, P. P., & Beaudouin-Lafon, M. (2013). High-precision pointing on large wall displays using small handheld devices. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Nancel, M., Pietriga, E., & Beaudouin-Lafon, M. (2011a). Precision pointing for ultra-high-resolution wall displays.

Nancel, M., Wagner, J., Pietriga, E., Chapuis, O., & Mackay, W. (2011b). Mid-air pan-and-zoom on wall-sized displays. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Nielson, G. M., & Olsen Jr, D. R. (1987). Direct manipulation techniques for 3D objects using 2D locator devices. Paper presented at the Proceedings of the 1986 workshop on Interactive 3D graphics.

Nugues, P., Godéreaux, C., El Guedj, P. O., & Revolta, F. (1996). A conversational agent to navigate in virtual worlds. Paper presented at the Proceedings Dialogue Management in Natural Language Systems. Twente Workshop on Language Technology.

O'Hara, K., Sellen, A., & Harper, R. (2011). Embodiment in brain-computer interaction. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Olwal, A., Feiner, S., & Heyman, S. (2008). Rubbing and tapping for precise and rapid selection on touch-screen displays. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Ortega, M. (2013). 3d object position using automatic viewpoint transitions. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 193-196). ACM.

Ouramdane, N., Otmane, S., Davesne, F., & Mallem, M. (2006). FOLLOW-ME: a new 3D interaction technique based on virtual guides and granularity of interaction. Paper presented at the Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications.

Oviatt, S., & Cohen, P. (2000). Perceptual user interfaces: multimodal interfaces that process what comes naturally. Communications of the ACM, 43(3), 45-53.

Paczkowski, P., Dorsey, J., Rushmeier, H., & Kim, M. H. (2014). Paper3D: bringing casual 3D modeling to a multi-touch interface. Paper presented at the Proceedings of the 27th annual ACM symposium on User interface software and technology.

Perelman, G., Serrano, M., Raynal, M., Picard, C., Derras, M., & Dubois, E. (2015). The Roly-Poly Mouse: Designing a Rolling Input Device Unifying 2D and 3D Interaction. Paper presented at the Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems.

Pfeiffer, T. (2008). Towards gaze interaction in immersive virtual reality: Evaluation of a monocular eye tracking set-up. Paper presented at the Virtuelle und Erweiterte Realität-Fünfter Workshop der GI-Fachgruppe VR/AR.

Pierce, J. S., Forsberg, A. S., Conway, M. J., Hong, S., Zeleznik, R. C., & Mine, M. R. (1997). Image plane interaction techniques in 3D immersive environments. Paper presented at the Proceedings of the 1997 symposium on Interactive 3D graphics.

Piumsomboon, T., Clark, A., Billinghurst, M., & Cockburn, A. (2013). User-defined gestures for augmented reality. Human-Computer Interaction–INTERACT 2013 (pp. 282-299): Springer.

Potter, R. L., Weldon, L. J., & Shneiderman, B. (1988). Improving the accuracy of touch screens: an experimental evaluation of three strategies. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.

Poupyrev, I., Billinghurst, M., Weghorst, S., & Ichikawa, T. (1996). The go-go interaction technique: non-linear mapping for direct manipulation in VR. Paper presented at the Proceedings of the 9th annual ACM symposium on User interface software and technology.

Ramos, G., Cockburn, A., Balakrishnan, R., & Beaudouin-Lafon, M. (2007). Pointing lenses: facilitating stylus input through visual-and motor-space magnification. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.

Reisman, J. L., Davidson, P. L., & Han, J. Y. (2009). A screen-space formulation for 2D and 3D direct manipulation. Paper presented at the Proceedings of the 22nd annual ACM symposium on User interface software and technology.

Ren, G., & O'Neill, E. (2013). 3d selection with freehand gesture. Computers & Graphics, 37(3), 101-120.

Rogers, S., Williamson, J., Stewart, C., & Murray-Smith, R. (2011). AnglePose: robust, precise capacitive touch tracking via 3d orientation estimation. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Roth, V., & Turner, T. (2009). Bezel swipe: conflict-free scrolling and multiple selection on mobile touch screen devices. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Roudaut, A., Huot, S., & Lecolinet, E. (2008). TapTap and MagStick: improving one-handed target acquisition on small touch-screens. Paper presented at the Proceedings of the working conference on Advanced visual interfaces.

Roudaut, A., Lecolinet, E., & Guiard, Y. (2009). MicroRolls: expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Ruiz, J., & Li, Y. (2011). DoubleFlip: a motion gesture delimiter for mobile interaction. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Ruiz, J., Li, Y., & Lank, E. (2011). User-defined motion gestures for mobile interaction. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Scheurich, D., & Stuerzlinger, W. (2013). A One-Handed Multi-Touch Method for 3D Rotations. Human-Computer Interaction–INTERACT 2013 (pp. 56-69): Springer.

Schmidt, R., Singh, K., & Balakrishnan, R. (2008). Sketching and composing widgets for 3d manipulation. Paper presented at the Computer Graphics Forum.

Schmidt, R. A., Zelaznik, H., Hawkins, B., Frank, J. S., & Quinn Jr, J. T. (1979). Motor-output variability: a theory for the accuracy of rapid motor acts. Psychological review, 86(5), 415.

Schwarz, J., Marais, C. C., Leyvand, T., Hudson, S. E., & Mankoff, J. (2014). Combining body pose, gaze, and gesture to determine intention to interact in vision-based interfaces. Paper presented at the Proceedings of the 32nd annual ACM conference on Human factors in computing systems.

Seipp, K., & Devlin, K. (2014). BackPat: one-handed off-screen patting gestures. Paper presented at the Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services.

Serrano, M., Lecolinet, E., & Guiard, Y. (2013). Bezel-Tap gestures: quick activation of commands from sleep mode on tablets. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Shoemake, K. (1992). ARCBALL: a user interface for specifying three-dimensional orientation using a mouse. Paper presented at the Graphics Interface.

Song, J., Sörös, G., Pece, F., Fanello, S. R., Izadi, S., Keskin, C., & Hilliges, O. (2014). In-air gestures around unmodified mobile devices. Paper presented at the

Proceedings of the 27th annual ACM symposium on User interface software and technology.

Song, P., Goh, W. B., Hutama, W., Fu, C. W., & Liu, X. (2012). A handle bar metaphor for virtual object manipulation with mid-air interaction. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Spindler, M., Schuessler, M., Martsch, M., & Dachselt, R. (2014). Pinch-drag-flick vs. spatial input: rethinking zoom & pan on mobile displays. Paper presented at the Proceedings of the 32nd annual ACM conference on Human factors in computing systems.

Steinicke, F., Ropinski, T., & Hinrichs, K. (2006). Object selection in virtual environments using an improved virtual pointer metaphor Computer Vision and Graphics (pp. 320-326): Springer.

Sternberger, L. (2006). Interaction en réalité virtuelle. Strasbourg 1.

Strauss, P. S., & Carey, R. (1992). An object-oriented 3D graphics toolkit. Paper presented at the ACM SIGGRAPH Computer Graphics.

Strauss, P. S., Issacs, P., & Shrag, J. (2002). The design and implementation of direct manipulation in 3D. SIGGRAPH Course Notes, 12.

Sturman, D. J., Zeltzer, D., & Pieper, S. (1989). Hands-on interaction with virtual environments. Paper presented at the Proceedings of the 2nd annual ACM SIGGRAPH symposium on User interface software and technology.

Su, X., Au, O. K. C., & Lau, R. W. (2014). The implicit fan cursor: a velocity dependent area cursor. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Sugiura, A., & Koseki, Y. (1998). A user interface using fingerprint recognition: holding commands and data objects on fingers. Paper presented at the Proceedings of the 11th annual ACM symposium on User interface software and technology.

Sun, Q., Lin, J., Fu, C. W., Kaijima, S., & He, Y. (2013). A multi-touch interface for fast architectural sketching and massing. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Tsandilas, T., Appert, C., Bezerianos, A., & Bonnet, D. (2014). Coordination of tilt and touch in one-and two-handed use. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Turner, J., Bulling, A., Alexander, J., & Gellersen, H. (2014, March). Cross-device gaze-supported point-to-point content transfer. In Proceedings of the Symposium on Eye Tracking Research and Applications (pp. 19-26). ACM.

Turner, J., Bulling, A., & Gellersen, H. (2011). Combining gaze with manual interaction to extend physical reach. Paper presented at the Proceedings of the 1st international workshop on pervasive eye tracking & mobile eye-based interaction.

Valli, A. (2005). Notes on natural interaction. http://naturalinteraction.org.

Vanacken, L., Grossman, T., & Coninx, K. (2007). Exploring the effects of environment density and target visibility on object selection in 3D virtual environments. Paper presented at the IEEE Symposium on 3D User Interfaces, 2007. 3DUI'07.

Vanacken, L., Grossman, T., & Coninx, K. (2009). Multimodal selection techniques for dense and occluded 3d virtual environments. International Journal of Human-Computer Studies, 67(3), 237-255.

Vatavu, R. D., Anthony, L., & Wobbrock, J. O. (2012). Gestures as point clouds: a $ P recognizer for user interface prototypes. Paper presented at the Proceedings of the 14th ACM international conference on Multimodal interaction.

Vogel, D., & Balakrishnan, R. (2005). Distant freehand pointing and clicking on very large, high resolution displays. Paper presented at the Proceedings of the 18th annual ACM symposium on User interface software and technology.

Vogel, D., & Baudisch, P. (2007). Shift: a technique for operating pen-based interfaces using touch. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.

Wagner, J., Huot, S., & Mackay, W. (2012). BiTouch and BiPad: designing bimanual interaction for hand-held tablets. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Wagner, J., Lecolinet, E., & Selker, T. (2014). Multi-finger chords for hand-held tablets: Recognizable and memorable. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Wagner, J., Nancel, M., Gustafson, S. G., Huot, S., & Mackay, W. E. (2013). A body-centric design space for multi-surface interaction. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Wang, F., Cao, X., Ren, X., & Irani, P. (2009). Detecting and leveraging finger orientation for interaction with direct-touch surfaces. Paper presented at the

Proceedings of the 22nd annual ACM symposium on User interface software and technology.

Wang, R., Paris, S., & Popović, J. (2011). 6D hands: markerless hand-tracking for computer aided design. Paper presented at the Proceedings of the 24th annual ACM symposium on User interface software and technology.

Wang, R. Y., & Popović, J. (2009). Real-time hand-tracking with a color glove. ACM Transactions on Graphics (TOG), 28(3), 63.

Weiss, M., Wagner, J., Jansen, Y., Jennings, R., Khoshabeh, R., Hollan, J. D., & Borchers, J. (2009). SLAP widgets: bridging the gap between virtual and physical controls on tabletops. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Wigdor, D., Forlines, C., Baudisch, P., Barnwell, J., & Shen, C. (2007). Lucid touch: a see-through mobile device. Paper presented at the Proceedings of the 20th annual ACM symposium on User interface software and technology.

Wigdor, D., Leigh, D., Forlines, C., Shipman, S., Barnwell, J., Balakrishnan, R., & Shen, C. (2006). Under the table interaction. Paper presented at the Proceedings of the 19th annual ACM symposium on User interface software and technology.

Wigdor, D., & Wixon, D. (2011). Brave NUI world: designing natural user interfaces for touch and gesture. Elsevier.

Wilson, A. D. (2006). Robust computer vision-based detection of pinching for one and two-handed gesture input. Paper presented at the Proceedings of the 19th annual ACM symposium on User interface software and technology.

Wilson, A. D. (2009). Simulating grasping behavior on an imaging interactive surface. Paper presented at the Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces.

Wilson, A. D. (2010). Using a depth camera as a touch sensor. Paper presented at the ACM international conference on interactive tabletops and surfaces.

Wilson, A. D., Izadi, S., Hilliges, O., Garcia-Mendoza, A., & Kirk, D. (2008). Bringing physics to the surface. Paper presented at the Proceedings of the 21st annual ACM symposium on User interface software and technology.

Wingrave, C., & Bowman, D. (2005). Baseline factors for raycasting selection. Paper presented at the Proceedings of HCI International.

Wobbrock, J. O., Morris, M. R., & Wilson, A. D. (2009). User-defined gestures for surface computing. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Wobbrock, J. O., Wilson, A. D., & Li, Y. (2007). Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. Paper presented at the Proceedings of the 20th annual ACM symposium on User interface software and technology.

Worden, A., Walker, N., Bharat, K., & Hudson, S. (1997). Making computers easier for older adults to use: area cursors and sticky icons. Paper presented at the Proceedings of the ACM SIGCHI Conference on Human factors in computing systems.

Wren, C. R., Azarbayejani, A., Darrell, T., & Pentland, A. P. (1997a). Pfinder: Real-time tracking of the human body. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 19(7), 780-785.

Wren, C. R., Sparacino, F., Azarbayejani, A. J., Darrell, T. J., Starner, T. E., Kotani, A., . . . Pentland, A. P. (1997b). Perceptive spaces for performance and entertainment: untethered interaction using computer vision and audition. Applied artificial intelligence, 11(4), 267-284.

Wu, M., & Balakrishnan, R. (2003). Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. Paper presented at the Proceedings of the 16th annual ACM symposium on User interface software and technology.

Wu, S., Otmane, S., Moreau, G., & Servières, M. (2013). Design of a visual query language for geographic information system on a touch screen. In Human-Computer Interaction. Interaction Modalities and Techniques (pp.530-539). Springer Berlin Heidelberg.

Wu, S., Chellali, A., Otmane, S. (2014a). FingerOscillation: Clutch-free Techniques for 3D Object Translation, Rotation and Scale. The 20th ACM Symposium on Virtual Reality Software and Technology.

Wu, S., Chellali, A., Otmane, S., Moreau, G. (2014b). HorizontalDragger: a Freehand Remote Selector for Object Acquisition. Les 9èmes journées de l'Association Française de Réalité Virtuelle (AFRV), AFRV 2014.

Wu, S., Chellali, A., Otmane, S., Moreau, G. (2015a) HorizontalDragger: a Freehand Remote Selector for Object Acquisition. In the proceedings of the Virtual Reality conference, IEEE VR 2015, Arles, France.

Wu, S., Chellali, A., Otmane, S., Moreau, G. (2015b). LayerStroke: a Layer based Selector for Small Target Acquisition. La 27ème Conférence Francophone sur l'interaction Homme-Machine. (Accepted)

Wu, S., Chellali, A., Otmane, S., Moreau, G. (2015c). TouchSketch: a touch-based interface for 3D object manipulation and editing. The 21th ACM Symposium on Virtual Reality Software and Technology. (Accepted)

Wu, M., Shen, C., Ryall, K., Forlines, C., & Balakrishnan, R. (2006). Gesture registration, relaxation, and reuse for multi-point direct-touch surfaces. Paper presented at the Horizontal Interactive Human-Computer Systems, 2006. TableTop 2006. First IEEE International Workshop on.

Yatani, K., Partridge, K., Bern, M., & Newman, M. W. (2008). Escape: a target selection technique using visually-cued gestures. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

Yoon, D., Lee, J. H., Yeom, K., & Park, J. H. (2011). Mobiature: 3d model manipulation technique for large displays using mobile devices. Paper presented at the Consumer Electronics (ICCE), 2011 IEEE International Conference on.

Zhai, S., Morimoto, C., & Ihde, S. (1999). Manual and gaze input cascaded (MAGIC) pointing. Paper presented at the Proceedings of the SIGCHI conference on Human Factors in Computing Systems.

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Etude et conception des techniques d'interaction pour faciliter la sélection et la manipulation d'objets dans des environnements virtuels sur interface mobile

Mots clés : interaction 2D/3D, interface naturelle utilisateur, environnement virtuel, interface mobile

Résumé : Les avancées dans le domaine des NUIs (interfaces utilisateur naturelles) permettent aux concepteurs de développer de nouvelles techniques efficaces et faciles à utiliser pour l'interaction 3D. Dans ce contexte, les interfaces mobiles attirent beaucoup d'attention sur la conception de techniques d'interaction 3D pour une utilisation ubiquitaire.
Nos travaux de recherche se focalisent sur la proposition de nouvelles techniques d'interaction pour faciliter la sélection et la manipulation d'objets dans des environnements virtuels s'exécutant sur des interfaces mobiles.
En effet, l'efficacité et la précision de la sélection des l'objets sont fortement affectés par la taille de la cible et la densité de l'environnement virtuel. Pour surmonter le problème d'occlusion du bout des doigts sur les Smartphones, nous avons conçu deux techniques de sélection reposant sur le toucher. Nous avons également conçu deux techniques hybrides à main levée pour la sélection à distance de petits objets. Pour effectuer une manipulation d'objets contraints sur les Tablet-PC, nous avons proposé une technique bimanuelle basée sur un modèle asymétrique. Les deux mains peuvent être utilisés en collaboration, afin de spécifier la contrainte, déterminer le mode de manipulation et de contrôler la transformation. Nous avons également proposé deux autres techniques de manipulation à une seule main en utilisant les points de contacts identifiés. Les évaluations de nos techniques démontrent qu'ils peuvent améliorer l'expérience des interactions utilisateurs sur des interfaces mobiles. Nos résultats permettent aussi de donner quelques lignes directrices pour améliorer la conception de techniques d'interactions 3D sur des interfaces mobiles.

Title: Study and design of interaction techniques to facilitate object selection and manipulation in virtual environments on mobile devices

Keywords: 2D/3D interaction, natural user interfaces, virtual environment, mobile devices

Abstract: The advances in the field of NUIs (Natural User Interfaces) can provide more and more guidelines for designers to develop efficient and easy-to-use techniques for 3D interaction. In this context, mobile devices attract much attention to design 3D interaction techniques for ubiquitous usage.
Our research work focuses on proposing new techniques to facilitate object selection and manipulation in virtual environments on mobile devices. Indeed, the efficiency and accuracy of object selection are highly affected by the target size and the cluster density.
To overcome the fingertip occlusion issue on Smartphones, we have designed two touch-based selection techniques. We have also designed two freehand hybrid techniques for selection of small objects displayed at a distance.
To perform constrained manipulation on Tablet-PCs, we have proposed a bimanual technique based on the asymmetrical model. Both hands can be used in collaboration, in order to specify the constraint, determine the manipulation mode, and control the transformation. We have also proposed two other single-hand manipulation techniques using identified touch inputs.
The evaluations of our techniques demonstrate that they can improve the users' interaction experience on mobile devices. Our results permit also to give some guidelines to improve the design of 3D interactions techniques on mobile devices.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France