UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

HENRIQUE GALVAN DEBARBA

Selection in 2D and 3D Environments with Levels of Precision and Progressive Refinement

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Profa. Dra. Luciana Nedel
Advisor

Prof. Dr. Anderson Maciel
Coadvisor

Porto Alegre, August 2012

CIP – CATALOGING-IN-PUBLICATION

Galvan Debarba, Henrique

Selection in 2D and 3D Environments with Levels of Precision and Progressive Refinement / Henrique Galvan Debarba. – Porto Alegre: PPGC da UFRGS, 2012.

107 p.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2012. Advisor: Luciana Nedel; Coadvisor: Anderson Maciel.

1. 2D interaction. 2. 3D interaction. 3. Selection. 4. User studies. I. Nedel, Luciana. II. Maciel, Anderson. III. Título.

“In the beginning there was nothing,
which exploded.”
Lords and Ladies
— TERRY PRATCHETT

AGRADECIMENTOS

Em primeiro lugar eu gostaria de agradecer a minha família e namorada, sempre compreensíveis e pacientes com minha ausência e rotina de estudo e trabalho em horário incomum.

Gostaria também de agradecer a ajuda prestada pelos colegas do grupo de computação gráfica, processamento de imagem e interação da UFRGS. Dentre muitos favores, estes facilitaram minha integração na UFRGS, auxiliaram em conteúdos onde meu conhecimento era insuficiente, voluntariamente revisaram meus artigos, disponibilizaram materiais, estimularam minha curiosidade, ativamente propuseram ideias para melhorar este trabalho e promoveram discussões nas quais aprendi muito. Em especial, agradeço aos colegas Marilena Maule, Leonardo Fischer, Vitor Jorge, Vitor Pamplona, Juliano Franz, Jerônimo Grandi e Leandro Fernandes.

Agradeço também aos meus orientadores, Luciana Nedel e Anderson Maciel, os quais me ajudaram a manter o foco e me ensinaram metodologia de pesquisa e escrita precisa, como requer a ciência. Também possibilitaram meu breve intercâmbio na Université Catholique de Louvain em 2011, e apoiaram minha iniciativa de começar um doutorado no exterior.

Por fim, agradeço a CAPES pela bolsa de mestrado, ao Instituto de Informática da UFRGS pelo financiamento de viagens, e especialmente a secretaria do PPGC pela eficiência e atenção.

Henrique Galvan Debarba

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

CD	Control-Display
DOF	Degree-of-freedom
FOV	Field of View
HMD	Head Mounted Display
LOP	Levels of Precision
PPI	Pixels Per Inch
HCI	Human-Computer Interaction

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Selection is one of the four fundamental forms of interaction in a virtual world. It is the ability of the user to specify objects in the virtual environment for subsequent actions. The literature is rich in immediate selection techniques; however, this class of technique is exposed to problems of selection accuracy, ambiguity and complexity.

These issues can be addressed using techniques of selection by: *progressive refinement*, which consists of reducing the amount of selectable objects through refinement steps until the desired object is selected; or *levels of precision*, which consists of increasing accuracy of pointing by manipulating the *control-to-display* parameters as well as by the combination of pointing approaches. *Levels of precision* is a class of selection techniques we are proposing in this dissertation. We also present a survey that comprehends literature on these two classes of selection techniques.

Furthermore, we propose the *LOP-cursor* and the *disambiguation canvas* selection techniques. The first rely on multiple levels of precision to achieve very high control over the cursor position, while the second uses the high resolution of input provided by a mobile device touchscreen to quickly disambiguate a selection among hundreds of objects. Finally, we present the *two-legged cursor* metaphor, a novel approach for simultaneously pointing of two distinct locations. The *two-legged cursor* is intended to be used to quickly perform composite tasks, such as drag and drop.

User evaluation shows that our approaches are promising, and that the design space of techniques using mobile devices can lead to numerous possibilities. Users were able to select targets as little as $\approx 0.1^\circ$ of visual angular size with *LOP-cursor*. In one of its evaluations, *disambiguation canvas* achieved an error rate < 0.01 per selection trial. Finally, Both techniques performed faster than *ray-casting* for small targets, and overall they were preferred by users.

Keywords: 2D interaction, 3D interaction, selection, user studies.

Seleção em Ambientes 2D e 3D utilizando Níveis de Precisão e Refinamento Progressivo

RESUMO

Seleção é uma das quatro formas fundamentais de interação em mundos virtuais. É a habilidade de especificar objectos em ambientes virtuais para ações posteriores. A literatura é rica em técnicas de seleção imediata; no entanto, esta classe de técnicas está exposta a problemas de precisão, ambiguidade e complexidade de seleção.

Os problemas mencionados podem ser abordados com técnicas de seleção por: *refinamento prograssivo*, que consiste na redução da quantia de objetos selecionáveis através de etapas de refinamento, até que o objeto desejado seja selecionado; ou *níveis de precisão*, que consiste em aumentar a precisão de apontamento através da manipulação de parametros do mapeamento *controle para display*, assim como na combinação de abordagens de apontamentio. *Níveis de precisão* é uma classe de técnicas de seleção que estamos propondo neste dissertação. Neste documento também apresentamos uma pesquisa com literatura relacionada a estas duas classes de técnicas de seleção.

Adicionalmente, propomos as técnicas de seleção *LOP-cursor* e *tela de desambiguação*. A primeira conta com múltiplos níveis de precisão para proporcionar maior controle sobre a posição do cursor, enquanto a segunda usa a alta resolução de entrada proporcionada pela tela sensível ao toque de disposiiticos móveis para rapidamente desambiguar a seleção de um alvo entre centenas de outros objetos. Por fim, apresentamos a metáfora de um *cursor com duas pernas*, uma nova abordagem para apontar duas posições distintas simultaneamente. O *cursor de duas pernas* é projetado para rapidamente realizar tarefas compostas, como arrastar e soltar em uma área de trabalho.

Avaliações com usuários mostram que nossas abordagens são promissoras, e que o espaço de *design* de técnicas utilizado dispoitivos móveis pode produzir inúmeras possibilidades. Os usuários foram capazes de selecionar alvos com $\approx 0.1^\circ$ de tamanho angular visual com o *LOP-cursor*. Em um dos experimentos, a *tela de desambiguação* obteve a proporção de erro < 0.01 por tentativa de seleção. Por fim, ambas as técnicas foram mais rápidas que o tradicional *ray-casting* para seleção de objetos pequenos, sendo sempre preferidas pelos usuários.

Palavras-chave: interação 2D, interação 3D, seleção, estudos com usuários.

1 INTRODUCTION

Modern displays and graphics technology, as well as its global availability and drop in prices are widening the reach of non conventional visualization environments for the general population everyday life. Large displays are rapidly spreading in public spaces and even in homes and offices, while televisions are increasing in size and resolution, and are commonly allowing stereoscopy.

However, there are still gaps on how to make such visualization environments conveniently interactive. Ordinary input devices (i.e. mouse and keyboard) tend to require a surface to use and do not perform as well as usual when faced with larger screens. More natural and intuitive metaphors are required, such as the laser pointer. However, the increase in resolution also prompts the need to allow for high precision of input, something that laser pointer cannot deliver due to hand jitter. On the other hand, current mobile phones and similar devices incorporate a broad range of sensors that can hardly be found in any other product on the market. This quickly evolving product is also regarded as the first really pervasive computational device (BALLAGAS et al., 2006).

Researchers in 3D virtual environments categorizes interaction in four fundamental forms: selection, manipulation, navigation, and control system(BOWMAN et al., 2004; MINE, 1995). In this dissertation we focus mainly on selection. Selection is the ability of the user to specify the choise of objects in the environment for subsequent actions (STEED, 2006). Many applications are more critical on correctness of selection than in time of selection. However, ordinary selection techniques in use tend to lack in accuracy to favor performance. We intend to provide precise, yet fast, alternatives for selection. Therefore, we rely on *selection by progressive refinement*, as proposed by Kopper et al. (KOPPER; BACIM; BOWMAN, 2011), and on *selection by levels of precision*, which is a classification we propose in this dissertation.

Progressive refinement consists of reducing the amount of selectable objects through refinement steps until the desired object is selected. It generally happens through a tradeoff between accuracy and time. Splitting selection in subsequent steps makes the pointing in each of them easier. However, what once was performed immediately now consists of a procedure. *Levels of precision* consists of increasing accuracy of pointing through the manipulation of *control-to-display* parameters – a constant subject of study in HCI (BLANCH; GUIARD; BEAUDOUIN-LAFON, 2004) – as well as by the combination of pointing approaches. As a consequence, the user will have her movements remapped to achieve finer and more stable cursor response.

In this work, we chose a recent mobile phone as the input hardware. As argued before, there is a lack of a standard input device for large and high resolution displays. Based on recent research and the range of sensors available in mobile phones, we support that they are candidates to take this position. This claim is even more likely for the sporadic users,

who occasionally interact with a large or high resolution display and might not be willing to spend on an specialized device.

1.1 Contributions

The main contributions of this dissertation are:

- A proposition of a *levels of precision* design space.
- An extense survey of techniques for selection by *levels of precision* and *progressive refinement*.
- A technique called *LOP-cursor* for precise selection by levels of precision using a mobile device.
- A technique called *disambiguation canvas* for selection by progressive refinement using a mobile device. It allows the disambiguation among hundreds of objects in one step.
- A *two-legged cursor* metaphor, which allows the definition of two simultaneous positions using only one hand.

The *LOP-cursor* (acronym of levels of precision cursor), the *disambiguation canvas*, and the *two-legged cursor* are introduced in more details below.

LOP-cursor is a technique for high precision pointing and selection. It is controlled with a smartphone in our implementation. *LOP-cursor* combines the absolute pointing technique of *ray-casting* (LIANG; GREEN, 1994; MINE, 1995) using the mobile device movement sensors, with a second level of control relying on the device's touchscreen. The user subjectively switches between precisions with the *start* and the *take-off* of a touch. We evaluate the technique with targets appearing as little as $\approx 0.1^\circ$ of visual angular size, while the limit of 20/20 for normal visual acuity is $\approx 0.017^\circ$. This means that *LOP-cursor* was able to select targets 6 times bigger than the limit of normal vision perception.

Disambiguation canvas is a technique for quick disambiguation of selection. We use the observed high precision of control provided by the touchscreen to allow the disambiguation of the desired object among a subset of hundreds of other objects in only one step of refinement. User tests show that this technique performs faster than *ray-casting* for targets with $\approx 0.53^\circ$ of angular size, being also much more accurate for all the tested target sizes. Previous progressive refinement techniques do not scale as well as ours. Available techniques that disambiguate only in one step are limited to a small subset of objects, while those that refine among large subsets require multiple steps of disambiguation. In order to easily include the object to be selected on the subset that will be refined, *disambiguation canvas* uses a volume-casting technique.

Two-legged cursor: the selection techniques proposed above use two distinct input hardware, the motion sensors and the touchscreen. We noticed that users easily switch between these inputs, and are even able to perform some level of simultaneous control over both of them. Thus, we propose a cursor that allows the completion of two-location tasks such as drag and drop in only one step.

1.2 Organization

The remaining of this thesis is organized as follows. In order to contextualize the reader, Chapter 2 explores selection taxonomy, immediate selection techniques and major selection problems; Chapter 3 summarizes related works on the use of mobile devices to interact with large displays as well as selection techniques that consider levels of precision and progressive refinement; Chapter 4 presents the proposed techniques; Chapter 5 describes the current state of hardware technology and software implementation. In Chapter 6 we present the evaluation of the *LOP-cursor* and *disambiguation canvas*, comparing it against other selection techniques. We also present the respective results followed by a discussion; Finally, in Chapter 7 we highlight our findings and suggest future developments.

2 SELECTION

This chapter recalls Bowman et al. *selection taxonomy* (BOWMAN et al., 2004), and discusses some of the traditional immediate selection techniques. Furthermore, we expose the most common pointing issues regarding large and immersive displays. These explanations may be useful to the general understanding of this work.

2.1 Selection Taxonomy

Selection may be decomposed in a few complementary parts. We will follow the taxonomy presented by Bowman et al. (BOWMAN et al., 2004) which split selection in three building blocks: *Indication of object*, *confirmation of selection* and *feedback*. Thus, on a selection task, the user must be able to point out an object and perform a selection command. Complementarily, the system must provide feedback in order to keep the user aware of the pointing and selection state while performing the task.

The taxonomy is presented in Figure 2.1. From these building blocks, the most relevant to our work is the *indication of object*, which can be performed through *occlusion* (image plane techniques (PIERCE et al., 1997)), *object touch* (virtual hand (MINE, 1995)), *pointing* (ray-casting (LIANG; GREEN, 1994)) or *indirect selection*, which was not addressed in this work.

An immediate selection technique may be constructed by the combination of one from each of the three building blocks presented above. However, selection techniques with *progressive refinement* or *levels of precision* are usually built using more than three blocks. Since these may require multiple ways to indicate and confirm the selection of an object, as well as it may require distinct feedback for each step of the selection.

For the sake of objectivity, we will recall this taxonomy in several moments on this thesis.

2.2 Immediate Selection Techniques

Here, immediate selection techniques are presented according to the taxonomy on *indication of object* presented in Section 2.1. Relevant techniques of each kind of *indication of object* were preferably arranged in chronological order. It is also worth noticing that earlier techniques are usually the base for most of current techniques, probably because most of them rely on real life metaphors, being very intuitive and sometimes obvious. Above all, we emphasize that *ray-casting* based techniques are the more often used for virtual environments, and base of most of progressive refinement and levels of precision techniques covered in the related work (Chapter 3).

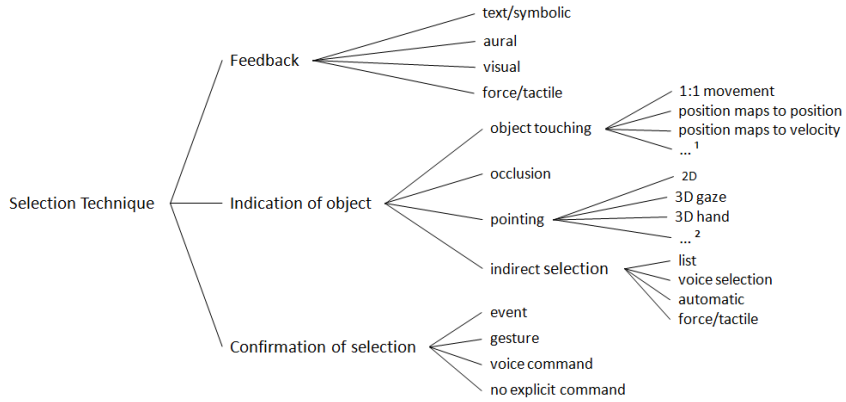

Figure 2.1: Selection techniques taxonomy proposed by Bowman et al. (BOWMAN et al., 2004) [1] [2] .

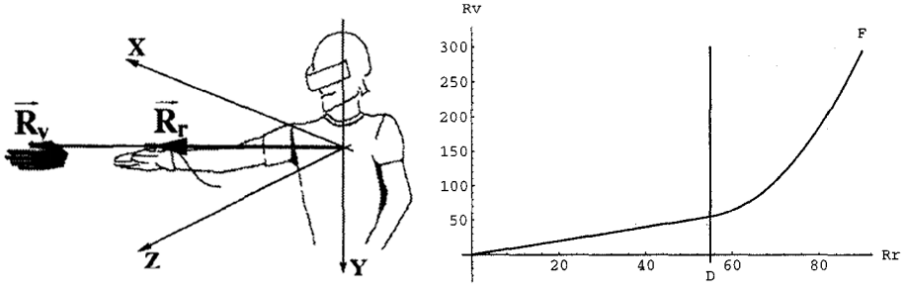

Figure 2.2: Go-Go interaction technique uses a non linear mapping between Rr and Rv. The mapping function is linear for Rr < D, and non-linear when Rr > D. (POUPYREV et al., 1996)

Object touching are position dependent techniques, usually applied for objects within a limited distance. The *virtual hand* (MINE, 1995) approach may be the most obvious technique. It uses an 1:1 mapping ratio between the real and virtual representation of the hand, mimicking the real behavior of the tracked hand inside the virtual environment. Furthermore, another worth noticing approach is the *Go-Go* technique (POUPYREV et al., 1996). *Go-Go* is a natural extension of the original Virtual Hand and tries to solve selection of objects beyond of the arm reach. It uses a non linear mapping function between the real and the virtual hand when the user stretch his arm more than 2/3 of its maximum reach. The mapping function is depicted in Figure 2.2. Other approaches use intersection with a scalable volume in order to select objects that are near the hand position, or rescale the cursor to intersect the nearest target. This is the case on the *Bubble Cursor 3D* (VANACKEN; GROSSMAN; CONINX, 2007), where the cursor is rescaled in order to always contain one (and only one) object (Figure 2.3).

Pointing is usually applied to allow the selection of objects at any distance. *Ray-casting* is the most common approach. It was first proposed by Liang and Green (LIANG;

[1] Steed (STEED, 2006) highlights that velocity, position and acceleration could be combined in much many ways than presented in (BOWMAN et al., 2004)

[2] Steed (STEED, 2006) highlights that any limb, or combination of limbs, can potentially be used for pointing

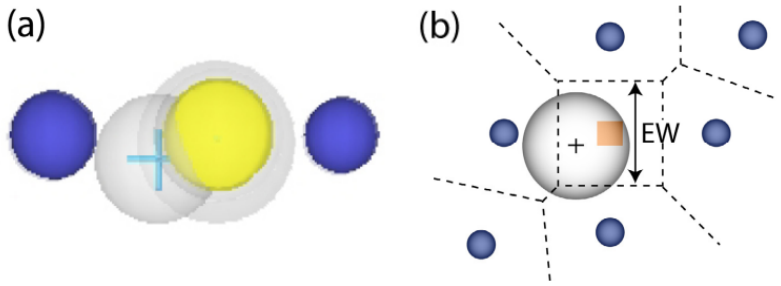

Figure 2.3: Bubble Cursor 3D selection technique: (a) The object nearest to the center of the cursor is always selectable. (b) Bubble cursor divides the space into 3D Voronoi regions in order to increase the effective width of each object. (VANACKEN; GROSSMAN; CONINX, 2007)

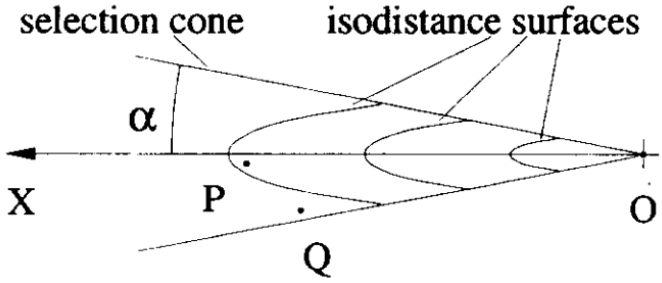

Figure 2.4: Spotlight cross section of the selection cone showing isodistance surfaces. The point P is closer than Q using this criteria. (LIANG; GREEN, 1994)

GREEN, 1994) under the name "laser gun" and consists of a ray intersection test, mimicking the real life laser pointer. This is a very commonly used approach and represents the basis for many of the techniques that rely in progressive refinement or levels of precision. Liang and Green have also presented the "Spotlight" selection as part of their 3D modeling system solution (LIANG; GREEN, 1994). *Spotlight* uses a cone projection in order to make selection soften then a mathematical ray intersection test. To be selectable, the object must fall inside its conic volume. If more than one object fall inside the volume, disambiguation is solved according to the anisotropic distance of each point P to the origin given by Equation (2.1), where $K = 1 \div \sin\alpha$ and $|P|$ is the Euclidian distance. Plots of the cut of some isodistance surfaces can be verified in Figure 2.4. Currently, this technique is better known as *cone-casting*.

$$|P|\frac{K}{\sqrt{1+(K^2-1)*(\frac{x}{|P|})^2}} \tag{2.1}$$

In *aperture*, Forsberg et al. (FORSBERG; HERNDON; ZELEZNIK, 1996) extend the *spotlight* technique placing the origin of the cone at the dominant eye virtual position, and using a tracked wand on the hand to define the cone direction. The spread angle of the cone can be controlled simply by moving the hand sensor closer or farther away, allowing for interactive control over the selection volume (Figure 2.5). Disambiguation can be automatically performed such as proposed by *spotlight*, or using the tracked wand and objects orientation similarities.

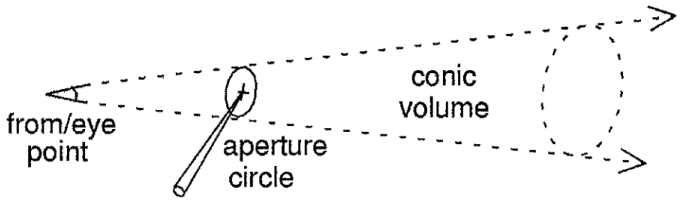

Figure 2.5: Aperture conic selection volume is resultant of the position relation between the eye and the aperture circle. (FORSBERG; HERNDON; ZELEZNIK, 1996)

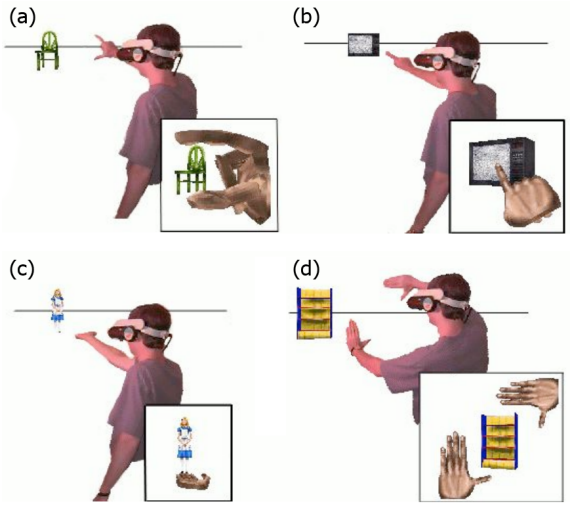

Figure 2.6: Image plane selection techniques: (a) head crusher, (b) sticky finger, (c) lifting palm and (d) framing hands. (PIERCE et al., 1997)

Concerning *Occlusion*, Pierce et al. (PIERCE et al., 1997) proposed a family of techniques using selection on the image plane. Selection on the image plane reduces the problem of pointing to 2 DOF, as the user is required to point on the image plane projection of the object. It may be compared to the mouse cursor on an usual 2D desktop. Techniques proposed by Pierce et al. were: the *head crusher*, where the thumb and forefinger should be positioned around the object, and a mid point between these is used for selection. The *sticky finger* consists of using an outstretched finger toward the object, the object underneath the finger on the image plane is selected. On the *lifting palm* the user must put the target object right above the palm of his hand. Finally, on the *framing hands* the user position his hands around the target object defining two corners of a frame, it may be used as an area selector, or the mid point inside the frame is used as a punctual selector. Figure 2.6 depicts the image plane techniques.

2.3 Selection Problems

This section discusses the major problems on selection, which we have took into account while designing the *LOP-cursor* and the *Disambiguation Canvas* techniques. As stated by Haan et al. (HAAN; KOUTEK; POST, 2005) regarding their observations while developing virtual reality visualization applications, the most common problems are selection accuracy, selection ambiguity and selection complexity. All of these are related to

the *indication of object* block of the selection taxonomy.

Additionally, less general problems related to *confirmation of selection* and *feedback* may occurs. As these are more specific or punctual, they are addressed within the proposed techniques chapter, while justifying our design choices.

2.3.1 Selection Accuracy

A major issue for pointing and selection at a distance is accuracy. Pointing techniques based on ray intersection may be very instable. These generally uses 6 DOF input devices, three – concerning device's position – are used to position the origin of the ray, and the other three – concerning device's orientation – are used to define the direction of the ray. As the angular control of the pointing device results in amplified movements at the intersection point of the ray, the orientation control is predominant over the position control in distant pointing selection. According to the object's visible size and distance, its angular size may be very small, making selection a hard and tiring task. Furthermore, jittering from the tracking system and/or user limbs will negatively affect ray casting techniques. Even though filtering and tracking quality have significantly improved in past decades. Human motor imperfections are not likely to be fully overcome. We have observed that users tend to support their elbow or forearm, and sometimes tend to manage the speed of their breathing, in order to reduce jittering when selecting objects of little angular size using *ray-casting*.

Furthermore, there is also a gap regarding the display output resolution and the interaction input resolution. On a tiled-display made from 90 PPI monitors, such as the one presented in Section 5.1.1, only one meter of distance is enough to overcome the normal eye acuity in resolution. However, users cannot achieve easy control over the information seen without switching between navigation, manipulation and selection when using common interaction metaphors.

2.3.2 Selection Ambiguity

Ambiguity happens when the system cannot solve which object or group of objects the user wants among a broader subset of indicated objects. For instance, when pointing on a cluttered environment, multiple targets can fall inside the volume of selection, or multiple targets can be intersected by the cast of a ray. Automatic disambiguation rules are frequently applied, such as using the nearest intersection for ray casting, the nearest target for positional cursors, and more specific/complex rules for volume casting techniques, such as proposed by the *spotlight* technique.

In the case of multiple objects intersecting a ray, the use of the nearest intersection may be inadequate due to cluttering and eye-hand visibility mismatch (ARGELAGUET; ANDUJAR; TRUEBA, 2008). Eye-hand mismatch refers to the fact that in regular *ray-casting* and similar pointing techniques, visibility from the eye position differs from visibility from the hand position. From the hand point of view, the intended object may be partially occluded, reducing the target angular size and requiring more accuracy, or even totally occluded, requiring the repositioning of the hand. If the environment is very cluttered, the user may have trouble finding a good position for his hand. Furthermore, disambiguation by first intersection in a *fishtank VR system* or a volumetric display is even more inadequate. In such case, eye-hand visibility mismatch tends to be more severe as the user usually sees the virtual environment from an up or front view, while the pointing device is frequently positioned in a sided angle (GROSSMAN; BALAKRISHNAN, 2006). Figure 2.8 depicts some consequences of eye-hand visibility mismatch. Addition-

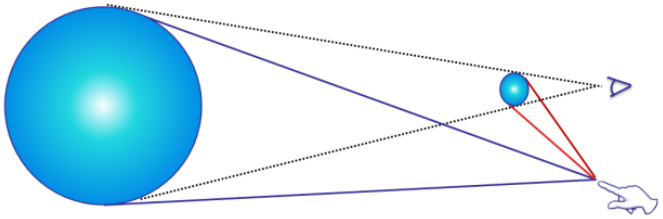

Figure 2.7: Eye-hand visibility mismatch: solid angle. The two spheres have the same angular size for the eye, but not for the hand. Figure from (ARGELAGUET; ANDUJAR; TRUEBA, 2008)

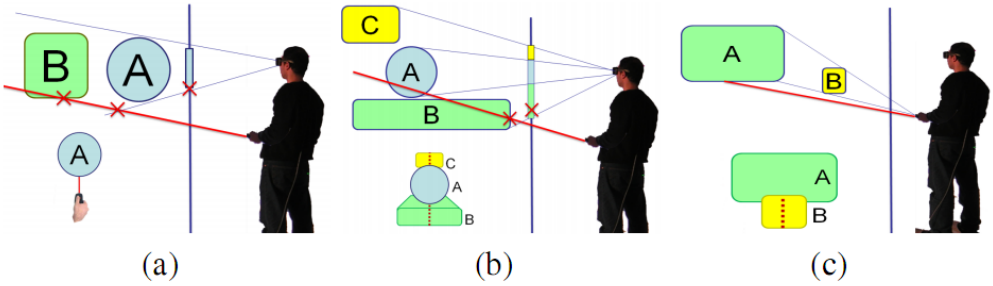

Figure 2.8: Eye-hand visibility mismatch: occlusion. In (a) the user can select an object hidden by another object. (b) The visible objects A and C cannot be selected from the current user hand position. (c) Object A is simultaneously visible from the eye and the hand, but none point on its boundary is simultaneously visible for eye and hand. Figure from (ARGELAGUET; ANDUJAR; TRUEBA, 2008)

ally, the solid angle may also vary significantly due to this mismatch, as depicted in figure 2.7.

Nevertheless, as stated by Vanacken et al. (VANACKEN; GROSSMAN; CONINX, 2007), the user may actually want to select a target occluded from the eyes point of view. He may be able to point at it (Figure 2.8a), but adequate feedback must be provided in order to make the user aware of the success of such task. Otherwise the user will need to change the head position and navigate. Vanacken et al. have used a magic lens (BIER et al., 1993) based metaphor to make intersected objects, and those on the line of sight to the intersected objects, transparent.

2.3.3 Selection Complexity

As stated by Haan et al. (HAAN; KOUTEK; POST, 2005), complexity of selection refers to animated objects, or objects that appear for a limited period of time. More generally, we see these as objects which can vary its state, attributes and pose over time. Additionally, we include to *selection complexity* the selection of a group of objects.

Animated objects may be difficult to select as they vary in angular size and may require accompanying hand movements and correct timing for selection triggering from the user. Additionally, they can also get out of the line of sight, requiring head movements and even virtual navigation. Furthermore, animated objects are likely to dynamically become occluded in a cluttered environment. Nonetheless, objects may vanish and become

unselectable after a given time, this may be the case for scientific visualization of simulations.

We consider the selection of a group of objects as a case of selection complexity. The user may need to manipulate several objects at the same time, or to perform the same treatment on multiple objects. Selecting and repeating a manipulation on one object at a time would be very laborious and time consuming. Furthermore, selection of single objects have been largely studied (BOWMAN et al., 2004), while multiple objects selection techniques receive less attention in 3D interaction literature. The deeper study available was proposed by Lucas (LUCAS, 2005), on which he suggests a design space and the PORT technique, which uses a series of movements and resizing actions to define the set of objects.

3 RELATED WORK

This chapter covers selection by *levels of precision* and selection by *progressive refinement* definitions and currently available techniques. We also present techniques based on mobile devices for the control of external displays that are not compatible with these classifications, but are relevant to our argument of mobile devices as standard input hardware.

3.1 Selection by Levels of Precision

We classify as levels of precision the techniques which increase the space of motor control, therefore increasing user's precision of pointing. This can be achieved by changing parameters of the *indication of object* selection building block, such as dinamically changing the *control-display gain* (*CD gain*) or the *control-display ratio* (*CD ratio*), or by building a selection technique that uses more than one *indication of object* block, such as switching from an absolute to a relative technique.

Levels of precision approaches have a strong connection with the *optimized initial impulse* motor control model (MEYER et al., 1988), considered the most successful and complete explanation for Fitts' law to date (ROSENBAUM, 2010; BALAKRISHNAN, 2004). In essence, it states that most aimed movements consist of an initial large and fast movement which puts the subject reasonably close to the target, followed by a shorter and slower corrective movement. Using a technique with *levels of precision*, the large movement, also known as the ballistic phase, is performed by a technique that supports the fast and large nature of the movement, while the corrective movements are performed by a pointing approach compatible with the expected application precision. We propose the design space in Figure 3.1 for *levels of precision* techniques, where techniques with n levels of precision may be build from $(n-1)$ iterations of this design space, with $n \geq 2$.

Transition between levels can be discrete, consisting of a well defined change on input control, or continuously, allowing intermediate control values between the two levels. It can be performed with an explicit command, such as with a button, or implicit. The strategy of transition can be enforced by the technique, where the user is required to combine levels, or determined by the user, so he is able to decide whether he wants to switch levels.

We have classified levels of precision techniques as: those which relies on the dynamic manipulation of the CD Gain or CD Ratio; and those which combines two techniques. In current literature, all the approaches which combines two techniques consists of absolute for the first level and relative for the second.

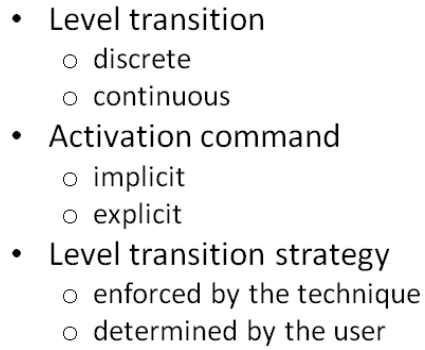

Figure 3.1: Selection by levels of precision design space.

3.1.1 Control-Display Gain and Control-Display Ratio

Most common techniques that increase the motor control space dynamically modify the *CD gain* of relative pointing techniques (such as used by Windows and OS X cursors). Casiez et al. have evaluated the impact of dynamic control of *CD gain* on mouse performance (CASIEZ et al., 2008), showing to be overall 3.3% faster than constant *CD gain*, and up to 5.6% faster for small targets. Furthermore, Casiez and Roussel have also compared the *CD gain* transfer functions of Windows, OS X and Xorg (CASIEZ; ROUSSEL, 2011). Although frequently used, this approach alone does not scale properly to large displays, where the user is likely to loose track of the cursor with relative mappings. Moreover, this approach does not take advantage of absolute relations of the user with the virtual environment, which is generally encouraged by immersive displays.

Some techniques address the lack of precision of *ray-casting* dynamically changing the *CD ratio*. PRISM do so automatically (FREES; KESSLER; KAY, 2007), according to the angular speed of the pointing device. Slower angular speed results in higher CD ratio, conferring more precision to the pointing task. Figure 3.2 shows a rough graphic and explanation of the system response. ARM (KOPPER et al., 2010) is another technique that dynamically controls the CD ratio. It uses a button to manually rescale the movement to a tenth for the duration of the button press. Although these techniques allow high level of precision, the mismatch between the real pointing direction and the virtual ray direction may be distracting. In addition, they also require the user to interact carefully and with attention.

3.1.2 Combination of Absolute and Relative Pointing

Most techniques combine absolute and relative modes of input. In all the cases, absolute control is used to approach the location of the object, while a relative approach is used to fine tune over the desired object. The relative pointing is often implemented with a transfer function for dynamic control of the CD gain, being similar to ordinary mouse cursors and adding a third level of precision.

Vogel and Balakrishnan (VOGEL; BALAKRISHNAN, 2005) explored natural input using the naked hand. The proposed technique allows switching across absolute and relative input performing two different hand poses. The user can point to a region at the large screen performing an absolute hand pose, and then, change the hand pose to switch to the relative mode. Similar in concept, Forlines et al. proposed an analogous for pen interaction with large and high resolution touch surfaces (FORLINES; VOGEL;

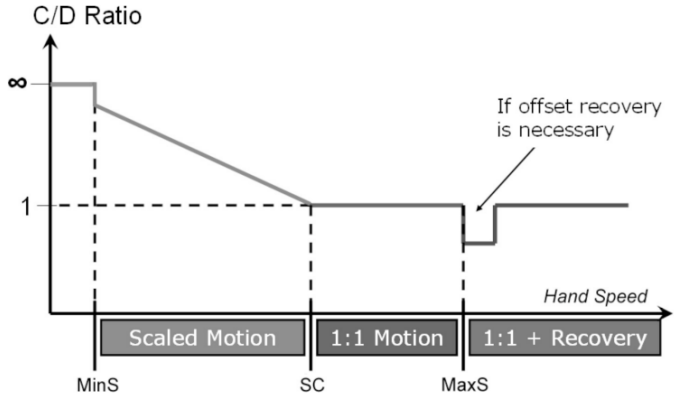

Figure 3.2: PRISM Control Display ratio mapping: movements below the *MinS* threshold are considered non intentional and are ignored. Movements between *MinS* and *SC* have the CD ratio linearly rescaled to maximize control. Movements between *SC* and *MaxS* have a 1:1 CD ratio. Finally, movements above *MaxS* result on the offset recovery (accumulated error resulting from movements below *SC* recovery). After the correction occurs, it uses the 1:1 CD ratio again. (FREES; KESSLER; KAY, 2007)

BALAKRISHNAN, 2006). Users could switch from absolute to relative input mode to achieve targets out of their arms reach area. Both techniques rely on high cost apparatus and a prepared interaction environment (VICON tracking system).

Nancel et al. (NANCEL; PIETRIGA; BEAUDOUIN-LAFON, 2011) also proposed a technique using the VICON system. They combine a VICON-based ray casting mode with a precise relative pointing sliding the finger on an Apple iPod Touch device touchscreen. The touch surface is divided in two areas: an upper zone for tracking, and a lower zone for clicking. Touching the upper zone switches to precise mode. Authors claim it is possible to select objects with 4 millimeters standing 2 meters away from the display, which is comparable to some results we show in this paper using only smartphone built-in sensors.

The *ARC-Pad* (MCCALLUM; IRANI, 2009) implements an absolute plus relative cursor controller using a mobile phone touchscreen. The movement of the finger while holding the touch triggers a relative movement, identical to an ordinary touchpad, but a quick tap on the screen triggers a jump of the cursor to the location defined by an absolute mapping with the external display (Figure 3.3 depicts the mentioned procedure). We implemented this approach and further compared with *LOP-cursor* for 2D interfaces (Section 4.1.1) at the Section 6.1.2.

3.1.3 Classification Within the Design Space

Table 3.1 classifies levels of precision selection techniques according to the proposed design space. The last row shows the *LOP-cursor*, our proposed selection by levels of precision technique. Its detailed description is presented in the next chapter.

3.2 Selection by Progressive Refinement

Selection by progressive refinement proposes the breakdown of a selection task in "effortless" subtasks. It aims to avoid the attention and precision usually required by tra-

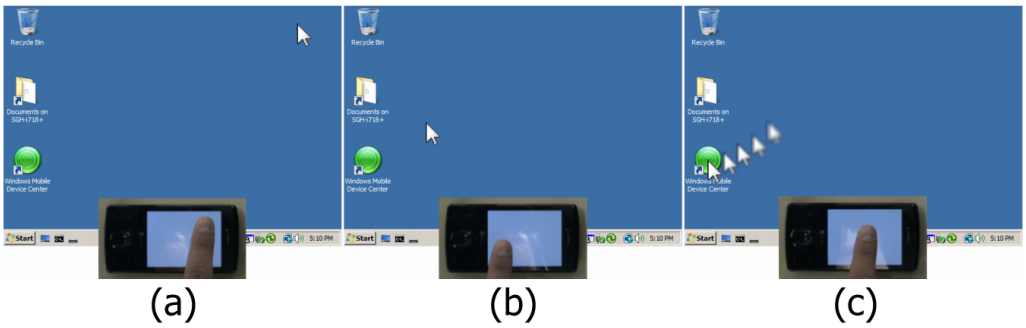

Figure 3.3: ARC-pad progressive refinement technique: (a) the cursor is on the top right corner of the screen. (b) A tap anywhere on the screen of the mobile device warps the cursor to the equivalent position. (c) Sliding the finger the user can accurately position the cursor (MCCALLUM; IRANI, 2009).

Technique	Levels	Level transition	Activation command	Level transition strategy
mouse/touchpad	$1 \rightarrow 2$	continuous	implicit	enforced by technique
PRISM	$1 \rightarrow 2$	continuous	implicit	enforced by technique
ARM	$1 \rightarrow 2$	discrete	explicit	determined by user
Vogel and Balakrishnan	$1 \rightarrow 2$ $2 \rightarrow 3$	discrete continuous	explicit implicit	enforced by technique enforced by technique
Forlines et al.	$1 \rightarrow 2$ $2 \rightarrow 3$	discrete continuous	explicit implicit	determined by user enforced by technique
Nancel et al.	$1 \rightarrow 2$ $2 \rightarrow 3$	discrete continuous	explicit implicit	determined by user enforced by technique
ARC pad	$1 \rightarrow 2$ $2 \rightarrow 3$	discrete continuous	explicit implicit	enforced by technique enforced by technique
LOP-cursor	$1 \rightarrow 2$	discrete	explicit	determined by user

Table 3.1: Techniques classification according to the selection by levels of precision design space

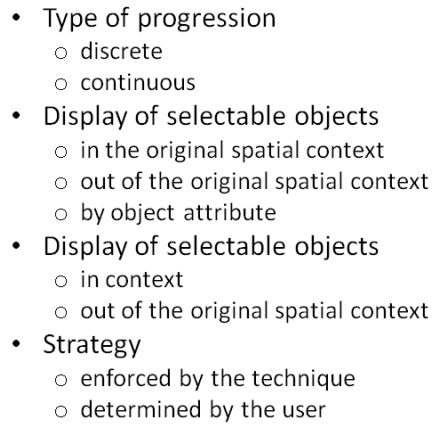

Figure 3.4: Selection by progressive refinement design space.

ditional selection techniques, so called immediate selection techniques. However, there is an inevitable tradeoff between immediate and progressive refinement selection techniques. Progressive refinement requires a process to complete a selection. This process usually consists of more than one quick subtask, resulting on higher accuracy. On the other hand, immediate selection techniques consist of performing the selection in only one step, slower and less accurate.

The expression selection by progressive refinement has been presented by Kopper et al. (KOPPER; BACIM; BOWMAN, 2011), described as an approach to progressively reduce the group of selectable objects and hence reduce required precision of pointing. The design space for selection by progressive refinement techniques was further expanded by Bacim et al. (BACIM; KOPPER; BOWMAN, 2013). It is presented in Figure 3.4.

3.2.1 Menu Disambiguation

Menu disambiguation techniques generally use a volume of selection on the initial phase, in order to reduce the effort of pointing into the desired object. Objects that fall inside or intersect the volume are then presented as a subset of objects using some sort of menu for disambiguation.

Dang et. al. presented earlier techniques that rely on menu disambiguation (DANG; LE; TAVANTI, 2003). These are called *transparent sphere* and *transparent cylinder*. With the *transparent sphere* technique, a positional cursor similar to the *virtual hand* metaphor is used to place a sphere volume of selection in space. Objects inside or intersecting the sphere have their name shown in a menu. Disambiguation is performed by selecting the desired object name. The only difference from transparent sphere to transparent cylinder is that in the latter technique a *ray-casting* based approach is used, where a cylindrical volume is attached along the casted ray in order to define the subset of objects. These techniques are depicted in Figure 3.5. Transparent sphere and cylinder use classical menu interface, presenting only the name of the target for disambiguation. Therefore, its original design is unsuitable for a series of applications.

Grossman and Balakrishnan (GROSSMAN; BALAKRISHNAN, 2006) proposed the *flower ray* for interaction with a volumetric display. *Flower ray* uses *ray-casting*, and disambiguates using a marking menu. When entering on menu disambiguation step, intersected objects animate towards the user viewport and spread as a marking menu, as shown in Figure 3.6. However, this technique still requires precision of pointing as relies

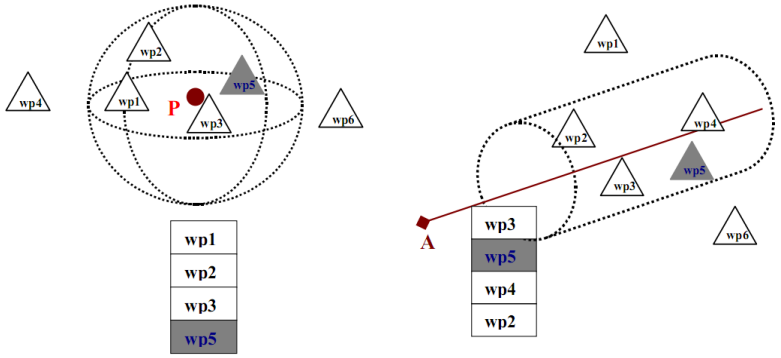

Figure 3.5: Transparent Sphere/Cylinder progressive refinement techniques. (DANG; LE; TAVANTI, 2003)

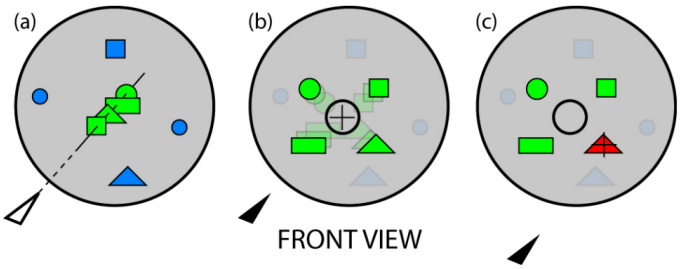

Figure 3.6: Flower Ray progressive refinement technique: (a) Intersected targets are highlighted. (b) Starting a button press rearrange targets in a marking menu. (c) Input device is used to point the desired target on the marking menu. (GROSSMAN; BALAKRISHNAN, 2006)

on Ray-Casting, and would have problems to disambiguate among a big subset of objects.

When proposing the taxonomy for progressive refinement selection techniques, Kopper et al. (KOPPER; BACIM; BOWMAN, 2011) also presented the SQUAD technique, sphere-casting refined by QUAD-menu. SQUAD consists of defining a subset of objects through their intersection with a sphere volume, and further refine the subset through QUAD menus, until only one object remains. The sphere position is controlled using sphere-casting, where the first intersection of a ray-casting defines the position of the center of the sphere in space. Figure 3.7 demonstrates the use of the sphere-casting and the QUAD menu. As SQUAD relies in several steps of disambiguation, we believe the major drawback of this approach is that the visual search will be repeated in each step. If the desired object is similar to others, visual search can be even more time consuming than the pointing task itself. This question was not addressed by the original study.

3.2.2 Zoom

Bacim et al. (BACIM; KOPPER; BOWMAN, 2013) propose two techniques for progressive refinement based on zoom, *discrete zoom* and *continuous zoom*. In order to avoid deformation, point of view discrepancy and resolution quality problems, both techniques use manipulation of the view frustum to control the zooming. In the *discrete zoom*, the user defines a quadrant of the screen he wants to see in more detail (Figure 3.8), the frustum changes so that specific quadrant covers all the FOV. In the *continuous zoom*

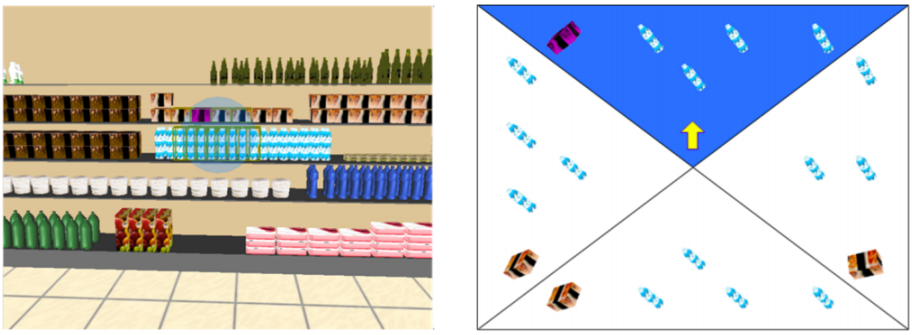

Figure 3.7: SQUAD progressive refinement technique: on the left, the casting of a sphere defines a subset of objects; on the right, QUAD menus are used to refine selection, in this case, at least one more disambiguation step will be required to select the purple object. (KOPPER; BACIM; BOWMAN, 2011)

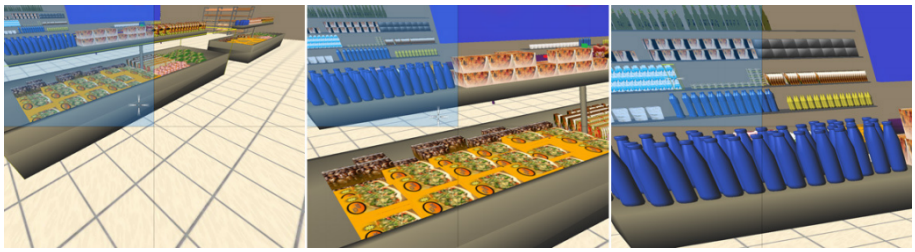

Figure 3.8: Discrete Zoom progressive refinement technique: the user can expand a quadrant of the screen so it occupy all the FOV. (BACIM; KOPPER; BOWMAN, 2013)

technique, the zoom is continuous at the pointed direction (Figure 3.9). Manually controlled zoom techniques tend to be more time consuming, indeed the evaluation presented by Bacim et al. showed worse performance than the SQUAD technique. However, it has the advantage of showing the objects in their original context.

3.2.3 Score Accumulation

Although not originally supported as progressive refinement selection techniques (KOPPER; BACIM; BOWMAN, 2011), we came to the conclusion that score accumulation techniques present the expected behavior described by the authors. These generally relies on the consistency of pointing, where objects that keep pointed for a larger duration

Figure 3.9: Continuous Zoom progressive refinement technique: the user can continuously zoom towards the pointed direction. (BACIM; KOPPER; BOWMAN, 2013)

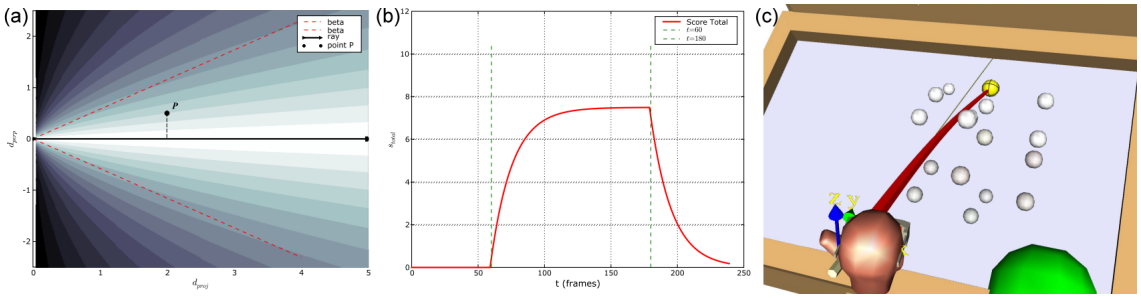

Figure 3.10: Intenselect technique: (a) plot of the scoring function relative to object distance from the cone center; (b) score accumulation according to frame, target enters in the cone in frame 60, and leaves the cone in frame 180; (c) visual feedback of the technique, the ray is snapped to the current high score object being represented by a curved cylinder. (HAAN; KOUTEK; POST, 2005)

will accumulate a larger score becoming the more likely to be the intended target of a selection.

Haan et al. (HAAN; KOUTEK; POST, 2005) use an approach similar to *lightspot* for the *intenselect* technique. However, on *intenselect* an alternative disambiguation function is applied and expanded to the dimension of time. Objects that fall inside a cone casting accumulate scores along time. The score increases according to its distance from the center of the cone, objects nearer to the center of the cone receive a bigger score. If the object stops intersecting the cone, its score is gradually lowered, as illustrated by Figure 3.10. Visual feedback of pointing is given by a bended ray connecting the start of the ray and the object with the higher score.

Grossman and Balakrishnan have implemented and evaluated the *smart ray* – technique based on (STEED, 2006) – on a volumetric display (GROSSMAN; BALAKRISHNAN, 2006). All objects intersected by the ray accumulate scores; to disambiguate the user moves the origin and direction of the ray so that it always intersects the intended object. As long as the user has been capable to maintain the ray over the desired object for more time than any other, he will be able to select it. *Smart ray* gradually decreases the score of the objects that have lost intersection with the ray. Therefore, it still maintains most of the score of objects that unintentionally lost contact with the ray for a short period of time.

3.2.4 Other Approaches

Steed and Parker have proposed *shadow cone-casting* (STEED; PARKER, 2004). *Shadow cone-casting* uses cone casting persistence of pointing along time to define a selection. When the user starts out a selection, all objects that are inside the cone are selectable. The user must disambiguate by moving the origin of the cone while trying to always maintain the desired object(s) inside the cone. If an object falls outside the cone, it will be cut out this selection process. This technique allows the selection of multiple targets. However, it relies too much on proximity of objects for this. Additionally, this technique may require high precision and be time consuming in a cluttered environment.

Grossman and Balakrishnan have proposed the *lock ray*, which expands their own technique of *depth ray*. The *depth ray* uses hand forward and backward movements to disambiguate which of the objects intersected by *ray-casting* will be selected. In *lock*

ray, these steps are performed in sequence, assuring higher precision control as the *ray-casting* becomes locked, while in *depth ray* they are performed simultaneously. Both techniques still require high precision of pointing in order to hit a target with the *ray-casting* metaphor.

3.2.5 Classification Within the Design Space

Table 3.2 classifies progressive refinement selection techniques according to the design space proposed by Bacim et al. (BACIM; KOPPER; BOWMAN, 2013). The last row classifies our proposed selection by progressive refinement technique, the *disambiguation canvas*.

Note that menu disambiguation techniques present the objects out of their original spatial context, while Zoom and accumulation techniques present and disambiguate on the original context.

Technique	Type of Progression	Refinement Criteria	Display of Selectable Objects	Strategy
Transp. Sphere	discrete	out of context	out of context	enforced by technique
Transp. Cylinder	discrete	out of context	out of context	enforced by technique
Flower Ray	discrete	out of context	out of context	enforced by technique
SQUAD	discrete	out of context	out of context	enforced by technique
Disc. Zoom	discrete	in context	in context	determined by user
Cont. Zoom	continuous	in context	in context	determined by user
Smart Ray	continuous	in context	in context	enforced by technique
Intenselect	continuous	in context	in context	enforced by technique
Shadow CC	continuous	in context	in context	enforced by technique
Lock Ray	discrete	in context	in context	enforced by technique
Disamb. Canvas	discrete	out of context	out of context	enforced by technique

Table 3.2: Techniques classification according to the selection by progressive refinement design space

3.3 Mobile Devices to Control Large Displays

Mobile devices, like smartphones and media players, are potentially the nearest to the ubiquitous *tab* concept (as proposed by Mark Weiser (WEISER, 1999)) we dispose nowadays. Ballagas assumed that the smartphone is the first really pervasive computational device (BALLAGAS et al., 2006), being an essential part of contemporaneous life and an always on pocket device. Therefore, its use as a general input/output device is quite obvious. Most relevant works using mobile devices to control external large displays are based on optical analysis, usually relying on optical flow pattern recognition and external optical tracking of the device . Other works also use mobile devices sensors on less specific tasks.

3.3.1 Optical Based Techniques

Ballagas et al. are precursors on mobile phone interaction with large displays, introducing the *Sweep* and *Point and Shoot* techniques (BALLAGAS; ROHS; SHERIDAN, 2005). Sweep uses the optical flow to move a cursor on the screen, with a central button as a clutching activator. Point and Shoot uses a quick blink of bi-dimensional tags on the controlling screen synchronized with the camera capture order, allowing reconstruction of camera pointing center. Jeon et al. have also implemented an optical flow control technique, but also provide other two marker based continuous tracking techniques: one based on a cursor placed marker, and the other based on object placed marker (JEON et al., 2006). Jiang et al. (JIANG et al., 2006) use the two last on screen cursor positions to define a coordinate system, allowing the cell phone to calculate the new position the cursor should assume.

Pears and Olivier introduced a technique for registration of mobile phone and external displays using four square markers (PEARS; JACKSON; OLIVIER, 2009). Registering allows direct mapping of every pixel at the large display on the mobile phone screen, and thus, direct control is provided. Boring et al. developed the Touch Projector, extending interaction with video to mobile devices (BORING et al., 2010). Touch Projector uses a polygon comparing algorithm to identify the screen where the user is aiming, thus allowing the control over multiple and spread displays. Touch Projector also suggested improvements to the video interaction concept, based on mobile device specific constraints and needs. Later, Boring et al. extended Touch Projector to interact with media facades (BORING et al., 2011). LightSense (OLWAL, 2006) uses a mobile phone with a back LED over a semitransparent table and track its two dimensional position using an external optical tracking system. The LED diffusion over the table is also used to distinguish across ten levels of distance between table and device.

None of the presented techniques deal with the precision issue at the level we are proposing. Presented pointing techniques are all based on camera tracking and/or ray casting with low cost approaches, thus resulting in less precision of pointing.

3.3.2 Movement Sensing Based Techniques

The motion sensors embedded in the newer smartphones allow for a number of gesture-based input modalities. Specific pointing devices (Wiimote, gyro mice etc.) make extensive use of these sensors, but generic smartphones have not been widely explored for pointing with embedded sensor tasks. WYSIWYF (SONG et al., 2011) uses accelerometer readings and prongs contact with a smart board to position and orient a volume cutting plane. Touching the smart board with the mobile device creates a virtual plane that starts from the contact points between the mobile device and the smart board which have the same orientation of the device. Katzakis and Hori (KATZAKIS; HORI, 2009) used mobile device accelerometers and digital compass to orient 3D objects applying a direct mapping of orientation across them. On a comparative test, the mobile device approach performed better than mouse and touchpen input. Other works used movement sensors for tasks like distinguishing devices on a multi-touch display wall using tilt correlation (HUTAMA et al., 2011).

4 PROPOSED TECHNIQUES

Here we will describe the *LOP-cursor* and the *disambiguation canvas*, our proposed techniques. The *LOP-cursor* is based on the earlier presented *levels of precision* concepts, it was first developed for general interaction with large and high resolution displays addressing pointing *accuracy* problems (DEBARBA; NEDEL; MACIEL, 2012) (paper available in Appendix A). After performing its evaluation, as presented in Section 6.1, *LOP-cursor* was further extended for interaction in immersive environments, for which we had addressed *complexity* of selection problems, such as described in Section 2.3. *Disambiguation canvas* is based on the *progressive refinement* concepts, and was also developed to overcome *accuracy*, *ambiguity* and *complexity* of selection problems. It allows the selection of a subset of objects and uses a menu like pointing for disambiguation. Our approach scales well, allowing the disambiguation among hundreds of objects with only one step.

Furthermore, this Chapter also presents the *two-legged cursor* metaphor, which is a metaphor we propose for easy and quick pointing of two simultaneous locations. It addresses completion of composed tasks with only one selection, and is mainly intended for multi-display and multi-computer environments.

4.1 LOP-cursor

As the *LOP-cursor* was initially developed for a monitor tiled-display and interaction with 2D interfaces, it is first explained regarding these situations. Furthermore, we present its adaptation for selection on 3D applications. However, when interacting with a head tracked environment or a stereoscopic display, these modifications also impose some discontinuities and inaccuracies that need further considerations, which are presented below.

4.1.1 Overview of the 2D LOP-cursor

The LOP-cursor technique is based on two levels of precision. In the first level, the user points the device towards the target on the screen, which results in moving a rectangular control canvas containing the cursor arrow to that location (see Figure 4.1a). If the user succeed on pointing at the desired target with the arrow, he can complete the selection by a quick *tap* gesture. Otherwise, a fine tuning can be done by sliding a finger on the device touchscreen, moving the cursor arrow inside the control canvas (see Figures 4.1b and 4.1c). Different from most multiple levels of input techniques, LOP-cursor uses absolute mapping in both levels, ray casting for the first level, and a rectangle representing the physical device touchscreen at the large display. The information contained in Figure 4.1d regards the *two-legged cursor* metaphor we are also proposing and is addressed

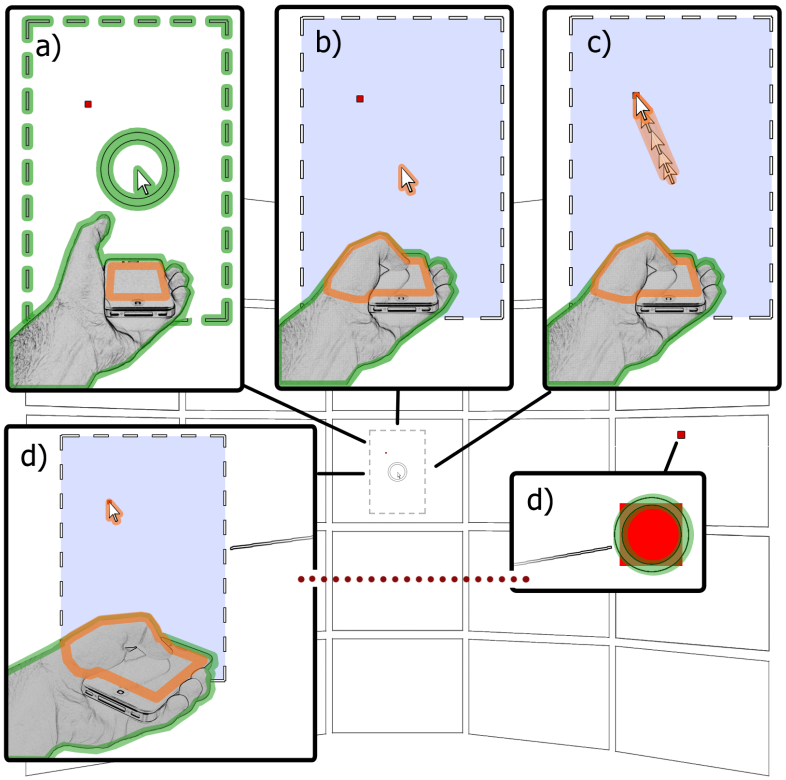

Figure 4.1: LOP-cursor usage to select and move objects on a tiled display: (a) the user hovers the target neighborhoods with an arrow pointer performing a 3D gesture with a mobile device in the hand; (b) touching the device touchscreen locks the position of a rectangular control canvas; (c) fine tune and precise selecting the target object with the arrow in the control canvas is achieved moving the thumb on touchscreen; (d) holding the control canvas locked, the user can move the arrow cursor by touchscreen on the mobile device while moving their hand to point at a second place with a secondary cursor (ring cursor) to define a destination target to the selected object. Green marks indicate actions made using movement sensors, while orange marks point out actions done on the touchscreen. The image background illustrates the LOP-cursor graphic representation on a tiled display.

in Section 4.3.

4.1.2 Technique Design

4.1.2.1 Graphic representation

Four complementary shapes are used to represent the mobile device at the external display. Two are needed to perform a selection, the arrow, which is a common point cursor metaphor for desktops, and the rectangle, which depicts the boundaries of the absolute mapping between the touch screen and an area of the external display. A ring and a line are used as additional shapes to avoid losing track of the pointing direction.

Arrow – as our techniques are intended to provide high precision of pointing and selection, a traditional point selector over an area selector was preferred for our main cursor design. We used the well known arrow shape to represent this cursor, which minimizes ambiguities. We believe that using the arrow to highlight the final position of selection helps users to immediately differentiate it from the other shapes we used.

The arrow opacity depends on the size of the nearest target to the cursor, the cursor size itself, and the distance between the target and the arrow center. Using adaptive opacity avoids the occlusion of very small targets by the arrow and allows users to keep track of the target whenever it is required. Figure 4.2 illustrates the arrow and its opacity function.

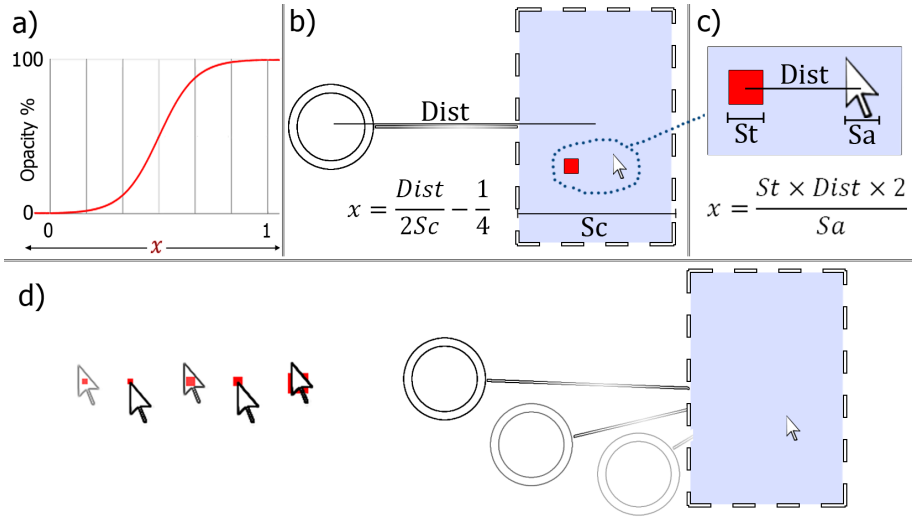

Figure 4.2: A logistic function maps the arrow and ring opacity according to the x variable value (a); x for the ring depends on distance between the center of the ring and the center of the rectangle $Dist$, and the width of the rectangle Sc (b); x for the arrow depends on distance between center of the target and center of the arrow $Dist$, and width of arrow Sa and target St (c); arrow and ring opacity sample results for a variety of target sizes and positions, and rectangle and ring distances (d).

Rectangle – a rectangle is used as the representation of the mobile device touchscreen on the display. It defines a control canvas within which the arrow is. An absolute mapping is used between the device's rectangular touchscreen and the control canvas, e.g. a touch at a location near a corner of the touchscreen results in placing the arrow at the same corner of the control canvas. This absolute approach makes the rectangle a natural choice for the virtual representation of the touchscreen. More important, this approach allows for the control canvas representation to contain an input resolution equivalent to that of the device's touchscreen sensing resolution (480 x 320 in our implementation). The rectangle

has also the same aspect ratio of the touchscreen, making the correspondence explicit to the user.

One could note that there is a discontinuity of orientation between the rectangular representation and the touch screen. The user tends to hold the device with the *touchscreen* facing up, while the rectangular shape will be facing the user. It is the same discontinuity noticed on *touchpad* and *mouse* devices. During the performed evaluations, including preliminary evaluations, none of the participants have reported problems or any difficulty regarding this $\approx 90^{\circ}$ difference. Thus we have assumed that there is no need to address this as an issue.

We have also made considerations regarding whether relative or absolute mapping would be the best approach. Relative mapping is expected to provide higher precision as it uses dynamic control of CD gain, thus refining the motor space again (it adapts the cursor response according to input speed). Relative mapping would not need the rectangular representation and could also allow the access of the whole displays space if clutching is implemented, but it prompts once more problems such as loosing track of the cursor. On the other hand, absolute mapping is more intuitive as finger input to cursor response is easier to predict (by means of matching the touchscreen shape with the rectangular representation), thus it is expected to perform faster. Additionally, CD ratio may be set by rescaling the rectangular representation, the bigger the rectangle is, the bigger the cursor movement range becomes (reducing precision), and vice versa. Both alternatives were implemented and subjectively evaluated by our group, we found that the absolute mapping is very consistent, but yet less explored, thus we preferred to stick with this approach. Section 4.1.2.3 describes our rectangle rescale approaches that affect the CD ratio of the absolute mapping.

Ring – the ring indicates the physical pointing direction of the device. It always follows the device's orientation, enabling the user to keep track of where they are pointing to. This is particularly relevant when the user enters the second level of precision of the LOP-cursor. In this occasion, when the focus of the user is on the *arrow*, involuntary hand movements eventually take the ring to locations far from the control canvas and the arrow. After release, they may lose his reference as the main cursor becomes, once again, controlled by the pointing direction, being warped to the current position of the ring pointer. Experiments show that while this jump does not seem to upset the user, it can cause disorientation when the ring pointer is not displayed.

To avoid distraction caused by continuous movement of the ring near the arrow, we used an adaptive opacity factor. The ring opacity is proportional to its distance from, and size of, the control canvas. The nearer the ring is from the control area, more transparent it is. Figure 4.2 illustrates the concepts above.

Line – a line is used to connect the control canvas (rectangle) and the ring. It aids the user to keep track of the relative position of the arrow and the ring, keeping awareness of where they are currently pointing to. The line is especially useful for the *two legged* cursor metaphor described in Section 4.3. Its geometry guides searching direction, while its color intensity inform the distance between legs.

While in the first level of precision, which consists of pointing through *ray-casting*, both the arrow and ring pointers follow the mobile device pointing direction. The arrow is kept at the center of the ring, and a semi-transparent rectangle with dashed borders is shown to keep the user aware of the size of the control canvas. As the control canvas and the ring cursor are superposed and move together, the line is not visible in this mode.

When switching from first to the second level of precision, a transition effect is played

to guide user attention across states. The effect renders the control canvas with more opacity, and ring transparency is controlled as described in Figure 4.2, giving a hint of switching modes. The inverse effect is played when returning to the first level of precision.

In preliminary tests, due to absolute pointing within the control canvas, the arrow cursor often jumped when entering in the second level of control (Figure 4.3b). This caused a discontinuity that led the users to report some disorientation. To overcome this issue, in the final design of the LOP-cursor the arrow is kept in place and the rectangle (canvas) is placed accordingly to compensate the distance of the touch from the center (Figure 4.3c). Although this causes a small motion discontinuity for the canvas (it is small because the users tend to touch at the center of the device touchscreen), it was preferred than a discontinuity of the arrow as the user focus is on the arrow.

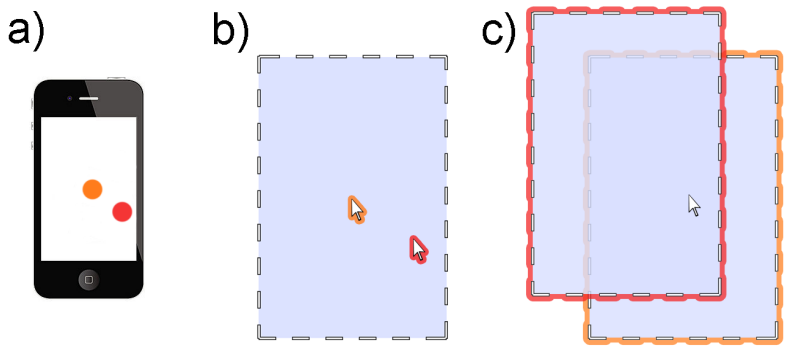

Figure 4.3: LOP-cursor first to second level switching discontinuity: (a) represents the start position of two distinct touches, a red and an orange highlighted; (b) demonstrates what happens on each touch if the rectangle is kept in a fixed position, forcing the arrow to warp inside the rectangle; (c) demonstrates what happens to each touch if the arrow is kept fixed and the warp is applied to the rectangle. We have found that the discontinuity in (c) is less detrimental to the LOP-cursor.

4.1.2.2 Confirmation of Selection

The use of touchscreen for fine tuning on the second level added constraints to the selection design. For instance, as the user is sliding the thumb over the touchscreen, we cannot use a *tap* gesture on the screen to confirm a selection, which is the most obvious action. To explore and understand such constraints, we implemented and tested five different confirmation of selection actions, three of them based on software (*take-off*, *second-finger-tap*, and *tap-after-take-off*), and two using software combined with hardware (*tap-on-back-touch-surface*, and *back-click-on-adapted-mouse*).

The *second-finger-tap* requires the use of two fingers: while one finger is used for fine tuning, another is used to confirm the selection, tapping on the touchscreen. Since the fine tuning is a very precise action, tapping with another finger of the same hand or from the other hand at the same time is very difficult without moving the first finger, leading to mistakes. Thus, this technique was abandoned.

The *take-off* performs a selection on the last position defined before the end of the touch. It has the drawbacks of always resulting on the release of the second level of precision and being susceptible to accidental selections. With *LOP-cursor*, if the second level of precision starts over an object that covers all the mapped rectangle and no escape method is provided, the selection command becomes inevitable.

In the *tap-after-take-off* technique the user touches the screen, slide the finger to select the target and then take off the finger and perform a tap right after to confirm the selection. The drawback is that the control canvas is unlocked by this action. These three described *confirmation of selection* techniques do not need any attachment to the mobile device, such as depicted by Figure 4.4a.

With the *tap-on-back-touch-surface* technique, the user confirms the selection by tapping on a touch surface on the back of the mobile device. This requires hardware adaptation, we did it using a second mobile device on the back of the main one (Figure 4.4b). This design is inspired by the recent rise of mobile devices with backtouch surfaces, such as the PS vita.

The *back-click-on-adapted-mouse* is quite similar to the previous technique. The idea of using the back of the device to confirm a selection is kept. However, the touch surface was replaced by a button (from a classic mouse), and the confirmation is made by pressing this button (Figure 4.4c). Furthermore, we were inspired by some touch devices that allows a "click" by pressing its surface, such as the Apple MacBook touchpad and a few mobile devices. The closest we came to this latter idea was attaching an Apple Magic Mouse to the back of the device, but the area of click got limited and false positive often occurred, thus we stick with the ordinary mouse button (Figure 4.4c). Back of device button input was earlier explored by Wigdor and Balakrishnan, using a chording keyboard (WIGDOR; BALAKRISHNAN, 2004).

We investigate and evaluate the latter three configurations in detail in Section 6.1.1.

Nonetheless, *LOP-cursor* also supports the immediate selection of an object if the user decides that he can perform it without the second level of precision (e.g. no fine tuning). For this, a confirmation of selection trigger needs to be available at this first level of pointing. We have used a *tap* gesture to allow the immediate selection, which consists of starting and finishing a touch in less than 250 milliseconds, without significantly move the thumb over the touchscreen for the duration of the touch. This observation is also true for the selectin technique presented in Section 4.2.

Figure 4.4: Three hardware configurations for selection action: (a) for *take-off*, *second-finger-tap* and *tap-after-take-off*. (b) iPod attachment for *tap-on-back-touch-surface*. (c) Adapted mouse attachment for *back-click-on-adapted-mouse*.

4.1.2.3 Rectangle Size Control

As we are using an absolute mapping between user's finger position on the tochscreen and the cursor position within the rectangle, the user have his reach of the total scene limited to the size of the rectangular representation, as well as precision limited to an unit of touch sensing remapped to the display. Furthermore, there is a tradeoff involved with these two constraints, that is, favoring one harms the second. The bigger the rectangle is

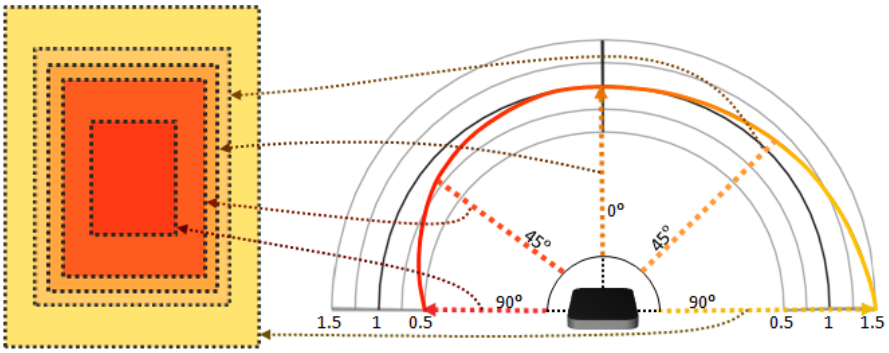

Figure 4.5: LOP-cursor rectangle can be dynamically rescaled while using the first level of precision. The quadratic function used for scale is plotted in a radial graphic aligned with its originating orientation.

– and as consequence the CD ratio between touchscreen and the rectangle –, the easier is to include the intended object inside the rectangle, while pointing using the touchscreen becomes harder. On the other hand, the smaller the rectangle is, the harder it is to include the intended object in the rectangle, while touch precision increases.

To address these limitations we have chosen a balanced standard setting, and propose two simple rescaling techniques. The standard size is based on the data provided by Argelaguet and Andular (ARGELAGUET; ANDUJAR, 2009), who states that in their user evaluation, no more than $4°$ of pointing error has happened during the ballistic pointing phase. Thus, we decided to use the angular size of $8°$ horizontal, and $12°$ vertical (to maintain the aspect ratio).

Concerning the two rescale techniques. The first uses *pinch* and *stretch* gestures, while the second uses the rotation around the mobile device vertical axis. With *pinch* and *stretch* gestures, a relative mapping is used, for each update, the difference from the past distance to the current distance between the fingers is used proportional to the diagonal of the screen to increase the current size, described as follows: $newSize = currentSize * (1 + \Delta distance/diagonal)$. Regarding the second rescale technique, as the rotation around the mobile device vertical axis is not necessary for pointing, we decided it could be used for fast control of the rectangle size, similar to the idea proposed by Forlines et al (FORLINES et al., 2005). The device's resting orientation is setted as the touchscreen facing up; an offset can be applied through calibration if the resting orientation is uncomfortable to the user. The resting orientation is equal to $0°$ in the mapping depicted by Figure 4.5. Counterclockwise rotation scales down the rectangle up to a factor of 0.5, while clockwise rotation scales it up to the factor of 1.5. We have used the non linear function $f(x) = 1 + (x * |x|)/2$ to define the rescale factor, x ranges from -1 to 1, using a linear mapping from $90°$ clockwise until $90°$ counter clockwise from the resting orientation. Figure 4.5 illustrates the function in a radial graph, aligning the orientation and the result obtained with this mapping.

The rescaling through orientation feature is only possible while controlling the first level of the *LOP-cursor*. As the second level is intended for high precision, rescaling the rectangle would cause the touchscreen/rectangle mapping to dynamically changes, resulting on undesired movements of the arrow.

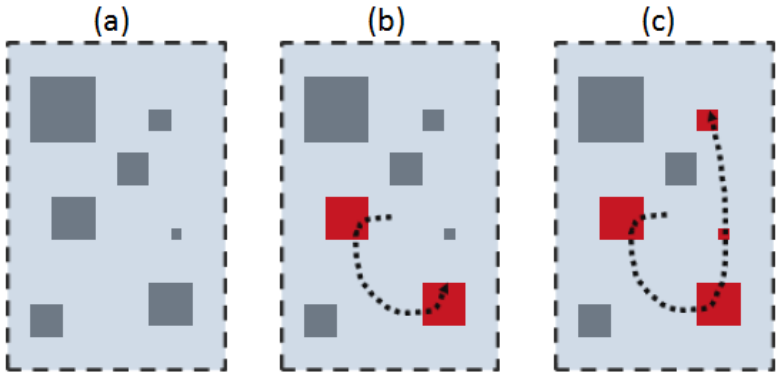

Figure 4.6: LOP-cursor free form selection for multiple objects: (a) the user enters the second level of precision, and hold the *click* button; (b) dragging the thumb around the screen draws a free form selector on the screen, objects intersected by the form are selected; (c) when the user is satisfied with his selection, he releases the click button.

4.1.2.4 Selection of Multiple Objects

For the simultaneous selection of multiple objects, we propose a freeform tracing tool controlled by the touchscreen. This technique only works for the *back-click-on-adapted-mouse* confirmation of selection technique. While controlling the second level of precision, the user may start and maintain a *click*, and draw a freeform. When the *click* is released, objects intersecting the drawn freeform are selected. Drawing over the touchscreen is a common task on mobile devices, although with our technique the user will not be looking to the mobile device screen (but instead to the external display), the absolute mapping still gives intuitiveness on predicting the result of the thumb movement for drawing. An expansion of our cursor for a drawing tool has been proposed by Silva and Nedel (SILVA; NEDEL, 2011). Their work allowed the replication of the image inside the rectangle on the mobile device display. The procedure to perform a free form selection is depicted by Figure 4.6.

4.1.3 LOP-cursor 3D Generalization

Although we demonstrated the LOP-cursor on a 2D virtual environment using a 2D display, the *indication of object* for the first level of precision is performed using the ray-casting metaphor, which is a 3D pointing technique. However, as all the objects are disposed over the same plane, all of them have the same depth, facilitating the control of this 3D pointing technique. For instance, the eye-hand mismatch problem reported in Section 2.3.2 is not a major issue for the *LOP-cursor* 2D.

When interacting with a 3D virtual environment, objects will be located behind the screen, having multiple depths. We may still use ray-casting, and refine its angle of pointing with the second level of precision. However, we preferred to use a combination of ray-casting over the screen plane – or a virtual plane at a predefined distance from the user –, and indication of object by *image plane occlusion* using the tip of the arrow. This is equivalent to cast a ray from the eye of the user, passing through a midpoint controlled by the LOP-cursor intersection with a plane. Figure 4.7 demonstrate the discrepancy of pointing between the ray-casting from device directly into the objects, and the image plane occlusion point of the tip of the arrow.

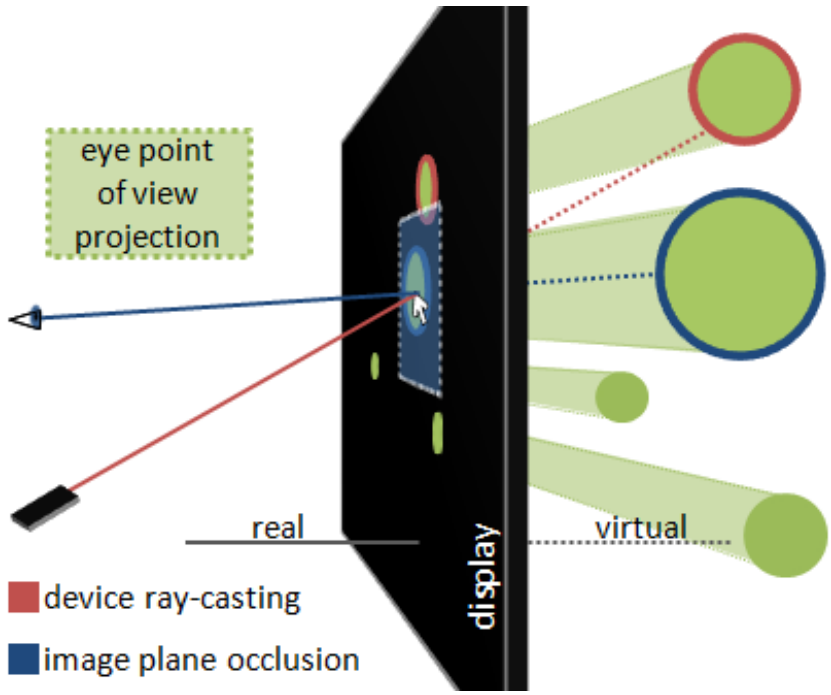

Figure 4.7: LOP-cursor 3D overview. Note that indication of object is performed by occlusion. The pointing direction of the device is used to intersect the display plane – or another predefined plane – which gives a midpoint for a ray leaving the eye. Final pointed object is the one occluded by the tip of the arrow, not the target intersected by the prolongation of the ray.

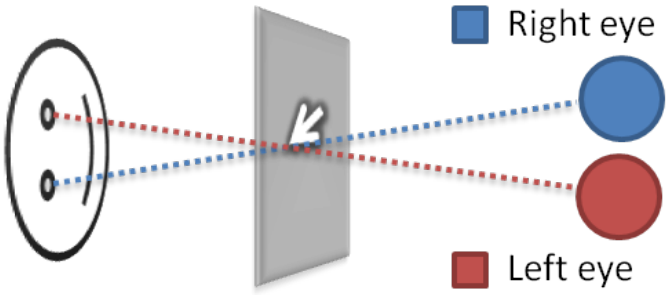

Figure 4.8: Stereoscopy mismatch when using image plane selection. As stereoscopy generates two distinct image planes, selection is usually solved favoring the pointing performed on the image of the dominant eye rather than on the image of the non-dominant eye, or drawing the cursor only on the dominant eye image.

The description above addresses the indication of object for a non immersive 3D virtual environment. There are two main issues that must be considered for the generalization of the LOP-cursor for immersive 3D virtual environments. The first is related to the stereoscopic display, as we use a plane to position the LOP-cursor in the virtual space, selection is performed through a second ray that leaves from the eye, and passes through the current position of the arrow cursor. However, two distinct images are generated, one for each eye, which eye should the ray be casted from? The second is the head tracking, which would cause instability of selection even when using the second level of precision of the LOP-cursor due to perspective change. These issues are solved using the *viewfinder* technique (ARGELAGUET; ANDUJAR, 2009), and an extension to it that we have named as *snapshot*.

4.1.3.1 Viewfinder

As the LOP-cursor floats on a fixed depth – intersecting the display plane, or an imaginary pre defined plane –, the object underneath the graphical representation of the cursor could vary on a stereoscopic display, result of generating an image plane for each eye. Solutions usually rely on favoring the dominant eye, as implemented by the Aperture technique (FORSBERG; HERNDON; ZELEZNIK, 1996). In such case, selection is correctly perceived by the dominant eye, while the non-dominant eye sees the arrow in an inconsistent position (Figure 4.8). Some authors suggests that the user may close the non dominant eye if the discontinuity is disturbing, or that the cursor can be drawn only for the dominant eye, which would result in visual inconsistency (PIERCE et al., 1997; FORSBERG; HERNDON; ZELEZNIK, 1996).

To overcome the stereoscopic display discontinuity, we have adopted the *viewfinder* technique (ARGELAGUET; ANDUJAR, 2009). As proposed by Argelaguet and Andujar, *viewfinder* solves the problem of stereoscopic vision pointing mismatch by rendering a floating rectangle that presents the same image for both eyes. It reuses that rectangular portion of the image generated for the dominant eye on the image of the non-dominant eye, as depicted in Figure 4.9. In practice, *viewfinder* mimics a digital camera visor aligned with the dominant eye. *Viewfinder* have also outperformed other commonly used forms of visual feedback and take advantage of the *LOP-cursor* already in use rectangular shape.

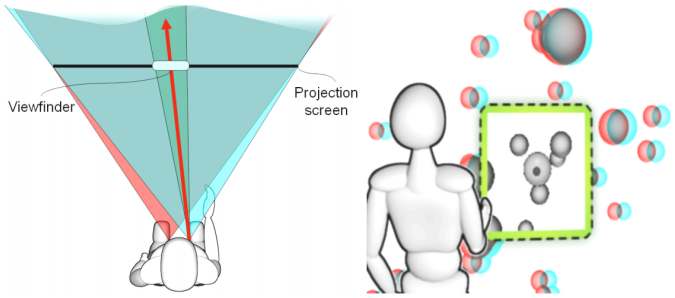

Figure 4.9: Viewfinder technique creates a window where the scene is rendered without stereoscopy, the image of the dominant eye is replicated for both of the eyes in that window. It mimics the behavior of a digital camera on the real world.(ARGELAGUET; ANDUJAR, 2009)

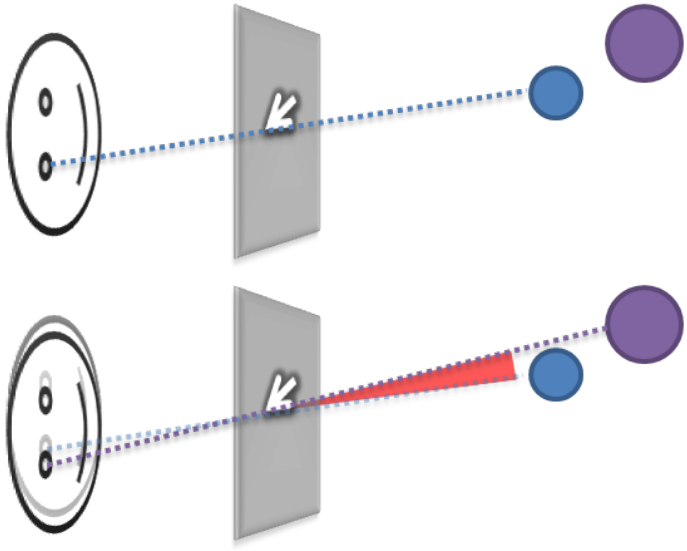

Figure 4.10: Head tracking inaccuracy may cause the pointing of the incorrect target.

4.1.3.2 Snapshot

Furthermore, head tracking is another factor that can cause accuracy problems. As we are using *indication of object* by occlusion, modifying the point of view would constantly affect the final point of selection, as depicted by Figure 4.10. To overcome this issue we decided to go further with the digital camera metaphor, so that when entering in the second level of precision the user will take a *"snapshot"* of the scene portion inside the *viewfinder*. The *snapshot* is kept until leaving the second level of precision. This approach is also used to overcome selection of animated objects, as reported in Section 2.3.3.

4.2 Disambiguation Canvas

Disambiguation canvas is a selection by *progressive refinement* technique that consists of defining a subset of objects on the first phase, with a subsequent step for disambiguation. It also uses the orientation sensors and the touchscreen of the mobile device.

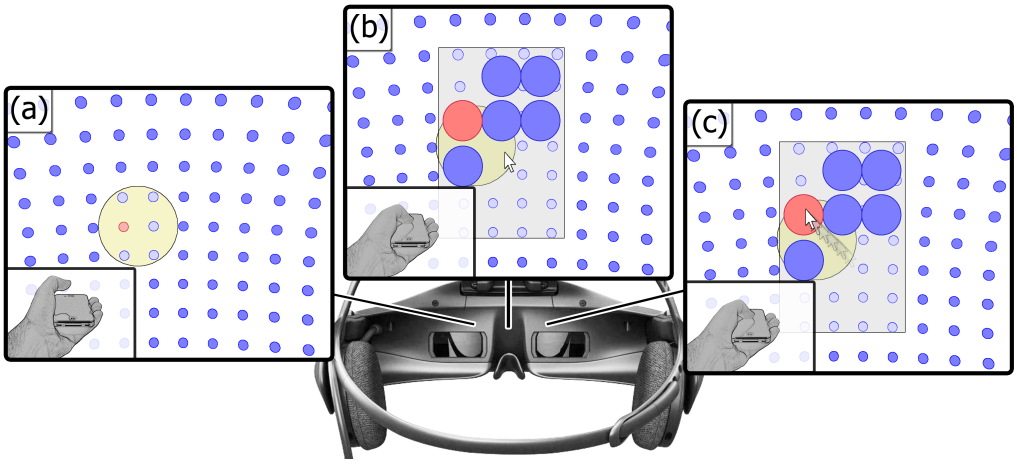

Figure 4.11: Disambiguation canvas walkthrough: (a) the user points to the region where the desired object is located; (b) starting a touch rearrange the subset of objects pointed by the volume casting technique over a selection canvas; (c) the canvas has an absolute mapping to the mobile device touchscreen, the user slides his thumb in order to point out the desired object. As the user may not see his hand when using an immersive display, hand inserts illustrates the performed hand gestures.

4.2.1 Overview

For explaining purposes, the following overview assumes the use of the *disambiguation canvas* on an immersive display. However, it is applicable to other visualization environments.

The *disambiguation canvas* is based in two steps. In the first step, the user uses a volume casting technique to point in the direction of the selected object (Figure 4.11a). Just like with the *LOP-cursor*, the user may trigger a selection if he was able to point out the desired object, the disambiguation of objects for immediate selection is performed according to its distance from the center of the pointing direction. When the intended object is inside or intersecting the volume of selection, the user may start a touch to enter the second step of the selection. A rectangle appears at the distance of 70 cm from the user, it is aligned parallel with the image plane – or with the mid orientation of the two image planes when stereoscopic rendering is in use – all the pointed subset of objects moves in an animation to form a matrix inside this rectangle (Figure 4.11b). The rectangle has a 1:1 mapping with the mobile device touchscreen, sliding the thumb over the touchscreen allows the superposition of the desired object by the arrow (Figure 4.11c). Selection is performed by a *take-off* gesture, returning to the first step of the technique. If the user wants to leave the disambiguation phase without selecting any object, he simply performs a *take-off* gesture over an empty point.

While a regular immediate selection technique usually has its difficulty of pointing increased by the reduction in size of the desired object, using *disambiguation canvas* the difficulty increases according to how many objects have been pointed by the volume-casting during the first step. As in many other techniques of selection by progressive refinement, this will make the refinement slower and harder. However, as the *disambiguation canvas* rely on the mobile device touchscreen for disambiguation, we are able to align hundreds of objects for an unique disambiguation step while still assuring high precision. During the technique evaluation, we have used three distinct object densities, such as depicted

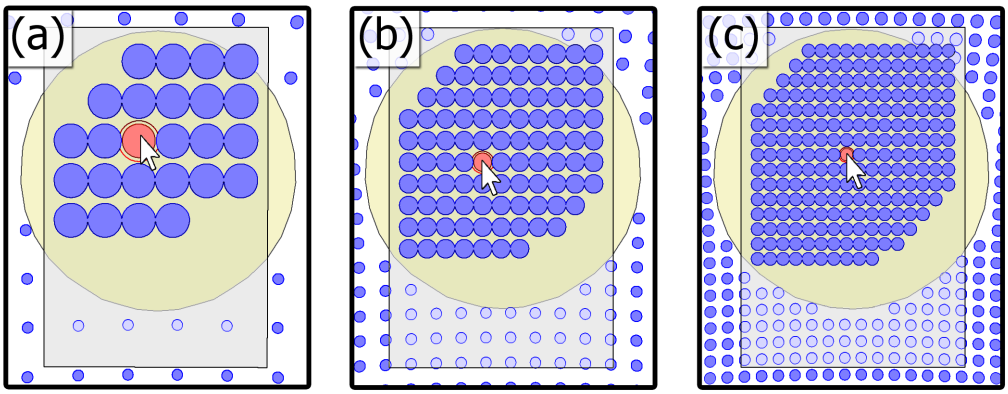

Figure 4.12: Density of objects on the disambiguation canvas: difficulty of selection is proportional to the amount of objects in the subset defined by the first step; in (a) there are 25 objects on the canvas; in (b) there are 97 objects; in (c) there are 224 objects.

by figure 4.12. For the worst case depicted in the Figure 4.12c, 224 objects went to the disambiguation phase. Yet, our technique still offers a sensing space of 19×19 of the total resolution of input provided by the mobile device touchscreen, which is 320×480. Using *SQUAD* for instance (described in Section 3.2.1), the user would need to complete four disambiguation steps to obtain the desired object from a subset of 224 objects.

4.2.2 Technique Design

4.2.2.1 Volume casting techniques

The most common approaches for volume casting are the *cone-casting* and the *sphere-casting* techniques, which one of these is the best fit for our technique may depend on the application. Thus we decided to support both volume casting approaches for our techniques.

Using the cast of a sphere it is likely that the amount of objects intersecting the volume will be smaller, as the sphere has a limited depth. However, it is also harder to control the first step of the selection as the depth must be somehow provided; there are two common approaches to determine the sphere depth. The first uses the near intersect of the sphere, and set its distance to the intersect position. The second casts a ray through the center of the sphere; the distance of the first intersection of this ray is used to set the sphere depth. In *SQUAD* for instance, Kopper et al. (KOPPER; BACIM; BOWMAN, 2011) favored the *sphere-casting* with the depth of the sphere determined by *ray-casting*, but this may be due to the type of environment for which they developed the technique, which was a virtual supermarket application. In the supermarket environment the objects were very cluttered and organized as stacks in many shelves.

Cone-casting overcomes most of the problems reported in Section 2.3. It allows reaching objects even if an intersection occludes them from the device point of view, which sphere-casting is unable to do. It also allows more sophisticated disambiguation if the user wants to perform an immediate selection – selection not using the second step –, such as using the *intenselect* technique for the first step (HAAN; KOUTEK; POST, 2005). On the other hand, *cone-casting* may intersect too many objects if the scene is very cluttered. On the supermarket case, *cone-casting* would require constraints in order not to select objects behind the shelves, otherwise a huge amount of objects may fall inside the conic

volume.

For the standard angular size of the sphere and the cone we decided to use $12°$. The sphere always rescales to achieve the angular size of $12°$ to the casting position point of view. The cone uses a $12°$ of angular opening from its casting point. We allow the same rescale approaches described in Section 4.1.2.3, this way the angle may vary from $6°$ to $18°$ degrees through the orientation of the mobile device, and the standard size can be modified using *pinch* and *stretch* gestures on the touchscreen.

This technique can be easily generalized for 2D virtual environments using a planar shape instead of a volume, such as a circle or a rectangle, on the intersection point.

4.2.2.2 Graphic representation

As for the *LOP-cursor*, we use an *arrow* shape for the cursor, and a *rectangle* shape for the working canvas. These are only visible during the second step, which is the disambiguation step. Rather than using the *ring* for mobile device pointing direction, it is represented by a sphere or a cone, depending on the used volume casting technique. As for the *ring* shape on the *LOP-cursor*, the volume casting shape in use is always visible.

The rectangle uses an absolute mapping with the mobile device touchscreen. It is drawn to use $30°$ of the total $45°$ standard vertical FoV of the camera. We are drawing it $70cm$ away from the camera on our immersive display implementation, thus we get the size of $\approx 38cm$. However, in order to make better use of the FoV, this size should be decided according to the available display, allowing the user to inspect the objects more efficiently, and therefore reducing the visual search time.

4.2.2.3 Mapping Objects to the Canvas

When entering the second step of the *disambiguation canvas*, the subset of objects must be reorganized side by side over the canvas plane. As most users cannot reach the whole touchscreen area with the thumb, we propose the reduction of the useful area in which the objects will be reorganized, so that objects are brought within users reach. Consequences are that objects become smaller (harder to hit) and a significant sensing area of the touchscreen is useless (around half of it). However, as we still obtained very low error rates when disambiguation among 200 objects, we concluded this is a worth tradeoff.

We implemented two layouts: the first consists of $\approx 53.4\%$ of the total area, and is oriented to handedness, Figure 4.13a depicts this layout for right and left handed users. This layout takes 5% from right, left and up, and 25% from the bottom out of the useful area, as well as $1/8$ and $1/32$ of the remaining that is too near from the palm and far from the thumb respectively. The second layout consists of ≈ 42.4 of the total touch screen area. This layout takes 5% from right and left, 10% from up, and 30% from the bottom out of the useful area. The final layout consists of a circle inside the remaining area, as illustrated in Figure 4.13b. For the user evaluation we used the first layout.

When entering the disambiguation phase, a matrix fitting every pointed object inside the layout is computed, and each object is designated to a slot of the matrix. In order to fit every object inside their designed slot, the objects are rescaled so their bounding box do not trespass that space. However, this may make some visual attributes less apparent or even impossible to be perceived, such as when the user wants to select a target of specific size within a group of similar objects. To overcome this issue the rescale factor could also be proportional to the bigger target, or linearly remapped between a minimum and maximum size. Then if an object is a ten times smaller than other, this factor can

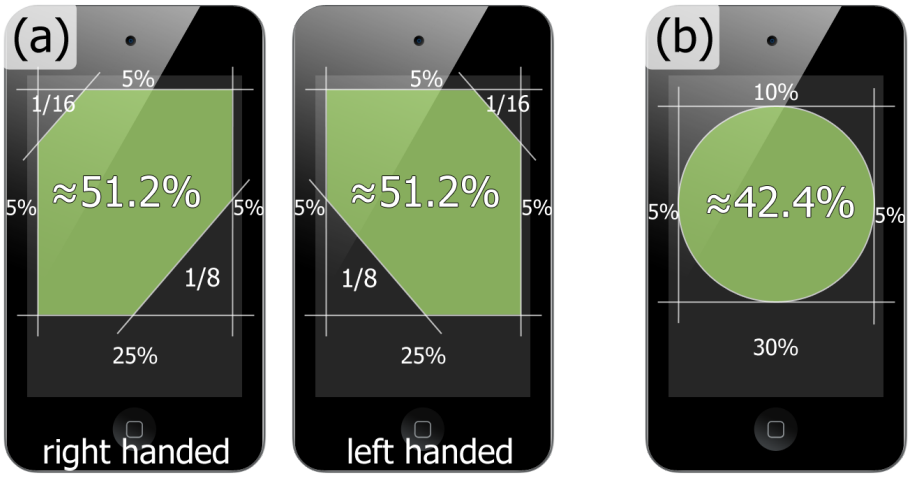

Figure 4.13: Standard layouts of useful touchscreen area proposed for the disambiguation canvas technique.

be lowered to a maximun of half of the size, and both objects will be still visible. By standard, our current implementation rescales all the objects to the same bounding box size.

Finnally, as for being a progressive refinement technique based in menu disambiguation, objects that go from the first to the second phase loose their original context. This could make it difficult to distinguish the intended object in real applications if they are very similar in shape or if the selection depends on their original topology. We propose three possible solutions for such limitation of the menu disambiguation approach.

The first solution is to control the instant of interpolation that animates the objects while bringing them over to the disambiguation menu. To cast a ray or a volume for pointing, only 2 degrees of freedom (DOF) – among the 3 provided by the device orientation – are required. Our proposal is to use the 3rd DOF to dynamically control the instant of the interpolation. The mapping from orientation into instant of interpolation can be achieved with an absolute relation, where a certain orientation always results on the same instant, or with a relative relation, where after a threshold the orientation controls acceleration forward or backward on the interpolation instant. This strategy was implemented, and showed to be functional; however it was not yet evaluated.

The second technique consists on duplicating the original object into the canvas – instead of moving the original – and using the copy superposed by the arrow cursor to highlight the original object. It can indicate whether the user is pointing to the desired object when there is more than one with the same or similar shape and color.

The third approach is to draw a trajectory curve to connect the original position of an object with its final position on the canvas. This approach allows the simultaneous observation of all connections between original and final positions at the same time. However, this can result in cluttering and may overwhelm the user with information if too many objects are taken for disambiguation.

Notice that these suggestions are not exclusive and can be combined among them. Given that this is a general problem of menu based progressive refinement selection techniques, we intend to investigate these approaches further in future works.

4.3 Two-Legged Cursor

Both of the techniques presented here use two distinct input hardware. These are the mobile device movement sensors and its touchscreen. During the development of the technique, we have noticed that users have no problem switching among them, furthermore, they are able to maintain some level of simultaneous control over both input methods. This observation gave rise to what we call here as *two-legged cursor*, which consists of allowing completion of composed tasks, such as drag and drop, defining two distinct pointing directions in one action.

The fine tuning/disambiguation by touchscreen freezes a control canvas for the duration of the touch, but do not stop the direct 3D pointing feature, which results in two simultaneous cursors under the control of the user. This design allows two simultaneous pointing locations (the two legs of the cursor) to quickly perform composed tasks as, for instance, drag & drop and distance measurements. Figure 4.1(a, b and d) presents the walkthrough on the simultaneous use of both legs of the cursor on the *LOP-cursor*. Ninja Cursors (KOBAYASHI; IGARASHI, 2008) is the only other approach that allows simultaneous control over multiple cursors. However, it only replicates the mouse movement across cursors aiming to reduce the index of difficulty of selection tasks, while we aim at allowing simultaneous pointing at two specific places at the same time.

For the *LOP-cursor* and the *disambiguation canvas*, the *arrow* represents the *leg-1* of the cursor, while the *ring* and the *sphere/cone* represents the *leg-2* of the cursor. The *line* present in the *LOP-cursor* is also useful to guide visual search when switchin attention among the legs of the cursor.

In order to facilitate the use of the *two-legged cursor*, we have defined a *pin* state for our selection techniques. The *pin* state consists of fixing the canvas on a pointed position for the *LOP-cursor*, or with the pointed subset of objects for the *disambiguation canvas*. To enter the *pin* the user must perform a *double tap* gesture, the same gesture is used to leave the pin state. We believe tasks that require the user to frequently point back and forth, such as arranging items in folders or spreading photos on a large display, will benefit from this approach. We also think this is a promising approach for interaction with a multi computer environment, allowing the user to *pin* the control canvas over the display of one computer, and point the device to another display in order to transfer files, import settings, etc.

Although the *two-legged cursor* was not deeply explored yet, preliminary results presented in Section 6.1.3 indicates it is promising.

5 PROTOTYPES IMPLEMENTATION

Our techniques were implemented for two distinct visualization hardware, on a 16 LCD monitors tiled-display and a Sensics zSight Integrated SXGA HMD (Sensics, 2011). Details of software and hardware are treated below by each of these implementations.

5.1 Hardware and Software

5.1.1 Tiled Display

Our display wall is a 16 LCD monitors tiled-display, disposed on a 4 x 4 matrix. Each monitor has 1,680 x 1,050 pixels and 22 inches of diagonal size. The total pixel count is 6,720 x 4,200 = 28,224,000 pixels (≈28 megapixels). The display wall is controlled by four Core2Quad PCs, with two NVIDIA GTX 285 each. See Figure 5.1 for an overview photograph of the prototype in operation.

The software running on the tiled-display is implemented in C++, using OpenGL for graphics. Each computer runs an instance of the program, and a control PC was used to send commands and control tests, e.g. to generate and to distribute challenge targets.

5.1.2 Immersive Display

Our immersive display is an Sensics zSight Integrated SXGA HMD (Figure 5.2a). It provides stereoscopic vision using two 1280 x 1024 displays and has a FoV of 60 degrees. This HMD also provides the orientation of the head. We used a Intel Core i7 computer, equipped with two AMD Radeon HD 5870 Eyefinity 6.

The software running on the zSight HMD was implemented in C++, using Ogre3D for graphics (Ogre 3D, 2012). Stereoscopy is achieved using a side-by-side image, where two distinct images are rendered and then used to generate a 1280 x 1024 image, so the effective resolution on each eye is 640 x 1024 (stretched by the HMD into two 1280 x 1024 images).

5.1.3 Mobile Device

The smartphone used in our main implementation is an Apple iPhone 4. An iPod Touch 4 was also used for preliminary and comparative testing. As the smartphones used do not present a general purpose embedded back button, we adapted a Microsoft Wireless Mobile 3500 Mouse, which offered a reliable wireless communication. The mouse was adapted to fit on a smaller enclosure box and use a lighter battery (Figure 4.4).

The mobile device software is an *app* implemented in Objective-C. It acquires the sensor readings and communicate them over a wi-fi infrastructure through UDP protocol.

Figure 5.1: Tiled-display prototype overview. This photograph depicts an user interacting with the two legs of LOP-cursor. Notice that while leg-1 (arrow) selects a square on the left, the leg-2 (ring) indicates the location where the square should be placed.

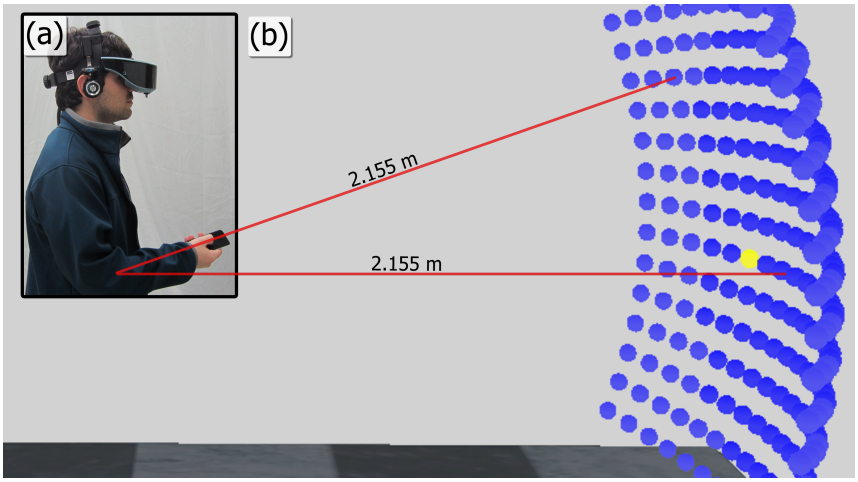

Figure 5.2: Disambiguation canvas experiments setup: (a) user wearing the Sensics zSight Integrated SXGA HMD (Sensics, 2011), (b) the virtual environment used for the experiments.

5.2 Orientation acquisition

In our prototype, we adapted the strategy proposed by Madgwick (MADGWICK; HARRISON; VAIDYANATHAN, 2011) which combines gyroscope, accelerometer and digital compass information to gather a more robust orientation. This orientation acquisition method relies on the *gyroscope* to provide instantaneous orientation changes (providing 3 axes angular rate of change), while perform gradual corrections using two distinct vectors with absolute frames of reference, given by the *accelerometer* and *magnetometer* (gravity and magnetic north pole directions). In practice, we correct gyroscope cumulative rotational error in two axis using the accelerometer (roll and pitch) and in a third axis using the magnetometer (yaw).

Magnetometer readings are less precise and more error prone than *accelerometer* readings, suffering from magnetic interference from nearby metal structures (display wall, attached mouse, metal furniture and floor). Raw readings from the iPhone oscillate within the range of 10% in our environment. Although it may not seem too much, it is crucial to understand that, as a ray is casted using device orientation, any variation is greatly magnified. Because of that, two additional steps were implemented to allow long term use of the *smartphone* as a pointing device. In the first step, to force the *magnetometer* to only adjust yaw drift error, the tridimensional vector provided is projected into a plane orthogonal to the accelerometer vector. In the second, to ignore high frequency *magnetometer* reading variations, instead of using a filter – which would result in poorer interaction due to delay – we are using a redundancy controller based on the gyroscope updates. More specifically, in this step we ignore small variations from the magnetometer readings whenever the gyroscope readings cannot confirm that a rotation actually occurred around that axis.

For the iPod touch, which does not contain a *magnetometer*, we have used the standard orientation provided by the iOS SDK.

5.3 Position calibration

As smartphones and other *every-pocket* mobile devices are not equipped with position tracking, we used only orientation to define pointing and assumed a constant position of the user while testing our prototype. To define this position and reconstruct rotational zero position of the device, our system uses a calibration step. Notice that this step is not necessary when position tracking is used.

For magnetometer calibration, the device must be placed face up pointing orthogonal to the plane defined by the display. A calibration command registers an orientation offset. Next, the calculation of approximate user position (P_3) is achieved registering two orientations of the device. The user is asked to aim the device at a blue point P_1 at one corner of the screen, and then at a red point P_2 at the opposite corner. This calibration method assumes that the real distance between blue and red dots is known, and that the display is perpendicular to the world Z axis.

The two registered orientations are used to retrieve two normalized vectors $\vec{v_1}$ and $\vec{v_2}$. A third vector $\vec{v_3}$ is calculated as $\vec{v_3} = P_2 - P_1$. The angles α_1 between $\vec{v_1}$ and $\vec{v_2}$, α_2 between $-\vec{v_1}$ and $\hat{v_3}$, and α_3 between $-\vec{v_2}$ and $-\hat{v_3}$ are computed by dot product. The length of the segments given by $\vec{v_1}$ and $\vec{v_2}$, respectively, are calculated as

$$l_1 = \frac{|\vec{v_3}|sin(\alpha_2)}{sin(\alpha_1)}, l_2 = \frac{|\vec{v_3}|sin(\alpha_3)}{sin(\alpha_1)}. \tag{5.1}$$

Then, we estimated device positions $P_3' = P_1 + l_1(\vec{v}_1)^{-1}$ and $P_3'' = P_2 + l_2(\vec{v}_2)^{-1}$. A mid point between P_3' and P_3'' is used as the final P_3.

Limitations of this calibration approach is that significant changes in the position of the user will result in the need for a new calibration, or at least some cursor offset control. Also, the rotational $zero$ will depend on which of the joints (shoulder, elbow, wrist) the user performs the rotations.

6 EVALUATION

This chapter details the tests we have performed so far to assess the *LOP-cursor*, the *disambiguation canvas*, and the *two-legged cursor*. As the *two-legged cursor* was implemented over the LOP-cursor, it is evaluated in the same section.

6.1 LOP-cursor Evaluation

We conducted three sets of user tests for the *LOP-cursor*. *Preliminary evaluation* was done to sustain the design decisions and evaluate the *confirmation of selection* techniques. We tested LOP-cursor with three of the five proposed confirmation of selection techniques configurations presented on section 4.1.2.2 (see Figure 4.4). The preferred configuration was then used in two *detailed evaluations*: a comparative evaluation, and a deeper exploration of the LOP-cursor capabilities. The *task 3* of the deeper exploration regards the use of the *tow-legged cursor* metaphor.

Preliminary and detailed evaluations used the tiled-display prototype presented in Section 5.1.1. Any characteristic that do not follow the presented prototype is described on evaluation specific details. Target start and desired final position was constrained to never intercept a monitor bezel, as stated by (BI; BAE; BALAKRISHNAN, 2010), this can be detrimental to some user interaction aspects. None of the evaluations allowed the rescale of control canvas.

6.1.1 Confirmation of selection evaluation

As a preliminary evaluation, we tested the three selection techniques identified as *tap-after-take-off*, *tap-on-back-touch-surface*, and *back-click-on-adapted-mouse* to find the one that best fits the LOP-cursor. The task consisted on using the LOP-cursor to point and select simple objects (here called *targets*). We tested with 11 subjects (mean age of 25, all males), all of them Computer Science students or researchers working in our lab. Thus, we expected to receive constructive feedback, report of issues on each technique, new ideas, and suggestions of improvements.

Additionally to the *confirmation of selection* technique, independent variables were, *Size*: 1.5cm, 3cm, 6cm; and *Distance* between targets: 25cm, 50cm, 100cm. *Technique* presentation was counter-balanced, while *Size* and *Distance* were randomly presented. Four *Trials* of each possible combination were presented, being divided in 2 *Blocks*. The training *Block* had 1 *Trial*, while the evaluation *block* had 3 trials for each condition. A *Trial* consisted on pointing to and selecting a *Target*. A *Trial* only ended when selection was successful, but users were asked to favor accuracy over time. We collected a total of 891 valid *Trials*. All *Blocks* started with a non valid target. Users were allowed to

use only one hand and were asked to stand at a mark 150cm far from the center of the tiled-display.

A quick oral introduction of the LOP-cursor was given for each participant individually. Participants were allowed to experiment until they felt comfortable using both levels of input (usually between 2≈3 minutes), and then performed the trials. After the trials, each participant filled a questionnaire asking they preferred technique, issues and opinions on each technique, as well as suggestions for improvements. Each participant took between 15≈25 minutes to complete the test.

Concerning the *tap-after-take-off* technique, even if it is the only technique ready to use on hardware availability, the user must release the control canvas to perform a selection. This requires the definition of a new canvas position for each selection interaction. Normally, we assume the user want to hold the control canvas while, for example, performing multiple selection tasks. We also noticed that *tap-after-take-off* added in complexity, being the more difficult to understand and perform for most users. The good points on this technique are size and weight of device, while the drawback is poor visual feedback. Take off of the touch caused the canvas to jump back to the leg-2 ring, and although selection occurred at the take off position, the cursor was already on another place. Even so, three participants preferred this technique.

Most frequently reported problems on *tap-on-back-touch-surface* technique were the weight and size of the device which require bigger physical effort and limit thumb movement on the frontal display. Since the frontal touch surface is used for precise movements, the users grab the iPhone in a way that is comfortable to interact with this surface. Because of this, the position of the finger that performs the back tap is not good. We observed that users tended to back-tap with the side of the finger, very often causing an unstable and not recognizable tap. Only three of the eleven participants preferred the *tap-on-back-touch-surface* technique. However, we believe that the use of a back touch surface for selection is the most promising interface, once it allows several back gestures (BAUDISCH; CHU, 2009; WIGDOR et al., 2007), and is rising as a trend on some mobile devices, as the new Sony PlayStation Vita, for instance.

The preferred technique was the *back-click-on-adapted-mouse*, chosen by five of the eleven participants. This technique provides better feedback, and is easier to learn. Drawbacks were the overall size of the prototype, the size and shape of click buttons, and the easy to perform undesired selections (false positives). Buttons were slighted reduced after that, and other design relying on physical buttons were experimented. One of the new designs used a presenter instead of a mouse, which is smaller, but buttons were harder to press and demanded bigger effort. The other prototype used an Apple Magic Mouse, that provides a smooth pressure surface for selection and could allow other gesture recognitions – similar advantages achieved with the *tap-on-back-touch-surface* –, but the batteries and the iPhone weight made false positive easy to occur. Also, selection at the lower end of the front touchscreen was almost impossible to perform, once the Magic Mouse does not provide an uniform pressure click distribution over all of its surface.

Once the *back-click-on-adapted-mouse* was preferred by users and presented significant best times (see Figure 6.1), we chose this as the standard selection technique for further evaluation.

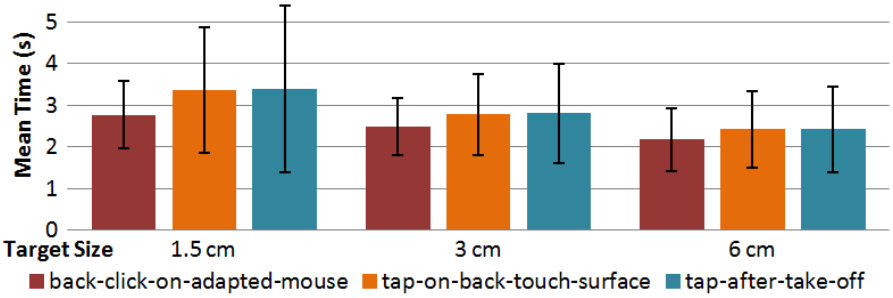

Figure 6.1: Mean time spent to select targets of different sizes using the three confirmation of selection techniques tested.

6.1.2 Comparative evaluation

6.1.2.1 Design

To validate the LOP-cursor technique, we conducted a comparative evaluation on a selection task. LOP-cursor was compared to an *ARC-pad* (MCCALLUM; IRANI, 2009) implementation, and a *ray-casting* using only device orientation (*ORayCasting*). To remove subjective bias from choosing the desired level of precision, the LOP-cursor used for comparison was constrained to force users to always use the two levels of precision for each selection. We call this implementation *CLOP-cursor*. CLOP is actually the same as the original LOP but the first level of precision is not effective for selection triggering. Thus, with the CLOP-cursor, users had to always perform steps *a* through *c* as described in Figure 4.1.

Independent variables are: *Technique*: ARC-Pad, CLOP-Cursor and ORayCasting. Target *Size*: 1cm, 2cm, 4cm and 8cm. Target *Distance*: 25cm, 50cm, 100cm. Dependent variables are: *Time* and *Error rate*. We used a *within-subject* design. Technique exposure was counter-balanced, while size and distance of target were randomly presented. Pointing to a target and triggering a selection counted as a *Trial*. There were a total of 10 *Trials* for each independent variable combination. Each technique evaluation was divided in 5 *Blocks*, where a *Block = Sizes x Distances x 2 Trials*, giving a total of 24 *Trials* per *Block*. The first 2 *Blocks* were used for practicing, and thus are not considered on further analysis. Participants were allowed to take a non-mandatory break between blocks.

After the comparative evaluation, another selection test where the complete *LOP-cursor* is used was taken. In this case, subjects were not constrained to always use the higher level of precision. They could decide when and if they want to use two levels of precision, or only one. This was intended to evaluate for which sizes of target users preferred to use only the ray-casting level of precision, or both of them, thus allowing us to infer if participants subjective preference match to the previously applied comparative evaluation best times per size. Six target sizes were used: 0.5cm, 1cm, 2cm, 4cm, 8cm, 16cm. Each test had 5 *Blocks* of 18 *Trials*, 3 trials per size condition. First 2 *Blocks* were used as training, thus being discarded for analysis.

Active targets were drawn in red on a black background. To avoid bias from visual search, the following target was shown in a dark grey tone. There was an additional starting target at each *Block*. During all evaluation, the user was positioned centered to the tiled-display, at a constant distance of 150cm. Users who completed all the evaluation were then asked for their preferred interaction technique (CLOP-cursor, LOP-cursor, ARC-pad or ORayCasting), and allowed to leave general comments and observations

about the evaluation. Users were asked to favor precision over speed.

6.1.2.2 Implementation details

Comparative tests were carried using an *iPod Touch 4*. As it does not have a magnetometer, orientation reconstruction may drift on *Yaw* over time (see Section 5.2), but the ray casting was too imprecise when using yaw corrections because of the high variability of the magnetometer readings from the iPhone. As this would put the ORayCasting technique in disadvantage relating the other techniques tested we disabled yaw corrections. To limit the effect of yaw drifting we preferred to arrange short blocks of trials (24 trials) and frequently correct any drift between blocks.

The ARC-pad paper (MCCALLUM; IRANI, 2009) does not make clear how a confirmation of selection is performed, and seems to suggest its use over a desk. Our test design, instead, proposes the users to stand in front of the screen, and thus they had to hold the device with both hands to maintain consistent aspect ratio orientation across the tiled display and the mobile device. Selection was implemented using a *Tap* gesture from the secondary hand while holding arrow position with the primary hand. See Section 3.1.2 for a brief description of the ARC-pad technique.

Cursor position was filtered using a *dynamic low-pass filter*, interpolating between cutoffs of 0.2Hz and 5Hz, with 60Hz sample rate. Cutoff is defined according to cursor speed: when $<$ 1cm/sec, lower cutoff is used (0.2Hz); when $>$ 50cm/sec, higher cutoff is used (5Hz). For any speed between these, a linearly interpolated cutoff value is used.

6.1.2.3 Comparative evaluation results

Eleven participants took place on this evaluation, (mean age 22, 2 women and 2 left handed) all of them Computer Science students (10 undergraduate, 1 master student). Most of them had previous experience with pointing devices (mostly Wii gaming), all of them had at least some experience with a mobile device touchscreen, only two had significant experience with very large displays.

From the comparative test we had 2,376 total valid *Trials*. A trial was considered a false positive (accidental selection triggering) when selection occurred at any of the following cases: *20cm* or farther from the target; less then *200ms* after the previous trial was completed. CLOP-cursor, ARC-pad and ORayCastig presented 2, 6 and 9 false positives respectively. These *Trials* were overlooked on further analysis.

All subjects completed the comparative evaluation, but three of them could not complete the last evaluation (using the non-constrained LOP-cursor). This was due to the long time taken by the comparative evaluation (from 30 to 40 minutes to complete), resulting on schedule and fatigue issues for some subjects. This test had a total of 432 trials, 2 of which false positives.

General mean time for a selection with CLOP-cursor, ORayCasting and ARC-pad were respectively: 2.55, 2.46 and 3.05 *seconds*. One-way ANOVA showed that ARC-pad was significantly slower than CLOP-cursor ($F(1.1574)=76.58$, $p<0.0001$) and ORayCasting ($F(1.1567)=84.81$, $p<0.0001$). See Figure 6.2 for detailed mean time and standard deviation for each target size.

Error rate for ORayCasting is significantly higher than CLOP-cursor and ARC-pad ($F(1.1571)=109.08$, $p<0.0001$ and $F(1.1567)=120.6$, $p<0.0001$). Error difference between CLOP-cursor and ARC-pad is not significant ($F(1.1574)=0.48$, $p<0.49$), even considering only the 1cm targets ($F(1.391)=1.93$, $p<0.17$). Error rates are presented on Figure 6.3.

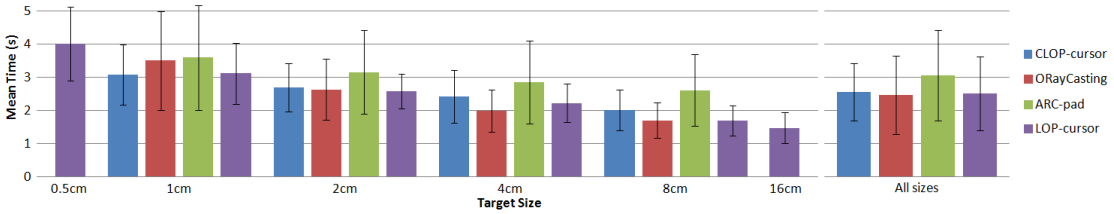

Figure 6.2: Mean time spent to select targets of different sizes using CLOP-cursor, ORayCasting, ARC-pad and LOP-cursor, and their respective standard deviations.

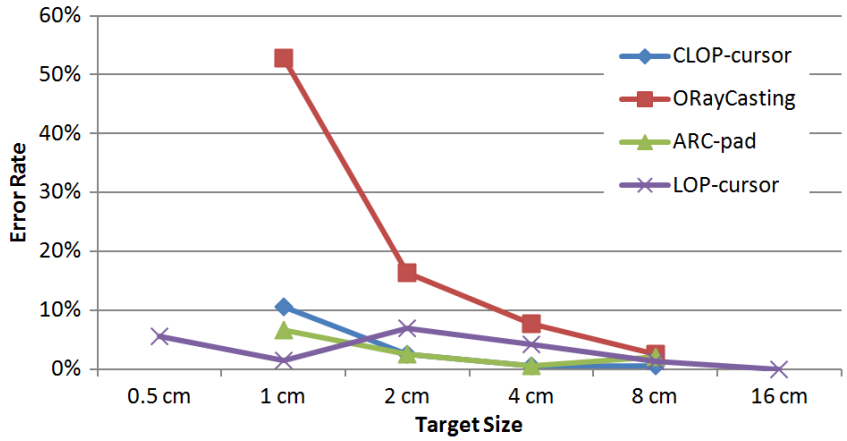

Figure 6.3: Error rate to select targets of different sizes using CLOP-cursor, ORayCasting, ARC-pad and LOP-cursor.

We also found significant learning effect on mean time across blocks for CLOP-cursor ($F(2.787)=4$, $p<0.019$) and ARC-pad ($F(2.783)=3.04$, $p<0.0484$). Although a reduction of error rate did occurred for CLOP-cursor (Block1=12, Block2=11 and Block3=5) there was no significance ($F(2.787)=1.61$, $p<0.2$).

On the LOP-cursor trials (post comparative evaluation) error rate followed a tendency between CLOP-cursor and ORayCasting (see Figure 6.3). This unconstrained LOP-cursor presented lower error rates for small sizes than the CLOP-cursor, even though they rely on the same fine tuning technique. As this technique was always presented to users as the last one, we believe learning effect did affect error rate, but there was not enough data to statistically prove such variation. Detailed error rate for each level showed consistency with CLOP-cursor and ORayCasting previously applied comparative test for most target sizes (Figure 6.4).

Subjective user preference between 1 or 2 levels of precision according to target size is presented in Figure 6.5. For target sizes of 0.5 and 1 cm users always used 2 levels.

After the evaluation, subjects who participated on all procedures were asked for their preferred interaction technique. The LOP-cursor implementation allowing 2 levels of interaction was preferred by seven users, while ARC-pad was preferred by only one subject.

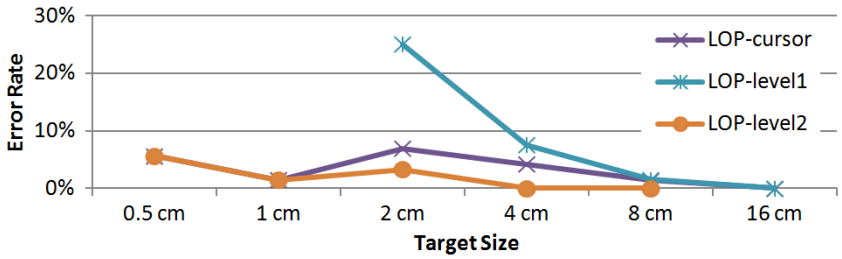

Figure 6.4: Error rate to select targets of different sizes using LOP-cursor, and individual error rate for each level of precision.

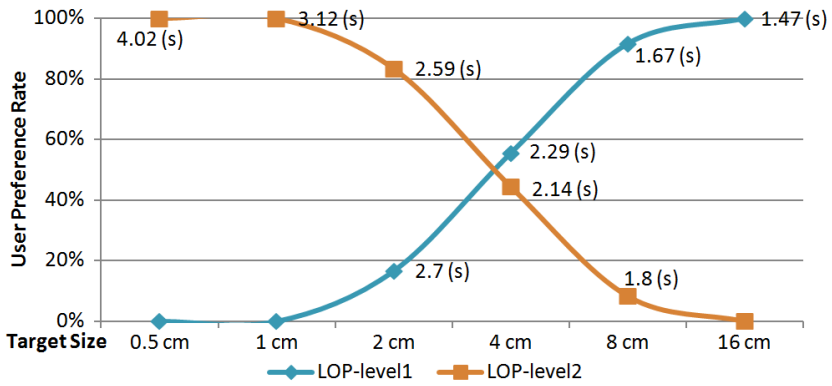

Figure 6.5: Users preferred level of precision for a variety of target sizes when using LOP-cursor, and their respective trials mean time.

6.1.3 Deeper LOP-cursor evaluation

6.1.3.1 Design

We conducted this deeper evaluation to assess the limits of our technique in terms of precision, and to initiate the study on simultaneous cursors. Experiments consisted on completing three different tasks using LOP-cursor:

Task 1. Pointing and selecting targets. A trial is complete when the target is hit.

Task 2. Pointing, selecting, dragging and docking targets. Users should point and select the target in the same way they do in Task 1: drag it over a grey object with the same shape and size of the target located at a fixed distance of 100 cm, and dock the target over the grey object the most precisely as possible. A trial is complete when the target is released (docked).

Task 3. Using the two legs of the LOP-cursor to select a target and put it into a predefined stock area on the right side of the display (as depicted in Figure 5.1). The target should be selected with the leg-1 using the two levels of precision allowed by the selection technique, and moved to its respective stock area (with the same color of the target) using leg-2. The selection results on a transfer between the legs. The trial is complete when the user successfully trigger a selection while simultaneously aiming with both cursors, on both target and goal area.

The presented tasks add in complexity, and thus, they were always applied on the same order, task 1 through 3. We used four different *Sizes* of targets (0.3, 0.8, 2 and 4 cm). The initial position of targets were randomly defined. For *Task 3*, targets appeared only at the left-half of the tiled display.

We also found that the 0.3 cm target could introduce delay due to visual search. Then a green target was shown between each trial for tasks 1 and 2. Green target and the trial target(s) appeared at the same time. Having found the trial target(s), the user selected the green target to start the trial.

The procedure consisted on two practice *Blocks* of two *Trials* for each *Size* (total of *eight trials per training block*) and one evaluation *Block* with six targets per *Size* (total of *24 trials*) for each *Task*. The first practice block was used to explain task goal, and to demonstrate the LOP-cursor capabilities that would be used during the task. Users were asked to complete trials as quick as possible, but without sacrificing precision over time. On the second practice block users interacted by themselves, with minor hints of the test conductor when needed. During the evaluation block, subjects were left by themselves, and no help was given. After the evaluation, users filled a System Usability Scale (SUS) questionnaire.

An iPhone 4 disposing of a magnetometer was used on these tests. This allowed orientation to be gathered as described in Section 5.2, allowing longer blocks to be taken. As this evaluation aimed to assess the limits of our technique, users were constrained to always use the two levels of the LOP-cursor (the so called CLOP-cursor).

6.1.3.2 Deeper LOP-cursor evaluation results

Eleven undergraduate students in Computer Science participated on this experiment (mean age of 23, 2 females, all right handed). Most subjects had some experience with pointing devices, all of them had at least some experience with mobile device touch screens. Each test took from 25 to 40 minutes to be concluded.

All users were able to complete the tasks, i.e., they were all able to select every 0.3 cm target at the distance of 150 cm with our technique. The chart on Figure 6.6 shows

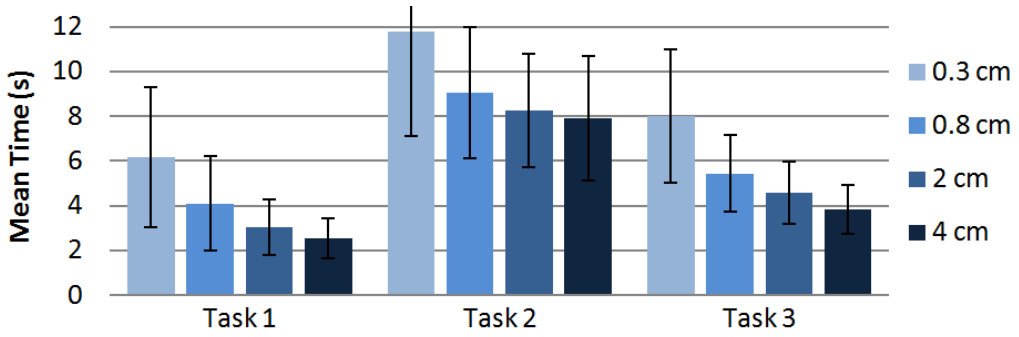

Figure 6.6: Mean time for completion of each trial of each task.

the mean time for completion of the 3 tasks. Notice that although the larger targets allow for faster task completion, the time difference is always inside the standard deviation, regardless of the huge difference (more than one degree of magnitude) in objects size.

There were 5 false positives (incidental docking) on Task 2, which were discarded for further results. Mean docking distance error was 0.17 cm, being 0.158 cm, 0.162 cm, 0.176 cm and 0.188 cm for target sizes with 0.3 cm, 0.8 cm, 1 cm, and 2 cm, respectively. Docking distance error difference per size showed not to be significant ($F(3.255) = 1.38$, $p < 0.254$). The most frequent user complain was related to prototype physical ergonomics (device size and back button position). We noticed that some users slightly moved touch position when triggering a selection command. We believe the ergonomic issue was the main cause of this effect. This was specifically detrimental when selecting the 0.3 cm targets, where subjects needed more than one selection triggering to complete the trial: 42% and 39% for Task 1 and Task 3, respectively. CLOP-cursor scored 77 on the SUS questionnaire, with standard deviation of 12.8.

Task 3 results showed that users could successfully point at two simultaneous positions. The other existent multiple cursor technique, Ninja Cursors (KOBAYASHI; IGARASHI, 2008), points to only one location at a time. It is difficult to compare performance results between the LOP-cursor and the Ninja Cursors as their evaluation is based on a 2 screens desktop environment, while ours pursuit interaction with much larger and higher resolution displays, away from the desktop and with potential for interaction in 3D environments. Nevertheless we could notice that while the performance of the LOP-cursor is independent of screen target density, the performance of the Ninja Cursors is significantly affected by both the number of cursors and the target density.

We have noticed that, eventually, users consciously performed confirmations of selection when the tip of the arrow was not over the 3mm target. We believe we may have challenged their visual acuity, or that the arrow may causes some uncertainties of where its tip is exactly located. To address this possibility, we plan to perform a comparative evaluation of the use of a crosshair against the arrow. We believe a crosshair may provide better perception of the alignment.

6.2 Disambiguation Canvas Evaluation

We conducted two sets of user tests for the *disambiguation canvas* technique. In both we have compared *disambiguation canvas* with *ray-casting*. As in the LOP-cursor evaluation, the implemented ray-casting only rely on the orientation of the device and

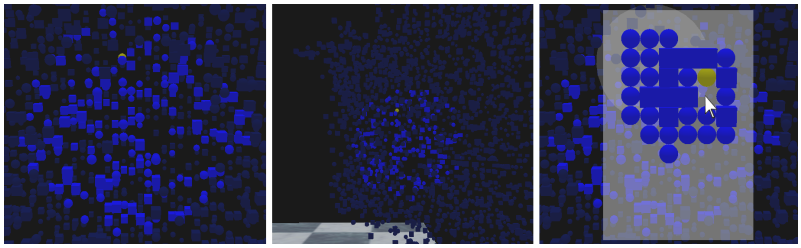

Figure 6.7: Disambiguation canvas performing a 3D selection.

therefore it will be referred as *ORayCasting*, the *disambiguation canvas* will be referred as *DCanvas*. Most design decisions are common for both evaluations, which were based on the one performed by Kopper et al. for the SQUAD technique evaluation (KOPPER; BACIM; BOWMAN, 2011).

We used *sphere-casting* for the first step of selection of DCanvas. Instead of using the standard size (suggested in Section 4.2.2.1) for the casted sphere, we used the angular size of $\approx 26^\circ$ ($1m$), so more objects would be pointed for the disambiguation phase. The rescale of the casted sphere was not allowed.

The user goal was to select a yellow sphere among several distracters of same size, represented by blue spheres. These objects were arranged as a matrix. To position these objects, we have used a main sphere of $2.155m$ radius. The origin of the ray/sphere-casting was set to the center of the main sphere, which guarantees the same angular pointing size for all the objects. The virtual camera was positioned $50cm$ above the ray/sphere-casting origin. Figure 5.2 shows this setup. This task happens in a 3D immersive environment, but objects do not have a proper 3D distribution. We chose this design in order to comply with the SQUAD experiment (KOPPER; BACIM; BOWMAN, 2011), and have a baseline for comparisons. Furthermore, we also used *disambiguation canvas* on objects with less uniform distribution, as depicted by Figure 6.7. It is suitable as long as the correct volume casting technique is used, if too dense or with too many occlusions, cone-casting would be preferred for the first stage of selection.

Independent variables are matrix of objects *density* (distractors), 9 x 9, 18 x 18 and 27 x 27, and *size*, $2cm$, $4cm$ and $6cm$ (effective angular size of $\approx 0.53^\circ$, $\approx 1.06^\circ$ and $\approx 1.6^\circ$ respectively). The matrix of objects on the surface of the sphere occupied the total angular size of 45°, horizontal and vertical, such as depicted in Figure 4.11a.

The same procedure and questionnaires were used, the procedure was as follows:

1. The subject was asked for any health issue or impairment that could prevent them from participating (such as epilepsy history and color blindness);

2. The subject filled a characterization questionnaire.

3. The subject was presented to the first technique on a common display (so the experimenter and the subject could share the view while explaining how the technique works);

4. The experimenter showed the HMD and how to adjust it to the head;

5. The subject performed practice blocks with the first technique;

6. The subject performed evaluation blocks with the first technique;

7. The subject answered a questionnaire on the first technique;

8. The subject repeated steps 3 through 7 for the second technique;

9. The subject filled a post experiment questionnaire comparing both techniques.

We used blocks of 10 trials, 9 for density x size possible combinations, which were randomly presented, and an initial target which is used by the subject to start the block. Training consisted of 5 blocks, while the evaluation consisted of 10 blocks. Technique presentation was counterbalanced. Each subject took from 25 to 40 minutes to complete the evaluation. They were also allowed to take the HMD off and rest between the blocks if desired.

A trial consisted of a selection task, ending with an activation of selection, which could be successful or not. The casted sphere and rectangular canvas were represented by a white semitransparent sphere. The ORayCasting was represented by a red cylinder with 1cm of diameter.

An *iPod touch 4* was used in both evaluations. As mentioned earlier, it is not equipped with a magnetometer, and thus it is subject to drift on *yaw*, losing its correct orientation. Therefore we have used blocks with no more than 11 targets. Mobile device orientation was filtered using a *dynamic low-pass filter*, interpolating between cutoffs of 0.2Hz and 50Hz, with 60Hz sample rate. Cutoff is defined according to the angular change speed in degrees: when $< 1^\circ/sec$, lower cutoff is used (0.2Hz); when $> 50^\circ/sec$, higher cutoff is used (50Hz). For any speed between these, a linearly interpolated cutoff value is used. We have achieved better results with this approach than the one described in Section 6.1.2.2, which filters the cursor position in 2D (instead of the device orientation).

6.2.1 First Evaluation

This was a preliminary evaluation performed to verify how DCanvas sustain the design decisions and evaluate whether the technique was comprehensive and easy to use or not. In this evaluation, the target was randomly positioned among the matrix of objects.

Ten undergraduate students in Computer Science participated on this experiment (mean age of 24, eight right handed). Seven of them very experienced managing mobile devices touchscreens, and three with no experience using natural pointing devices. None of them reported any significant experience with virtual reality equipments. Each test took from 25 to 40 minutes to be concluded.

General mean time for a selection with *DCanvas* and ORayCasting were respectively: 2.67 and 2.56 *seconds*. One-way ANOVA showed that *DCanvas* was significantly slower than *ORayCasting* ($F(1.1783)=5.83$, $p<0.016$). See Figure 6.8 for detailed mean time for each size, time for density is also shown for the *DCanvas*.

Error rate for ORayCasting is significantly higher than *DCanvas* ($F(1.1783)=135$, $p<0.0001$). Error rates for each size and density are presented in Figure 6.9. We noticed that two users had difficulty to reach the top of the touchscreen, indeed, they had the higher error rate for *DCanvas* in this test. This issue is related with the chosen layout and is discussed in more details in Section 6.2.3. The error ratio per trial of the *DCanvas* was 0.038, while for *ORayCasting* it was 0.214.

6.2.2 Second Evaluation

While on the previous evaluation the target was randomly pick at the matrix of objects by the system, here only the targets that falls inside the range of 52 up to 77cm from the

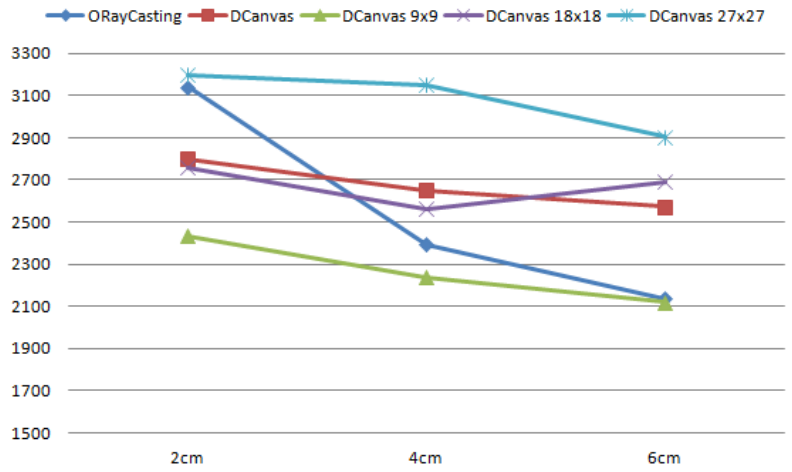

Figure 6.8: Mean trial completion time for each target size and density.

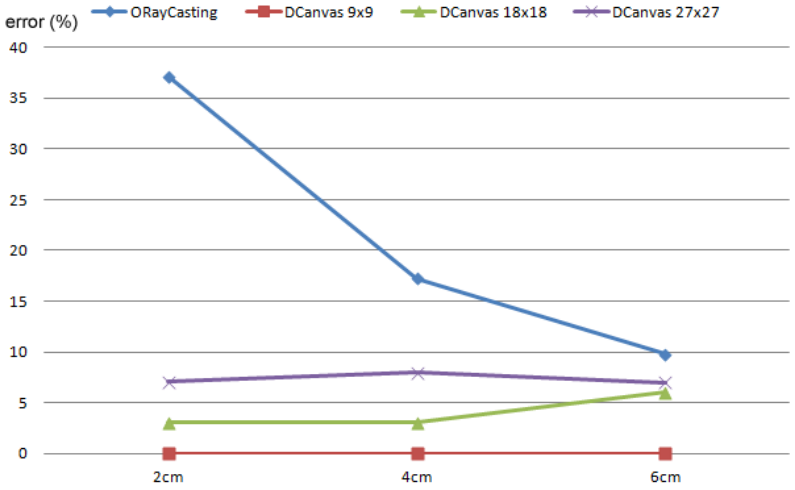

Figure 6.9: Error rate for each combination of target size and density.

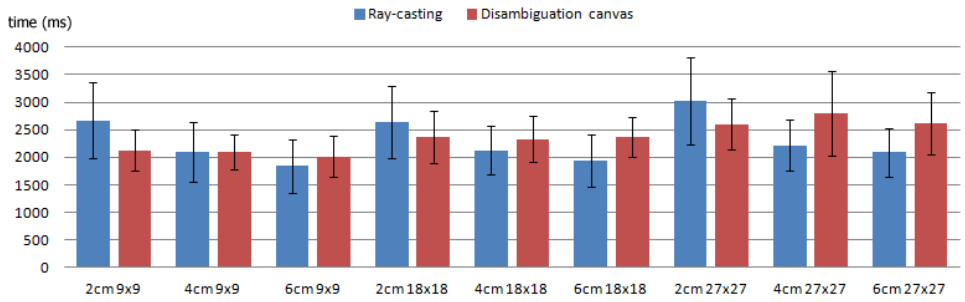

Figure 6.10: Mean trial completion time for each combination of target size and density.

center of the matrix are eligible as targets. We did so in order to reduce visual search bias, and to reduce the standard deviation, which was high for the previous evaluation. With the same purpose, we have also made objects beyond 77cm from the center of the matrix green, so the user knows they are not target candidates.

Six graduate students in Computer Science from our university participated on this experiment (mean age of 29, four right handed). All of them very experienced managing mobile devices touchscreens, and at least with some experience using natural pointing devices. Only two reported experience with virtual reality equipments of 3 or above out of a maximum of 7. Each test took from 25 to 40 minutes to be concluded. We have obtained a total of 1080 valid trials.

Overall mean time for a selection with *DCanvas* and ORayCasting were respectively: 2.37 and 2.29 *seconds*. One-way ANOVA showed that *DCanvas* was slower than *ORay-Casting* with statistical significance ($F(1.1078)=4.43$, $p<0.036$). See Figure 6.10 for detailed mean time and standard deviation for each combination of target size and density.

SQUAD has used 16, 64 and 256 densities of disambiguation, while on our approach we used ≈ 25, ≈ 100 and ≈ 225 (depicted in Figure 4.12). If we assume equivalence among these parameters of difficulty, *DCanvas* has it low density time of selection roughly equal to the SQUAD. However, medium density and high density have increased *DCanvas* times in $\approx 300ms$ for each level of density, while these factors increases SQUAD times in $\approx 700ms$ for level of density.

Error rate for ORayCasting is significantly higher than *DCanvas* ($F(1.1078)=70.34$, $p<0.0001$). Error rates for each combination of size and density are presented on Figure 6.11.

In this evaluation, none of the subjects had problems reaching the areas defined by the layout in this test, which may have been the reason why the error ratio lowered from 0.038 to 0.009. Almost as good as the achieved by SQUAD, which was 0.007. However *DCanvas* presented best times

6.2.3 Discussion

Questionnaires: The same questionnaires were used in both evaluations. Users took an intermediate questionnaire for each technique right after using it, and a comparative questionnaire at the end of the evaluation. All the questions were formatted as 7 point likert scale.

The intermediate questionnaire asked users to rate each technique concerning: ease of learn and ease of use; how well it performs for little, medium and large targets; fatigue felt on the wrist, hand, fingers, back and legs. Results are presented in Figure 6.12. Both were

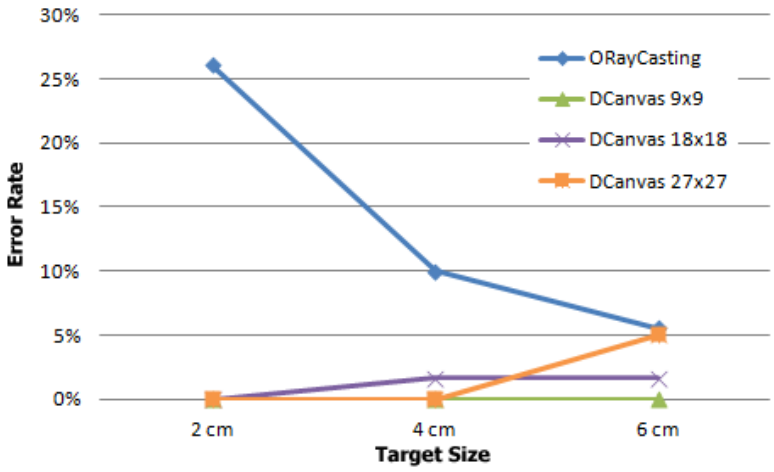

Figure 6.11: Error rate for each combination of target size and density.

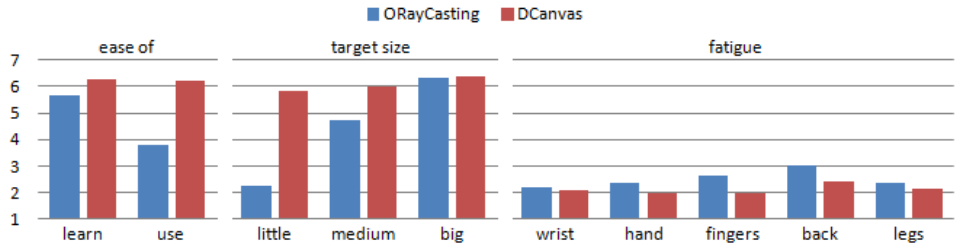

Figure 6.12: Subjective questionnaire scores for Disambiguation canvas and Ray-casting.

considered very easy to learn, while our technique was considered easier to use. *DCanvas* was preferred over *ORayCasting* for little and medium targets, while big targets received equivalent ratings for both techniques. Overall fatigue was lower for *DCanvas*, its mean of the 5 related questions was 2.1, against 2.5 of *ORayCasting*.

Regarding the comparative questionnaire, *DCanvas* presented higher scores for all the questions. It was considered more accurate (6.7 against 1.6), faster (5.9 against 2.7), less tiring (4.8 against 2.6) and easier to use (5.6 against 2.5). Curiously, users felt as *DCanvas* were faster, which is truth for $\approx .5^{\circ}$ angular size targets, but false for the overall evaluation. *DCanvas* was also preferred by all the users.

Transition to the canvas: The most recurrent feedback left by the users regards the transition of the subset of objects from its original context to the control canvas. Users frequently had the conviction that positioning the intended target near to the center of the sphere during the sphere-casting step would take that target near to the center of the canvas when switching to disambiguation. This intuition may arise from the arrangement of objects as a matrix, which would be easily fitted inside the layout. However, on a more complex scene, with targets spread in depth, such organization is not so obvious. We are currently working on this issue, as it could reduce user effort of reaching distant objects, as well as reduce visual search time.

Canvas layout: We found that although it was very functional to a majority of the users – which holds the mobile device in the center of the hand (Figure 6.13a) – some of them tends to hold the mobile device supported by the little finger (Figure 6.13b). Based

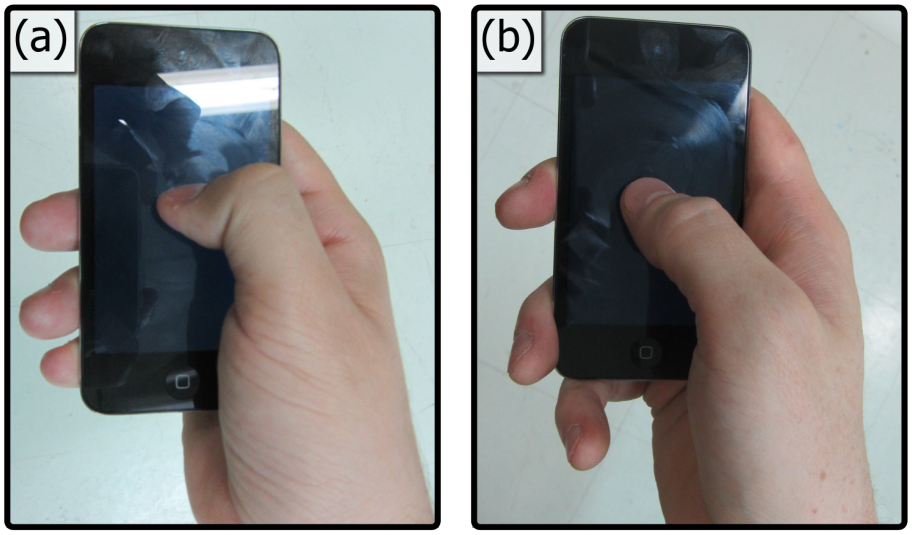

Figure 6.13: Observed hand postures while holding a mobile device: (a) most users hold the device with the whole palm, reaching the upper area better than the lower; (b) some users support the mobile device with the little finger, having trouble to reach the upper area of the screen.

on observations during the evaluation, we concluded that holding such as depicted in Figure 6.13b decreases users reach of the upper side of the screen while increases the reach of the lower part. We believe this limitation might be overcome with two approaches: the layouts can be vertically scrolled to best fit hand position; or alternatively, the subject could perform a quick calibration in order to detect the area that he can easily reach, generating a personal layout.

7 CONCLUSIONS AND FUTURE WORK

In this chapter we present a summary of our contributions, as well as how our techniques distiguish thenselves from other approaches. We also present some considerations regarding our opinion of *levels of precision* and *progressive refinement*. Finally, we identify our following steps for this project.

7.1 Conclusions

In this dissertation we have presented the *LOP-cursor* and the *disambiguation canvas* techniques for fast and high accuracy selection. We have also provided a design space for techniques relying in multiple levels of precision, as well as performed a survey on selection by *levels of precision* and *progressive refinement* techniques. To the best of our knowledge, this is the most complete survey available on these specific subjects. Finally, we have proposed a new cursor metaphor which allows the simultaneous definition of two distinct positions to perform composite tasks, the *two-legged cursor*.

Levels of precision and *progressive refinement* provide selection techniques that address the lack in accuracy of immediate selection techniques. They frequently do so through the combination of more than one *indication of objects* approach. While literature on these classes of technique covers quite well the combination of techniques using only one hardware of input, techniques that rely on distinct input hardware for each technique – such as combining the movement sensors and the touchscreen of an ordinary mobile phone – are still overlooked. *LOP-cursor* and *disambiguation canvas* pay attention not only on designing selection techniques for accurate selection, but also on providing the best hardware for each phase of control in order to speed up the process. Movement sensors are used to travel long distances and approximate the cursor to the target, while the touchscreen is used to fine tune/disambiguate the selection.

By using the touchscreen on the second level of precision of the *LOP-cursor*, users were able to select objects represented in a motor area of $\approx 0.6mm^2$ of the touchscreen surface during evaluation. If we transfer this parameter to the *disambiguation canvas*, the whole screen surface would allow the disambiguation among 6144 objects in one step. Such submillimetric precision is not perceived when using ordinary mobile device interfaces, because the finger occludes the content thus preventing accuracy.

The advantages of *LOP-cursor* and *disambiguation canvas* are not limited to aspects from accuracy, ambiguity and complexity of selection problems. We have also kept performance of immediate techniques and user control. We argue that, in order to design a successful levels of precision or progressive refinement technique, one should sum the lower accuracy approach with the refinement approach, and let the user decide which one he will use for each selection task. The user might prefer not to progress to the

next level/disambiguation phase if the intended target is easy enough to select. This was verified on LOP-cursor evaluation, where users have always used *ray-casting* to select $16cm$ targets, and *LOP-cursor* to select $0.5cm$ and $1cm$ targets. Our techniques allow the confirmation of selection at any instant, not requiring the level progression.

Regarding the *disambiguation canvas*, we emphasize the user's subjective rating assumes that our technique is faster than *ray-casting*, while it has in fact performed slightly slower. This might be a clue of how unpleasant it is to perform a difficult selection with full attention, on which even the breath have to be controlled some times.

This dissertation also introduces a *two-legged cursor* modality which can be controlled with only one hand. This is a major breakthrough as many everyday user actions with a computer involve quickly defining two locations on the screen to complete an operation (copy files, arrange photographs, etc.).

7.2 Future work

We have obtained exciting results which prompts the sequence of this work. We are already planning several new evaluations, as well as some design changes.

We want to find out the limit of precision of *LOP-cursor*. Thus we will evaluate whether the arrow is really the best representation for the cursor, and if visual acuity may lead to misleading judgment of its exact position. Moreover, we want to know how users will deal with the rescale functionality, and how the *LOP-cursor* will perform in immersive environment.

Moreover, we want to implement a dynamic zoom functionality inside the *LOP-cursor* rectangular screen representation, to be controlled through the orientation of the device – instead of rescaling the control canvas as stated in Section 4.1.2.3 –, providing a *fish-eye* focus-and-context visualization and selection technique. This metaphor will allow to increase the size of the central region of the rectangular representation (of more interest to the user), while the peripheral region of the rectangle (of less interest) is progressively squeezed. This zoom will also compensate for eye acuity limitations when using very high resolution displays.

Furthermore, we plan to perform an evaluation comparing *disambiguation canvas* with SQUAD. As discussed before, our preliminary results indicate best performance while maintaining similar precision with our technique. As our current implementation of the *disambiguation canvas* was tested only for spheres selection, it uses the arrow occlusion of the object mesh to indicate the selection. We will improve this by assigning a square area within the canvas for each object, this will simplify collision tests as well as increase the effective size of the object's area of selection.

On the tiled-display, we plan to investigate how the user distance to the screen and the screen size affects the optimal scale for the control canvas. As stated by Peck et at. (PECK; NORTH; BOWMAN, 2009), users tend to interact in different scales according to their distance from screen. Naturally, when users are near the display, they can see and point with more precision, and thus, a smaller control canvas allowing more precision would also make sense. We will also investigate the possibility of perspective correction of the rectangular shape. To do so, the device needs to be constantly tracked in space with a 3D tracking system. We will use the Bratrack, a system of relative low cost developed in our university, it prejects infrared to track reflective markers with two cameras.

Moreover, we also want to assess the cognitive load of our *two-legged cursor* metaphor, and determine which two-location tasks may benefit from this metaphor. We plan to apply

our concept on distributed computing scenarios, with many computers and screens that do not necessarily respect a position pattern, but rather are distributed across the room. Such scenarios can benefit from a two-legged cursor for tasks such as file transfer across computers, where the user can pin control canvas on a high resolution display (computer 1), while controlling the file destination pointing to another display (computer 2, 3, 4 ...) with leg-2 cursor. Future works also include the study of leg-2 serving as a controller for another dimension, such that, its distance from (or circular movement around) the leg-1 can control selected object attributes (scale/depth/orientation).

We recently started a collaboration with the mining engineering research group of our institution (UFRGS), in which we will use the *LOP-cursor* and the *two-legged cursor* metaphor to interact with high resolution satellite images on our tiled-display. Our work will be used for tasks such as quick measurement of distance between two locations, setting parameters, arranging information, and annotating images. The adaptation of our tool for this specific application will allow a more ecological validation, evaluating our technique in a real life application, as well as comparing it with other available options on practical and useful tasks. Finally, this collaboration may also suit as a testbed environment to evaluate the collective and collaborative use of the *LOP-cursor*.

8 RESUMO EXPANDIDO

Pesquisadores em ambientes virtuais 3D categorizam interação em quatro formas fundamentais: *seleção*, *manipulação*, *navegação* e *controle do sistema* (BOWMAN et al., 2004; MINE, 1995). Neste trabalho nos focamos principalmente na *seleção*. Seleção é a capacidade do usuário de especificar a escolha de objetos no ambiente virtual para ações subseqüentes (STEED, 2006). Muitas aplicações são mais críticas quanto a precisão da seleção do que quanto ao tempo requerido para seleção. No entanto, técnicas de seleção comuns tendem a sacrificar precisão para reduzir o tempo de seleção. Neste trabalho oferecemos alternativas para seleção precisa, mas ainda assim rápida. Para tanto investimos em seleção por *refinamento progressivo* (proposto por Kopper et al. (KOPPER; BACIM; BOWMAN, 2011)), e em seleção por *níveis de precisão* (classificação que propomos nesta dissertação).

Refinamento progressivo consiste em reduzir a quantidade de objetos selecionáveis com passos de refinamento, até que o objeto desejado seja selecionado. Isso geralmente acontece através de uma troca que favorece precisão, mas sacrifica em tempo de seleção. A divisão de uma seleção em passos subsequentes torna o apontamento em cada passo mais fácil. No entanto, o que uma vez era realizado imediatamente agora consiste em um procedimento. *Níveis de precisão* consiste no aumento da precisão de apontamento por meio da manipulação de parâmetros de entrada para saída (controle sobre proporção e ganho para este mapeamento são comuns na área de interação humano-computador (BLANCH; GUIARD; BEAUDOUIN-LAFON, 2004)) e/ou combinação de abordagens de apontamento. Por consequência, o usuário terá seus movimentos remapeados de forma a alcançar maior controle e estabilidade sobre o cursor.

Neste trabalho optou-se por um telefone celular atual como dispositivo de controle. Identificamos a falta de um dispositivo de entrada padrão para telas grandes e de alta resolução, que vem se tornando populares em lares e ambientes públicos. Com base na gama de sensores disponíveis e em pesquisas recentes (BALLAGAS et al., 2006; BORING et al., 2010, 2011; PEARS; JACKSON; OLIVIER, 2009; SONG et al., 2011), apoiamos que o telefone celular é forte candidato a assumir essa posição. Esta afirmação é ainda mais provável para os usuários esporádicos, que interagem apenas ocasionalmente com telas grande e de alta resolução, e portanto não estão dispostos a gastar em um aparelho especializado.

As contribuições dessa dissertação são: a proposta de um espaço de *design* para técnicas de seleção com *níveis de precisão*; o levantamento e classificação das técnicas de seleção por *níveis de precisão* ou *refinamento progressivo* (texto em inglês apenas, Seções 3.1 e 3.2); a técnica de seleção *LOP-cursor* – abreviação para *levels of precision cursor*(*cursor de níveis de precisão*) –, baseada em *níveis de precisão*; a técnica de seleção *DCanvas* – abreviação para *disambiguation canvas* (*tela de desambiguação*) –, baseada

em *refinamento progressivo*; a metáfora de um *cursor de duas pernas*, que permite definir duas posições simultaneamente com apenas uma mão.

8.1 Seleção de Objetos

Segundo a taxonomia de Bowman et al. (BOWMAN et al., 2004), uma seleção pode ser decomposta em três partes: *indicação do objeto*, *confirmação da seleção* e *realimentação*. Desta forma, em uma tarefa de seleção o usuário deve ser capaz de apontar um objeto e executar um comando de seleção. Complementariamente, o sistema deve proporcionar realimentação de modo a manter o utilizador ciente do estado da seleção. A taxonomia completa é apresentado na Figura 2.1. Dentre estas três partes, a mais relevante para este trabalho é a *indicação do objeto*, Bowman et al. classificam as formas de *indicação do objeto* como: *oclusão*, *toque do objeto*, *apontamento* e *seleção indireta*.

Neste trabalho tratamos como técnicas de *seleção imediata* aquelas que necessitam apenas de uma forma de *indicação do objeto* e *confirmação da seleção*, sem a necessidade de repetir estas etapas durante uma seleção. Por outro lado, as técnicas de seleção por *refinamento progressivo* ou *níveis de precisão* são elaboradas ou usando mais que uma forma de *indicação do objeto* e *confirmação da seleção*, ou pela repetição destes passos no processo de seleção, isso acontece a fim de reduzir o conjunto de objetos selecionáveis ou aumentar a precisão da etapa de *indicação do objeto*. Abaixo segue uma breve revisão de técnicas de seleção imediata seminais, de acordo com a abordagem de *indicação do objeto* (toque do objeto, apontamento ou oclusão).

8.1.1 Seleção Imediata

Toque do objeto geralmente depende da posição absoluta de um cursor no ambiente virtual, útil para selecionar objetos próximos ao usuário. A abordagem da *mão virtual* (MINE, 1995) é provavelmente a técnica mais comum e óbvia. Ela usa mapeamento 1:1 entre a representação real e virtual da mão, a mão real é rastreada e a mão virtual imita seu comportamento no ambiente virtual.

Apontamento permite a seleção de objetos a qualquer distância. *Ray-casting* é a abordagem mais comum, proposta por Liang e Green (LIANG; GREEN, 1994) consiste em um teste de interseção entre um raio controlado pelo usuário e os objetos da cena, análogo a um apontador laser no mundo real. Liang e Green também apresentaram a técnica *Spotlight* (LIANG; GREEN, 1994) que usa um cone a fim de fazer a seleção a distância mais tolerante do que um mero teste de interseção entre raio e objeto. Para ser selecionado o objeto deve cair dentro do volume do cone, se houver ambiguidade ela é resolvido de acordo com a distância anisotrópica de cada objeto até a origem do cone (Figura 2.4). A noção do uso de um volume cônico para seleção é atualmente referida como *cone-casting*.

Oclusão permite a indicação pela sobreposição do objeto. Pierce et al. (PIERCE et al., 1997) desenvolveram uma família de técnicas que utilizam seleção no plano de imagem. Seleção no plano da imagem reduz a tarefa de *indicação de objeto* para 2 dimensões, como o usuário indica o objeto no plano de projeção da cena (*rendering*). Ele pode ser comparado com o cursor do *mouse* sobre uma área de trabalho habitual. As técnicas de seleção no plano de imagem propostas por Pierce et al. foram: *head crusher*, onde o polegar e o indicador são posicionados ao redor do objeto alvo, e um ponto médio entre estes é usado para a seleção; *sticky finger*, que consiste na utilização de um dedo estendido em direção ao alvo, o objeto sob o dedo no plano da imagem é selecionada; *lifting palm*, onde o usuário deve colocar o objeto alvo logo acima da palma da sua mão; *framing*

hands, o usuário posiciona as mãos em torno do objecto alvo, definindo dois cantos de uma moldura, o objeto no centro do retângulo é selecionado. A Figura 2.6 retrata as técnicas mencionadas acima.

8.1.2 Problemas de Seleção

As técnicas de seleção imediata, tais como as acima descritas, estão mais sujeitas a problemas de *precisão*, *ambiguidade* e *complexidade* de seleção. As técnicas baseadas em *refinamento progressivo* ou *níveis de precisão*, objeto de estudo deste trabalho, surgem para contornar estes problemas. Segue uma breve descrição destes problemas.

A *precisão de seleção* pode ser comprometida em técnicas como *ray-casting*. Esta geralmente usa 6 graus de liberdade de um dispositivo rastreado, três – posição do dispositivo – são usados para posicionar a origem do raio, e os outros três – orientação do dispositivo – são usados para definir a direção do raio. O controle angular do dispositivo apontador resulta em movimentos amplificados no ponto de intersecção do raio, sendo portanto predominante sobre o controle de posição para apontamento de objetos distantes. De acordo com o tamanho e distância do objeto, seu tamanho angular pode ser muito pequeno, fazendo com que a seleção se torne uma tarefa difícil e cansativo. Além disso, a instabilidade do sistema de rastreamento e/ou membros do usuário (geralmente a mão) afetará negativamente a precisão de técnicas baseadas em *ray-casting*.

Ambiguidade de seleção acontece quando o sistema não pode resolver qual objeto (ou grupo de objetos) que o usuário deseja, entre um subconjunto mais amplo de objetos indicados. Por exemplo, ao apontar em um ambiente sobrecarregado, múltiplos alvos podem cair no interior do volume de seleção ou ser intersectados pelo raio de apontamento, qual destes o usuário deseja selecionar? Regras de desambiguação automática são frequentemente adotadas, como o uso da interseção mais próxima para *ray-casting*, do alvo mais próximo para cursores de posição, e de regras mais específicas/complexas para *cone-casting* (como a de distância anisotrópica da técnica *spotlight*).

Complexidade de seleção: de acordo com Haan et al. (HAAN; KOUTEK; POST, 2005), uma seleção pode ser complexa quando conta com objetos animados ou que existem por um período limitado de tempo. Objetos animados variam seu tamanho angular e requerem sincronia nas ações do usuário, podem sair do campo de visão criando a necessidade de reposicionar o ponto de vista. De forma generalizada, estes são objetos que podem variar seu estado, atributos e posição no decorrer do tempo. Adicionalmente, incluímos a seleção de um grupo de objetos ao problema de complexidade de seleção.

8.1.3 Seleção por Níveis de Precisão

Compreende técnicas que aumentam o espaço de controle motor, melhorando assim a precisão de indicação de objetos do usuário. Pode ser implementado com a alteração dos parâmetros da etapa de *indicação do objeto*. Por exemplo, controle sobre o *ganho/proporção de controle-visualização* (*CD gain/CD ratio*), ou através da elaboração de uma técnica de seleção com mais que uma forma de *indicação do objeto*, como a mudança de uma abordagem de mapeamento absoluto para mapeamento relativo.

Abordagens com *níveis de precisão* têm uma forte ligação com o modelo de controle motor *impulso inicial otimizado* (MEYER et al., 1988), tido como a explicação mais completa e bem sucedida para a lei de Fitts (ROSENBAUM, 2010; BALAKRISHNAN, 2004). Na sua essência, defende que a maioria dos movimentos de indicação consistem em duas fases, de *movimento balístico* (grande e rápido, para próximo do objeto alvo), seguido pela fase de *finalização refinada* (movimentos curtos e lentos para correção). Us-

ando uma técnica com *níveis de precisão*, o movimento balístico pode ser realizado com uma técnica de indicação que possibilite cobrir grandes distâncias em um curto período, enquanto os movimentos de correção são realizadas por um método compatível com a precisão esperada pela aplicação. Na Figura 3.1 propomos um espaço de design para técnicas com *níveis de precisão*, onde as técnicas com n níveis de precisão podem ser construídas a partir de $(n-1)$ iterações deste espaço de design, com $n \geq 2$.

As técnicas com *níveis de precisão* variam desde mapeamento dinâmico de atributos do cursor em sistemas operacionais atuais – ganho/proporção controlados pela aceleração/velocidade (CASIEZ et al., 2008; CASIEZ; ROUSSEL, 2011; FREES; KESSLER; KAY, 2007) – até técnicas que combinam formas de mapeamento completamente diferentes (VOGEL; BALAKRISHNAN, 2005; FORLINES; VOGEL; BALAKRISHNAN, 2006; NANCEL; PIETRIGA; BEAUDOUIN-LAFON, 2011). A Tabela 3.1 classifica estas técnicas de acordo com o espaço de *design* proposto nesse trabalho.

Apenas a técnica *ARCpad* é descrita neste resumo (MCCALLUM; IRANI, 2009), visto que a utilizamos na avaliação comparativa do *LOP-cursor*. *ARCpad* combina controle absoluto e relativo de um cursor através da tela táctil de um telefone celular. O movimento do dedo enquanto mantendo o toque desencadeia o controle relativo sobre o cursor, análogo ao uso de um *touchpad* de computadores portáteis. Um toque rápido na tela táctil desencadeia um salto do cursor para uma posição equivalente no monitor externo, caracterizando um mapeamento absoluto (a Figura 3.3 ilustra esse procedimento).

8.1.4 Seleção por Refinamento Progressivo

Seleção por *refinamento progressivo* propõe a quebra de uma tarefa de seleção em subtarefas "sem esforço". O objetivo é evitar a atenção e precisão normalmente exigidas pelas técnicas de seleção imediata. No entanto, existe uma troca inevitável entre técnicas de seleção imediata e por refinamento progressivo. Refinamento progressivo requer um processo para completar a seleção, este processo geralmente consiste em mais do que uma subtarefa rápida, e resulta em maior precisão e maior tempo somado. Por outro lado, as técnicas de seleção imediata consistem em realizar a seleção em um único passo, mais lento e menos preciso.

A expressão *refinamento progressivo* foi apresentado por Kopper et al. (KOPPER; BACIM; BOWMAN, 2011), descrita como uma abordagem para reduzir progressivamente o grupo de objetos selecionáveis, e consequentemente reduzir a precisão necessária para indicar um objeto. O espaço de design para a seleção por meio de técnicas de refinamento progressivo foi expandida por Bacim et al. (BACIM; KOPPER; BOWMAN, 2013), e é apresentado na Figura 3.4. Neste trabalho agrupamos as técnicas pelo método de desambiguação (*menu*, *zoom*, *acumulação de pontos* e *outras abordagens*), apresentadas somente na Seção 3.2. A Tabela 3.2 classifica estas técnicas de acordo com o espaço de *design* proposto por Kopper, Bacim e Bowman.

Ao propor o espaço de *design* de técnicas de seleção por *refinamento progressivo*, Kopper et al. (KOPPER; BACIM; BOWMAN, 2011) também apresentaram a técnica *SQUAD*, *sphere-casting* refinado por *menu quádruplo*. Utilizamos esta técnica e a metodologia de avaliação como referência para os estudos da técnica *DCanvas*. *SQUAD* consiste em definir um subconjunto de objetos através da intersecção com um volume esférico, e refinar este subconjunto através de repetidos menus com quatro opções (onde os objetos são distribuídos), até que sobre apenas apenas o objeto alvo em uma das opções do menu. A posição da esfera é controlada utilizando *sphere-casting*, em que a primeira intersecção de um raio define a posição do centro da esfera no espaço. A Figura 3.7 demonstra o uso

da técnica.

8.2 Técnicas Propostas

8.2.1 LOP-cursor

O *LOP-cursor* foi inicialmente desenvolvido para interação com interfaces 2D em uma parede de monitores, ele será explicado neste contexto, sua adaptação para ambientes 3D e imersivos é discutida no final dessa seção. A técnica *LOP-cursor* usa dois níveis de precisão. No primeiro, o usuário aponta o dispositivo para o alvo nos monitores, o cursor é então levado ao ponto de intersecção da direção de apontamento com os monitores (Figura 4.1a). Se o usuário for capaz de apontar para o alvo desejado, ele pode confirmar a seleção com um gesto de *tap* (toque rápido) na tela táctil. Caso contrário, pode passar para o segundo nível de precisão iniciando e mantendo um toque na tela táctil, o ajuste fino da posição do cursor é realizado com o deslizar do dedo sobre a tela (ver Figuras 4.1b e 4.1c). Diferente da maioria das técnicas com *níveis de precisão*, *LOP-cursor* usa mapeamento absoluto em ambos os níveis, *ray-casting* no primeiro nível, e um retângulo que representa a tela táctil do dispositivo móvel nos monitores externos no segundo nível.

Quatro formas complementares são usadas para representar o cursor no monitor externo. Duas são necessárias para realizar a seleção, a *flecha* (metáfora comum de cursor para seleção pontual), e o *retângulo*, que torna explícito o mapeamento absoluto entre a tela táctil do celular com uma área do monitor externo (e.g. um toque em uma extremidade da tela táctil resulta no posicionamento equivalente da *flecha* dentro do *retângulo*). Adicionalmente, um *anel* e uma *linha* são utilizados para evitar a perda da direção de apontamento do dispositivo móvel no segundo nível de precisão. O *anel* sempre indica a direção apontada pelo dispositivo móvel, enquanto a *linha* conecta o *anel* e a *flecha*, guiando a mudança de atenção do usuário entre os dois.

Enquanto no primeiro nível de precisão, que consiste em apontar com *ray-casting*, a *flecha* e o *anel* seguem a direção de apontamento do dispositivo móvel. A *flecha* é mantida no centro do anel, e um *retângulo* semitransparente com bordas tracejadas é mostrado para manter o utilizador ciente do alcance de seleção ao passar para o segundo nível. Ao mudar de primeiro para o segundo nível de precisão, um efeito de transição orienta a atenção do usuário entre os estados. O efeito torna o *retângulo* mais opaco, e a transparência da *flecha* e do *anel* são controladas conforme descrito na Figura 4.2. O efeito inverso é usado ao retornar para o primeiro nível de precisão.

O uso da tela sensível ao toque para ajuste fino no segundo nível de precisão trouxe restrições a etapa de *confirmação de seleção*. Por exemplo, enquanto o usuário desliza o dedo sobre a tela sensível ao toque, não podemos usar o gesto *tap* (iniciar e finalizar um toque com mínima movimentação do dedo, análogo ao clic de um mouse) para confirmar a seleção, que seria a ação mais óbvia. Para explorar e compreender tais restrições, foram implementadas e testadas cinco técnicas de confirmação de seleção: três delas sem necessidade de modificar o dispositivo móvel, *decolagem* (encerramento do toque), *tap-com-segundo-dedo* e *tap-após-decolagem*; duas técnicas anexando dispositivos auxiliares ao celular, *tap-na-superfície-táctil-traseira* e *clic-traseiro-em-mouse-adaptado* (Figuras 4.4 b e c respectivamente). Com base em avaliação preliminar comparando as três técnicas de *confirmação de seleção* mais promissoras (omitido nesse resumo, Seção 6.1.1), escolhemos a *clic-traseiro-em-mouse-adaptado* como padrão. Adicionalmente, o *LOP-cursor* também suporta a seleção imediata de um objeto (se o usuário decidir que é capaz de selecionar o alvo sem o segundo nível de precisão), nesse caso adotamos o gesto *tap* para

confirmar a seleção.

O tamanho do retângulo pode ser controlado pelo usuário, resultando no controle sobre a proporção controle-visualização do segundo nível de precisão do *LOP-cursor* (como o mapeamento é absoluto, o aumento/redução das dimensões do retângulo resulta em menor/maior precisão). O redimensionamento pode ser feito com a rotação do dispositivo móvel em torno de seu eixo vertical (rotação esta desnecessário para o apontamento, Figura 4.5), ou por gestos de *pinch* e *stretch*. O retângulo pode ser redimensionado apenas enquanto na primeira fase de apontamento.

Para a seleção de múltiplos objetos com o *LOP-cursor*, propomos o traçado de uma forma livre utilizando a tela táctil. Enquanto no segundo nível de precisão, o usuário pode iniciar e manter um clic no *mouse* conectado ao dispositivo, e desenhar a forma livre durante este clic. Quando o clic é finalizado, objetos que intersectam esta forma livre são selecionados (Figura 4.6). O mapeamento absoluto torna intuitivo o controle da *flecha* com o polegar enquanto desenhando a forma livre.

Para seleção em ambientes 3D – em que objetos podem estar localizados a frente e atrás da tela (Figura 4.7) – propomos trabalhar com a projeção das imagens na tela, caracterizando portanto uma técnica de seleção no plano de imagem (Seções 2.2 e 8.1). Dois problemas surgem se considerarmos que o ambiente virtual 3D utiliza estereoscopia e correção perspectiva. Primeiro, estereoscopia produz dois planos de imagem, podendo resultar em ambiguidade (Figura 4.8). Segundo, a correção da projeção perspectiva de acordo com a posição do usuário modifica a projeção no plano de imagem (Figura 4.10) tornando o segundo nível de precisão instável. Para tratar o primeiro problema, adotamos a técnica *viewfinder*, proposta por Argelaguet e Andujar (ARGELAGUET; ANDUJAR, 2009). Esta técnica consiste em um retângulo de monoscopia, neste retângulo a imagem produzida para o olho dominante é reutilizada para o olho não dominante, enquanto o resto da cena é renderizada com estereoscopia (Figura 4.9). Na prática, *viewfinder* é analogo ao uso de uma câmera digital no mundo real. A solução para o segundo problema estende essa semelhança do *viewfinder* com uma câmera digital, utilizando a transição do primeiro para o segundo nível para gerar uma imagem paralisada da cena (fotografia), onde a seleção pode ser feita com precisão, esta imagem é descartada ao fim do toque. Esta abordagem também facilita a seleção de objetos animados, uma vez que sua imagem será mantida estática dentro do retângulo pela duração do toque.

8.2.2 DCanvas

DCanvas é uma técnica de seleção por *refinamento progressivo* que utiliza a direção de apontamento do dispositivo móvel para definir um conjunto de objetos, o objeto alvo a ser selecionado neste conjunto é apontado com auxílio da tela táctil (desambiguação), consistindo portanto em duas fases. Na primeira fase o usuário utiliza a técnica de volume de seleção para apontar na direção do objeto alvo (Figura 4.11a). Quando o objecto alvo estiver dentro ou intersectando o volume de seleção, o usuário pode iniciar um toque na tela táctil para entrar na segunda fase, de desambiguação. Um retângulo aparece a uma distância de 70 cm do utilizador, alinhado paralelo ao plano de imagem, e os objetos do conjunto movem-se numa curta animação para formar uma matriz dentro deste retângulo (Figura 4.11b). O retângulo tem um mapeamento absoluto com a tela táctil do dispositivo móvel, deslizar o polegar sobre a tela permite a sobreposição do objeto desejado pela seta (Figura 4.11c). A seleção é realizada pela retirada do polegar (decolagem), o objeto sob a *flecha* é selecionado, enquanto os outros retornam as suas posições originais e a técnica retorna ao primeiro nível. Se o usuário deseja sair da fase de desambiguação sem

selecionar nenhum dos objetos do conjunto, pode simplesmente finalizar o toque quando não houver nenhum objeto sob a *flecha*. Essa técnica foi desenvolvida para ambientes virtuais imersivos, mas sua adaptação para ambientes virtuais 2D e 3D não imersivo é trivial.

Duas abordagens podem ser usadas na fase de volume de seleção, *cone-casting* e *sphere-casting*, qual a mais adequada depende da tarefa e tipo de ambiente virtual. Com *sphere-casting*, a quantia de objetos a ser desambiguada deve ser menor, uma vez que a esfera possui sempre profundidade limitada. Por outro lado, a profundidade do volume cônico com *cone-casting* é potencialmente ilimitada. Idealmente este cone deve ser interrompido por obstáculos que definem divisões do espaço, tal como paredes e prateleiras.

Assim como para o *LOP-cursor*, uma *flecha* é usada como cursor e um *retângulo* mapeia a tela táctil para o ambiente virtual. Estas representações são visíveis apenas durante a segunda fase, de desambiguação. No lugar de utilizar o *anel* para representar a direção de apontamento do dispositivo móvel, um cone/esfera (de acordo com a técnica de volume de seleção adotada) é utilizado. Assim como acontece com o *anel* no *LOP-cursor*, a representação do volume de seleção sempre segue a direção apontada pelo dispositivo móvel, sendo sempre visível.

Ao transitar para a segunda fase da técnica *DCanvas*, o conjunto de objetos apontados deve ser reorganizado sobre o retângulo que representa a tela táctil. Uma matriz é formada de acordo com a quantia de objetos para subdividir o espaço do retângulo, e cada objeto recebe uma das posição na matriz. Para que os objetos não ultrapassem os limites de cada nó da matriz, eles podem ser redimencionados de acordo com o maior objeto, ou maximizados de forma a utilizar todo o espaço disponível no nó da matriz. Outra observação relevante é que a maioria dos usuários é incapaz de alcançar toda a área da tela táctil com o polegar, então propomos a redução da área útil em que os objetos vão ser reorganizados de modo a facilitar seu alcance. Como conseqüência, o tamanho efetivo para seleção de objetos é reduzido e uma grande porção da tela táctil é inutilizada. No entanto, como ainda assim obtivemos taxas de erro baixas nestas condições – mesmo na desambiguação entre mais de 200 objetos (Figura 4.12c) –, concluímos que esta troca é benéfica. Propomos duas segmentações distintas para a área útil da tela táctil (Figura 4.13). Nos os experimentos utilizamos a primeira segmentação (Figura 4.13a).

8.2.3 Two-legged cursor

Ambas as técnicas apresentadas usam dois equipamentos de entrada distintos, os sensores de movimento e a tela táctil de um celular. Durante o desenvolvimento da técnica, percebemos que os usuários não têm problema para alternar entre eles, e além disso, são capazes de manter algum nível de controle simultâneo sobre ambos meios de entrada. Esta constatação deu origem ao que chamamos de *two-legged cursor*, um cursor que permite a realização de tarefas compostas, como arrastar e soltar, com apenas uma *confirmação de seleção*.

O ajuste fino/desambiguação (*LOP-cursor/DCanvas*) pela tela táctil congela o retângulo pela duração do toque, mas não elimina o recurso de apontamento com a orientação do dispositivo, o que resulta em dois cursores sob o controle do usuário. Este *design* permite o apontamento de dois locais simultaneamente (as duas pernas do cursor) para executar rapidamente tarefas compostas como, por exemplo, medir distância ou arrastar e soltar objetos. Figura 4.1 (a, b e d) apresenta o passo a passo sobre o uso simultâneo de ambas as pernas do cursor com a técnica *LOP-cursor*. Para o *LOP-cursor* e *DCanvas* a *flecha* representa a *perna-1* do cursor, enquanto o *anel* e a *esfera/cone* representam

a *perna-2* do cursor. A *linha* presente no *LOP-cursor* é útil para guiar a busca visual quando o usuário alterna a atenção entre as pernas do cursor.

8.3 Experimentos e Resultados

Foram realizados dois experimentos com cada uma das técnicas apresentadas. Um terceiro experimento (preliminar) para decidir a técnica de *confirmação de seleção* para o *LOP-cursor* foi realizado mas não é apresentado neste resumo (Seção 6.1.1). A metáfora do *two-legged cursor* é brevemente explorada em um dos experimentos do *LOP-cursor*.

8.3.1 Avaliação da Técnica LOP-cursor

Avaliação comparativa: o *LOP-cursor* foi comparado com as técnicas *ARCpad* (MCCALLUM; IRANI, 2009) e *ray-casting*. O *LOP-cursor* utilizado requer o uso de ambos os níveis de precisão para uma seleção e é referido aqui como *CLOP-cursor* (passos descritos na Figura 4.1abc). A apresentação das três técnicas foi contrabalanceada. Ao final do teste comparativo, usuários utilizaram a técnica *LOP-cursor* sem a restrição mencionada acima, afim de determinar para que tamanho de objeto o usuário prefere utilizar o nível de menor ou de maior precisão. A tarefa para esta avaliação consistiu em selecionar quadrados de diferentes tamanhos em nossa parede de monitores (Figura 5.1). Para a análise dos resultados utilizamos o tempo de seleção, a proporção de erros, e a preferência subjetiva de técnica dos usuários. Mais detalhes referentes a implementação e ao *design* do experimento estão disponíveis nas Seções 5 e 6.1.2.

Resultados: onze estudantes de Ciências da Computação participaram do experimento, três dos quais não puderam participaram da etapa com o *LOP-cursor* sem restrição (etapa pós comparativa). Os tempos médios para seleção com *CLOP-cursor*, *ORayCasting* e *ARCpad* foram respectivamente 2,55, 2,46 e 3,05 segundos. ANOVA unidirecional mostrou que *ARCpad* foi significativamente mais lento que *CLOP-cursor* ($F(1.1574)=76.58$, $p<0.0001$) e *ORayCasting* ($F(1.1567)=84.81$, $p<0.0001$), mais detalhes na Figura 6.2. A proporção de erros com *ORayCasting* foi significativamente maior do que com *CLOP-curor*($F(1.1571)=109.08$, $p<0.0001$) e *ARCpad*($F(1.1567)=120.6$, $p<0.0001$), a diferença de erros não foi significante entre *CLOP-cursor* e *ARCpad* ($F(1.1574)=0.48$, $p<0.49$). Mais detalhes são apresentados na Figura 6.3. No etapa pós-comparativa, com o *LOP-cursor* sem restrição de nível para seleção, sua proporção de erro seguiu uma tendência coerente com os resultados das técnicas *CLOP-cursor* e *ORayCasting* (Figuras 6.3 e 6.4). A preferência subjetiva dos usuários quanto ao uso de um ou dois níveis de precisão para seleção de acordo com o tamanho do alvo é apresentada na Figura 6.5. Para alvos com 0,5 e 1 cm os usuários sempre utilizados dois níveis. O *LOP-cursor* apresentado no final foi preferido por 7 dos 8 participantes que realizaram o teste completo.

Avaliação aprofundada: nessa avaliação foram exploradas três tarefas distintas com o *LOP-cursor*. A primeira é similar a tarefa do teste anterior, mas utiliza alvos ainda menores (até 0,3 cm). A segunda consiste em selecionar e reposicionar objetos sobre objetos sombra de mesma dimensão, da forma mais precisa possível. A terceira tarefa usa a metáfora do *two-legged cursor*, onde o usuário deve apontar com a *flecha* (*perna 1*, de maior precisão) para um alvo no lado esquerdo da tela ao mesmo tempo em que aponta para uma tela de cor correspondente no lado direito com o *anel* (*perna 2*, de menor precisão), tal como representado na Figura 5.1. As três tarefas foram aplicadas sempre nessa ordem, ao fim do experimento os usuários responderam ao questionário de *Escala de Usabilidade do Sistema* (*SUS*). Mais detalhes quando a implementação, metodologia

e *design* deste experimento estão disponíveis nas Seções 5 e 6.1.3.

Resultados: onze estudantes de Ciências da Computação participaram do experimento. Todos foram capazes de selecionar todos os alvos. A Figura 6.6 demonstra o tempo médio para completar as tarefas. O erro médio na tarefa 2 (posicionamento do alvo sobre sua sombra) foi de 0,17 cm. A seleção de alvos com 0,3 cm obteve elevada proporção de erros, 42% e 39% para as tarefas 1 e 3 respectivamente. Com base em relatos e observações dos usuários, acreditamos que fatores ergonômicos, mais especificamente o tamanho do dispositivo (celular com o mouse acoplado), foi o principal responsável pela instabilidade. Novos testes seriam necessários para verificar se existe melhora na precisão com a redução do protótipo. Nossa técnica obteve a pontuação de 77 no questionário SUS. A metáfora de *two-legged cursor* se mostrou funcional através da tarefa 3, mas ainda é preciso verificar sua eficiência, assim como validar seu uso em tarefa específicas que tirem vantagem do conceito.

8.3.2 Avaliação da Técnica DCanvas

Os experimentos com a técnica *DCanvas* foram semelhantes entre eles. Ambos consistiram na comparação de nossa técnica com *ray-casting*, tendo o *design* baseado no experimento realizado por Kopper et al. para avaliação da técnica *SQUAD* (KOPPER; BACIM; BOWMAN, 2011). *Sphere-casting* foi usado para a primeira fase de seleção com *DCanvas*. A tarefa do usuário era selecionar alvos de diversos tamanhos e cercado por quantias diferentes de distraçoes (outros objetos). Quanto mais distrações, mais objetos vão para a fase de desambiguação, tornando a seleção mais difícil (Figura4.12). A figura 5.2 demonstra a cena virtual e o posicionamento do usuário, mais detalhes quanto a implementação, procedimento e *design* dos experimentos são apresentados na Seção 6.2.

Resultados da avaliação 1: dez estudantes de Ciência da Computação participaram do experimento. O tempo médio para seleção com *DCanvas* e *ORayCasting* foi de 2,67 e 2,56 segundos respectivamente, diferença significativa (ANOVA unidirecional, $F(1.1783)=5.83$, $p<0.016$). A proporção de erros com *ORayCasting* foi significativamente maior ($F(1.1783)=135$, $p<0.0001$). Dois usuários enfrentaram dificuldade para alcançar a parte superior da tela táctil, apresentando piores tempos e mais erros. A média de erros foi de 3.8% com nossa técnica, e 21.4% com *ORayCasting*. Maiores detalhes quanto ao tempo e proporção de erros de acordo com fatores estão disponíveis nas Figuras 6.8 e 6.9.

Resultados da avaliação 2: seis pós-graduandos em Ciências da Computação participaram do experimento. O tempo médio para seleção com *DCanvas* e *ORayCasting* foi de 2,37 e 2,29 segundos respectivamente, diferença significativa (ANOVA unidirecional, $F(1.1078)=4.43$, $p<0.036$). Por outro lado a proporção de erros com *ORayCasting* foi significativamente maior ($F(1.1078)=70.34$, $p<0.0001$). Ao contrário do experimento anterior, nenhum usuário teve problemas em alcançar o segmento da tela táctil onde os objetos selecionáveis foram dispostos. Com isso a proporção de erros da *DCanvas* foi reduzida para 0,9%, compatível com os 0,7% obtido pela técnica *SQUAD* (KOPPER; BACIM; BOWMAN, 2011). Maiores detalhes quanto ao tempo e proporção de erros de acordo com fatores estão disponíveis nas Figuras 6.10 e 6.11.

Questionários: os questinários aplicados nos experimentos foram os mesmos, e o resultado somado é apresentado na Figura 6.12. Destacamos que nossa técnica causou menos fadiga, foi considerada mais fácil de usar, mais precisa, e até mesmo mais rápida, o que contraria os resultados do teste (*DCanvas* foi mais rápido apenas para objetos pequenos). Atribuímos esta incorreta avaliação do tempo a atenção (e tensão) exigida para

controlar o *RayCasting*, observamos usuários gerenciando até mesmo a respiração para estabilizar o apontamento.

8.4 Conclusões e Trabalhos Futuros

Nesta dissertação apresentamos as técnicas *LOP-cursor* e *DCanvas* para seleção rápida e precisa em ambientes virtuais. Propusemos também um espaço de *design* para as técnicas que dependem de *níveis de precisão*, bem como classificamos as principais técnicas que utilizam seleção por *níveis de precisão* ou *refinamento progressivo*. Por fim, apresentamos a metáfora de cursor *two-legged cursor*, que permite definir simultaneamente duas posições distintas, podendo ser usada para realização de tarefas compostas.

Usando a tela táctil no segundo nível de precisão do *LOP-cursor*, usuários puderam selecionar objetos representados em uma área motora de $\approx 0.6mm^2$ da tela táctil durante a avaliação. Se transferir esse parâmetro para a *DCanvas*, a superfície da tela táctil permitiria acomodar uma desambiguação entre 6.144 objetos em uma única etapa. Esta precisão submilimétrica não é percebida no uso diário de dispositivos móveis, já o dedo obstrui o conteúdo da tela táctil impedindo a percepção do local tocado. Enfatizamos também a avaliação subjetiva dos usuários de que a técnica *DCanvas* foi mais rápida que *ray-casting*, sendo que na verdade foi um pouco mais lenta. Isso pode ser um indício de quão desagradável é realizar uma seleção difícil, que requer tanta atenção que até mesmo a respiração tem que ser controlado algumas vezes.

Queremos descobrir o limite de precisão do *LOP-cursor*. Para tanto pretende-se reduzir o tamanho do protótipo, e verificar o uso de outras formas para substituir a *flecha* como cursor principal. Adicionalmente, queremos saber como os usuários irão lidar com a funcionalidade de redimencionamento do retângulo, e se o *LOP-cursor* terá também boa performance em ambientes imersivo. Pretendemos ainda investigar como a distância do usuário para a tela afeta a interação com o *LOP-cursor*. Como demonstrado por Peck et al. (PECK; NORTH; BOWMAN, 2009), os usuários tendem a interagir em diferentes escalas de acordo com a distância dos monitores.

Com relação a técnica *DCanvas*, pretendemos realizar uma comparação direta com a técnica *SQUAD* e algumas otimizações. Alguns usuários tiveram dificuldade para alcançar toda a área de desambiguação da tela táctil definida pela segmentação padrão, permitiremos que o usuário calibre sua própria segmentação da região de desambiguação, garantindo alcance total e estável.

Recentemente iniciamos uma colaboração com o grupo de pesquisa de engenharia de minas da nossa instituição (UFRGS), em que vamos usar o *LOP-cursor* e a metáfora *two-legged cursor* para interagir com imagens de satélite de alta resolução em nossa parede de monitores. Nossas técnicas serão usadas para tarefas como: medição rápida da distância entre dois locais; definição de parâmetros; organização de informações; inserção de anotações nas imagens. A adaptação das ferramentas apresentadas para esta aplicação permitirá a validação ecológico, avaliando a técnica em uma aplicação real, e permitindo compará-la com outras opções disponíveis nesse contexto. Por fim, esta colaboração pode possibilitar também a avaliação do uso coletivo e colaborativo de nossas técnicas.

REFERENCES

ARGELAGUET, F.; ANDUJAR, C. Visual feedback techniques for virtual pointing on stereoscopic displays. In: ACM SYMPOSIUM ON VIRTUAL REALITY SOFTWARE AND TECHNOLOGY, 16., New York, NY, USA. **Proceedings...** ACM, 2009. p.163–170. (VRST '09).

ARGELAGUET, F.; ANDUJAR, C.; TRUEBA, R. Overcoming eye-hand visibility mismatch in 3D pointing selection. In: ACM SYMPOSIUM ON VIRTUAL REALITY SOFTWARE AND TECHNOLOGY, 2008., New York, NY, USA. **Proceedings...** ACM, 2008. p.43–46. (VRST '08).

BACIM, F.; KOPPER, R.; BOWMAN, D. Design and evaluation of 3D selection techniques based on progressive refinement. **International Journal of Human-Computer Studies**, [S.l.], 2013.

BALAKRISHNAN, R. Beating Fitts law: virtual enhancements for pointing facilitation. **International Journal of Human-Computer Studies**, [S.l.], v.61, n.6, p.857 – 874, 2004. <ce:title>Fitts' law 50 years later: applications and contributions from human-computer interaction</ce:title>.

BALLAGAS, R. et al. The Smart Phone: a ubiquitous input device. **IEEE Pervasive Computing**, Piscataway, NJ, USA, v.5, p.70–, January 2006.

BALLAGAS, R.; ROHS, M.; SHERIDAN, J. G. Sweep and point and shoot: phonecam-based interactions for large public displays. In: CHI '05 EXTENDED ABSTRACTS ON HUMAN FACTORS IN COMPUTING SYSTEMS, New York, NY, USA. **Anais...** ACM, 2005. p.1200–1203. (CHI EA '05).

BAUDISCH, P.; CHU, G. Back-of-device interaction allows creating very small touch devices. In: HUMAN FACTORS IN COMPUTING SYSTEMS, 27., New York, NY, USA. **Proceedings...** ACM, 2009. p.1923–1932. (CHI '09).

BI, X.; BAE, S.-H.; BALAKRISHNAN, R. Effects of interior bezels of tiled-monitor large displays on visual search, tunnel steering, and target selection. In: HUMAN FACTORS IN COMPUTING SYSTEMS, 28., New York, NY, USA. **Proceedings...** ACM, 2010. p.65–74. (CHI '10).

BIER, E. A. et al. Toolglass and magic lenses: the see-through interface. In: COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 20., New York, NY, USA. **Proceedings...** ACM, 1993. p.73–80. (SIGGRAPH '93).

BLANCH, R.; GUIARD, Y.; BEAUDOUIN-LAFON, M. Semantic pointing: improving target acquisition with control-display ratio adaptation. In: SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, New York, NY, USA. **Proceedings...** ACM, 2004. p.519–526. (CHI '04).

BORING, S. et al. Touch projector: mobile interaction through video. In: HUMAN FACTORS IN COMPUTING SYSTEMS, 28., New York, NY, USA. **Proceedings...** ACM, 2010. p.2287–2296. (CHI '10).

BORING, S. et al. Multi-user interaction on media facades through live video on mobile devices. In: HUMAN FACTORS IN COMPUTING SYSTEMS, 2011., New York, NY, USA. **Proceedings...** ACM, 2011. p.2721–2724. (CHI '11).

BOWMAN, D. A. et al. **3D User Interfaces: theory and practice**. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 2004.

CASIEZ, G. et al. The Impact of Control-Display Gain on User Performance in Pointing Tasks. **HUMAN-COMPUTER INTERACTION**, [S.l.], v.23, n.3, p.215–250, 2008.

CASIEZ, G.; ROUSSEL, N. No more bricolage!: methods and tools to characterize, replicate and compare pointing transfer functions. In: ACM SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 24., New York, NY, USA. **Proceedings...** ACM, 2011. p.603–614. (UIST '11).

DANG, N.-T.; LE, H.-H.; TAVANTI, M. Visualization and interaction on flight trajectory in a 3D stereoscopic environment. In: DIGITAL AVIONICS SYSTEMS CONFERENCE, 2003. DASC '03. THE 22ND. **Anais...** [S.l.: s.n.], 2003. v.2, p.9.A.5 –91–10 vol.2.

DEBARBA, H.; NEDEL, L.; MACIEL, A. LOP-cursor: fast and precise interaction with tiled displays using one hand and levels of precision. In: D USER INTERFACES (3DUI), 2012 IEEE SYMPOSIUM ON, 3. **Anais...** [S.l.: s.n.], 2012. p.125–132.

FORLINES, C. et al. Zoom-and-pick: facilitating visual zooming and precision pointing with interactive handheld projectors. In: ACM SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 18., New York, NY, USA. **Proceedings...** ACM, 2005. p.73–82. (UIST '05).

FORLINES, C.; VOGEL, D.; BALAKRISHNAN, R. HybridPointing: fluid switching between absolute and relative pointing with a direct input device. In: ACM SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 19., New York, NY, USA. **Proceedings...** ACM, 2006. p.211–220. (UIST '06).

FORSBERG, A.; HERNDON, K.; ZELEZNIK, R. Aperture based selection for immersive virtual environments. In: ACM SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 9., New York, NY, USA. **Proceedings...** ACM, 1996. p.95–96. (UIST '96).

FREES, S.; KESSLER, G. D.; KAY, E. PRISM interaction for enhancing control in immersive virtual environments. **ACM Trans. Comput.-Hum. Interact.**, New York, NY, USA, v.14, n.1, May 2007.

GROSSMAN, T.; BALAKRISHNAN, R. The design and evaluation of selection techniques for 3D volumetric displays. In: ACM SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 19., New York, NY, USA. **Proceedings...** ACM, 2006. p.3–12. (UIST '06).

HAAN, G. D.; KOUTEK, M.; POST, F. H. IntenSelect: using dynamic object rating for assisting 3d object selection. In: IN VIRTUAL ENVIRONMENTS 2005. **Anais...** [S.l.: s.n.], 2005. p.201–209.

HUTAMA, W. et al. Distinguishing multiple smart-phone interactions on a multi-touch wall display using tilt correlation. In: HUMAN FACTORS IN COMPUTING SYSTEMS, 2011., New York, NY, USA. **Proceedings...** ACM, 2011. p.3315–3318. (CHI '11).

JEON, S. et al. Interaction techniques in large display environments using hand-held devices. In: ACM SYMPOSIUM ON VIRTUAL REALITY SOFTWARE AND TECHNOLOGY, New York, NY, USA. **Proceedings...** ACM, 2006. p.100–103. (VRST '06).

JIANG, H. et al. Direct pointer: direct manipulation for large-display interaction using handheld cameras. In: SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, New York, NY, USA. **Proceedings...** ACM, 2006. p.1107–1110. (CHI '06).

KATZAKIS, N.; HORI, M. Mobile Phones as 3-DOF Controllers: a comparative study. In: DEPENDABLE, AUTONOMIC AND SECURE COMPUTING, 2009. DASC '09. EIGHTH IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2009. p.345 –349.

KOBAYASHI, M.; IGARASHI, T. Ninja cursors: using multiple cursors to assist target acquisition on large screens. In: PROCEEDING OF THE TWENTY-SIXTH ANNUAL SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, New York, NY, USA. **Anais...** ACM, 2008. p.949–958. (CHI '08).

KOPPER, R.; BACIM, F.; BOWMAN, D. A. Rapid and accurate 3D selection by progressive refinement. In: IEEE SYMPOSIUM ON 3D USER INTERFACES, 2011., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2011. p.67–74. (3DUI '11).

KOPPER, R. et al. A human motor behavior model for distal pointing tasks. **Int. J. Hum.-Comput. Stud.**, Duluth, MN, USA, v.68, n.10, p.603–615, Oct. 2010.

LIANG, J.; GREEN, M. JDCAD: a highly interactive 3d modeling system. **Computers & Graphics**, [S.l.], v.18, n.4, p.499–506, 1994.

LUCAS, J. F. Design and evaluation of 3D multiple object selection techniques. In: MATER'S THESIS, VIRGINIA POLYTECHNIC INSTITUTE AND STATE UNIVERSITY, Blacksburg, VA, USA. **Anais...** [S.l.: s.n.], 2005.

MADGWICK, S. O. H.; HARRISON, A. J. L.; VAIDYANATHAN, R. Estimation of IMU and MARG orientation using a gradient descent algorithm. In: REHABILITATION ROBOTICS (ICORR), 2011 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2011. p.1 –7.

MCCALLUM, D. C.; IRANI, P. ARC-Pad: absolute+relative cursor positioning for large displays with a mobile touchscreen. In: ACM SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 22., New York, NY, USA. **Proceedings...** ACM, 2009. p.153–156. (UIST '09).

MEYER, D. E. et al. Optimality in human motor performance: ideal control of rapid aimed movements. **Psychological Review**, [S.l.], v.95, p.340–370, 1988.

MINE, M. **Virtual Environment Interaction Techniques**. [S.l.]: UNC Chapel Hill CS Dept, 1995.

NANCEL, M.; PIETRIGA, E.; BEAUDOUIN-LAFON, M. **Precision Pointing for Ultra-High-Resolution Wall Displays**. [S.l.]: INRIA, 2011. Research Report. (RR-7624).

Ogre 3D. **Ogre 3D**. Available at <http://www.ogre3d.org/>. Access in July 30, 2012.

OLWAL, A. LightSense: enabling spatially aware handheld interaction devices. In: IEEE AND ACM INTERNATIONAL SYMPOSIUM ON MIXED AND AUGMENTED REALITY, 5., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2006. p.119–122. (ISMAR '06).

PEARS, N.; JACKSON, D. G.; OLIVIER, P. Smart Phone Interaction with Registered Displays. **IEEE Pervasive Computing**, Piscataway, NJ, USA, v.8, p.14–21, April 2009.

PECK, S. M.; NORTH, C.; BOWMAN, D. A multiscale interaction technique for large, high-resolution displays. In: IEEE SYMPOSIUM ON 3D USER INTERFACES, 2009., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2009. p.31–38. (3DUI '09).

PIERCE, J. S. et al. Image plane interaction techniques in 3D immersive environments. In: INTERACTIVE 3D GRAPHICS, 1997., New York, NY, USA. **Proceedings...** ACM, 1997. p.39–ff. (I3D '97).

POUPYREV, I. et al. The go-go interaction technique: non-linear mapping for direct manipulation in vr. In: ACM SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 9., New York, NY, USA. **Proceedings...** ACM, 1996. p.79–80. (UIST '96).

ROSENBAUM, D. A. **Human Motor Control (2. ed.)**. [S.l.]: Academic Press, 2010. I-XVI, 1-505p.

Sensics. **zSight Integrated SXGA HMD**. Available at <http://sensics.com/products/head-mounted-displays/zsight-integrated-sxga-hmd>. Access in July 30, 2012.

SILVA, T. R. d.; NEDEL, L. Um estudo de interação com displays grandes usando dispositivos iOS. In: BACHELOR'S THESIS, UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL, Porto Alegre, Brasil. **Anais...** [S.l.: s.n.], 2011.

SONG, P. et al. WYSIWYF: exploring and annotating volume data with a tangible hand-held device. In: HUMAN FACTORS IN COMPUTING SYSTEMS, 2011., New York, NY, USA. **Proceedings...** ACM, 2011. p.1333–1342. (CHI '11).

STEED, A. Towards a General Model for Selection in Virtual Environments. In: D USER INTERFACES, 3., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2006. p.103–110. (3DUI '06).

STEED, A.; PARKER, C. 3D Selection Strategies for Head Tracked and Non-Head Tracked Operation of Spatially Immersive Displays. In: INTERNATIONAL IMMERSIVE PROJECTION TECHNOLOGY WORKSHOP, 8. **Anais...** [S.l.: s.n.], 2004.

VANACKEN, L.; GROSSMAN, T.; CONINX, K. Exploring the Effects of Environment Density and Target Visibility on Object Selection in 3D Virtual Environments. In: D USER INTERFACES, 2007. 3DUI '07. IEEE SYMPOSIUM ON, 3. **Anais...** [S.l.: s.n.], 2007.

VOGEL, D.; BALAKRISHNAN, R. Distant freehand pointing and clicking on very large, high resolution displays. In: ACM SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 18., New York, NY, USA. **Proceedings...** ACM, 2005. p.33–42. (UIST '05).

WEISER, M. The computer for the 21st century. **SIGMOBILE Mob. Comput. Commun. Rev.**, New York, NY, USA, v.3, n.3, p.3–11, July 1999.

WIGDOR, D.; BALAKRISHNAN, R. A comparison of consecutive and concurrent input text entry techniques for mobile phones. In: SIGCHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, New York, NY, USA. **Proceedings...** ACM, 2004. p.81–88. (CHI '04).

WIGDOR, D. et al. Lucid touch: a see-through mobile device. In: ACM SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 20., New York, NY, USA. **Proceedings...** ACM, 2007. p.269–278. (UIST '07).

APPENDIX A - ARTICLES PUBLISHED DURING THIS WORK

During the development of this work, two related articles were published. The following pages contains the full text of these articles.

The Cube of Doom: a Bimanual Perceptual User Experience

Henrique Debarba* Juliano Franz† Vitor Reus‡ Anderson Maciel§ Luciana Nedel¶

Instituto de Informática (INF)
Universidade Federal do Rio Grande do Sul (UFRGS)

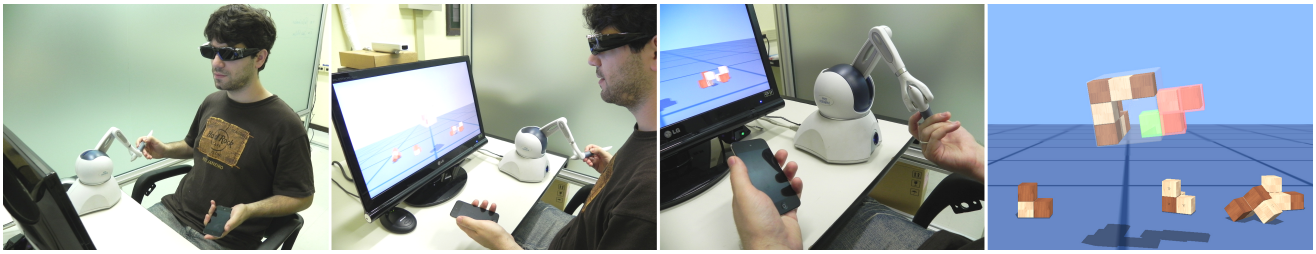

Figure 1: From left to right: the user wearing shutter-glasses and holding the interaction devices; the system overview including the monitor with the Wiimote on the top; a highlight of the iPod Touch and Phantom Omni devices used for interaction; a screenshot of the system.

Abstract

This paper presents a 3D user interface to solve a three-dimensional wooden blocks puzzle. Such interface aims at reproducing the real scenario of puzzle solving using involving devices and techniques for interaction and visualization which include a mobile device, haptics and enhanced stereo vision. The paper describes our interaction approach, the system implementation and user experiments.

Index Terms: H.5.2 User Interfaces [Input devices and strategies]: 3D Interaction—

1 Introduction

Simple three-dimensional tasks of the real world, as playing with wooden blocks, may become very hard when performed within a virtual environment with virtual blocks. While in the real world a person can benefit from the complete set of human sensory and motor skills to deal with the problem, a suitable human-computer interface must be developed to make the task even feasible in a VE.

This paper presents a 3D user interface to solve a three-dimensional wooden blocks puzzle. The puzzle problem is defined as a virtual environment showing a free area like a tabletop, where all puzzle pieces are equally selectable. The user is able to select any piece and move it to a working area where the final composition is assembled. The system identifies when the solution is achieved and shows the time for completion.

In the remaining of the paper we describe our interaction approach, the system implementation and user experiments performed with a magic cube puzzle.

2 System Overview

Three-dimensional puzzles have the particularity that part of the problem to be solved is hidden from the user view. Thus, we wanted to conceive a system which allowed great mobility for easily turning and inspecting all sides of the objects. Our system consists of an interaction and control model which is similar to the real world object manipulation of such 3D puzzles. It integrates bimanual manipulation, haptic feedback, stereoscopy and head tracking.

For more than a decade, bimanual manipulation has proven to be effective [2]. In our system it occurs with the user's primary hand being used to select, translate and rotate objects, while the user's secondary hand rotates a three-dimensional working area. The primary hand controls a 3D cursor with 6 dof (degrees of freedom). The cursor is used to pick, move and orient objects – virtual wooden blocks for instance. At the same time, the secondary hand holds a mobile device with 3 dof tracking capability. With this hand, the user controls the orientation – position is fixed – of a working area which has the shape of the final object of the puzzle. As the blocks are placed into this area, they become part of the solution and start to be controlled by the secondary hand, leaving the primary hand free to pick another block. This allows for quickly inspecting the status of the partial solution.

The primary hand is also stimulated by haptic information in the form of force feedback. Information as weight, collision and impingement are rendered to the user through a haptic device. Force feedback also aids in reinforcing the positioning rules of the environment. For example, collision forces will avoid that the user's hand proceed on a trajectory which would otherwise take a block to a portion of the space already occupied by another block in the working area or in the free area.

Besides selection and manipulation, another difficulty in 3D interaction is visualization. More specifically, depth information is difficult to obtain. To help the user in acquiring more accurate depth information, in our system we provided two additional features: stereoscopy and parallax effect [1]. While stereoscopy is widely known and understood , the parallax effect is less explored. Parallax occurs when, as the user moves their head to the sides, nearer objects appear to move faster than farther objects. A positive side effect of parallax is that very close objects can be viewed through a variety of different angles simply moving the user's head. We use parallax in our system by tracking the user's head and updating the virtual camera position accordingly.

*e-mail:hgdebarba@inf.ufrgs.br
†e-mail:jmfranz@inf.ufrgs.br
‡e-mail:vureus@inf.ufrgs.br
§e-mail:amaciel@inf.ufrgs.br
¶e-mail:nedel@inf.ufrgs.br

3 Implementation

The system has been implemented in C++ based on the Ogre3D library for graphics and using the following hardware: Phantom Omni; iPod Touch; stereo shutter glasses; 120Hz monitor; Wii remote and infra-red LEDs. Also, physics-simulation of rigid bodies has been implemented using the NVIDIA PhysX library [3].

The primary hand uses a Phanton Omni to interact with 6 dof input and 3 dof force output. The assynchronous HD API of the OpenHaptics library has been used to implement the communication with the Omni. Collision detection is calculated by PhysX between objects, and feedback force is calculated following the god object approach using a penalty force.

An Apple iPod Touch is the mobile device held by the secondary hand. This type of device is becoming ubiquitous an we believe in its potential as a 3D interaction device as it offers a number of integrated motion sensors. In our system we use the gyroscope and the accelerometers to provide three accurate rotational degrees of freedom. The working area is then rotated using this information in such a way that it mimes the iPod orientation. iPod to system communication is implemented through WiFi network using UDP.

Stereoscopy is obtained with an NVIDIA 3D vision kit of shutter glasses and driver, coupled with a 120Hz LCD monitor. The parallax depth clue is produced by tracking a couple of IR LEDs we placed at the sides of the shutter glasses. We use the IR sensor of a Wii remote for tracking the user head position. The complete system setup can be seen in Figure 1.

4 User tests

A user study was performed according to the following protocol. We considered two groups of subjects – novice and experts – that were firstly invited to fill a pre-test form that characterizes their profile. Then, they received a short introduction to the interface and the task to be accomplished. Novices also had a practice session of about 1 minute. After that, they started the test without any other guidance. Novices solved the virtual puzzle only once, while experts had to solve the same puzzle three times in a row. Finally, subjects had also solved a real wood puzzle (equal to the virtual one) and filled a post-test form rating satisfaction and fun.

Along the tests, we logged a whole set of dependent variables for each subject which we are not detailing here for lack of space. The ones we analyze in section 5 are the time form completion and completed or not. Fifteen subjects took part in the test, all of them right-handed, male and with a background in Computer Science (professors at the Department, undergraduate and graduate students, most of them on Computer Graphics), with a mean age of 30 years old (standard deviation of 11). Ten subjects were novices and five experts.

5 Results

Analyzing the data acquired during the tests of the novices, we notice that 5 subjects did not complete the puzzle using our system, and 3 out of these 5 did not complete the wooden puzzle neither. This indicates that their main dificulty is with puzzles and not 3D interfaces. Considering only the subjects that have completed the task, the minimum and maximum time spent was respectively 340 and 1,551 seconds. The mean time was 945s with a standard deviation of 457s (180 and 132s are the mean time and standard deviation to complete the wooden puzzle). We have also calculated the mean time for all 10 subjects. In this case, we added a penalty of 1,000s (two standard deviation) for the participants who did not accomplish the task. The mean time was then 1,344s with a standard deviation of 1,267s. Figure 2 shows a chart with all individual times.

Concerning the experts, all subjects completed the virtual puzzle three times, and only one did not complete the wooden puzzle. The minimum and maximum times spent were respectively 130 and

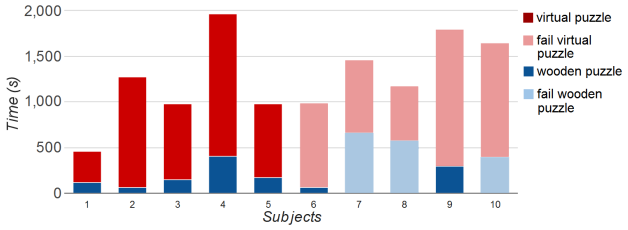

Figure 2: Completion time for novices.

2,074s. The mean time was 625s with a standard deviation of 442s (considering all of the 15 tries). In Figure 3 you can observe the learning curve of the 5 subjects involved in this user study.

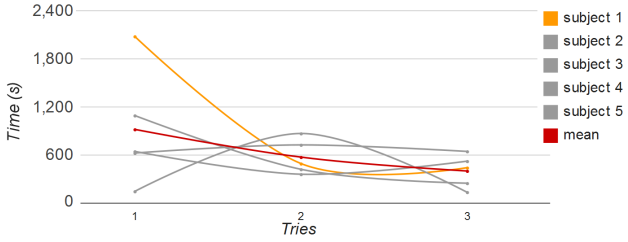

Figure 3: Completion time for experts. Subject 1, in yellow, did not complete the wooden puzzle, what could explain the low performance in his first try.

The subjective data captured with the post-test questionnaire indicate that 12 participants are satisfied or very satisfied with the experience. In their comments, subjects mentioned that, even if they are safistied, the experiment is very long and involves two different and difficult cognitive tasks: the use of a new two-hands 3D interface, and the solution of a complex puzzle.

6 Final Comments

We presented an involving and robust solution for complex manipulations of objects in a 3D virtual environment. Our solution proposes a user experience with two hands and the head movement in a natural way, and was well accepted by the most part of the users. As future work, we are planning further user tests with simpler objects to reduce the cognitive load imposed by the puzzle to a minimum and focusing only on the interaction issue. Also, a greater number of subjects should allow for statistical analysis of the data.

Acknowledgements

Thanks are due to the volunteers, CAPES and CNPq for grants PIBIC, 481762/2008-6, 309092/2008-6, 509160/2010-7, 483814/2010-5, 483947/2010-5 and 302679/2009-0.

References

[1] J. LaViola, A. Forsberg, J. Huffman, and A. Bragdon. Poster: Effects of head tracking and stereo on non-isomorphic 3d rotation. In *3D User Interfaces, 2008. 3DUI 2008. IEEE Symposium on*, pages 155 –156, 2008.

[2] A. Leganchuk, S. Zhai, and W. Buxton. Manual and cognitive benefits of two-handed input: an experimental study. *ACM Trans. Comput.-Hum. Interact.*, 5:326–359, December 1998.

[3] A. Maciel, T. Halic, Z. Lu, L. P. Nedel, and S. De. Using the physx engine for physics-based virtual surgery with force feedback. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 5:341–353, 2009.

LOP-cursor: Fast and Precise Interaction with Tiled Displays Using One Hand and Levels of Precision

Henrique Debarba* Luciana Nedel† Anderson Maciel‡

Instituto de Informatica (INF)
Universidade Federal do Rio Grande do Sul (UFRGS)

ABSTRACT

We present levels of precision (LOP) cursor, a metaphor for high precision pointing and simultaneous cursor controlling using commodity mobile devices. The LOP-cursor uses a two levels of precision representation that can be combined to access low and high resolution of input. It provides a constrained area of high resolution input and a broader area of lower input resolution, offering the possibility of working with a two legs cursor using only one hand. LOP-cursor is designed for interaction with large high resolution displays, e.g. display walls, and distributed screens/computers scenarios. This paper presents the design of the cursor, the implementation of a prototype, and user evaluation experiments showing that our method allows both, the acquisition of small targets, and fast interaction while using simultaneous cursors in a comfortable manner. Targets smaller than 0.3 cm can be selected by users at distances over 1.5 m from the screen with minimum effort.

Index Terms: H.5.2 [User Interfaces]: Input devices and strategies (e.g., mouse, touchscreen)—

1 INTRODUCTION

With the increasing availability of larger displays – both in size and resolution – desktop conventional interaction is no longer an efficient option in a variety of use cases [15, 20]. Such larger displays are giving rise to new working scenarios in which users are not sitting in front of the screen nor they have a table upon which to pose their mice. A widespread example of this situation are the multi-display screens in operation centers, public spaces and scientific facilities. While they offer great visualization possibilities, efficient interaction with such displays is still a major challenge.

The literature is rich in direct pointing based techniques, many of which do not necessarily rely on conventional mouse-and-keyboard input (next section comments on a few examples). Although those usually offer fast and intuitive input for pointing tasks while interacting with large and distant displays, precision of pointing is limited due to hand jittering. Moreover, most interaction approaches used today in this context do not offer the same resolution for interaction input as the resolution offered by the displays. This problem is related to input device limitations but also to the cursor metaphor in use.

On the other hand, small multi-use devices as cell phones and, more generally, smartphones with sophisticated high-resolution small screens are more ubiquitous than specific devices. Those mobile devices are complete devices, with a number of sensors (GPS, accelerometers, gyroscope, magnetometer, multi-touch screens, cameras, etc.) and networking capabilities (3G, wi-fi, bluetooth).

*e-mail: hgdebarba@inf.ufrgs.br
†e-mail:nedel@inf.ufrgs.br
‡e-mail:amaciel@inf.ufrgs.br

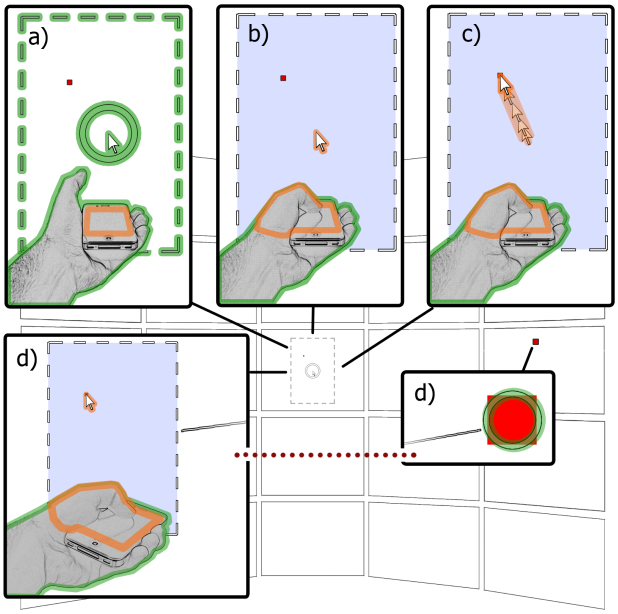

Figure 1: LOP-cursor usage to select and move objects on a tiled display: (a) the user hovers the target neighborhoods with an arrow pointer performing a 3D gesture with a mobile device in the hand; (b) touching the device touchscreen locks the position of a rectangular control canvas; (c) fine tune and precise selecting the target object with the arrow in the control canvas is achieved moving the thumb on touchscreen; (d) holding the control canvas locked, the user can move the arrow cursor by touchscreen on the mobile device while moving their hand to point at a second place with a secondary cursor (ring cursor) to define a destination target to the selected object. Green marks indicate actions made using gestures, while orange marks point out actions done on the touchscreen. The image background illustrate the LOP-cursor graphic representation on a tiled display.

Different combinations of sensor sets enable a number of positioning and orientation possibilities for interaction with external facilities by means of personal mobile devices. While literature covers quite well the use of camera and touch based mobile device interaction techniques [2, 8, 4, 5, 9, 17, 13], sensors of movement and their possible combinations are less explored [19, 10, 7].

In this paper we introduce the LOP-cursor, a multilevel and two-legged cursor for interaction with large high-resolution displays based on a combination of 3D and 2D interaction metaphors. By freely walking in front of the display area, the user changes their position and distance to the screen, emulating pan and scale of the visualization in relation to himself.

LOP is a cursor metaphor in which the accurate cursor positioning is based on two levels of precision. The two levels are necessary

to address the lack of precision observed in many single level direct pointing techniques. With LOP-cursor, the user first points the device towards the target on the screen using the laser pointing 3D metaphor. If higher precision is needed, the user can fine tune the cursor position by sliding a finger on the device's touchscreen, at this point using a 2D interaction technique. Figure 1(a to c) illustrates a walkthrough on positioning the cursor as a combination of the two levels.

The fine tuning by touchscreen freezes a control canvas through all the duration of the touch, but does not stop the direct 3D pointing feature, which results in two simultaneous cursors under the control of the user. This design allows two simultaneous pointing locations (the two legs of the cursor) to quickly perform composed tasks as, for instance, selection and move. Figure 1(a, b and d) presents the walkthrough on the simultaneous use of both legs of the cursor. Ninja Cursors [11] is the only other approach that allows simultaneous control over multiple cursors. However, it only replicates the mouse movement across cursors aiming to reduce the index of difficulty of selection tasks, while we aim at allowing simultaneous pointing at two specific places at the same time.

The contributions of this paper are twofold:

- An approach for high resolution interaction with large high resolution displays based on two levels of precision using 3D and 2D interaction metaphors
- A two-legged cursor concept defining two simultaneous positions using only one hand

The remaining of the paper is organized as follows. Section 2 summarizes related works on the use of mobile devices to interact with large displays as well as input techniques that consider levels of precision. Section 3 presents the decisions and details of the LOP-cursor design. Section 4 describes the current state of hardware technology and software implementation. In Section 5 we present the evaluation of the LOP-cursor comparing it against other selection techniques. Finally, in Section 6 we present and discuss the results achieved and in Section 7 we highlight our findings and suggest future developments.

2 Related Work

2.1 Mobile device control of large displays

Ballagas assumed that the smartphone is the first really pervasive computational device [1], being an essential part of contemporaneous life and an always on pocket device. Therefore, its use as an input/output device is quite obvious. Most relevant works using mobile devices to control external large displays are based on optical analysis, usually relying on optical flow [2, 8], pattern recognition [2, 4, 5, 8, 9, 17], and external optical tracking of the device [16]. Other works also use mobile devices sensors on less specific tasks [19, 10, 7].

Ballagas et al. are precursors on mobile phone interaction with large displays, introducing the *Sweep* and *Point and Shoot* techniques [2]. Sweep uses the optical flow to move a cursor on the screen, with a central button as a clutching activator. Point and Shoot uses a quick blink of bi-dimensional tags on the controlling screen synchronized with the camera capture order, allowing reconstruction of camera pointing center. Jiang et al. [9] use the two last on screen cursor positions to define a coordinate system, allowing the cell phone to calculate the new position the cursor should assume.

Pears and Olivier introduced a technique for registration of mobile phone and external displays using four square markers [17]. Registering allows direct mapping of every pixel at the large display on the mobile phone screen, and thus, direct control is provided. Boring et al. developed the Touch Projector, extending interaction with video to mobile devices [4]. Touch Projector uses a polygon comparing algorithm to identify the screen where the user is aiming, thus allowing the control over multiple and spread displays. Touch Projector also suggested improvements to the video interaction concept, based on mobile device specific constraints and needs. Later, Boring et al. extended Touch Projector to interact with media facades [5]. LightSense [16] uses a mobile phone with a back LED over a semitransparent table and track its two dimensional position using an external optical tracking system. The LED diffusion over the table is also used to distinguish across ten levels of distance between table and device.

None of the presented techniques deal with the precision issue at the level we are proposing. Presented pointing techniques are all based on camera tracking and/or ray casting with low cost approaches, thus resulting in less precision of pointing. The motion sensors embedded in the newer smartphones allow for a number of gesture-based input modalities. Specific pointing devices (Wiimote, gyro mice...) make extensive use of these sensors, but generic smartphones have not been widely explored for pointing tasks. WYSIWYF [19] uses accelerometer readings and prongs contact with a smart board to position and orient a volume cutting plane. Touching the smart board with the mobile device creates a virtual plane that starts from the contact points between the mobile device and the smart board which have the same orientation of the device. Katzakis and Hori [10] used mobile device accelerometers and digital compass to orient 3D objects applying a direct mapping of orientation across them. On a comparative test, the mobile device approach performed better than mouse and touchpen input. Other works used movement sensors for tasks like distinguishing devices on a multi-touch display wall using tilt correlation [7].

2.2 Levels of precision input

Switching between absolute and relative modes of input, in some sense, allows the use of more than one level of precision for interaction. Absolute pointing is used for quick traveling long distances, while relative movements of the cursor are used for fine tuning, thus also providing fast and high precision pointing.

Vogel and Balakrishnan [20] explored natural input using the naked hand. The proposed technique allows switching across absolute and relative input performing two different hand poses. The user can point to a region at the large screen performing an absolute hand pose, and then, change the hand pose to switch to the relative mode. Similar in concept, Forlines et al. also implemented an analogous for pen interaction with large and high resolution touch surfaces [6]. Users could switch from absolute to relative input mode to achieve targets out of their arms reach area. Both techniques rely on high cost aparatus and a prepared interaction environment (VICON tracking system).

Nancel et al. [14] also proposed a technique using the VICON system. They combine a VICON-based ray casting mode with a precise relative pointing sliding the finger on an Apple iPod Touch device touchscreen. The touch surface is divided in two areas: an upper zone for tracking, and a lower zone for clicking. Touching the upper zone switches to precise mode. Authors claim it is possible to select objects with 4 millimeters standing 2 meters away from the display, which is comparable to some results we show in this paper using only smartphone built-in sensors.

The ARC-Pad [13] implements an absolute plus relative cursor controller using a mobile phone touchscreen. The movement of the finger while holding the touch triggers a relative movement, identical to an ordinary touchpad, but a quick tap on the screen triggers a jump of the cursor to the location defined by an absolute mapping relation with the external display. We implemented this approach and further compared with LOP-cursor at the Section 5.1.

3 DESIGN

3.1 The conception of the LOP-cursor

LOP-cursor was conceived for high-precision pointing on high-resolution displays. The technique is based on two levels of precision. In the first level, the user points the device towards the target on the screen, which results in moving a rectangular control canvas containing the cursor arrow to that location (see Figure 1a). If the user succeed on selecting the desired target with the arrow, the task is achieved. Otherwise, a fine tuning can be done by sliding a finger on the device touchscreen, moving the cursor arrow inside the control canvas (see Figures 1b and 1c). Different from most 2 levels of input techniques, LOP-cursor uses absolute mapping in both levels, ray casting for the first level, and a rectangle representing the physical device touchscreen at the large display. During the fine tuning, the position of the control canvas is locked, but the coarse pointing using the ray casting metaphor remains active (see Figure 1d). This feature – the simultaneous use of the two legs of the cursor – inspired the use of LOP-cursor to perform composed tasks, as select-and-move, for instance.

3.2 Cursor States

The three-dimensional interaction space of our cursor metaphor combined with its two legs allows for an increased pointing capability. To ensure user control while keeping flexibility in this context, we defined three states of interaction for the cursor: (1) free-pointing state; (2) hold-control-canvas + free-pointing state; (3) pin-control-canvas + free-pointing state. The states are detailed below and depicted in Figure 2.

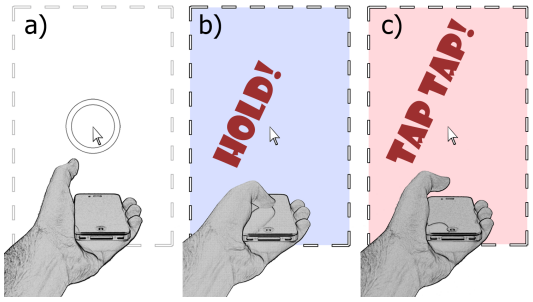

Figure 2: LOP-cursor states: (a) *free-pointing*; (b) holding a touch leads to *hold-control-canvas + free-pointing*, (c) a double tap activate and deactivate *pin-control-canvas + free-pointing*.

3.2.1 Free-pointing

This is a one-legged cursor mode (Figure 2a). While on free-pointing state, cursor positioning is allowed on a direct-pointing only manner. In this state, our metaphor is very similar and, in average, not worse than a ray casting based technique, allowing large targets to be easily acquired. Small differences in performance can appear depending on the way position and direction are gathered, and difficulties related to small targets remain an issue.

3.2.2 Hold-control-canvas + free-pointing

In this state, the two legs of the cursor are active and can be defined (Figure 2b). The cursor enters this state when a touch on the mobile device touchscreen is hold. While in hold, the control canvas keeps the initial touch pointing position as its anchor, so that the user can fine tune pointer positioning by sliding the finger on the touchscreen. Unlike most touchpads, here touch is mapped with an absolute relation between the devices touchscreen and the control canvas.

While the leg-1 of the cursor is finely controlled inside the very stable control canvas, wrist movements can still displace and rotate the device, giving rise to the leg-2 of the cursor. Leg-2 is then a cursor pointer controlled by free-pointing that can be used as a lower precision secondary and simultaneous cursor. Interaction scenarios for leg-2 are described later in this paper. Touch take-off results in the release of the control canvas. On releasing, the control canvas silhouette assumes the current position of leg-2, and the two-legged cursor reunifies, returning to the free-pointing only state.

3.2.3 Pin-control-canvas + free-pointing

Starting from a free-pointing state, a double tap on the device touchscreen pins down the control canvas (Figure 2c). Pinning position is defined by the pointing position at the start of the first tap, which minimizes the occurrence of pinning on an undesired position due to device displacement while tapping.

While pinned, the touch take-off does not trigger a state change. The state is maintained until the user performs another double tap. This allows for a number of touchscreen actions to be performed. For example, a single tap can be used for selection. Meanwhile, sliding a finger is used to fine pointing inside the control-canvas analogously to the hold-control-canvas state above. This state allows more stable use of simultaneous cursors.

3.3 Graphic representation

Four complementary shapes are used to represent the LOP-cursor which are combined to maximize usability. They are described here in association with the terms already used above to introduce the cursor states.

3.3.1 Arrow: the leg-1 pointer

To benefit from the high precision positioning capability of LOP-cursor, a traditional point selector over an area selector was preferred for our main cursor design. We used the well known arrow shape to represent this cursor, which minimizes ambiguities. We believe that using the arrow to highlight the leg-1, which is the ultimate pointer in LOP-cursor, helps users to immediately differentiate it from the other shapes we used.

The arrow opacity depends on the size of the nearest target to the cursor, the cursor size itself, and the distance between the target and the arrow center. Using adaptive opacity avoids the occlusion of very small targets by the arrow and allows users to keep track of the target whenever it is required. Figure 3 illustrates the arrow and its opacity function.

3.3.2 Rectangle: the control canvas

A rectangle is used as the representation of the mobile device touchscreen on the display. It defines a control canvas within which the arrow is. An absolute mapping is used between the device's rectangular touchscreen and the control canvas, e.g. a touch at a location near a corner of the touchscreen results in placing the arrow at the same corner of the control canvas. This absolute approach makes the rectangle a natural choice for the virtual representation of the touchscreen. More important, this approach allows for the control canvas representation to contain an input resolution equivalent to that of the device's touchscreen sensing resolution (480 x 320 in our implementation). The rectangle has also the same aspect ratio of the touchscreen, making the correspondence explicit to the user.

To allow user control over the leg1 cursor precision, *pinch* and *stretch* gestures resize the control canvas proportionally to its current size. In our implementation, the default input scale is 1:2, where each pixel on the touchscreen is represented by a 2x2 area of pixels at the large screen. For instance, to achieve higher precision, the user can resize the control canvas to a 10:1 precision, and thus, a 10x10 area of pixels on the touchscreen would be mapped to only 1 pixel of the large screen.

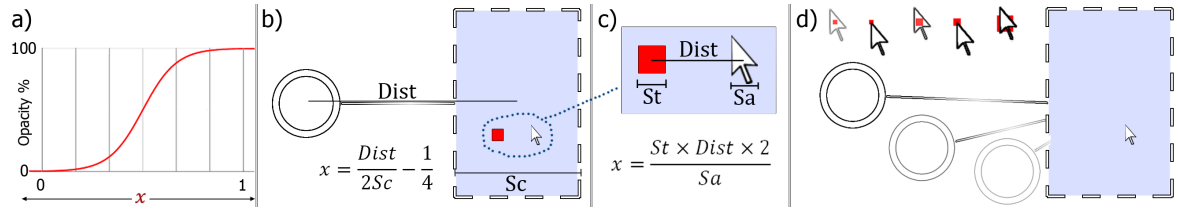

Figure 3: A logistic function maps the arrow and ring opacity according to the x variable value (a); x for the ring depends on distance between the center of the ring and the center of the rectangle $Dist$, and the width of the rectangle Sc (b); x for the arrow depends on distance between center of the target and center of the arrow $Dist$, and width of arrow Sa and target St (c); arrow and ring opacity sample results for a variety of target sizes and positions, and rectangle and ring distances (d).

3.3.3 Ring: the leg-2 pointer

We chose a relatively large ring shape instead of an arrow for leg-2 because it is subject to jittering and precision cannot be guaranteed.

Leg-2 indicates the physical pointing direction of the device. It always follows the device's orientation, enabling the user to keep track of where they are pointing to. This is particularly relevant when the control canvas is in hold or pinned and the two legs are separate. In this occasions, when the focus of the user is on leg-1, involuntary hand movements eventually take the ring to locations far from the control canvas and the leg-1 arrow. After release, the control canvas jumps back to follow the device's orientation and consequently the location of the leg-2 ring pointer. Experiments show that while this jump does not seem to upset the user, it can cause disorientation when the ring pointer is not displayed.

To avoid distraction caused by continuous movement of the ring near the arrow, we used an adaptive opacity factor. The ring opacity is proportional to its distance from, and size of, the control canvas. The nearer the ring is from the control area, more transparent it is. Figure 3 illustrates the concepts above.

3.3.4 Line: a bridge between leg-1 and leg-2

A line is used for connecting the control canvas (rectangle) and the leg-2 (ring). This aids the user to keep track of the relative position of legs 1 and 2 and to warn where they are currently pointing to. The line is specially useful when the two legs are separated by a large distance. For example, while simultaneously working with the two cursor legs, the user will need to quickly switch attention between them. The line guides searching direction, while the color intensity guides on the distance between legs.

3.4 Transitions between states

While in free-pointing state, both the arrow and ring pointers follow the mobile device pointing direction. The arrow is kept at the center of the ring, and a semi-transparent rectangle with dashed borders is shown to keep the user aware of the size of the control canvas. As the control canvas and the ring cursor are superposed and move together, the line is not visible in this mode.

When switching from free-pointing to hold-control-canvas + free-pointing, or from free-pointing to pin-control-canvas + free-pointing, a transition effect is played to guide user attention across states. The effect renders the rectangle with more opacity, and ring transparency is controlled as described in Figure 3, giving a hint of switching modes. The inverse effect is played when returning to the free-pointing-only state. To inform the user about the current state, *holding* fills the rectangle with a blue transparent tone, and *pinning* fills the rectangle with a red transparent tone.

In preliminary tests, due to absolute pointing within the control canvas, the arrow cursor often jumped when entering in hold-control-canvas + free-pointing state. This caused a discontinuity that led the users to report some disorientation. To overcome this issue, in the final design of the LOP-cursor the arrow is kept in place and the rectangle (canvas) is placed accordingly to compensate the distance of the touch from the center. Although this causes a small motion discontinuity for the canvas (it is small because the users tend to touch at the center of the device touchscreen), it was preferred than a discontinuity of the arrow as the user focus is on the arrow.

4 Prototype Implementation

We implemented the LOP-cursor in the context of an experimental interface. Our prototype consists of interaction challenges that cover many of the interaction needs when working with large high resolution displays. The challenges are simple selection, drag and classification tasks with low cognitive load. Real world applications, which can also involve 3D environments, are not addressed here.

4.1 Hardware and Software

Input and output hardware in our implementation consist respectively of a smarphone and a tiled display wall.

Our display wall is a 16 LCD monitors tiled-display, disposed on a 4 x 4 matrix. Each monitor has 1,680 x 1,050 pixels and 22 inches diagonal. The total pixel count is 6,720 x 4,200 = 28,224,000 pixels (≈28 megapixels). The display wall is controlled by four PC Core2Quad, with two NVIDIA GTX 285 each. See Figure 4 for an overview photograph of the prototype in operation.

The smartphone used in our main implementation is an Apple iPhone 4. An iPod Touch (fourth generation) was also used for preliminary and comparative testing. As the smartphones used do not present a general purpose embedded back button, we adapted a Microsoft Wireless Mobile 3500 Mouse, which offered a reliable wireless communication.

4.2 Orientation acquisition

In our prototype, we adapted the strategy proposed by Madgwick [12] which combines gyroscope, accelerometer and digital compass information to gather a more robust orientation. This orientation acquisition method relies on the *gyroscope* to provide instantaneous orientation changes (providing 3 axes angular rate of change), and gradual adjustments using two distinct vectors related to absolute frames of reference, the *accelerometer* and the *magnetometer* (respectively related to gravity and magnetic north pole). In practice, we correct gyroscope cumulative rotational error in two axis using the accelerometer (roll and pitch) and in a third axis using the magnetometer (yaw).

Magnetometer readings are less precise and more error prone than *accelerometer* readings, suffering from magnetic interference from near metal structures. Raw readings from the iPhone oscillate within the range of 10% in our environment. Although it does not seem too much, it is crucial to understand that, as a ray is casted using device orientation, any variation is greatly magnified. Because of that, two additional steps were implemented to allow long

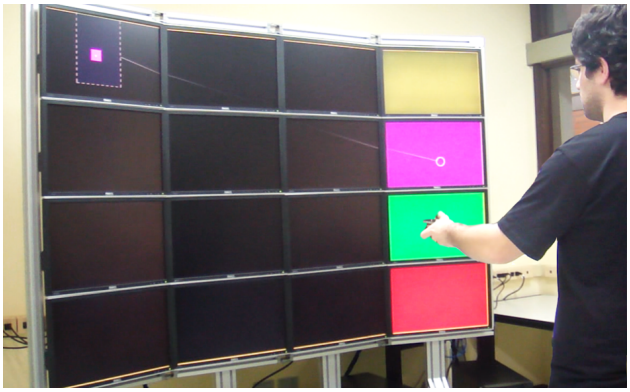

Figure 4: Prototype overview. This photograph depicts a user interacting with the two legs of LOP-cursor. Notice that while leg-1 selects a square on the left, the leg-2 ring indicates the location on the right to where the square will be moved.

term use of the *smartphone* as a pointing device. In the first step, to force the *magnetometer* to only adjust yaw drift error, the tridimensional vector provided is projected into a plane orthogonal to the accelerometer vector. In the second, to ignore high frequency *magnetometer* reading variations, instead of using a filter – which would result in poorer interaction due to delay – we are using a redundancy controller based on the gyroscope updates. More specifically, in this step we ignore small variations from the magnetometer readings whenever the gyroscope readings cannot confirm that a rotation actually occurred around that axis.

4.3 Position calibration

As smartphones and other *every-pocket* mobile devices are not equipped with position tracking, we used only orientation to define pointing and assumed a constant position of the user while testing our prototype. To define this position and reconstruct rotational zero position of the device, our system uses a calibration step. Notice that this step is not necessary when position tracking is used.

For magnetometer calibration, the device must be placed face up pointing orthogonal to the plane defined by the display. A calibration command registers an orientation offset. Next, the calculation of approximate user position (P_3) is achieved registering two orientations of the device. The user is asked to aim the device at a blue point P_1 at one corner of the screen, and then at a red point P_2 at the opposite corner. This calibration method assumes that the real distance between blue and red dots is known, and that the display is perpendicular to the world Z axis.

The two registered orientations are used to retrieve two normalized vectors $\vec{v}_1$ and $\vec{v}_2$. A third vector $\vec{v}_3$ is calculated as $\vec{v}_3 = P_2 - P_1$. The angles α_1 between $\vec{v}_1$ and $\vec{v}_2$, α_2 between $-\vec{v}_1$ and $\hat{v}_3$, and α_3 between $-\vec{v}_2$ and $-\hat{v}_3$ are computed by dot product. The length of the segments given by $\vec{v}_1$ and $\vec{v}_2$, respectively, are calculated as

$$l_1 = \frac{|\vec{v}_3| sin(\alpha_2)}{sin(\alpha_1)}, l_2 = \frac{|\vec{v}_3| sin(\alpha_3)}{sin(\alpha_1)}. \quad (1)$$

Then, we estimated device positions $P_3' = P_1 + l_1(\vec{v}_1)^{-1}$ and $P_3'' = P_2 + l_2(\vec{v}_2)^{-1}$. A mid point between P_3' and P_3'' is used as the final P_3.

Limitations of this calibration approach is that significant changes in the position of the user will result in the need for a new calibration, or at least some cursor offset control. Also, the rotational *zero* will depend on which of the joints (shoulder, elbow, wrist) the user performs the rotations.

5 Evaluation

We conducted two sets of user tests, both using the prototype presented in Section 4: a comparative evaluation, and a deeper exploration of the LOP-cursor capabilities.In all tests, target start and goal positions were constrained to never intercept a monitor bezel since, as stated by [3], this can be detrimental to some user interaction aspects.

5.1 Comparative evaluation

5.1.1 Design

To validate the LOP-cursor technique, we conducted a comparative evaluation on a selection task. LOP-cursor was compared to an *ARC-pad* [13] implementation, and a Ray Casting using only device orientation (*ORayCasting*), a technique equivalent to limiting LOP-cursor to the use of the lower precision level of pointing. To remove subjective bias on choosing the level of precision, the LOP-cursor implementation used for evaluation constrained users to always use the two levels of precision for selections. We call this implementation (*CLOP-cursor*). CLOP is actually the same as the original LOP but free-pointing is not effective for selection triggering. Thus, with the CLOP-cursor, users had to always perform steps *a* through *c* as described in Figure 1.

Independent variables are: *Technique*: ARC-Pad, LOP-Cursor and ORayCasting. Target *Size*: 0.5cm, 1cm, 2cm, 4cm. Target *Distance*: 25cm, 50cm, 100cm. Dependent variables are: *Time* and *Error rate*. We used a *within-subject* design. Technique exposure was counter-balanced, while size and distance of target were randomly presented. Pointing to a target and triggering a selection counted as a *Trial*. There were a total of 10 *Trials* for each independent variable combination. Each technique evaluation was divided in 5 *Blocks*, where a *Block = Sizes x Distances x 2 Trials*, giving a total of 24 *Trials* per *Block*. The first 2 *Blocks* were used for practicing, and thus are not considered on further analysis. Participants were allowed to take a non-mandatory break between blocks.

After the comparative evaluation, another selection test where the complete *LOP-cursor* is used was taken. In this case, subjects were not constrained to always use the higher level of precision. They could decide when and if they want to use two levels of precision, or only one. This was intended to evaluate for which sizes of target users preferred to use only the ray casting level of precision, or both of them, thus allowing us to infer if participants subjective preference match to the previously applied comparative evaluation best times per size. Six target sizes were used: 0.5cm, 1cm, 2cm, 4cm, 8cm, 16cm. Each test had 5 *Blocks* of 18 *Trials*, 3 trials per size condition. First 2 *Blocks* were used as training, thus being discarded for analysis.

Active targets were drawn in red on a black background. To avoid bias from visual search, the following target was shown in a dark grey tone. There was an additional starting target at each *Block*. During all evaluation, the user was positioned centered to the tiled-display, at a constant distance of 150cm. Users who completed all the evaluation were then asked for their preferred interaction technique (CLOP-cursor, LOP-cursor, ARC-pad or ORayCasting), and allowed to leave general comments and observations about the evaluation. Users were asked to favor precision over speed.

5.1.2 Implementation details

Comparative tests were taken using an *iPod Touch 4th generation*. Orientation without the magnetometer recalibration may drift on *Yaw* over time (see Section 4.2), but the ray casting was too imprecise when using yaw corrections because of the high variability of the magnetometer readings from the iPod. As this would put the ORayCasting technique in disadvantage relating the other techniques tested we disabled yaw corrections. To limit the effect of yaw drifting we preferred to arrange short blocks of trials (24 trials) and frequently correct any drift between blocks.

The ARC-pad paper [13] does not make clear how a selection is triggered, and seems to suggest its use over a desk. Our test design, instead, proposes the users to stand in front of the screen, and thus they had to hold the device with both hands to maintain consistent aspect ratio orientation across the tiled display and the mobile device. Selection was implemented using a *Tap* gesture from the secondary hand while holding arrow position with the primary hand. See Section 2.2 for a brief description of the ARC-pad technique.

Cursor position was filtered using a *dynamic low-pass filter*, interpolating between cutoffs of 0.2Hz and 5Hz, with 60Hz sample rate. Cutoff is defined according to cursor speed: when $<$ 1cm/sec, lower cutoff is used (0.2Hz); when $>$ 50cm/sec, higher cutoff is used (5Hz). For any speed between these, a linearly interpolated cutoff value is used.

5.2 In depth LOP-cursor evaluation

5.2.1 Design

We conducted this deeper evaluation to assess the limits of our technique in terms of precision, and to initiate the study on simultaneous cursors. Experiments consisted on completing three different tasks using LOP-cursor:

Task 1. Pointing and selecting targets. A trial is complete when the target is hit.

Task 2. Pointing, selecting, dragging and docking targets. Users should point and select the target in the same way they do in Task 1: drag it over a grey object with the same shape and size of the target located at a fixed distance of 100 cm, and dock the target over the grey object the most precisely as possible. A trial is complete when the target is released (docked).

Task 3. Using the two legs of the LOP-cursor to select a target and put it into a predefined stock area on the right side of the display (as depicted in Figure 4). The target should be selected with the leg-1 using the two levels of precision allowed by the selection technique, and moved to its respective stock area (with the same color of the target) using leg-2. The selection results on a transfer between the legs. The trial is complete when the user successfully trigger a selection while simultaneously aiming with both cursors, on both target and goal area.

The presented tasks add in complexity, and thus, they were always applied on the same order, task 1 through 3. We used four different *Sizes* of targets (0.3, 0.8, 2 and 4 cm). The initial position of targets were randomly defined. For *Task 3*, targets appeared only at the left-half of the tiled display.

We also found that the 0.3 cm target could introduce delay due to visual search. Then a green target was shown between each trial for tasks 1 and 2. Green target and the trial target(s) appeared at the same time. Having found the trial target(s), the user selected the green target to start the trial.

The procedure consisted on two practice *Blocks* of two *Trials* for each *Size* (total of *eight trials per training block*) and one evaluation *Block* with six targets per *Size* (total of *24 trials*) for each *Task*. The first practice block was used to explain task goal, and to demonstrate the LOP-cursor capabilities that would be used during the task. Users were asked to complete trials as quick as possible, but without sacrificing precision over time. On the second practice block users interacted by themselves, with minor hints of the test conductor when needed. During the evaluation block, subjects were left by themselves, and no help was given. After the evaluation, users filled a System Usability Scale (SUS) questionnaire.

An iPhone 4 disposing of a magnetometerwas used on these tests. This allowed orientation to be gathered as described in Section 4.2, allowing longer blocks to be taken. As this evaluation aimed to assess the limits of our technique, users were constrained to always use the two levels of the LOP-cursor (the so called CLOP-cursor).

6 Results and Discussion

6.1 Comparative evaluation

Eleven participants took place on this evaluation, (mean age 22, 2 women and 2 left handed) all of them Computer Science students (10 undergraduate, 1 master student). Most of them had previous experience with pointing devices (mostly Wii gaming), all of them had at least some experience with a mobile device touchscreen, only two had significant experience with very large displays.

From the comparative test we had 2,376 total valid *Trials*. A trial was considered a false positive (accidental selection triggering) when selection occurred at any of the following cases: *20cm* or farther from the target; less then *200ms* after the previous trial was completed. CLOP-cursor, ARC-pad and ORayCastig presented 2, 6 and 9 false positives respectively. These *Trials* were overlooked on further analysis.

All subjects completed the comparative evaluation, but three of them could not complete the last evaluation (using the non-constrained LOP-cursor). This was due to the long time taken by the comparative evaluation (from 30 to 40 minutes to complete), resulting on schedule and fatigue issues for some subjects. This test had a total of 432 trials, 2 of which false positives.

General mean time for a selection with CLOP-cursor, ORayCasting and ARC-pad were respectively: 2.55, 2.46 and 3.05 *seconds*. One-way ANOVA showed that ARC-pad was significantly slower than CLOP-cursor ($F(1.1574)=76.58$, $p<0.0001$) and ORayCasting ($F(1.1567)=84.81$, $p<0.0001$). See Figure 5 for detailed mean time and standard deviation for each target size.

Error rate for ORayCasting is significantly higher than CLOP-cursor and ARC-pad ($F(1.1571)=109.08$, $p<0.0001$ and $F(1.1567)=120.6$, $p<0.0001$). Error difference between CLOP-cursor and ARC-pad is not significant ($F(1.1574)=0.48$, $p<0.49$), even considering only the 1cm targets ($F(1.391)=1.93$, $p<0.17$). Error rates are presented on Figure 6.

We also found significant learning effect on mean time across blocks for CLOP-cursor ($F(2.787)=4$, $p<0.019$) and ARC-pad ($F(2.783)=3.04$, $p<0.0484$). Although a reduction of error rate did occurred for CLOP-cursor (Block1=12, Block2=11 and Block3=5) there was no significance ($F(2.787)=1.61$, $p<0.2$).

On the LOP-cursor trials (post comparative evaluation) error rate followed a tendency between CLOP-cursor and ORayCasting (see Figure 6). This unconstrained LOP-cursor presented lower error rates for small sizes than the CLOP-cursor, even though they rely on the same fine tuning technique. As this technique was always presented to users as the last one, we believe learning effect did affect error rate, but there was not enough data to statistically prove such variation. Detailed error rate for each level showed consistency with CLOP-cursor and ORayCasting previously applied comparative test for most target sizes (Figure 7).

Subjective user preference between 1 or 2 levels of precision according to target size is presented in Figure 8. For target sizes of 0.5 and 1 cm users always used 2 levels.

After the evaluation, subjects who participated on all procedures were asked for their preferred interaction technique. The LOP-cursor implementation allowing 2 levels of interaction was preferred by seven users, while ARC-pad was preferred by only one subject.

6.2 In depth LOP-cursor evaluation

Eleven undergraduate students in Computer Science participated on this experiment (mean age of 23, 2 females, all right handed). Most subjects had some experience with pointing devices, all of them had at least some experience with mobile device touch screens. Each test took from 25 to 40 minutes to be concluded.

All users were able to complete the tasks, i.e., they were all able to select every 0.3 cm target at the distance of 150 cm with our technique. The chart on Figure 9 shows the mean time for completion of

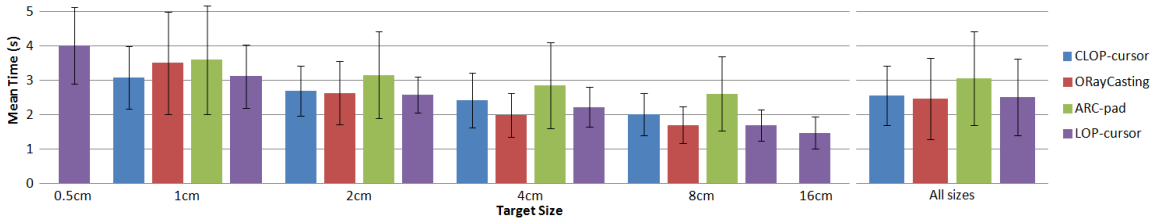

Figure 5: Mean time spent to select targets of different sizes using CLOP-cursor, ORayCasting, ARC-pad and LOP-cursor, and their respective standard deviations.

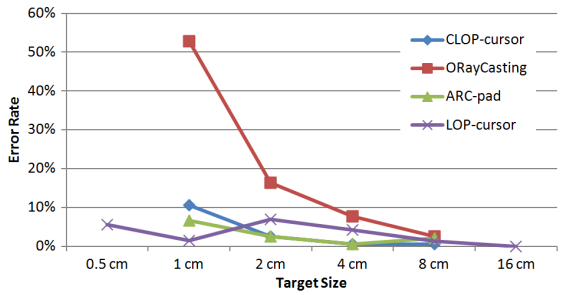

Figure 6: Error rate to select targets of different sizes using CLOP-cursor, ORayCasting, ARC-pad and LOP-cursor.

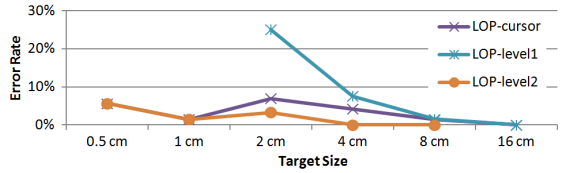

Figure 7: Error rate to select targets of different sizes using LOP-cursor, and individual error rate for each level of precision.

the 3 tasks. Notice that although the larger targets allow for faster task completion, the time difference is always inside the standard deviation, regardless of the huge difference (more than one degree of magnitude) in objects size.

There were 5 false positives (incidental docking) on Task 2, which were discarded for further results. Mean docking distance error is 0.17 cm, being 0.158 cm, 0.162 cm, 0.176 cm and 0.188 cm for target sizes with 0.3 cm, 0.8 cm, 1 cm, and 2 cm, respectively. Docking distance error difference per size showed not to be significant ($F(3.255) = 1.38$, $p < 0.254$). The similarity observed on the mean errors obtained with all target sizes tested indicates that the LOP-cursor is a technique suitable for tasks involving different sizes of objects, with more or less the same accuracy, which is a positive result. The most frequent user complain was related to prototype physical ergonomics (device size and back button position). We noticed that some users slightly moved touch position when triggering a selection command. We believe the ergonomic issue was the main cause of this effect. This was specifically detrimental when selecting the 0.3 cm targets, where subjects needed more than one selection triggering to complete the trial: 42% and 39% for Task 1 and Task 3, respectively. CLOP-cursor scored 77 on the SUS questionnaire, with standard deviation of 12.8.

Task 3 results showed that users could successfuly point at two simultaneous positions. The other existent multiple cursor technique, Ninja Cursors [11], points to only one location at a time. It is difficult to compare performance results between the LOP-cursor and the Ninja Cursors as their evaluation is based on a 2 screens desktop environment, while ours pursuit interaction with much larger and higher resolution displays, away from the desktop and with potential to interaction in 3D environments. Nevertheless we could notice that while the performance of the LOP-cursor is independent of screen target density, the performance of the Ninja Cursors is significantly affected by both the number of cursors and the target density.

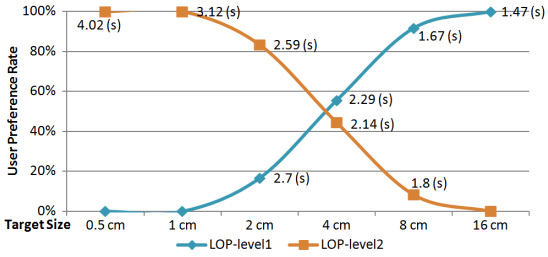

Figure 8: Users preferred level of precision for a variety of target sizes when using LOP-cursor, and their respective trials mean time.

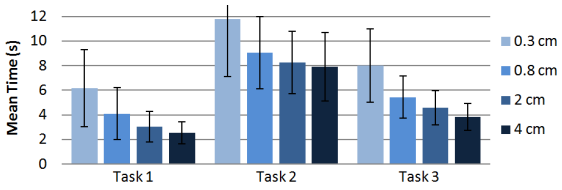

Figure 9: Mean time for completion of each trial of each task.

7 Conclusions and Future Work

While high-resolution displays are a widely explored subject, in this paper we addressed the issues involving high-resolution *interaction* with such displays. As such displays are used away from the desktop, 3D interaction techniques can be helpful even when the application deals with 2D data. We introduced the concept of the LOP-cursor, a two levels pointing metaphor greatly adapted to interact with large high-resolution displays.

We have shown through user studies that ordinary smartphones implementing the LOP-cursor can be a valuable device for interaction with such displays. Our tests demonstrate that using direct pointing to first define the raw location of a cursor, and then a mechanism for fine tuning the raw location to a very precise point, enables the users to quickly and precisely select and drag objects.

The LOP-cursor also introduces a two-legged cursor modality controlled with only one hand. This is a major breakthrough as many everyday user actions with a computer involve quickly defin-

ing two locations on the screen to complete the operation (copy files, arrange photographs, etc.).

Concerning the devices, our pointing method requires both the orientation and position of the mobile device. Orientation is retrieved using device built-in movement sensors. In our prototype, position is assumed or given by a simple calibration step. A tracking system for position would perfectly fit and enhance our concept, but as tracking requires more complex equipment, we preferred to keep the solution available for everyone who owns a smartphone.

We continue working on LOP-cursor, investigating the use of simultaneous cursors on real life tasks and alternative screen configurations. Further investigations will be held in order to determine to which level users control two cursors simultaneously or one at a time. In addition, we want to access two-legged cursor cognitive load and determine which two-location tasks may benefit from this metaphor. We plan to apply our concept on distributed computing scenarios, with many computers and screens that do not necessarily respect a position pattern, but rather are distributed across the room. Such scenarios can benefit from LOP-cursor for tasks such as file transfer across computers, where the user can pin control canvas on a high resolution display (computer 1), while controlling the file destination pointing to another display (computer 2, 3, 4 ...) with leg-2 cursor. Future works also include the study of leg-2 serving as a controller for another dimension, such that, its distance from (or circular movement around) the leg-1 can control selected object attributes(scale/depth/orientation).

Although the LOP is based on 3D interaction techniques, the evaluation in this paper focuses on a controlled 2D task. Nevertheless, we see a great potential for using the LOP in 3D environments. One possibility we are investigating is to control the canvas orientation in $\mathbb{R}_3$ using the device's orientation while pinned. We believe in this approach based on works in the literature showing that users quickly map 2D displacements from a given plane to any arbitrary plane, as for instance, the mouse on a table to the arrow cursor on a screen. Non-desktop 3D environments, as a CAVE, could benefit from the bases of LOP-cursor with some incremental development to add these arbitrary planes.

We also plan to investigate how the user distance from the screen and the screen size affects the optimal scale for the control canvas. As stated by Peck et at. [18], users tend to interact in different scales according to their distance from screen. Naturally, when users are near the display, they can see and point with more precision, and thus, a smaller control canvas allowing more precision would also make sense (Section3.3.2).

ACKNOWLEDGEMENTS

This work was supported by CNPq-Brazil under the projects 309092/2008-6, 483947/2010-5, 483814/2010-5 and 302679/2009-0. Thanks are due to all the volunteers who kindly tested the system.

REFERENCES

[1] R. Ballagas, J. Borchers, M. Rohs, and J. G. Sheridan. The smart phone: A ubiquitous input device. *IEEE Pervasive Computing*, 5:70–, January 2006.

[2] R. Ballagas, M. Rohs, and J. G. Sheridan. Sweep and point and shoot: phonecam-based interactions for large public displays. In *CHI '05 extended abstracts on Human factors in computing systems*, CHI EA '05, pages 1200–1203, New York, NY, USA, 2005. ACM.

[3] X. Bi, S.-H. Bae, and R. Balakrishnan. Effects of interior bezels of tiled-monitor large displays on visual search, tunnel steering, and target selection. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 65–74, New York, NY, USA, 2010. ACM.

[4] S. Boring, D. Baur, A. Butz, S. Gustafson, and P. Baudisch. Touch projector: mobile interaction through video. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 2287–2296, New York, NY, USA, 2010. ACM.

[5] S. Boring, S. Gehring, A. Wiethoff, A. M. Blöckner, J. Schöning, and A. Butz. Multi-user interaction on media facades through live video on mobile devices. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 2721–2724, New York, NY, USA, 2011. ACM.

[6] C. Forlines, D. Vogel, and R. Balakrishnan. Hybridpointing: fluid switching between absolute and relative pointing with a direct input device. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, pages 211–220, New York, NY, USA, 2006. ACM.

[7] W. Hutama, P. Song, C.-W. Fu, and W. B. Goh. Distinguishing multiple smart-phone interactions on a multi-touch wall display using tilt correlation. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 3315–3318, New York, NY, USA, 2011. ACM.

[8] S. Jeon, J. Hwang, G. J. Kim, and M. Billinghurst. Interaction techniques in large display environments using hand-held devices. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '06, pages 100–103, New York, NY, USA, 2006. ACM.

[9] H. Jiang, E. Ofek, N. Moraveji, and Y. Shi. Direct pointer: direct manipulation for large-display interaction using handheld cameras. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, CHI '06, pages 1107–1110, New York, NY, USA, 2006. ACM.

[10] N. Katzakis and M. Hori. Mobile phones as 3-dof controllers: A comparative study. In *Dependable, Autonomic and Secure Computing, 2009. DASC '09. Eighth IEEE International Conference on*, pages 345 –349, dec. 2009.

[11] M. Kobayashi and T. Igarashi. Ninja cursors: using multiple cursors to assist target acquisition on large screens. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 949–958, New York, NY, USA, 2008. ACM.

[12] S. O. H. Madgwick, A. J. L. Harrison, and R. Vaidyanathan. Estimation of imu and marg orientation using a gradient descent algorithm. In *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*, pages 1 –7, 29 2011-july 1 2011.

[13] D. C. McCallum and P. Irani. Arc-pad: absolute+relative cursor positioning for large displays with a mobile touchscreen. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pages 153–156, New York, NY, USA, 2009. ACM.

[14] M. Nancel, E. Pietriga, and M. Beaudouin-Lafon. Precision Pointing for Ultra-High-Resolution Wall Displays. Research Report RR-7624, INRIA, May 2011.

[15] T. Ni, G. S. Schmidt, O. G. Staadt, M. A. Livingston, R. Ball, and R. May. A survey of large high-resolution display technologies, techniques, and applications. In *Proceedings of the IEEE conference on Virtual Reality*, VR '06, pages 223–236, Washington, DC, USA, 2006. IEEE Computer Society.

[16] A. Olwal. Lightsense: enabling spatially aware handheld interaction devices. In *Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '06, pages 119–122, Washington, DC, USA, 2006. IEEE Computer Society.

[17] N. Pears, D. G. Jackson, and P. Olivier. Smart phone interaction with registered displays. *IEEE Pervasive Computing*, 8:14–21, April 2009.

[18] S. M. Peck, C. North, and D. Bowman. A multiscale interaction technique for large, high-resolution displays. In *Proceedings of the 2009 IEEE Symposium on 3D User Interfaces*, 3DUI '09, pages 31–38, Washington, DC, USA, 2009. IEEE Computer Society.

[19] P. Song, W. B. Goh, C.-W. Fu, Q. Meng, and P.-A. Heng. Wysiwyf: exploring and annotating volume data with a tangible handheld device. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 1333–1342, New York, NY, USA, 2011. ACM.

[20] D. Vogel and R. Balakrishnan. Distant freehand pointing and clicking on very large, high resolution displays. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, UIST '05, pages 33–42, New York, NY, USA, 2005. ACM.