# Pointing Facilitation Techniques for 3D Object Selection in Virtual Environments

## Doctoral Thesis

Fernando Argelaguet Sanz

Barcelona, May 2011

Tesi doctoral presentada per obtenir el grau del Doctor
amb Menció Europea per la Universitat Politècnica de Catalunya


Universitat Politècnica de Catalunya
Departament de Llenguatges i Sistemes Informàtics

Informàtica Gràfica
Programa de Doctorat de Software

Advisor : Carlos Andújar Gran

# Abstract

Selection is one of the fundamental tasks in virtual reality applications and the initial task for most common user's interactions in a virtual environment. In this thesis we analyze major factors influencing selection performance, and propose new techniques for facilitating selection in 3D space. Considering the frequency of selection tasks in a typical virtual reality workflow, improving selection tasks often results in significant gains in the overall user performance.

A 3D selection task requires the user to gesture in 3D space, e.g. grabbing an object or pointing to something. The success or failure of the task depends mainly on the interaction technique, the dexterity of the user, and the spatial perception of the virtual environment. Since the dexterity of the user can be improved by training, we focus on how to take advantage of existing human control models to minimize the effort required to select an object, and how to enhance the user's spatial perception of the virtual environment to facilitate selection and referral tasks. We propose several selection techniques based on Fitts' Law and study how visual feedback can be used to overcome spatial perception limitations in virtual environments. The techniques proposed are not only oriented to achieve performance gains as we also account for user's preferences. During the development of this thesis we have conducted a number of user studies, both to validate our theoretical analyses, and to compare the proposed selection techniques to existing ones.

Although the major contributions of this thesis refer to the selection of 3D objects, we also provide new techniques for facilitating the interaction with 2D graphical user interfaces embedded in 3D space. Furthermore, we explore selection tasks in collaborative virtual environments. In CVEs pointing tasks often change their purpose and turn into referring tasks. Referential awareness can be compromised in complex environments, because a user can point to a feature in the environment which might be occluded for the other users. We analyzed how improvements on referential awareness increase the information exchange among users without violating social protocols in formal presentations.

# Acknowledgements

First, I would like to thank my advisor, Carlos Andújar, for his guidance and support throughout all my Ph.D. research, specially for his endless patience during the long revisions for all papers and this thesis. I also wish to thank all members of the Moving Group whose comments and suggestions have improved the quality of this research, specially to my lab mates for their help, ideas and fun. I would like to thank Dr. Bernd Froehlich and all the colleges from the Virtual Reality Systems group at the Bauhaus Universität for all the ideas we came up with and the great time spend during my stay at Weimar. I also want to thank many other people whose participation in the user studies made much of this work possible. This research was funded by a grant of the Spanish Ministry of Science throughout my work.

Finally, I want to send my biggest thanks to my parents, the rest of my family and friends for all of their encouragement, love, and support throughout this entire process, which would have not been possible without them.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Virtual reality (VR) has gained in popularity over the last years. The strong potential of VR has been acknowledged in multiple areas including design, prototyping, psychiatric treatment, scientific visualization, cultural heritage, virtual tourism and collaborative work. However, VR has also been criticized since it has not always lived up to its promises. The lack of usable user interfaces for virtual environments (VEs) has been a major factor preventing the deployment of effective VR systems outside research labs.

User Interfaces (UIs) are an essential component of any interactive application. The UI defines how the communication between the user and the application is done, being a critical issue during application design. The UI translates user actions (inputs) into application changes, and application state (output) into a representation the user can understand. A good UI must provide the user with efficient tools for driving the application while balancing expressiveness and simplicity.

Most interactive applications available today still rely on the well-established WIMP paradigm and the mouse-and-keyboard interface. Although these traditional UI components are well suited for personal computers, interaction techniques available for mouse-and-keyboard setups are inappropriate for most immersive virtual environments (IVEs). In these systems, a mouse-and-keyboard setup is practically unusable, as users might be standing in front of a projection screen or wearing a head mounted display. Furthermore, the displayed content is always 3D, so traditional 2D approaches might no longer apply; user interfaces involving 3D interaction are thus required.

User interfaces involving 3D interaction are called 3D user interfaces (3DUIs). By employing VR tracking technologies [109], 3DUIs allow users to interact directly in 3D space and reproduce everyday actions into the virtual environment. Users can e.g. interact with objects by grasping them with their hands or explore the virtual environment by just moving their heads.

A number of 3D interaction techniques have been developed to allow users accomplish typical tasks in virtual environments. Bowman et al. [12] proposed a task-driven taxonomy to classify 3D interaction techniques according to four main interaction tasks: selection, manipulation, navigation, and application control. While manipulation and navigation tasks have their clear counterpart in real world tasks, selection and application control are more specific to computer applications.

Manipulation tasks range from applying rigid transformations to 3D objects (translations and rotations), to modifying their physical properties or their shapes. Navigation tasks involve modifying the current viewpoint to explore the environment, search for some feature, travel from one location to another, or perform precise maneuvering tasks. Application control tasks allow the user to send specific commands to the application, changing state values, or requesting some functionality.

Concerning selection tasks, although they resemble real world tasks, their purpose is slightly different. In reality humans do no perform selection tasks directly; we just made choices. If we want to move an object, we think about the object we want to move, and then we perform an action to move it. In a computer application though, in addition to think about the object, we also have to inform the system about the object we want to interact with (selection task).

Although in this thesis we only focus on selection tasks, improvements on selection tasks will also improve manipulation, application control and navigation tasks, as they often depend on (as are preceded by) selection tasks. In this sense, efficient and error-proof selection techniques are critical because they allow the user to control the interaction flow between the above tasks.

Manipulation tasks and selection tasks are highly coupled; the user often has to select an object prior to its manipulation. In some situations, the same action used to manipulate an object can also be employed to select it. If the user has to translate an object, he can select it by directly grasping it, move it to the desired location and release it. However, as we shall see, grasping is not suitable for many VR applications. Application control tasks also benefit from improvements in selection tasks. Since they are performed by interacting with a 2D or 3D graphical user interfaces, efficient and error-proof selection mechanisms are also required. Finally, navigation can also benefit from efficient selection tasks when the navigation technique requires the user to select the destination.

In summary, despite selection tasks are conceptually simple, even small improvements will result in overall improvement of the usability of the entire application. By analyzing and

classifying existing selection techniques, we identified their major hindrances and explored how to overcome them.

## 1.1 Motivation

Developing appropriate 3D interaction techniques for immersive virtual environments is a challenging problem. On the one hand, interacting in free space with gestures greatly increases the richness and expressiveness of the interaction. On the other hand it might hinder interaction by increasing the dexterity required and it might raise fatigue levels. Consider for example the differences in selecting a 2D object using a mouse pointer, and grasping a 3D object in free space. Grasping an object in 3D space requires a complex arm movement (shoulder, arm, forearm and wrist) while using a mouse it only requires wrist and finger movements. The involvement of bigger muscle groups (and the extra degrees of freedom to control) often decreases the precision of the movement and increases the physical effort.

Moreover, immersive virtual environments (IVEs) cannot provide the same level of cues for understanding the environment, nor reproduce faithfully the physical constraints of the real world. For that reason, despite users always use 3D interaction in the real world, in a IVE users experiment difficulties in controlling multiple degrees of freedom simultaneously or understanding 3D spatial relationships. Furthermore, these problems are magnified due to the lack of standards for VR input and output devices.

There seems to be, in general, little understanding of human computer interaction (HCI) in three dimensions, and a lack of knowledge regarding the effectiveness of interaction in IVEs, although some recent work has begun to address these issues [86]. During decades, many researchers held to the intuitive notion that interaction in IVEs should replicate our interaction with the physical world. However, such interaction is never completely realistic, and severely limits the potential for productivity. In contrast, we can enhance the physical, cognitive, and perceptual capabilities of the user, allowing them to do things that are impossible in the real world.

Over the last decades, a number of interaction techniques have been proposed for object selection in virtual environments. We can identify two main approaches: virtual hand [96, 98] and virtual pointing metaphors [82, 56]. In the early days, virtual hand techniques were the most popular metaphor as they map identically the real task and the virtual task, giving a more natural interaction. Lately, it has been shown that virtual environments can overcome the physical constraints of the real world. For example, letting the user to select objects out of reach by enlarging the user's virtual arm [96] or using virtual pointing techniques such as

raycasting [82], which is one of the most popular techniques for 3D object selection tasks [9].

There are a number of user studies in the literature comparing different selection techniques [4] and, in overall, virtual pointing techniques result on better selection effectiveness than competing 3D selecting metaphors such as the virtual hand. Unlike virtual hand techniques, virtual pointing techniques allow the user to select objects beyond the area of reach and require less physical hand movement.

However, current pointing selection techniques still leave much room for improvement. Selection of small or distant objects is particularly difficult, and performance tends to degrade in high-density scenes. Some techniques attempt to solve the selection of small objects by increasing the size of the selection tool [40, 94], at the expense of requiring disambiguation mechanisms, for example, using metrics to guess which object the user wants to select [32].

Furthermore, the lack of physical support [71] and tracking noise have a severe negative impact on selection performance, specially for high precision selections. Current approaches to mitigate these effects include choosing appropriate selection devices or filtering the user's hand position and orientation [92]. Moreover, once the selection ray intersects the target, the user has to maintain the ray's orientation until the selection confirmation is triggered by, for example, pressing a button. When pressing a button, the user's hand might change its orientation involuntarily, changing slightly the pointing direction and causing a wrong selection. This effect, nicknamed Heisenberg effect [13], introduces a further difficulty in selecting small targets. All these problems contrast with mouse-based interaction where none of them arise.

In addition, occlusion is a big handicap for accomplishing spatial tasks [36], as most interaction techniques for 3D selection and manipulation require the involved objects to be visible from the user's viewpoint. A common solution for selecting occluded objects is to navigate to a different location so that the targets become unoccluded. However, this navigate-to-select approach is impractical for selection intensive applications. Three-dimensional occlusion management techniques are often essential for helping viewers understand the spatial relationships between the constituent parts that make up these data sets.

In essence, selection of small and partially occluded objects can cause user dissatisfaction due to increased error rates, discomfort due to the duration of corrective movements, which in the absence of physical support require an additional physical effort, and unconfidence on which object will be selected after triggering the confirmation, thus compromising usability.

Usability is a key factor from different points of view. From the user's perspective it can make the difference between performing a task accurately or not, and enjoying the task or

being frustrated. From the developer's perspective, usability is important because it can mean the difference between the success and the failure of a system. From a management point of view, software with poor usability can reduce productivity. In all cases, the lack of usability can cost time and effort.

Usability of current 3D selection techniques can be improved in several aspects. On the one hand, we can focus on user performance, which can be measured with objective metrics such as task completion time and error rate. An strategy to improve performance lays on applying human control models such as the optimized initial impulse model [81] and Fitts' Law [38, 39]. While the optimized impulse model refers to the accuracy a user can achieve given the movement required to perform the action, Fitts' Law estimates the time required to acquire a target, both in the physical and in the virtual world. However, as we are bounded by human motor skills, there is a natural trade-off between speed and accuracy. In a standard scenario, high-accuracy rates will produce high task completion times and vice-versa.

On the other hand, we can take into account user preferences. User preferences are mainly subjective and qualitative. We can consider a number of different measures, including ease of use, ease of learning, user satisfaction, user comfort and intuitiveness. In the context of the real usage of a VR application, the subjective impressions of the users about an interaction technique can play a much larger role than speed in controlled experiments. The inability to select precisely may prove to be overly annoying to the user and thus be a source of frustration and dissatisfaction. In addition, a reduction of selection time might not be always desirable, for example, if the reduction is achieved at the expense of increasing the cognitive load of the task, or requiring longer learning curves.

Finally, we cannot obviate the potential overhead of the interaction technique in the application performance; VR applications have to provide interactive frame rates (over 25-30 fps) and the end-to-end latency must be as low as possible.

## 1.2 Contributions

The main goal of this thesis is to contribute to the 3D user interfaces field by analyzing major factors influencing selection performance, and proposing new interaction techniques for 3D object selection in immersive virtual environments. We also aim at improving the interaction with 2D GUIs embedded within virtual environments. As they are a particular case of 3D object selection, we can develop specific selection techniques according to their needs.

In Chapter 2 we present a complete analysis of current selection techniques, and then, we follow with the contributions of this PhD. The contributions are subdivided into four main blocks:

- Analysis of visual issues in virtual pointing.
- Overcoming visual issues in virtual pointing.
- Apply Fitts' law to enhance 3D object selection.
- Improve the development and the usability of 2D GUIs embedded in virtual environments.

Now a short summary for all contributions is provided.

### 1.2.1 Analysis of visual issues in virtual pointing tasks

The user's perception of the virtual environment is crucial for an effective interaction. Effective object selection and manipulation requires the object to be clearly visible in its exact location. This mostly holds for 2D displays, but it does not hold for immersive virtual environments due to the limitations of the displayed content. For example, the deficiencies of the depth cues provided by stereoscopic output devices keep users from grabbing 3D objects effectively, requiring multiple feedback loops in order to accomplish the action.

#### Visual feedback for pointing on stereoscopic displays

One of our main concerns is visual feedback. Selection techniques require the application to provide appropriate visual feedback about the pointing tool and its spatial relationship with potential targets. In virtual pointing techniques, this is often provided by drawing a ray/cone extending out from the user's hand. Similarly, virtual hand techniques use a hand avatar. Visual feedback has to provide information about two key questions: which object is being intersected by the selection tool (if any), thus allowing users to identify the object that would be selected if the selection is confirmed, and which movement of the selection tool (translation, rotation, or a combination of both) is needed to aim at a particular target.

The usage of stereoscopic displays poses several problems to provide precise feedback on the two questions above. First, current stereoscopic displays are not able to reproduce well all the visual cues provided by real-world objects. A second problem is the limited ability of the human visual system to fuse objects with different retinal disparities.

We performed an exhaustive analysis of existing selection techniques and their visual feedback, studying how they behave when accurate pointing is needed. We analyzed the main cursor-based and ray-based visual feedback techniques in combination with multiple hand-to-device mappings. Both theoretical and experimental analysis showed that existing selection techniques have severe limitations regarding their visual feedback when precise selection tasks are required.

Moreover, the fact that most pointing techniques for 3D selection rely on a ray originating at the user's hand whose direction is controlled by the hand orientation introduces additional problems. The literature has largely ignored the effect of the misalignment between the user's viewpoint and the user's hand.

However, this misalignment in combination with poor visual feedback have a significant impact on selection performance. For objects that are visible but appear occluded from the user's hand position, visual feedback will misguide the user decreasing selection performance (see Figure 1.1). We will refer to this issue as the eye-hand visibility mismatch.

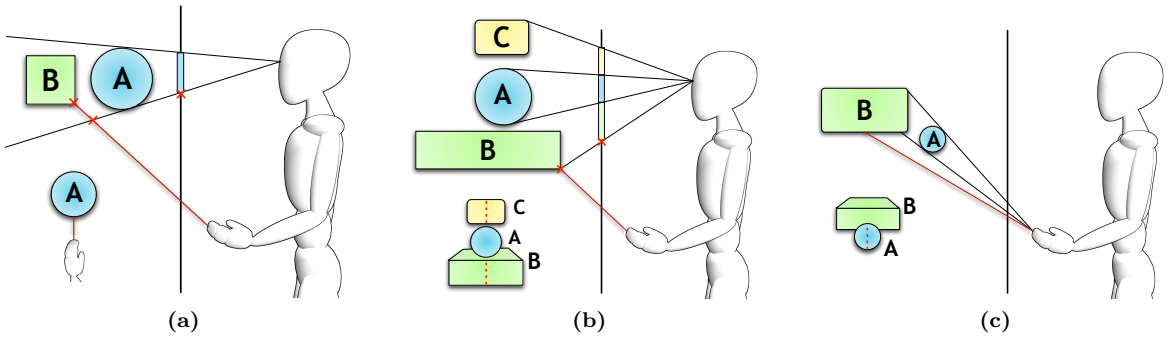



**Figure 1.1:** *Eye-hand visibility mismatch issues. (a) The user can selected an object which is hidden by another object. The last visible point on the ray, is projected over the screen projection of the occluding object, leading to misinterpretation: the ray appears to intersect object A, although the intersected object is behind. (b) The visible object A cannot be selected because it cannot be reached by a ray emanating from the user's hand. The dotted line shows the path followed by the ray-scene intersection as seen on the screen; it skips object A. (c) Object A is visible and selectable, but no point on its boundary is simultaneously visible and selectable. The dotted line shows the path followed by the ray-scene intersection as seen on the screen, any intersection point on the boundary of B is visible from the user's viewpoint.*

We designed an experiment to evaluate the impact of the eye-hand visibility mismatch for raycasting selection. We proposed two different test scenarios where all objects had the same theoretical index of difficulty (computed using Fitts' Law formulation). In the first scenario all the objects appeared unoccluded from the hand's position, but in the second scenario several objects suffered from eye-hand visibility mismatch. The results of the experiment showed a significant drop in selection performance when objects appear occluded from the hand's position.

The details of this contribution can be found in Chapter 3.

### 1.2.2 Overcoming visual issues in virtual pointing tasks

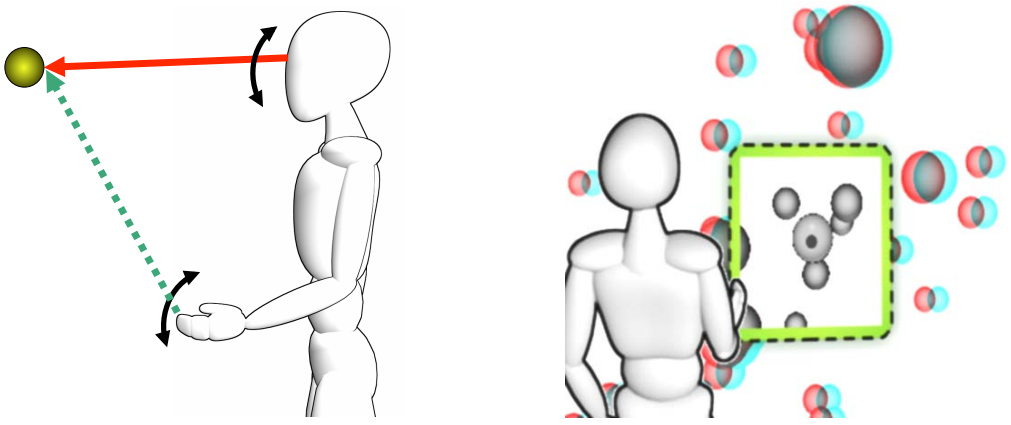
#### Eye-hand visibility mismatch and visual feedback

For selection tasks, the eye-hand visibility mismatch appears when the set of selectable objects and the set of visible objects differ. We can avoid this limitation by matching the origin of the selection tool with the user's viewpoint. The first option considered was employing image-plane techniques.

However, existing image-plane techniques do not allow controlling the selection tool with hand rotations (they are controlled only by hand position). As the selection tool is controlled by bigger muscle groups, it results in increased fatigue levels and decreased precision, . In addition our visual feedback analysis showed that existing visual feedback techniques are not enough accurate when selecting small 3D objects.

We proposed a new device-ray mapping, where the selection ray is controlled by hand rotations, but emanates from the eye position. This mapping combines the benefits of image-plane techniques (absence of visibility mismatch and continuity of the ray movement in screen-space) with the benefits of ray control through hand rotation (requiring less physical hand movement from the user). In this sense, it can be considered as a hybrid technique between raycasting and image-plane techniques.

Besides the device-ray mapping, adequate visual feedback must be provided. Since the selection ray originates at the eye position, the ray projects into a single point in the viewing plane. We successfully developed two different visual feedback techniques which better comply with stereoscopic output devices.



**(a)** *Hand-to-cursor ray* **(b)** *Viewfinder*

**Figure 1.2:** *Proposed visual feedbacks to use in combination with the Raycasting from the eye*

- The hand-to-cursor ray (see Figure 1.2a). The visual feedback provided is a ray (the display ray) described by the hand position and the intersection point of the selection ray with the virtual environment. The feedback ray clearly determines the object currently selected, and provides enough information about the movement required to aim a target.
- The Viewfinder (see Figure 1.2b). The visual feedback provided locally flatten potential targets in the vicinity of the pointing direction by projecting them onto a small virtual screen attached to the pointing direction itself. We call this technique viewfinder because the resulting effect is similar to looking a small part of the scene thought an LCD digital camera display. The visual feedback provides enough information of the object intersected (a small cursor in the middle of the viewfinder represent the selection ray), and provides enhanced information regarding the movement required to aim a target, as the cursor is continuous along the screen space.

The user evaluation showed that Raycasting from the Eye outperformed existing device-ray mappings (raycasting and image plane techniques) no matter which visual feedback was used. Our proposed mapping clearly outperformed raycasting especially for selections with significant levels of eye-hand visibility mismatch.

The details of the Raycasting from the eye and the Viewfinder can be found in Section 4.1.

**Supporting referential awareness in collaborative virtual environments**

In the common workflow of a virtual reality application, 3D selection techniques are used to determine the the object (or feature) of interest. Once defined, the user may transform or change its properties. However, in applications where several users interact in the same virtual environment (collaborative virtual environments, CVEs) selection tasks can change their purpose and become referral tasks. One user might want to show some feature of the environment to others.

One of the main concerns in CVEs is to keep awareness among users. All users should know where other users are and what are they doing. We have discussed how eye-hand visibility mismatch can play an important role for selection tasks, but in CVEs the potential viewpoint mismatch among several users can be extreme: one user might select an object but other users might be unable to locate the object (absence of referential awareness).

To solve this problem in the real world, people have to walk around the occluding objects to obtain a suitable viewing position. Often they move close to the person who is pointing in order to see the specified object (e.g. by looking over his shoulder). This issue is of

particular importance if the two users collaborate in the same physical environment because it may result in physical proximity. Close proximity among two users can potentially induce discomfort and does not comply with social protocols, especially in formal presentations.

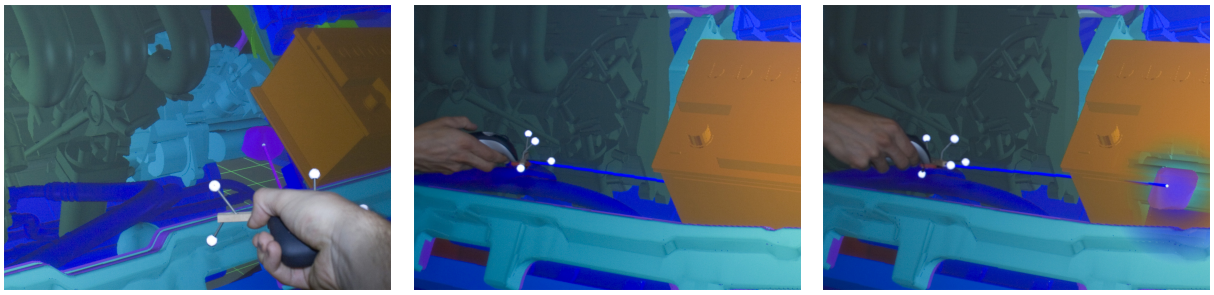
We have explored how users behave in these situations in a real scenario, by resembling a joint design review of an automotive engine model. One user (the presenter) was responsible to show some occluded features to another (the observer). We explored how users behave in that situation, keeping track of the distance between them and checking whether the distance among users comply with social protocols.

As we expected, users tend to keep closer to the presenter violating social protocols. In order to keep referential awareness observers had to continuously follow the presenter, thus keeping a similar viewpoint. While in reality this problem can only be solved by adapting the viewing position, specialized individual views of the shared virtual scene enable various other solutions (see Figure 1.3).

As one such solution we propose using virtual X-Ray techniques to ensure that referred objects can be seen by others. For each specialized individual view, potential occluders can be removed or turned semitransparent. X-Ray techniques ensure that users are able to see the referred object. However, the use of such augmentation techniques might compromise spatial perception and decrease context information because the removed content.

We analyzed the influence of such augmented viewing techniques on the spatial understanding of the scene, the rapidity of information exchange as well as the social behavior of users. The results of our user study revealed that X-Ray techniques in addition to allow users to keep more comfortable distances, they support spatial understanding on a similar level as walking around to achieve a non-occluded view of specified objects.

The details of this contribution can be found in Section 4.2.



**(a)** **(b)** **(c)**

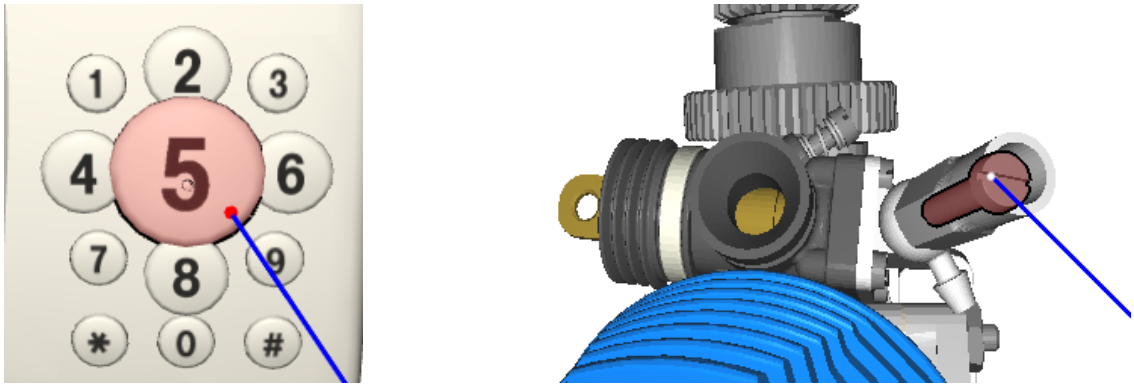
**Figure 1.3:** *Figures (a) and (b) illustrate the issue of interpersonal occlusion between two tracked users in a collaborative virtual environment: an object that is fully visible to one user (a), can or cannot be partially seen from other viewpoints (b). Virtual X-Ray techniques can improve target discovery in such situations by showing the indicated object through the occluding environment (c).*

### 1.2.3 Applying Fitts' law to enhance 3D object selection

Fitts' law is a human psychomotor behavior model which introduces several guidelines to improve selection performance. According to Fitts' law formulation ($MT = a + b\log(A/W + 1)$), the mean selection time of an acquisition task (MT) can be reduced by decreasing the amplitude of the movement (A), increasing the size of the target (W) or a combination of both.

These guidelines have been successfully applied to develop pointing facilitation techniques for WIMP interfaces, but have been hardly applied to 3D object selection. For 3D object selection, the efforts have been focused on increasing the area of influence of the selection tool, or modulating the control-display ratio [41, 63].

However, there are no studies on 3D object selection techniques which dynamically increase the size of objects to improve their selection. Increasing the effective size of objects, as Fitts' law predicts, may result in better selection times but it may introduce some drawbacks. For example, if we scale an object it might occlude neighboring objects making them unselectable. We have developed and evaluated two different methods to increase the size of small and occluded objects: Dynamic Scaling and Forced Dissoclusion.




(a) (b)

**Figure 1.4:** *(a) Dynamic Scaling approach scales the currently selected object and Neighbouring objects are rearranged to avoid occlusion. (b) The forced dissoclusion approach shows the selected object completely unoccluded.*

Dynamic Scaling (DS) increases the size of objects indicated by the selection tool and rearranges neighboring objects to minimize occlusion (see Figure 1.4a). The rearrangement is driven by an image-space graph which encodes neighboring information for each object in the environment. Neighboring objects are slightly scaled and moved apart to avoid occlusion and also for facilitating their selection in case the currently indicated target is not the intended one.

In contrast, Forced Disoclussion (FD) maximizes the number of visible pixels of the focus object by forcing it to appear completely unoccluded (see Figure 1.4b). An object which was partially visible becomes fully visible.

Given that the effort to select small and partially occluded objects is governed by the final corrective movements, in the best case scenario one could expect that Dynamic Scaling and Forced Disocclusion have a positive impact on selection performance. However, the transformation of neighboring targets to avoid occlusion could be potentially distracting to the users and negate the benefits of Dynamic Scalling. On the other hand, forcing the disocclusion of the object being pointed to might occlude neighboring objects and result in poor performance.

To evaluate their usability, we conducted a user study. In terms of task performance, the results showed that the drawbacks of increasing the effective size of targets may exceed their benefits. Although we did not found significant differences for task completion time among classic raycasting, task completion time for FD and DS tends to degrade as the complexity of the scene increases. On the other hand, error rates were significantly lower for DS and FD, and, the additional visual feedback provided made easier to users to recognize the object intersected by the selection ray.

The details of this contribution can be found in Section 5.

### 1.2.4 Interacting with 2D GUIs embedded in VEs

Application control is one of the fundamental tasks a Virtual Reality application must ensure. Given its flexibility and its ease of use, graphical user interfaces are typically employed for these purposes, but creating graphical user interfaces for virtual environments can pose several problems.

#### A cost-effective approach for embedding 2D GUIs in VEs

Existing GUI toolkits for VEs are still too simple; they allow only a limited number of GUI components and often lack visual authoring tools. In contrast, existing GUI toolkits for 2D desktop environments are mature, include powerful authoring tools and provide a wide range of widgets.

We proposed a new approach to improve GUI prototyping for virtual environments. This approach allows developers to re-use existing 2D GUIs and embed them into virtual envi-

ronments. It is based on monitoring and capturing the 2D displayed contents (windows) and embed them into the virtual environment as 3D windows. Once embedded, the user is able to interact with them using standard 3D interaction techniques (selection and manipulation), and in contrast to VNC and other frame-based approaches, the application has knowledge of the GUI structure. Our approach minimizes the number of lines of source code that need to be modified to migrate the GUI of an existing application to a VE. As migration, we refer not only to the graphical representation of the widgets, but also to their behavior.

The details of this contribution can be found in Section 6.1

However, although this approach reduces the development step as the 2D GUI remains unchanged, embedding an existing 2D GUI directly in a virtual environment may pose some usability problems. As 2D widgets are optimized for mouse and keyboard interaction, existing 3D interaction techniques might result in poor usability. Instead of having the 2D GUI redesigned, we proposed two alternative approaches to improve its usability.

**Anisomorphic ray-casting manipulation**

The selection of small GUI elements using raycasting pose several usability issues as it requires a high degree of accuracy. Small rotations of the wrist sweep out large arcs at the end of the selection ray. Therefore hand trembling and tracking errors are amplified with increasing distance, thus requiring a high level of angular accuracy. Accurate selection is also compromised by the hand instability caused by the absence of constraints on the hand movements (lack of physical support for manipulation). As a result, users attempting to select small buttons have to make a considerable effort to stabilize their wrist.

Using an anisomorphic mapping between the user's hand orientation and the selection ray orientation, we are able to scale down hand rotations and enable accurate selections and manipulations. The anisomorphic mapping modifies the control-display ratio to increase accuracy, so that the ray rotates more slowly than the user's hand, thus reducing the effect of hand instability. Our technique uses a curved representation of the ray providing visual feedback of the orientation of both the input device and the selection ray (see Figure 1.5a).

Altough the anisomorphic mapping increases the amplitude of movement, our experiments indicate that it outperforms significantly isomorphic ray-casting in task completion time, number of mistakes and manipulation accuracy, especially for high accurate selections.

The details of this contribution can be found in Section 6.2.

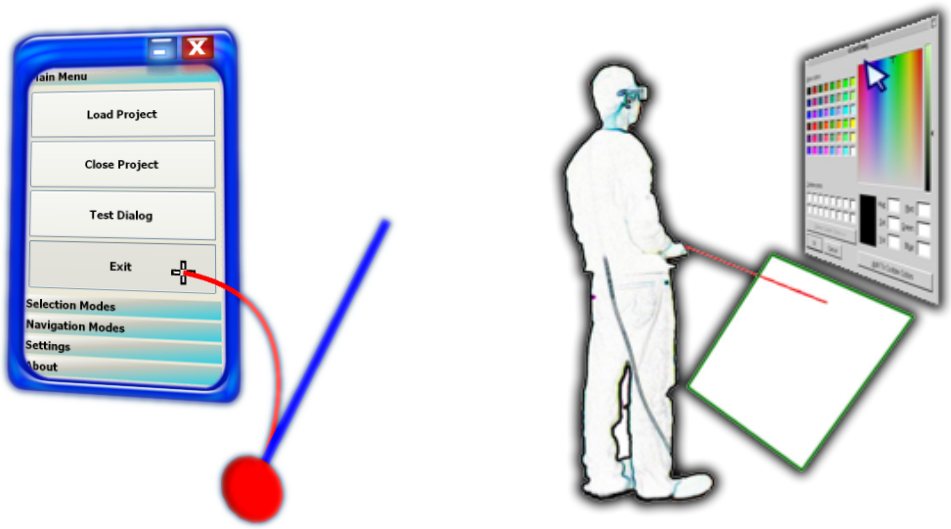
### Decoupling Motor Space and Visual Space

When interacting with WIMP interfaces in a personal computer, the motor space and the visual space are decoupled. The relative movement of the mouse on the table (motor space) is mapped to the cursor over the graphical user interface (visual space). We explored whether decoupling motor and visual space is beneficial also for 2D GUIs embedded in virtual environments.

The main advantage of this decoupling is than 2D GUI components can be selected and manipulated within a user-defined working volume, whose location and size is completely independent from the application's visual representation (see Figure 1.5b). This decoupling is accomplished through a virtual pad which receives user actions and maps them into cursor movements. The user can place the virtual pad freely on the environment, e.g. in a location that allows a more comfortable interaction. In addition, the virtual pad can be scaled to manually adjust the control-display ratio.

We designed an experiment to evaluate how the virtual pad metaphor behaves in terms of time-to-complete a task and accuracy. The results showed no significant differences in terms of selection performance and error rates between direct interaction and interaction through the virtual pad. However, our experiments indicate that the manipulation through the virtual pad technique increases user's comfort while providing dynamic management of speed/accuracy trade-off.

The details of this contribution can be found in Section 6.3.




**(a)** *Friction Surfaces* **(b)** *Virtual Pad*

**Figure 1.5:** *(a) Anisomorphic raycasting's visual feedback. The red way corresponds to the computed selection ray and the blue ray to the real hand orientation. (b) Using the virtual pad metaphor the user can decouple the working and the visual space.*

# Chapter 2

# Previous work

In this chapter we discuss the intrinsic and extrinsic limitations of 3D object selection. The ability to efficiently select an object is constrained by several factors: the properties of the object to select, the virtual environment, the input and output devices, and the users' skills. The knowledge of these limiting factors allows to design better selection techniques and improve existing ones.

Object selection techniques involving physical interaction are constrained by the human motor system as the speed and the accuracy of any gesture are limited by the nervous and muscular systems. In Section 2.1 we introduce two human control models, the Fitts' Law and the optimized initial impulse model. Both explain how selection performance is related to the object's size (accuracy) and the location of the object (amplitude of the movement required for its selection). On the one hand, Fitts' law determines the relationship between the time required to perform an acquisition task regarding the object's size and the amplitude of the movement required for its selection. On the other hand, the optimized initial impulse model explains how users performs acquisition tasks and how they trade-off speed an accuracy. The application of these principles to human computer interaction provide several guidelines to improve pointing performance.

In contrast, other limitations are explained in terms of the main characteristics of the selection technique. Each selection technique provides the user with a selection tool (e.g. a 3D cursor, a ray) and states how the user is able to control it (e.g. through hand translations, wrist rotations). Both determine the level of accuracy the user can achieve and the physical effort required to perform a selection task. In Section 2.2 we classify current selection techniques according to these characteristics. The resulting classification allows the identification of the worst and the best case scenarios for each selection technique.

Moreover, external factors, such as the virtual environment and the input and output devices, introduce additional limitations. For example, selections in cluttered environments require

additional accuracy due to occlusion, and inaccurate tracking devices can hinder selection tasks requiring high accuracy. In Section 2.3 we detail these limitations and present usability guidelines to deal with them.

In addition to this general perspective on selection techniques, we are also interested in two particular situations: the use of selection techniques to refer to objects in collaborative environments and the interaction with 2D graphical user interfaces embedded in virtual environments.

In Section 2.4 we discuss the importance of referential awareness in collaborative environments and how it has been traditionally addressed. Regarding the interaction with 2D graphical user interfaces, in Section 2.5 we review existing approaches to embed them in a virtual environment and analyze whether existing selection techniques are well suited to interact with them. Finally, in Section 2.6 we summarize current approaches for designing evaluations of 3D user interfaces.

## 2.1 Human pointing models

In order to point to (*acquire*) an object (*the target*), a user is required to perform a set of gestures (*movements*) to position the selection tool (e.g. his finger) over it. For each movement, the final position of the selection tool (*endpoint*) determines whether the acquisition is accomplished (the endpoint is inside the target) or not (the endpoint is outside the target). Once the target is acquired, the user has to trigger some selection mechanism to confirm the acquisition (e.g. pressing a button).

Pointing tasks involving physical interaction are constrained by the human psychomotor behavior. Several human pointing models have been presented in order to model these aiming movements, to allow a better understanding of the processes involved and providing reliable prediction models of performance. From all the existing human motor models, Fitts' law provides by far the most successful and complete explanation. Fitts' law estimates the time required to perform an aimed movement considering only the physical properties underlying the acquisition task (the size of the target and the amplitude of the movement required to acquire it).

However, Fitts' law does not explain the processes involved in the acquisition task. Several explanations appeared, like the *iterative corrections model* by Crossman and Goodeve [28], which stated that the entire movement towards the target is subdivided in a set of small movements, each taking the user closer to the target. However, only the *optimized initial*

*impulse model*, proposed by Meyer et al. [81], provided a complete explanation accounting from all the effects shown in the literature. It states that acquisition tasks are subdivided in a two-step movement phases. First a fast and inaccurate movement is made towards the target and then, iterative slow correction movements are executed until the target is acquired.

### 2.1.1 Fitts' Law

Fitts' law [38], which emerged from experimental psychology, is a well known human psychomotor behavior model which has been widely adopted in numerous areas, including human factors, ergonomics and human-computer interaction. The application of Fitts' law ranges from estimating the time required to perform an assembly operation, the time required to press a button with a mouse or to select an object in 3D space. Fitts' law is so well know because it provides one of the few quantitative measures for human-computer interaction research.

Fitts, originally, sought to establish the information capacity of the human motor system. His model mimics Shannon's Theorem 17 [129] (see Equation 2.1), which expresses that the information capacity of a channel (C) is determined by the bandwidth (B), the signal power (S) and the perturbations introduced by white thermal noise (N).

$$C = B \log_2 \left( \frac{S + N}{N} \right) \tag{2.1}$$

Fitts claimed that electronic signals are analogous to the distance or the amplitude of the movement to acquire a target (A) and noise is analogous to the tolerance or width (W) of the movement's endpoint during acquisition tasks.

Fitts' law stated that the information capacity of a given task, which he called the index of performance (IP) (see Equation 2.2), is obtained by dividing the index of difficulty (ID) of a motor task, by the movement time (MT) required to perform the task.

$$IP = ID/MT \tag{2.2}$$

Following Shannon's logarithmic expression, Fitts' proposed Equation 2.3 as the index of difficulty; the ID is considered in bits as it has no units. The formulation slightly differs from Shannon's theorem. The purpose of multiplying $A$ by two was to avoid negative IDs,

for all practical situations the minimum value for A is $W/2$. It also has the effect of adding one bit to the index of difficulty.

$$ID = -\log_2\left(\frac{W}{2A}\right) \tag{2.3}$$

Combining Equations 2.1,2.2 and 2.3, we can build the original Fitts' law formulation (see Equation 2.4.

$$MT = a + b\log_2\left(\frac{2A}{W}\right) \tag{2.4}$$

Where $a$ and $b$ are regression coefficients (see Figure 2.1). The intercept $a$ is sensitive to additive factors like reaction times (e.g. time to locate the target or time to trigger the selection confirmation) and the inverse of the slope $1/b$ is the index of performance (IP) expressed in seconds/bit.

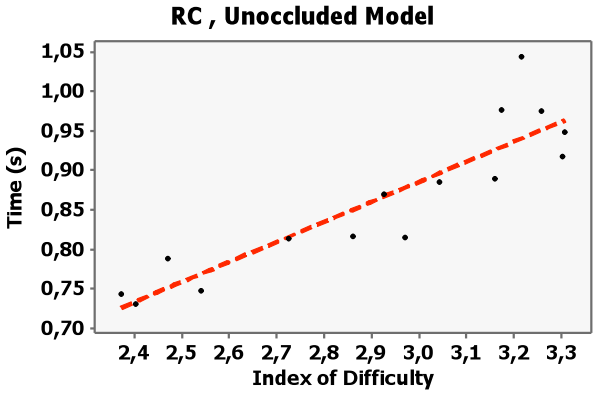
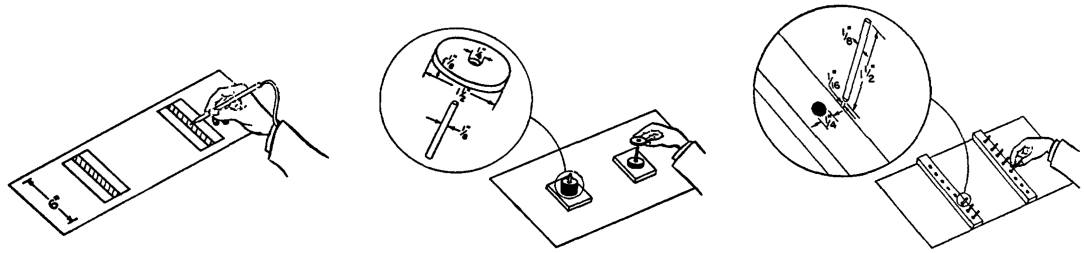



**Figure 2.1:** *Example of a Fitts' law regression study. Each dot represent a different acquisition task. The stripped line corresponds to the function* $f(\mathrm{ID}) = a + b\ \mathrm{ID}$.

If we forget about the Shannon derivation, the model still has a simple physical interpretation. The mean selection time increases when the amplitude of the movement (A) increases, decreases when the precision required to acquire the target (W) increases and vice-versa. In other words, tasks become more difficult when targets are smaller or farther away [116].

To validate his model, Fitts designed three different user studies. As a requirement, the experiments had to be simple enough to keep the cognitive load of the participants as low as possible. Each task involved successive repetitive actions covering the same amplitude, maximizing user's performance due to repetition and the participants were asked to perform the task as fast as possible.



(a) (b) (c)

**Figure 2.2:** *Original Fitts' experiments. (a) Reciprocal tapping task. Participants had to hit repeatedly both center plates (stripped), without hitting the error plates surrounding the center plates. (b) Disc transfer task. Participants had to transfer eight washers (one at a time) from the right to the left pin. (c) Pin transfer task. Participants had to transfer each pin from one side to another. Image from Fitts [38].*

With these requirements, Fitts assumed that the performance would be limited mainly by the capacity of the human motor system.

Fitts' successfully validated his model on a tapping and two transfer tasks (see Figures 2.2). He obtained high correlation values ($r = 0.99$), using Equation 2.4, between mean selection time and ID (ID values ranging from 1 to 7). A number of following studies from other authors also validated the model and proved its robustness.

In addition to the mean selection time estimation, if we experimentally obtain the mean movement time (MT), we can compute the index of performance (IP). In a study presented by Card, English and Burr [17] several input devices were evaluated by performing a text selection task. In that scenario they used the index of performance IP to rank devices taking into account the measured performance. Other examples can be found in [74, 76].

Fitts' law can also be used to evaluate the performance of different aiming movements. In a study by Stuard et al. [18] several aiming movements involving different muscle groups were evaluated (hand, forearm, arm). Their results showed than the bigger is the muscle group the lower is the IP. It is interesting to notice that the index of difficulty was the same for the different muscle groups and only the slope ($b$) changed. It also supports the thought that the intercept $a$ accounts for additive factors like reaction times and $b$ for task performance.

Successive iterations have been done in order to improve the data-to-model fit. The most common adopted formulation was proposed by Scott MacKenzie [75]. His formulation (see Equation 2.5) obtains a better fit for lower IDs ($< 3$). This formulation is also known as the Shannon formulation of Fitts' Law, as it totally resembles the Shannon's Theorem.

$$MT = a + b \log_2 \left( \frac{A + W}{W} \right) \tag{2.5}$$

Now we review additional studies that have been carried out focusing on the $W$ and $A$ adjustments and the extension of Fitts' law to higher dimensional tasks.

### Effective target width

MacKenzie explored in [74] how the ID can be adjusted for high values of $W$. Instead of computing $W$ directly from the task, MacKenzie proposed a method to compute the value of $W$ considering how the subjects performed, and thus $W$ becoming a dependent variable. The width computed by the MacKenzie method is called *effective target width* ($W_e$).

The effective width is computed considering the number of errors (the endpoint of the movement is outside the target) and the distribution of the distance from each endpoint to the midpoint of the target for each task (see Figure 2.3). The distribution of the distance is assumed to follow a normal distribution. If the percentage of errors is smaller than 4%, $W_e$ can be computed as $4.133\sigma_d$, where $\sigma_d$ is the standard deviation of the endpoint of the movements. Otherwise, if the percentage is greater than 4%, $W_e$ is equal to $W$. In essence, $W_e$ corresponds to the distance covering the 96% of endpoint distances. When $W_e$ is used, the index of difficulty will be referred as $ID_e$.

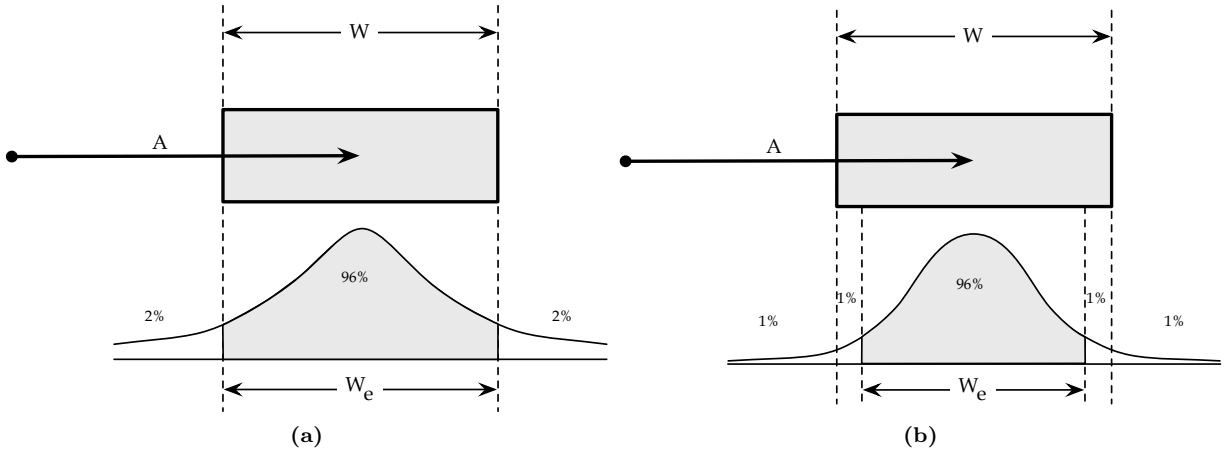



**Figure 2.3:** *Method proposed by MacKenzie in [74] to adjust the target's width. The plots represents the distribution of the endpoints for each acquisition task. They follow a normal distribution. (a) If the number of errors is greater or equal than 4% then $W = W_e$, (b) otherwise $W_e$ accounts for the 96% of the endpoints closer to the center of the target, $W_e = 4.133\sigma$.*

Fitts' in its original experiments allowed up to 4% of erroneous trials. If higher values are obtained, he stated that the task is not well designed or the level of difficulty is too high. In addition, Crossman and Goodev in [28] and Klapp in [62] showed that Fitts' law does not hold neither for low ID values. Their analysis showed that the mean selection time reached an asymptotic lower bound as the ID decreases. While the effective width does not provide any solution for tasks with high ID values, they typically result in more than 4%

of erroneous trials, it does for low ID values. If the low ID is due to increased target size, at some point the standard deviation of the endpoint of the movements will also reach an asymptotic lower bound. In these situations $W_e$ provides a better estimation of the endpoint tolerance determined by $W$. However, the main issue about this approach for the effective width computation is that we cannot estimate the performance of a task, we require to perform the task to model the performance.

Zhai et al. in [137] studied the bias introduced when providing different instructions to users. They explored how users trade-off speed and accuracy and how $ID_e$ can account for it. They described two different speed-accuracy trade-off layers, one defined by the task, and a second one determined by the user's behavior. In the original Fitts' law formulation user's behavior is explained by $a$ and $b$, but $ID_e$ also accounts for user's behavior, which may result in a better data fit.

Zhai observed differences for each condition on $MT$ when providing different instructions to users (e.g. "Perform as accurately as possible and do not worry about time or speed"; "as accurate as possible and as fast as possible"). The more accuracy is required the greater was $MT$. They performed two different regression analyses; grouping the trials among the different instruction's sets or considering all the trials together. The correlation between the selection time and the ID when considering the groups separately was better when using $ID$ rather than $ID_e$. In contrast, when considering all the groups together the correlation was significantly better using $ID_e$. The effective target width better explains the bias introduced by the speed-accuracy trade-off, although it was not totally explained.

### Noise

The distribution of the movement endpoints, as stated earlier, follow a normal distribution. The human motor system is unable to perform perfect aimed movements and introduces endpoint variability. This endpoint variability is considered noise and is referred as *neuromotor noise* [119]. Neuromotor noise is already considered in Fitts' law as $W$ is the measure of the allowed distribution of the endpoint of the movement.

However, in addition to the neuromotor noise, input devices introduce additional noise. Jagacinsky and Monk [58] explored how Fitts' law formulation can be modified to include the noise of input devices. They solution relied on subtracting the mean of the noise distribution ($W_0$) (considered as white Gaussian noise) to $W$. As the noise increases the index of difficulty also increases, which can be resembled as the width of the object shrinks. Equation 2.6 shows the formulation provided which resulted in a better data-to-model fit in their experiments.

$$ID = \log_2\left(\frac{2A}{W - W_0}\right) \tag{2.6}$$

### Latency

In human-computer interaction the communication between the user and the system is accomplished through input and output devices. This communication often introduces latency due to (a) the processes involved, (b) low refresh rate for the output devices and (c) limited input device sampling.

MacKenzie and Ware in [77] explored how latency effects can be introduced in Fitts' law formulation. Their experiments showed that task performance is reduced if the frame rate decreases or if the latency introduced by the device sampling increases. The effect was stronger for the latency introduced by the device sampling than the latency introduced due to low frame rates

The resulting model was Equation 2.7. The lag only affects the slope ($b$ in Equation 2.5) and it increases linearly if the lag increases. If there is no lag Equation 2.7 reduces to Equation 2.5.

$$MT = a + (b + c\,\mathrm{LAG})ID \tag{2.7}$$

Ware and Balakrishnan in [125] in reference to Equation 2.7, stated that $a$ include the initial reaction time plus the selection confirmation, $b$ represents the human processing time required to perform a corrective movement and $c\,\mathrm{LAG}$ represents the impact of latency during the corrective movement.

Equation 2.7 is useful when the latency is a known factor which varies among experiments and when comparing different studies with different latency conditions. If the latency is constant among the experiments, the $b$ factor can absorb the term $c\,\mathrm{LAG}$, resulting in the original Fitts' law formulation.

### Fitts' law extensions

Originally, Fitts' law was applied only for 1D acquisition tasks, but along the years a huge number of studies support it usage for 2D [75, 106, 84, 1] and 3D [125, 85, 47] acquisition tasks. However Fitts' law extensions only apply for simple scenarios; the user has to perform a 1D task embedded in a 2D or 3D space and point to a simple target (rectangular or spherical).

The main challenge of extending Fitts' law to higher dimensional tasks is how to compute $W$ and $A$ and determine whether changes in the formulation are required or not. Increasing the dimensions also increase the degrees of freedom and the muscle groups involved in the acquisition task.

The extension of $A$ to higher dimensional tasks is easier, as the Euclidean distance between the starting point and the target's midpoint is still a valid measure. However as we increase the number of dimensions, computing the size of the target ($W$) becomes more difficult. In 1D, $W$ is clearly defined, but for 2D and 3D is not longer the case; $W$ depends on the shape of the target and the approach angle of the movement ($\Theta$). If the targets considered are only circular (or sphere like) the 1D constraint still holds as the $W$ is directly related to the radius of the target.

In the literature only rectangular-shaped axis-aligned targets are considered, which allows for the interaction with words, buttons and simple 3D shapes. In order to compute the size for higher dimensional targets, additional measures have to be considered, typically the width (W), height (H) and depth (D) of the target are considered [74, 47]. In a 2D scenario, the width of the target is the dimension more aligned with the direction of the acquisition movement (see Figure 2.4) and the height is the remaining dimension [74]. For a 3D scenario, the same rule applies to determine the width of the target, but it is not clear how to determine the other two dimensions. Grossman and Balakrishnan [47] defined the height considering a up-down axis (Y-axis) and the depth was the remaining dimension, but no additional discussion was provided. However, as we will see, it is not relevant how the height and the depth are defined.

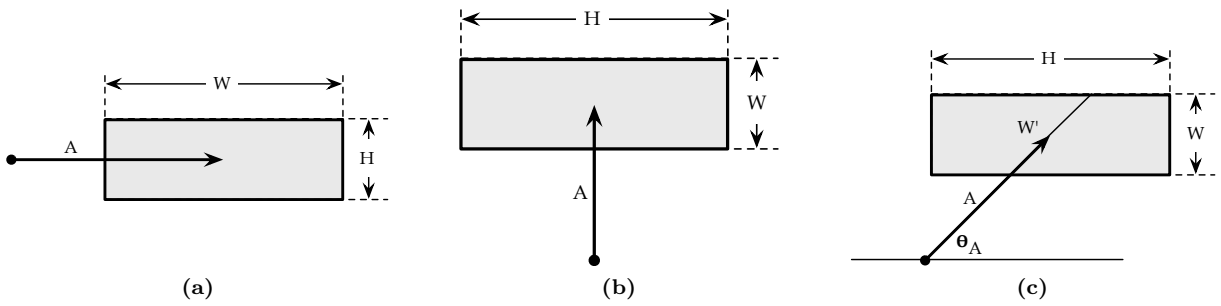



**Figure 2.4:** *Roles of target width and target height for 2D acquisition tasks proposed by MacKenzie in [75]. (a, b) Target width is aligned with the direction of movement. (c) If there is no perfect alignment alternatives rely on considering the width of the target along the approach vector ($W'$).*

Extensions of Fitts' law to higher dimensional tasks follow two different strategies, (a) compute a new target size considering the dimensions of the target ($W_c$) or (b) introduce all the dimensions into Fitts' law formulation.

MacKenzie in [75], proposed two different approaches to compute $W_c$ for 2D acquisition tasks, obtaining good data-to-fit results. First, considering that the smaller dimensions restricts the movement, $W_c$ was determined by the minimum of both dimensions $W_c = min(W, H)$. Second, considering $W_c = W'$, being $W'$ the distance of the segment delimited by the approach axis and the target (see Figure 2.4c).

However as stated by Accot and Zhai in [1], when considering $W_c = min(W, H)$, several target configurations can have the same $W_c$ which potentially can have different IDs. Moreover, considering the minimum does not account for the movement direction. Nevertheless it is still a good approximation and it is easy to extend to 3D acquisition tasks: $W_c = min(W, H, D)$.

Murata [84] improved the computation of $W_c$ by computing its effective value (as MacKenzie did for 1D acquisition tasks [74]). They compute the effective with in the axis parallel to the approach angle $W_{xe}$ and in the perpendicular axis $W_{ye}$ (see Figure 2.5). Considering both axes, they compute $W_c$ as the $min(W_{xe}, W_{ye})$. This approach can also be extended to 3D acquisition tasks.

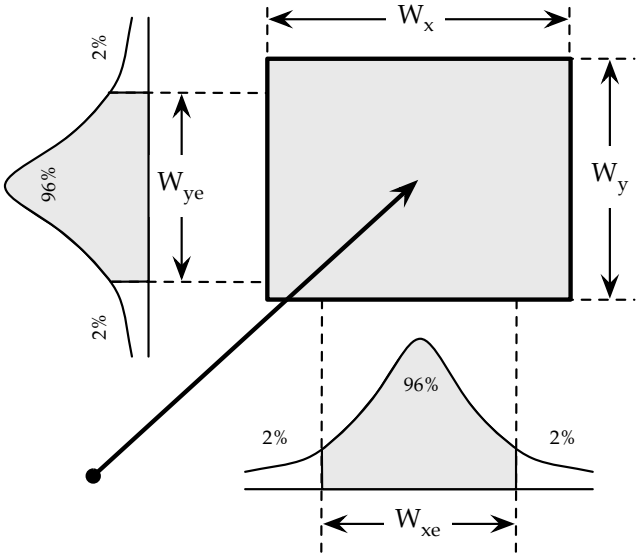



**Figure 2.5:** *Computation of the effective target size for two dimensional tasks, as proposed by Murata [84]. It considers the endpoint variability for each dimension separately.*

The second approach to extend Fitts' law to higher dimensional tasks is based on considering the dimensions involved in the acquisition task separately and introducing them into Fitts' law formulation. The first model was suggested by Crossman [27] (see Equation 2.8). Crossman observed that the restriction introduced by the height of the target also affected movement time, but in a lesser grade.

$$MT = a + b \log_2 \left( \frac{2A}{W} \right) + c \log_2 \left( \frac{2A}{H} \right) \tag{2.8}$$

Later, Accot and Zhai in [1] proposed an improved version employing only one logarithmic term (see Equation 2.9). The $\eta$ is an additional parameter and varies approximately in the range of $[1/3, 1/7]$.

$$MT = a + b \log_2 \left( \sqrt{\left(\frac{A}{W}\right)^2 + \eta \left(\frac{A}{H}\right)^2 + 1} \right) \tag{2.9}$$

Compared to Shannon's formulation of Fitts' law using $W_c = min(W, H)$, the model proposed by Accot and Zhai only obtains a better data-fit when $W \geq H$. However, the better fit can be explained due to the fact that when $W \geq H$, $W_e$ is typically smaller than $W$. Nevertheless, they showed how the computation of $W_c$ as $min(W, H)$ breaks when $H < W$ (remember that the movement is aligned with $W$). They observed that increasing $W$ results in performance improvements although $W_c$ remained constant (see Figure 2.6).

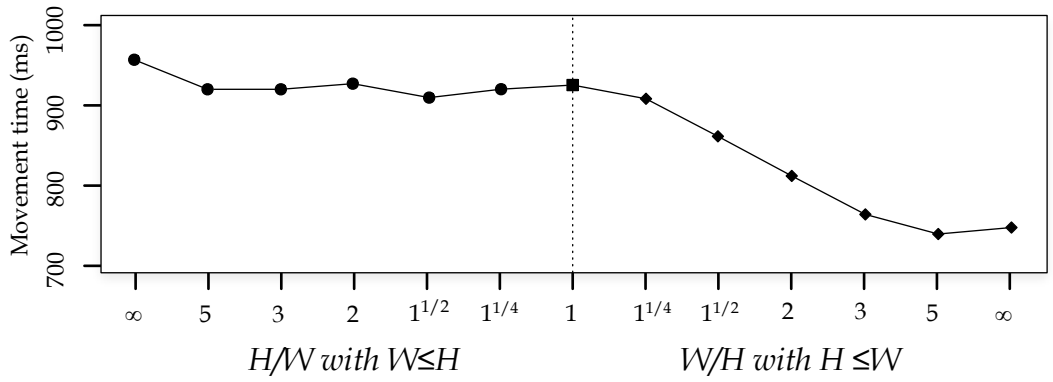



**Figure 2.6:** *Performance plot for different $W$ and $H$ configurations (same $W_c$) obtained by Accot and Zhai in the experiment presented in [1]. The computation of $W_c$ as* $\min(W, H)$ *breaks when $W > H$.*

Grossman and Balakrishnan in [47] applied a similar approach to introduce a new ID formulation for 3D acquisition tasks (see Equation 2.10). In addition to consider the three directions of movement (W,H,D), all three components are weighted by an additional parameter $f_{W,H,D}(\Theta)$. This parameter is determined experimentally and depends on the approach angle $\Theta$.

$$ID_{WtEuc\Theta} = \log_2 \left( \sqrt{f_W(\Theta)\left(\frac{A}{W}^2\right) + f_H(\Theta)\left(\frac{A}{H}^2\right) + f_D(\Theta)\left(\frac{A}{D}^2\right)} + 1 \right) \tag{2.10}$$

They performed a user evaluation comparing its model with four additional ID formulations (see Equations 2.11). These formulations were obtained applying the models proposed by MacKenzie [75] and Accot [1].

$$
\begin{aligned}
ID_{min} &= \log_2\left(\frac{A}{min(W,H,D)}+1\right)\\
ID_{Wtmin} &= \log_2\left(\frac{A}{min(W,\alpha H,\beta D)}+1\right)\\
ID_{WtEuc} &= \log_2\left(\sqrt{\left(\frac{A}{W}^2\right)+\alpha\left(\frac{A}{H}^2\right)+\beta\left(\frac{A}{D}^2\right)}+1\right)\\
ID_{Wtmin\Theta} &= \log_2\left(\frac{A}{min(f_W(\Theta)W,f_H(\Theta)H,f_D(\Theta)D)}+1\right)
\end{aligned}
\tag{2.11}
$$

The results showed that $ID_{WtEuc\Theta}$ ($R^2 = 0.912$) and $ID_{WtEuc}$ ($R^2 = 0.911$) provide the better explanation of the data obtained. Followed by $ID_{Wtmin\Theta}$ ($R^2 = 0.878$), $ID_{Wtmin}$ ($R^2 = 0.878$) and $ID_{min}$ ($R^2 = 0.799$). Although their results showed a slightly better data-to-model fit with its proposed formulation, it not clear whether the improvements of the fit are due to the additional degrees of freedom. The simplest model is still able to account for the 80% of the data.

Interestingly, during the evaluation they observed that acquisition tasks requiring forward and backward movements were significantly slower than left and right movements and were more prone to errors. This observation matches with the observations of Ware and Balakrishnan [125], which also showed higher error rates for movements involving depth perception. They justify performance and error rates because bigger muscle groups are involved, and due to limitations in the visual feedback provided. Visual feedback is crucial for virtual reality interaction, but depth perception available in virtual reality setups is limited. A more detailed discussion can be found in Section 3.1.

### Virtual pointing tasks

Until now we have considered only acquisition tasks where the user has to reach the target with its hand (or with a stylus). Wingrave and Bowman [131] showed that Fitts' law still holds for virtual pointing tasks in VEs. Instead of considering the size of the target, they observed that $W$ is related to the visual size of the target and $A$ to the amplitude of the movement considering the angle covered by hand rotations. Poupyrev et al. in [97] went further, by defining the size of an object $W$ according to the vertical $\varphi$ and horizontal $\phi$ angles an object occupies in the user's field of view. They also consider how occlusion between objects diminishes $W$. In Section 3.1.2 we will go deeper on how occlusion alters $W$.

### 2.1.2 Optimized initial impulse model

Fitts' law only accounts for movement time according to the target's characteristics and the empirical parameters $a$ and $b$. However it does not provide any insight on how subjects perform acquisition tasks. Different human performance models have been proposed to explain the logarithmic speed-accuracy trade-off defined by Fitts' law.

The human movement model which better accounts for Fitts' Law is the *optimized initial impulse model* proposed by Meyer et al. [81]. It has its origins in the model proposed by Woodworth [135] where acquisition tasks are subdivided in a two-step movement phases. In the first phase, called ballistic phase, a fast and inaccurate movement is made towards the target. If the target is not acquired, then, iterative slow correction movements are executed until the target is acquired (see Figure 2.7).

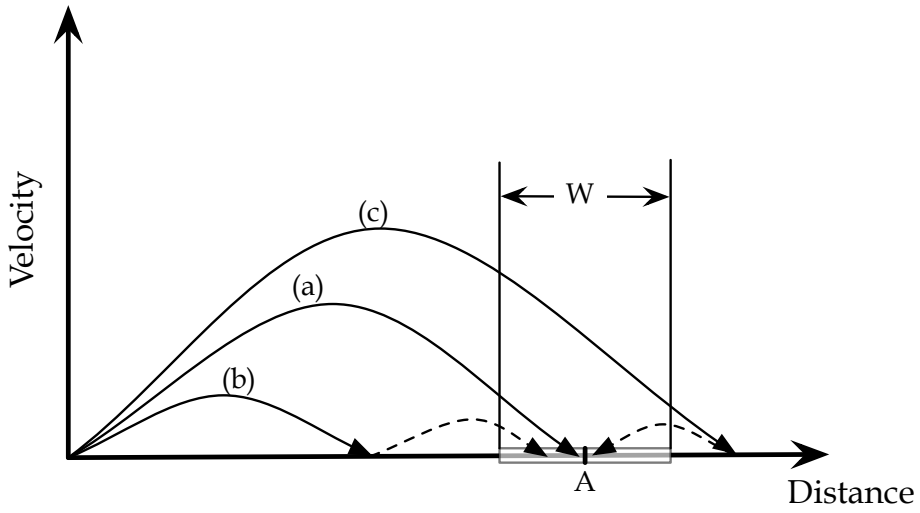



**Figure 2.7:** *Following the optimized initial impulse model, after the ballistic movement: (a) the target might be selected, (b) under shooted or (c) over shooted. For situations (b) and (c) subsequent corrective movements are required.*

*Ballistic movements* are intended to cover the whole distance towards the target, but due to limitations of the human motor system, the endpoint of the movement is randomly distributed over the desired endpoint [100]. This variability depends on the muscle groups involved in the movement [18], bigger muscle groups introduce higher variability than smaller ones. On the other hand, *corrective movements* are slow movements where precision is the main requirement. They are needed when the target is undershot or overshot.

In his experiments Meyer et al. [81] defined the speed-accuracy ratio for ballistic movements. They exposed that the standard deviation of the movement's endpoint is proportional to the average of the movement speed ($D/T$),

$$S = k\frac{D}{T} \tag{2.12}$$

Where $S$ is the standard deviation of the endpoint, $D$ is the distance covered and $T$ is the movement time. This relationship determines te trade-off between the speed of the movement and the precision needed. Faster movements result in higher endpoint variability, thus requiring more corrective movements to acquire the target. On the other hand, slow movements result in smaller endpoint variability and thus requiring fewer corrective movements.

Experimental observations showed than given a task, users minimize movement times by balancing the speed of the ballistic movement in relation with the required corrective submovements [119]. Other researchers have studied the velocity profiles of acquisition movements taking into account the index of difficulty of the task [73]. Their results also match with the *optimized initial impulse model*, the profiles showed two different movement phases (see Figure 2.8), few fast movements followed by a sequence of slower movements.

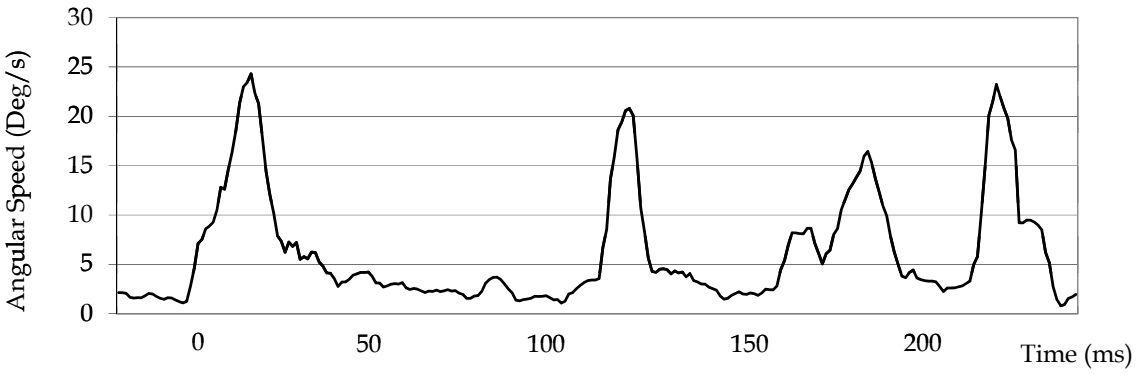



**Figure 2.8:** *Example of velocity profile of a 3D acquisition task using raycasting selection; the ballistic and corrective phases of the movement are clearly visible.*

MacKenzie et al. [73] concluded that velocity profiles depend on $W$ and $A$ and not only on the ID. During the acceleration phase, $A$ determines the maximum movement speed, regardless of target size. In contrast, $W$ determines the deceleration phase and the corrective movements required to acquire the target. The higher is $A$ the higher is the peek velocity, and the smaller is $W$ the longer is the deceleration phase. In other words, the shape of movement trajectories is directly influenced by the required precision of the movement endpoint.

Walker et al. [106] introduced two additional *movement* phases: the initiation phase and the verification phase. The initiation phase covers the time required until the user starts the acquisition process. During the verification phase the user ensures that he has acquired the desired target and confirms the selection. Both phases did not depend on the movement at all, and only introduce additive effects.

## 2.2 3D object selection techniques

Since the first 3D object selection techniques appeared [82], many researchers have focused on improving their usability. Typically small improvements are introduced at each iteration step by improving performance, precision or user comfort. Most selection techniques are related to other techniques and share several features.

Bowman et al. [11] proposed a classification based on task decomposition, where selection techniques are decomposed into subtasks, and classified according to them (see Figure 2.9). They state that a selection technique has to provide means to indicate an object (*object indication*), a way to confirm its selection (*confirmation of selection*) and visual, haptic or audio feedback to guide the user during the selection task (*feedback*). However, their classification does not consider that the purpose of the feedback may vary during indication and confirmation tasks. While feedback guides user's actions during indication tasks, it has to show if the selection was successful in confirmation tasks. In addition the feedback is highly coupled with the object indication which introduces redundancy into the classification.

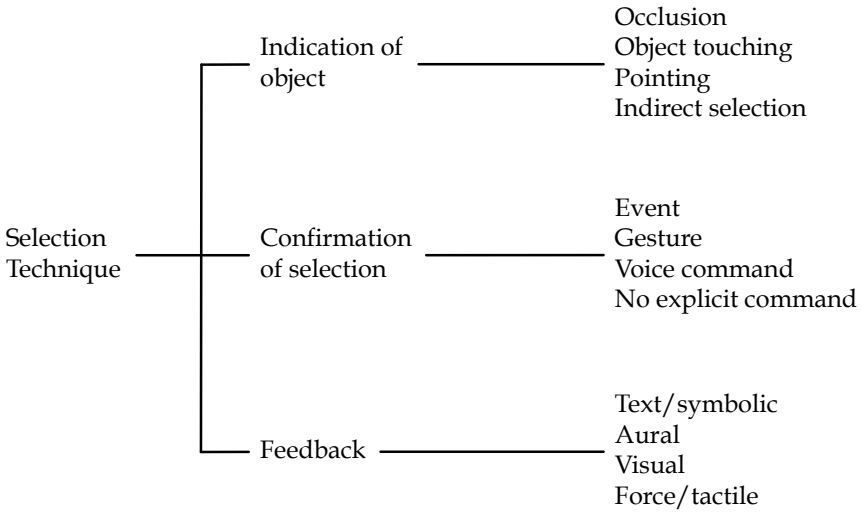



**Figure 2.9:** *Classification of selection techniques by task decomposition. Bowman et al. [11]*

Poupyrev et al. in [95] proposed an alternative classification based on interaction metaphors; each technique can be classified considering real world interactions (see Figure 2.10). The classification have several levels. The first level considers selection techniques as exocentric and egocentric. For exocentric techniques (god's-eye viewpoint) the user interacts from outside the environment, while for egocentric (first person viewpoint) the user interacts from inside the environment. In the second level, egocentric metaphors are further subdivided into virtual hand and virtual pointer metaphors and exocentric metaphors are subdivided into

World-in-miniature and automatic scaling. In contrast to Bowman's classification, Poupyrev classification's does not to consider small technique differences like feedback or selection confirmation mechanisms.

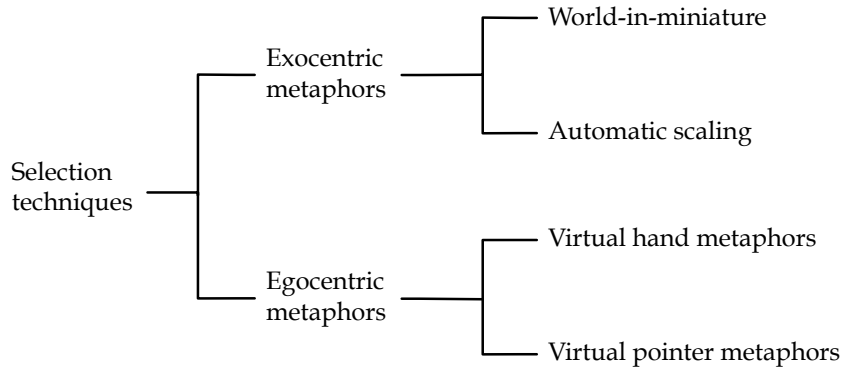



**Figure 2.10:** *Classification of selection techniques by interaction metaphor. Poupyrev et al. [95]*

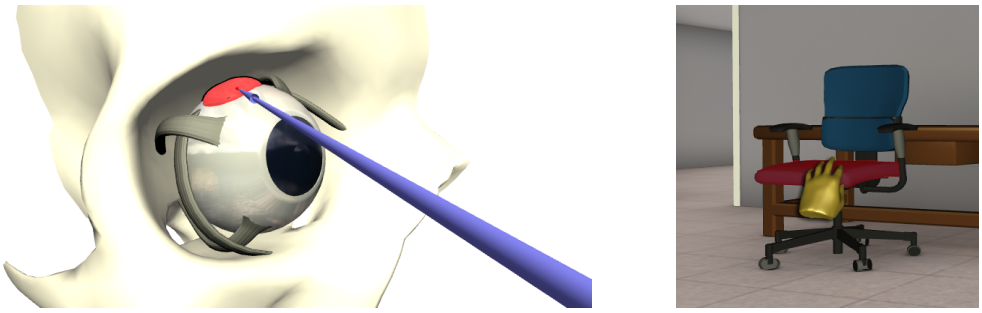
Notice the similarities between the classification for egocentric metaphors in Poupyrev's classification and the *indication of object* in Bowman's. Both classifications can be combined by considering that the object indication subtask in Bowman's classification can be further subdivided into exocentric and egocentric metaphors as in Poupyrev's classification.

Both taxonomies provide a broad view of selection techniques but consider a small number of design variables. Additional design variables relate to the characteristics of the selection tool and how the user is able to control it. Given a selection technique, it can be characterized considering how the user controls the 3D selection tool, the area of influence of the tool, whether the control-display ratio is modified or not, the mapping between the motor and visual space and automatic processes which guide user's actions.

For example, consider raycasting and virtual hand selection techniques (see Figure 2.11). For raycasting, the 3D selection tool is a 3D ray, controlled by the hand position and orientation, being the area of influence a segment determined by the ray, the control-display ratio is 1:1, the motor and visual space are coupled, no automatic process are provided and the feedback consists in the visualization of the 3D ray.

In contrast, for virtual hand, the 3D selection tool is a virtual hand avatar, controlled by the hand position, the area of influence is the volume defined by the virtual hand, the control-display ratio is 1:1, the motor and visual space are coupled, no automatic process are provided and the feedback provided is the visualization of the 3D virtual hand avatar.

According to our classification, the only differences appear in the selection tool and how the user is able to control. Obviously, the visual feedback differs as the visual representation of both tools also differ.



(a) (b)

**Figure 2.11:** *(a) Raycasting selection, the first object intersected by the ray is selected. (b) Virtual hand selection, the object intersected by the virtual hand is selected.*

In the rest of the section we describe the characteristics considered and how current selection techniques can be classified according to them.

### 2.2.1 Selection tool

Three-dimensional object selection techniques requiring physical interaction provide the user with a selection tool and define how user's actions are mapped onto the selection tool. The most common selection techniques employ rays, 3D cursors or simple 3D shapes. Once the user places conveniently the selection tool, an intersection or a proximity test is performed against the 3D environment. As we consider only the selection of single objects, if more than one object is selected, the selection technique has to provide mechanisms to disambiguate the result.

#### Virtual prop

Referring to Poupyrev's classification, virtual hand and virtual pointing techniques are determined by the selection tool. Virtual hand techniques rely on 3D cursor tools while virtual pointing techniques employ 3D rays (or cones). A classification of existing selection techniques against their selection tool is presented in Table 2.1.

For the majority of techniques, the selection tool does not change its shape during the interaction process, although there are techniques which provide mechanisms to alter the shape of the selection tool. For example, the Bubble cursor [120], which employs a sphere-like selection tool, automatically increase the radius of the sphere to reach object closest to it. Another example is aperture selection [40] which allow to manually adjust the size of the selection tool.

In addition to 3D Cursors and 3D rays other techniques employ alternative solutions. For example, the depth ray and the lock ray [48] are hybrid solutions which combine a ray with a 3D cursor constrained along the ray. When the selection trigger is activated the object intersected by the ray and closest to the 3D cursor is selected. Another example is the iSith technique [93] which employs two selection rays. Both rays determine a 3D point which is used to perform a proximity test against the scene. Finally, the flexible pointing [90] allows the user to bend the selection ray to select fully or partially occluded objects.

| **Metaphor** | **Selection Tool** | **Technique** |
|---|---|---|
| Virtual Hand | Hand avatar | Virtual-hand |
| | | Go-go [96] |
| | Adjustable sphere | Bubble-Cursor [120] |
| | Axis aligned box | Silk Cursor [136] |
| Virtual Pointing | Ray | RayCasting [82] |
| | | Direct Image plane [69] |
| | | Flow Ray [48] |
| | | Eye-gazed selection [115] |
| | | Occlusion Selection [94] |
| | | One-Eyed Cursor [126] |
| | | Two-handed Pointing [83] |
| | | PRISM [42] |
| | | Adaptive pointing [63] |
| | | IntenSelect [32] |
| | | Smart Ray [48] |
| | Cone | Sticky Ray [112] |
| | | Flashlight [56] |
| | | Sense Shapes [89] |
| | | Shadow Cone Selection [111] |
| | | Probabilistic Pointing [105] |
| | | Enhanced Cone Selection [110] |
| | Adjustable cone | Aperture [40] |
| | Two rays | iSith [93] |
| | Curved ray | Flexible Pointing [90] |
| Hybrid | Ray & 3D cursor | Depth Ray [48] |
| | | Lock Ray [48] |

**Table 2.1:** *Selection techniques classified according to their interaction metaphor and their selection tool.*

### Tool Control

Once defined the shape of the selection tool, the selection technique also determines how the user is able to control it. As virtual reality systems are equipped with tracking devices, the system is able to track users' actions and map them into the selection tool.

The user can control a 3D cursor by moving its dominant hand (virtual hand), control a virtual ray with the hand's position and wrist's orientation (raycasting) or define a virtual ray emanating from the head going along the hand's position, as in occlusion selection [94]. Other techniques determine the ray's orientation by the vector defined by both hands [83], bending the ray given the position of the non-dominant hand [90], or using eye tracking [115].

Virtual pointing techniques whose tool origin is located in the hand position will be referred as *hand-rooted techniques* and techniques having the origin at the eye position will be referred as *eye-rooted techniques*. An analysis of its implications is detailed in Section 3.1.

The control mechanism also determines the degrees of freedom (DoF) required to control the selection tool. A 3D cursor controlled by the hand position has three DoFs (one for each dimension), while a virtual ray controlled by hand position and orientation has five DoF, three to determine the ray's origin and two for the ray's orientation.

However, some DoFs are more relevant than others. For example, when selecting objects placed far away form the user with a virtual ray, selection can be accomplished without changing the ray's origin. Changes in the origin of the virtual ray result in relatively small variations of the ray's endpoint. Although a virtual ray has up to five DoFs in most situations it is enough using only two DoFs. An exception is the smart ray [48] which was conceived to select semitransparent objects in cluttered environments. To select an object, the user has to point to it from several directions. The depth ray and the lock ray [48] require an additional DoF, as the selection tool is a 3D cursor constrained along a 3D ray.

Two-handed techniques also increase the number of DoFs the user has to control. In Two-Handed pointing [83] the virtual ray is constrained by the position of both user's hands resulting in six positional DoFs. A variation of two-handed pointing is flexible pointing [90] which employs a Bézier curve as a selection tool. Two control points are determined by the position of the user's hands (six DoFs) and the remaining control point is computed regarding the orientation of both hands (four DoFs). Another two-handed technique is the iSith technique [93] in which the user has to control two virtual rays, requiring up to ten DoFs. In contrast to raycasting, the origin of the rays play an important role, as the selected object is determined by the intersection of both rays.

Table 2.2 shows the maximum degrees of freedom and the effective DoFs required for each selection technique. In Section 2.3.3 we discuss the implications of the number of DoF in usability.

| **Origin/Location DoFs** | **Orientation DoFs** | **Effective DoFs** | **Technique** |
|---|---|---|---|
| $(x, y, z)$ | None | $(x, y, z)$ | Virtual Hand<br>Go-go [96]<br>Silk Cursor [136]<br>Bubble Cursor [120] |
| | $(\theta, \varphi)$ | $(\theta, \varphi)$ | RayCasting [82]<br>Sticky Ray [112]<br>Flashlight [56]<br>Sense Shapes [89]<br>Shadow Cone Selection [111]<br>Probabilistic Pointing [105]<br>PRISM [42]<br>Adaptive Pointing [63]<br>Enhanced Cone Selection [110]<br>IntentSelect [32]<br>Flow Ray [48] |
| | | $(z, \theta, \varphi)$ | Depth Ray [48]<br>Lock Ray [48] |
| | | $(x, y, z, \theta, \varphi)$ | Smart Ray [48] |
| | None [1] | $(x, y)$ | Direct Image Plane [69] |
| | $(x_n, y_n, z_n)$ | $(x, y, z)$<br>$(x_n, y_n, z_n)$ | Two-Handed Pointing [83] |
| | $(x_n, y_n, z_n, \theta_n, \varphi_n)$<br>$(\theta, \varphi)$ | $(x, y, z, \theta, \varphi)$<br>$(x_n, y_n, z_n, \theta_n, \varphi_n)$ | Flexible Pointing [90] |
| $(x, y, z)$<br>$(x_n, y_n, z_n)$ | $(\theta, \varphi)$<br>$(\theta_n, \varphi_n)$ | | iSith [93] |
| $(x_e, y_e, z_e)$ | $(x, y, z)$ | $(x, y)$ | One-Eyed Cursor [126] |
| | | | Occlusion Selection [94] |
| | | $(x, y, z)$ [2] | Aperture Selection [40] |
| | $(\theta_e, \varphi_e)$ | $(\theta_e, \varphi_e)$ | Eye-gazed Selection [115] |

**Table 2.2:** *Selection techniques classified according to how the user controls the selection tool. $(x, y, z, \theta, \varphi)$ refers to the dominant hand position, and yaw and pitch angles. $(x_n, y_n, z_n, \theta_n, \varphi_n)$ refers to the user's non-dominant hand and $(x_e, y_e, z_e, \theta_e, \varphi_e)$ to the user's eye. The coordinate system is user-centered.* [1] *The orientation of the selection ray is determined by the screen normal.* [2] *The third DoF allows to adjust the apex angle of the selection cone.*

### Disambiguation Mechanisms

As discussed above, one of the common strategies to improve selection performance relies on volumetric selection tools [56, 111]. Volumetric tools are prone to indicate more than one object at once, specially in dense environments [120]. In these situations, the selection technique has to provide mechanisms to disambiguate the selection. Disambiguation mechanisms can be classified into three groups: manual, heuristic and behavioral.

For *manual approaches*, the user has to decide, among the indicated targets, which target is the desired one. Manual approaches provide the maximum flexibility at the expense of increasing the cognitive load of the user due to additional selection steps. The simplest solution consists on using a button to cycle among all indicated objects [54], but obviously, it does not scale well if too many objects are selected.

A better solution, proposed by Grossman et al. in [48] is the flow ray. Although it behaves as raycasting, all objects intersected by the virtual ray are selected. In a second step, selected objects are displayed in a list or a pie menu, thus letting the user select the desired one. Other options may require more complex interaction like iSith [93], which employs two selection rays; the object closest to both rays is selected.

Grossman et al. in [48] proposed another extension for raycasting selection called depth ray. In addition to the virtual ray, it provides a 3D cursor constrained along the virtual ray. The 3D cursor is controlled by pulling the hand forwards and backwards; the object intersected by the virtual ray and closest to the 3D cursor is the selected one.

The second group of disambiguation techniques employ *heuristics* to determine which is the object the user wants to select. Objects are ranked according to a heuristic and the higher ranked object is selected. The easiest approach considers the distance between objects and the center of the selection volume, as flashlight selection does [56]; the object closest to the axis of the selection cone is selected.

Schmidt et al. [105] extended this naive approach by proposing probabilistic pointing-based selection algorithms. All methods are based on computing a weighted pixel count of a circular area surrounding a 2D cursor. They considered three alternatives:

1. Each pixel has the same relevance (similar to [89]).
2. For each object, the pixel relevance is inversely proportional to the distance of the pixel to the barycenter of the visible portion of the object.
3. Use hierarchical information of the model to weight the pixel's relevance.

The first approach, obviously presents issues when selecting small objects surrounded by bigger ones, and the third is problematic as we need the hierarchical information of the model which may be unavailable. The user study presented in [105], showed that the second approach worked better. Users, when pointing to an object, tend to move the selection tool towards its center. However as they state, concave objects may pose some difficulties as the barycenter is not located inside the object. They report that it is less sensible to noise but inefficient for selecting small items.

Finally, *behavioral approaches* take into account user's actions prior selection confirmation. Instead of applying an heuristic when the user triggers the selection, they continuously rank objects during the selection process, gathering statistical information, such as IntenSelect [32], enhanced cone selection [110] and sense shapes [89].

The data they considered is related to: the time the object is inside the selection volume, its distance to the center of the volume, the number of times the object has entered the volume, the objects's visible pixels in the volume frustum and the average or minimum pixel distance to the center of the volume's center. These approaches are particularly useful for selecting moving targets; as if we track the moving target with the selection tool its selection weight will increase with respect to static objects.

In summary, manual approaches provide total control to users at the expense of increased cognitive load or additional selection steps. On the other hand, heuristic and behavioral techniques do not introduce any further selection steps, but as they are not completely accurate, they might result in unwanted selections and thus require the user to perform another selection. Table 2.3 shows the disambiguation mechanisms provided for each of the considered selection techniques.

### 2.2.2 Control display ratio

The Control-Display ratio (CD ratio) determines how translations and rotations of the input device ($x$) are transfered to the selection tool ($X$): CD ratio $= \Delta x / \Delta X$. In a system where there is an isomorphism between the pointing device and the display, the CD ratio is one, which means than the movement of the pointing device is the same as the movement of the selection tool in the virtual environment. When the CD ratio differs from one, there is an anisomorphism, the movement is scaled (greater than one) or downscaled (lower than one). A related concept is the Control-Display gain which provides a relationship between the velocity of the physical device and the selection tool.

| Disambiguation | Technique |
|---|---|
| N/A | Virtual-hand |
| | Go-go [96] |
| | Silk Cursor [136] |
| | RayCasting [82] |
| | Direct Image plane [69] |
| | Eye-gazed selection [115] |
| | Occlusion Selection [94] |
| | One-Eyed Cursor [126] |
| | Two-handed Pointing [83] |
| | Flexible Pointing [90] |
| | PRISM [42] |
| | Adaptive pointing [63] |
| Manual | Flow Ray [48] |
| | Depth Ray [48] |
| | Lock Ray [48] |
| | iSith (Two Handed) [93] |
| | Shadow Cone Selection [111] |
| | Smart Ray [48] |
| Heuristic | Probabilistic Pointing [105] |
| | Aperture [40] |
| | Sticky Ray [112] |
| | Flashlight [56] |
| | Bubble-Cursor [120] |
| Behavioral | Enhanced Cone Selection [110] |
| | Sense Shapes [89] |
| | IntenSelect [32] |

**Table 2.3:** *Selection techniques classified according to the disambiguation mechanism provided.*

The effect of a constant CD ratio on performance has been extensively explored in the literature but results are still inconclusive [4]. Considering Fitts' law, a constant CD ratio affects the amplitude of the movement and the target size at the same level, keeping the index of difficulty unchanged.

The first 3D object selection technique proposing a change on the CD ratio was the go-go technique [96]. Poupyrev et al. proposed a non-linear mapping for the CD ratio to increase the limited area of reach of virtual hand techniques. They defined two areas considering the distance between the user's hand and its torso. When the distance is smaller than a threshold, the CD ratio is one. But as the distance increases, user's movements are mapped non-linearly to the virtual hand. A factor $k$, where $0 < k < 1$, is used to adjust the non-linear component (see Figure 2.12). In some sense, the go-go technique allows the user to stretch its virtual arm to select objects placed far away.

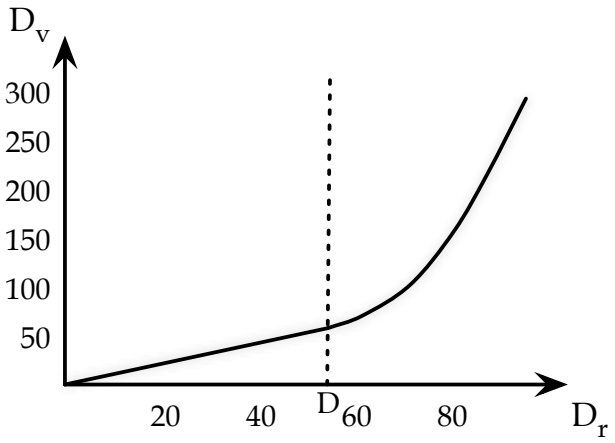



**Figure 2.12:** *Plot of the distance between the user's torso and the user's hand ($D_r$) in comparison to the virtual hand ($D_v$). If the user's hand is close to the user's torso ($D_r < D$) both are coupled ($D_v = D_r$). When the user moves the hand further ($D_r > D$), the virtual hand moves faster than the real hand ($D_v = D_r + k(D_r - D)^2$).*

However, on the downside, precision decreases as the user move its hand further (user's movements are magnified). In addition, although the CD ratio depends on the distance between the user's hand and torso, it may result unnatural and unpredictable. People tend to judge their hand movement mainly on the basis of the on-screen movement of the cursor and adapt their hand movement accordingly [63].

König et al. in [63] classified existing CD ratio based techniques into three groups, according to how they manage the CD ratio: manual switching, target oriented and velocity oriented techniques.

*Manual switching* techniques provide mechanisms allowing the user to manually control the CD ratio. The most common approach is based on reducing the CD ratio when additional accuracy is required. For example, Vogel et al. in [122], proposed the use of gestures to allow the user to switch between isomorphic raycasting and anisomorphinc raycasting with a CD ratio lower than one. Although they obtained higher selection times with their approach (mainly due to mode switches), they got lower error rates than standard raycasting.

The second group, *target oriented* techniques, are based on reducing the CD ratio when the selection tool enters or approaches an object, following a sticky metaphor. Although this approach is useful for the selection of isolated targets, its performance tends to degrade in cluttered environments.

Finally, *velocity oriented* techniques dynamically adjust the CD ratio according to the input device speed. Considering the optimized initial impulse model [81], accuracy can be decreased during ballistic phases (CD ratio higher than one) and increased during corrective

movements (CD ratio lower than one). As a result, during ballistic movements the amplitude of the movement $A$ decreases, while during corrective movements the size of the target $W$ increases. This approach has been widely adopted for 2D mouse interaction.

The PRISM technique, proposed by Frees and Kessler in [41], applied this concept for manipulation and selection tasks. Figure 2.13 shows the plot of how the CD ratio varies according to the speed of the input device. Movements below a minimum speed (MinS) are considered noise and thus ignored. Corrective movements (speeds between MinS and SC) are scaled down, increasing precision. For ballistic movements (speed higher than SC), they applied a 1:1 CD ratio.

However, changes in the CD ratio introduce an offset between the physical device and the virtual selection tool. After a sequence of corrective movements the position of the input device no longer matches the position of the virtual device. In order to solve this issue, when the speed exceeds a maximum speed (MaxS) the CD ratio is increased until the offset is recovered. A fast movement allows the user to recover the accumulated offset.

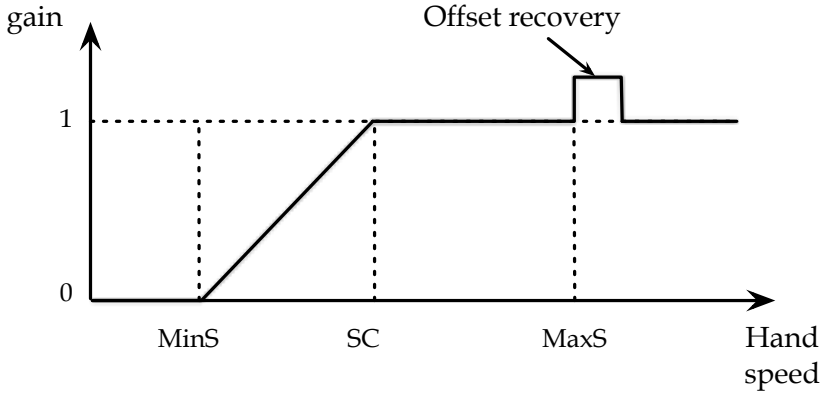



**Figure 2.13:** *Control display ratio function for PRISM.*

A similar approach was applied by König et al. in [63]. They proposed the technique named Adaptive Pointing which is based on a similar mapping as PRISM (see Figure 2.14), but it also takes into account the accumulated offset between the physical device and the virtual selection tool. They introduce to the CD ratio computation the accumulated offset, limiting the maximum offset allowed. Similar to PRISM, their user evaluation showed that Adaptive Pointing applied in combination with raycasting selection result in reduced error rates due to increased precision and slightly improved selection time.

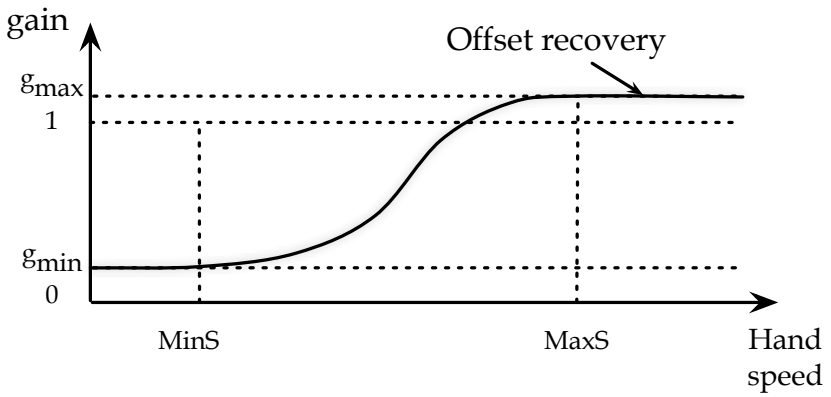



**Figure 2.14:** *Control display ratio function for Adaptive pointing. The influence of the offset is not considered in this plot.*

### 2.2.3 Motor and visual space

Two main spaces are involved during the interaction with an object on immersive virtual environments: the motor and the visual space. The *motor space* (or working space) is the physical space available for the user to operate, which is constrained to the degrees of freedom available and the virtual reality setup. For example, a user controlling a 6 DoFs hand-held device inside a CAVE is constrained by the 6 DoFs of the input device, the CAVE's walls and its own body limitations. In contrast, the *visual space* is the visual representation of the environment perceived by the user, it is independent to the selection technique employed and it is constrained by the field of view of the output device.

In a typical desktop PC setup, the motor space is decoupled from the visual space. The movement of the mouse on a horizontal plane is transformed to the movement of a 2D cursor on a vertical plane (screen). The motor space is thus mapped onto the visual space by a rotation and a translation.

In contrast, for the virtual hand technique both spaces are coupled (there is an absolute mapping); the selection is accomplished by simply *touching* the virtual object. Absolute hand pointing can be carried out with the proprioceptive feedback of the hand. However, when the motor and the visual space are decoupled proprioceptive feedback is broken, in that situation visual feedback is mandatory. The user has to sample the selection tool location with gaze to provide accurate corrective movements [107].

The selection tool together and its control determines which objects within the visual space may afford direct manipulation. In other words, these two components (selection tool and control) transforms the motor space into another space, which defines the scope of the user's actions. We refer to this transformed space as the *control space*. The intersection between the control space and the visual space determines which objects afford direct manipulation.

For the *isomorphic* virtual hand technique, as depicted in Figure 2.15a, the control space matches the motor space; objects outside the motor space are not selectable. In contrast, for virtual pointing techniques, the control space matches the visual space allowing to select objects outside the motor space.

Changes in the CD ratio modify the relationship between the motor and the control space. A CD ratio greater than one scales the control space increasing the area affording direct manipulation at the expense of decreasing the accuracy and vice-versa. For example, the non-linear mapping of the CD ratio provided by the go-go technique [96] increases the control space (see Figure 2.15b), although no uniformly.

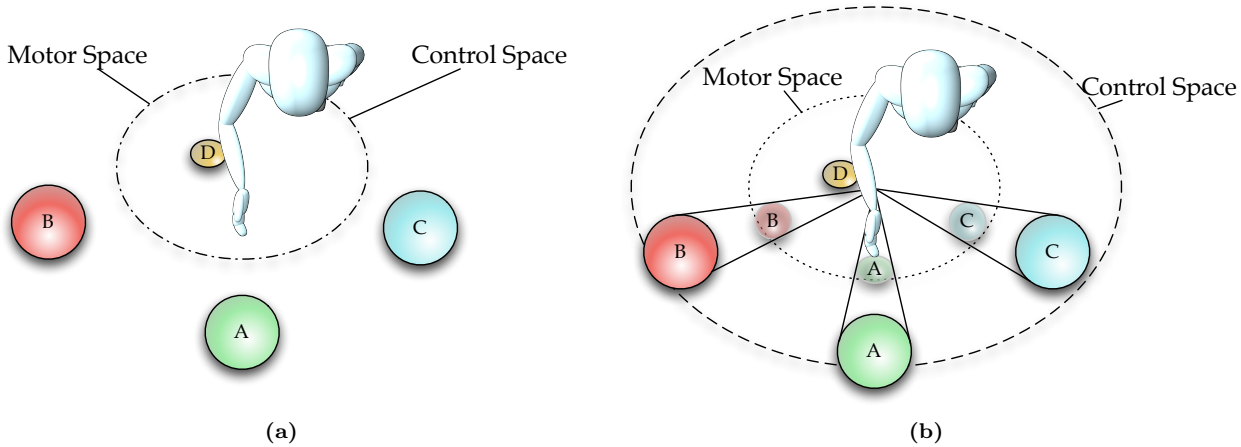



**Figure 2.15:** *Mapping between motor and control space for virtual hand and go-go. (a) In classic virtual hand selection, only objects inside the working space are selectable. (b) The go-go techniques alters the mapping between the motor space and the control space, so the control space matches the visual space. An alternative explanation is that the visual space shrinks to match the motor space.*

The alternative relies on variable CD ratios; reducing the control space when precision is required and vice-versa. However, as in PRISM [42] and adaptive pointing [63] an offset recovery mechanism is required to avoid an excessive decouple between the motor and the control space.

The motor and the control space can also be decoupled by introducing a constant offset and allowing the user to determine the transformation between both spaces (clutching).

For example, in occlusion selection [94], the pointing direction is defined by roughly aligning the hand with the eye position which requires the user to keep its arm extended. Introducing a vertical offset allows the user to keep its hand in a lower position, reducing fatigue levels (see Figure 2.16). Indeed, it is mandatory when using projection based systems as the real hand may occlude the projected content.

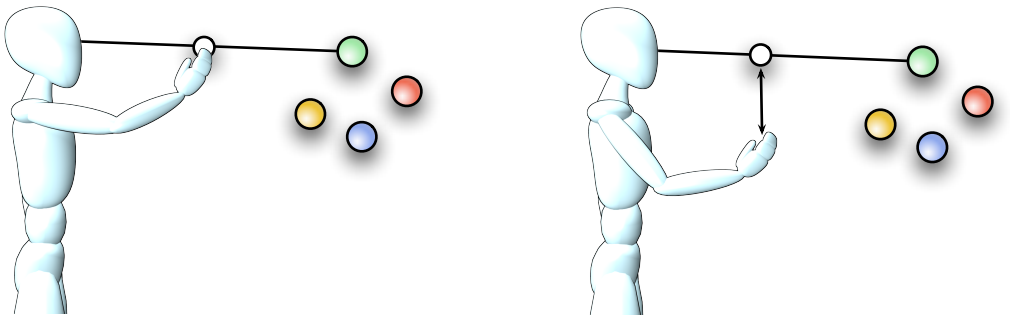



**Figure 2.16:** *Occlusion selection requires the user to keep its hand roughly aligned with the eye position (a). By introducing an offset between the motor and the visual space, the user can keep a more comfortable position (b).*

On the other hand, clutching mechanisms allow the user to relocate the control space accounting for hand repositioning [54], at the expense of introducing an offset between the selection tool and the physical device. Relocating the control space allows to select objects otherwise unreachable (see Figure 2.17) and avoids unwanted user movements to affect the selection tool at the expense of increased user attention. A trigger is needed to enable and disable the clutching. This trigger can be explicit like pressing a button, or implicit like a 2D mouse where the clutching is achieved by lifting the mouse.

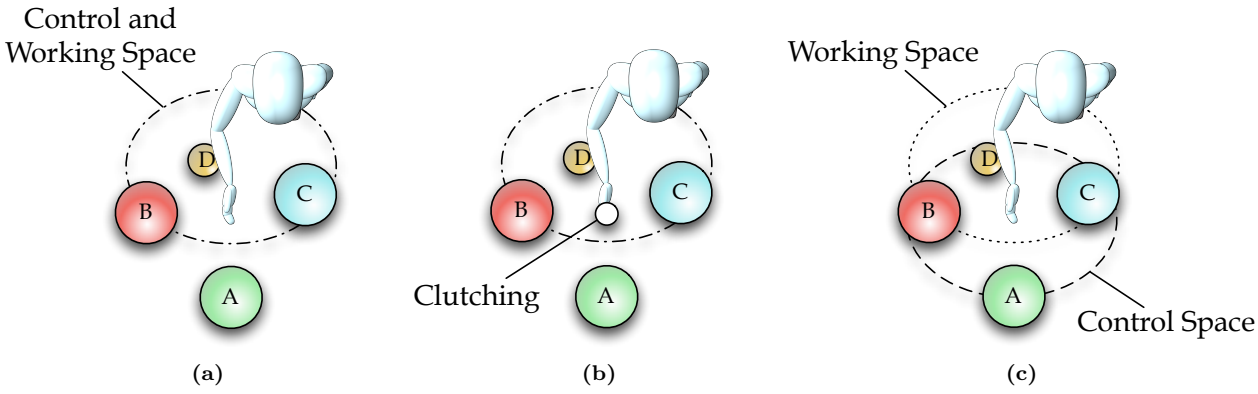



**Figure 2.17:** *A clutching mechanisms allows the user to relocate the control space. (a) The object A is not selectable as it is outside the control space. (b) The user places the 3D cursor and fixes its position with the clutching mechanism. (c) The user returns its hand to the center of the motor space and disengages the clutching mechanism. Now the control space is centered at the previous 3D cursor position and object A is now selectable.*

However, decoupling the motor and visual spaces may result in performance loss. Humans seem to achieve optimum manipulation performance when haptic and graphic displays of objects are superimposed [124], particularly during rotation tasks. However, moderate disparity in orientation between haptic and graphic displays appears to have no significant effect

on object translation. This higher tolerance to object translation with respect to rotations explains why most virtual hand techniques can provide clutching mechanisms while virtual pointing techniques do not.

| **CD Ratio** | **Motor and visual space relationship** | **Technique** |
|---|---|---|
| Isomorphic | Coupled | RayCasting [82]<br>Flexible Pointing [90]<br>Eye-gazed selection [115]<br>Flashlight [56]<br>Sense Shapes [89]<br>Sticky Ray [112]<br>Shadow Cone Selection [111]<br>Probabilistic Pointing [105]<br>Enhanced Cone Selection [110]<br>Flow Ray [48]<br>IntenSelect [32]<br>Smart Ray [48]<br>Lock Ray [48]<br>Depth Ray [48]<br>iSith [93]<br>Two-handed Pointing [83] |
| | Offset | Aperture [40]<br>Occlusion Selection [94] |
| | Offset / Clutching | Virtual-hand<br>Silk Cursor [136]<br>Bubble-Cursor [120]<br>One-Eyed Cursor [126]<br>Direct Image plane [69] |
| Anisomorphic (Distance Based) | CD ratio<br>Offset / Clutching | Go-go [96] |
| Anisomorphic (Velocity Based) | CD ratio | PRISM [42]<br>Adaptive pointing [63] |

**Table 2.4:** *Selection techniques classified according to the control display ratio and how the motor and the visual space can be decoupled. Offset and clutching mechanisms can be provided, or not, by the technique.*

### 2.2.4 Summary

A selection technique is characterized by the selection tool and the user's control of it. Both determine the affordances (what users can do) and the constraints (how users can do it). Affordances are determined by the selection tool itself, while the mapping between users' actions and the selection tool determines the constraints.

Unlike to the sub-task classification [11], we have not considered neither feedback nor selection confirmation mechanisms. Although they are critical issues, they are not related to a specific selection technique. The feedback and the selection confirmation mechanism might vary depending external factors, such as the pointing device or the output devices, which are discussed in the following section.

Regarding Popyrev's classification, interaction metaphors provide an overview of the selection technique, but they are too generic. To refine Popyrev's classification but keeping its abstraction layer, we propose a complementary classification where selection techniques are grouped according to its main purpose: (a) selection techniques providing alternative ways to map user's actions into the environment by improving the selection tool control (*Control*), (b) techniques which apply human motor control principles to increase performance (*Fitts*) and techniques which tracks user's actions and adapt its behavior (*User*). Notice that the three groups are not disjoint, a selection technique might fit in more than one. The raycasting and the isomorphic virtual hand techniques are considered *baseline* techniques, as they are the simplest representation of virtual pointing and virtual hand techniques. For completeness, techniques which do not involve 3D interaction, like menu or speech based selections, are considered in a fourth group (*Indirect*).

Table 2.5 shows the classification of the considered selection techniques. Furthermore, two additional classifications are provided. The first one is based on the origin of the selection tool (hand-rooted or eye-rooted) and the second is related to the principle of Fitts' law they exploit (increase width or decrease amplitude).

As we will discuss in Chapter 3 hand-rooted virtual pointing techniques present a mismatch between the visible objects and the selectable objects which significantly reduces selection performance. In contrast, eye-rooted techniques do not present this issue.

| Interaction Metaphor | Origin of the Selection tool | Purpose | Fitts' Law Basis | Technique |
|---|---|---|---|---|
| Virtual Hand | N/A | Baseline | None | Virtual-hand |
| | | Control | | Go-go [96] |
| | | Fitts' | Width | Silk Cursor [136] |
| | | | | Bubble-Cursor [120] |
| Virtual Pointing | Hand-rooted | Baseline | None | RayCasting [82] |
| | | Control | | Flow Ray [48] |
| | | | | iSith [93] |
| | | | | Two-handed Pointing [83] |
| | | | | Flexible Pointing [90] |
| | Eye-rooted | | | Direct Image plane [69] |
| | | | | Eye-gazed selection [115] |
| | | | | Occlusion Selection [94] |
| | | | | One-Eyed Cursor [126] |
| | | Control / Fitts' | Width | Aperture [40] |
| | Hand-rooted | Fitts' | | Sticky Ray [112] |
| | | | | Flashlight [56] |
| | | | | Sense Shapes [89] |
| | | | | Shadow Cone Selection [111] |
| | | | | Probabilistic Pointing [105] |
| | | Fitts'/User | Width Amplitude | PRISM [42] |
| | | | | Adaptive pointing [63] |
| | | | Width | Enhanced Cone Selection [110] |
| | | | | IntenSelect [32] |
| | | User | None | Smart Ray [48] |
| Hybrid | | Control | | Depth Ray [48] |
| | | | | Lock Ray [48] |

**Table 2.5:** *Selection techniques classified according to the classification by metaphor proposed by Poupyrev in [95], the origin of the selection tool, its main purpose and the Fitts' Law principles they exploit.*

## 2.3 Usability guidelines and limitations for 3D object selection

Usability, broadly speaking, encompasses everything about an artifact and how a person is affected by the use of the artifact. In human computer interaction, usability is considered as a measure of the user experience when interacting with a user interface. A usable user interface is easy to use, is engaging and efficiently supports user's tasks.

Despite a number of usability guidelines exists for 2D user interfaces, they are not directly applicable to 3D user interfaces. 3D user interfaces are significantly more difficult to design, implement and use than their 2D counterparts. 3DUIs are based upon real-world characteristics such as naive physics, body awareness, environmental awareness, social awareness and social skills [57].

While designing a new interaction technique, we first have to determine its purpose, and consider its requirements at a higher level. For a selection technique, we can ask about the selectable objects, the object layout in the virtual environment, the amount of objects to be selected or which are the input and output devices available.

Usability issues might arise due to intrinsic factors, mainly determined by the nature of the selection task, and due to extrinsic factors introduced by input and output devices. There are not too many works explicitly focusing only on usability guidelines for 3D user interfaces; being the work of Gabbard [44], Hal [54] and Bowman [12] notable exceptions. Usability guidelines are useful during the first stages of the design as they avoid known usability issues and speed up the whole design process.

VR community has been more focused on providing novel selection techniques, addressing specific problems or considering only a subset of the requirements, lacking completeness. On the other hand, having so many well known selection techniques allows us to choose the selection technique that better fulfills our specific requirements. From a usability point of view, a selection technique has to:

- Provide reduced selection time.
- Be accurate and error-proof.
- Support for precise selection.
- Be easy to understand and control.
- Produce low levels of fatigue.

Additional requirements depend on the application, e.g. support sparse or dense environments, provide with mechanisms to select semi-transparent or occluded objects, and do not interfere with the user's immersion in the virtual environment.

In the previous section we focused on classifying existing selection techniques considering its distinctive characteristics. In contrast, in this section we focus on the main factors limiting selection tasks and additional requirements a selection technique must ensure. The requirements considered range from common requirements for 3D user interfaces to specific requirements for 3D selection tasks.

### 2.3.1 Target size and location

Different applications have different requirements depending on the potential objects the user has to select. Selection performance is highly coupled with the geometrical properties (size, shape and location) of the target. As in reality, its properties determine the level of difficulty required to select it (e.g, grasping a cup is much easier than grasping a needle).

Object's size and location have a direct relation with selection performance. In Section 2.1, we reviewed Fitts' law which estimates the selection time considering the object size and the amplitude of the movement. Recall that Fitts' law asserts that the time $T$ to acquire a target of width $W$ which lies at a distance $A$ is governed by the relationship:

$$T = a + b \log_2 \left( \frac{A}{W} + 1 \right) \tag{2.13}$$

Selection time increases if the amplitude of the movement $A$ and/or the object size $W$ decreases, and vice-versa. It has been corroborated by several studies [120, 92, 6, 98].

According to Fitts' law and the optimized initial impulse model, several guidelines can be proposed to increase user's performance in selection tasks. As proposed by Balakrishnan [4], options rely on decreasing $A$, increasing $W$, or modifying both at the same time.

Techniques attempting to reduce $A$ include designs where graphical elements are laid out to minimize the distance to the cursor [29]. However, these approaches are limited to GUI elements where potential targets are known a priori. A more general approach is to reduce $A$ only in control space, thus preserving the original layout of the elements in the visual space. For example, it can be accomplished by ignoring the empty space between targets (see Figure 2.18a).

In contrast, techniques attempting to increase $W$ focus on increasing the area of influence of

the selection tool, increasing the activation area of targets in control space or dynamically increasing the size of targets.

The area of influence can be increased by using volumetric tools such as the flashlight [56] or aperture selection [40]. They allow for fast selections in sparse environments (see Figure 2.18b), but in cluttered environments their performance tends to degrade as disambiguation mechanisms are needed. As previously discussed, they performance is bounded by the efficiency of the disambiguation mechanism provided.

Increasing the activation area of a target can be done only in control space. For example, the bubble-cursor [120], subdivides the control space into Voronoy cells according to the layout of the objects in the virtual environment. Instead of selecting the object by placing the selection tool over it, the object is selected if the selection tool is inside the Voronoy cell defined by the object (see Figure 2.18c). In other words, the object selected is the object closest to the virtual hand. Another example is the sticky ray [112], the object selected is the object closest to the selection ray. However, improvements on selection performance depends on the area of the control space that do not belong to any activation area, which depends on the density of the virtual environment. So, again improvements are more apparent in sparse environments. Furthermore, as the visual representation is kept unmodified, additional feedback is required to show changes in the control space.

The last approach, increasing $W$ in control and visual space, known as expanding targets, relies on dynamically increasing the size of targets closer to the selection tool. By dynamically increasing the size of targets, users are provided with a larger target area to interact with.

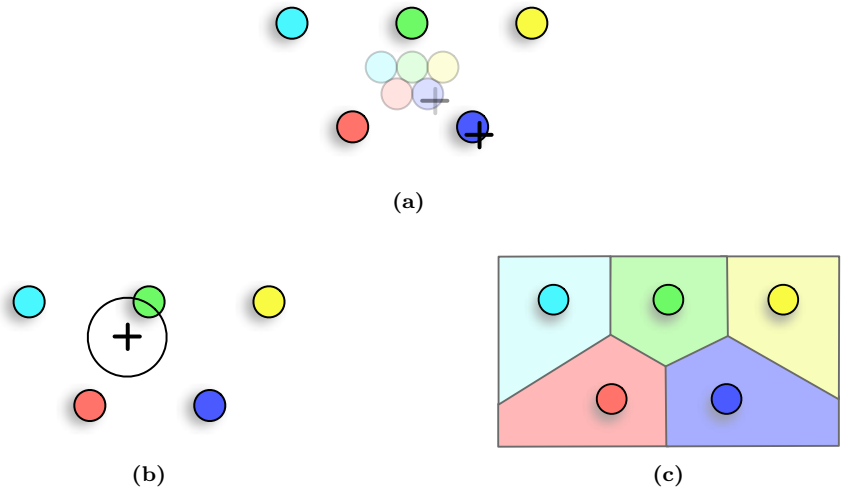



**Figure 2.18:** *Three approaches to improve the acquisition of small targets without changing their visual representation. (a) Reduce the distance between targets only in control space. (b) Increase the size of the selection tool. (c) Increase the area of influence of each target.*

Considering that the time to acquire isolated targets depends mostly on the final target size and not on the initial one [4], selection performance is improved. Several studies support the use of expanding targets for 2D acquisition tasks [23], but no work has addressed 3D acquisition tasks. We explore the viability of 3D expanding target approaches in Chapter 5.

Another approach to increase $W$ is to apply a distortion pattern in image space by using magic lens effects [46]. To ensure the compatibility with VR display systems, correct parallax values of the projected stereo images are required. This means that standard techniques based on 2D image distortion cannot be used as they completely disrupt parallax values.

A general drawback of increasing $W$, as reported by Wingrave et al. in [132], is that it will induce users to decrease their accuracy as they no longer need to be precise. In other words, the index of difficulty will be lower, but the index of performance ($1/b$) will also decrease, resulting in similar selection times.

Finally, techniques which increase $W$ and reduce $A$ at the same time, are techniques which in essence change the CD ratio (already detailed in Section 2.2.2). They have the best of both worlds, they can reduce $A$ by increasing the CD ratio and increase $W$ by decreasing the CD ratio. According to how the user performs aimed movements, ballistics movements covers most of the distance towards the target ($A$), while slow corrective movements depend on the size of the target ($W$), the CD ratio can be adjusted according to the speed of the user's actions [42].

#### Area of Reach

As stated in the previous section, the control space determines which objects can be selected. Virtual hand techniques only allow to select the objects inside the working space if no decoupling mechanism is employed. Although clutching or CD ratio based techniques can be used to extend the control space, clutching mechanisms introduce additional cognitive overhead and CD ratio based techniques, like the go-go [96], cannot provide enough precision when selecting distant objects.

In contrast, the control space for virtual techniques matches the visual space, thus all objects in the visual space are selectable. However, as the selection tool is mainly governed by hand's rotations, its precision is limited to the user's hand angular accuracy and stability. The further away an object is the higher the accuracy is required. Although the theoretical control space matches the visual space, the precision slightly decreases as the distance increases. Nevertheless its precision is higher than that provided by the go-go technique.

Furthermore, depth perception become an additional limiting factor when selecting distant objects. The level of required depth perception varies from one selection technique to another. For example, virtual hand metaphors require higher depth perception as the hand's depth is used to control the virtual tool. In contrast it is less important for virtual pointing techniques and even less for image plane techniques.

At this point, we have to ask whether it is necessary to provide techniques being able to select small or distant objects. The alternative relies on navigating towards the target to obtain an easier selection, but does the time spend during navigation compensates for a "potentially" difficult selection?

Two main issues arise. First, navigate to obtain an easier selection can be also "potentially" difficult. Navigation in cluttered environments require proper navigation techniques and the selection of small objects will require to scale the virtual environment. Furthermore, the user has to be provided with mechanisms to easily switch between selection and navigation tasks.

Second, when navigating in homogeneous virtual environments, such as a molecular model, the user can lose context information. For example, while navigating towards the target, the user might lose track of it. If this happens, the user has to go back to the starting position, locate again the target and start the navigation task again.

In conclusion, although for some situations the navigation approach will be more efficient, we believe than for selection intensive scenarios, the ability to select small and distant objects is necessary.

### 2.3.2 Environment limitations

Until now, we considered selection tasks when the objects are aisled, but in a standard scenario objects might be surrounded by other objects. As the object density increases, occlusions between objects is likely to increase.

Occlusion is always present in reality and provides important depth cues for spatial perception. However, occlusion is not always a desired feature. It can have a detrimental impact on tasks requiring to locate an object (discovery), obtain information encoded in the object (access) or obtain spatial information of the object and its context (spatial relationship) [36]. Occlusion is a common issue for cluttered environments [120]. High object density leads to occluded objects from the user viewpoint, reducing user's selection performance.

Selection performance can be reduced at different levels. First, occlusion might increase the time required to discover an object in the virtual environment. In the worst case scenario the object can be fully occluded and the user will need to navigate to locate it. We assume that the user knows where the target is located or at least, the object is partially visible.

Furthermore, although the user sees the target in the environment, occlusion still results in reduced object visual sizes and restricted access to targets [97] which impacts on user performance [110, 120]. In these situations the user has two choices, navigate to find a unoccluded view of the target or perform the selection from the occluded viewpoint. Improvements can be focused on the discovery phase or on the access phase, although it remains unclear whether improvements in the discovery phase will also improve access phase.

To avoid occlusion, in controlled situations, we can rearrange the environment [110]. However, in most situations it is not possible as the environment is fixed and context information should be preserved.

Volumetric selection tools (e.g. flashlight selection [56]) are more efficient when selecting objects in sparse environments. But, as the density increases, ensure that only one object is inside the selection volume becomes harder. Even if the selection technique provides disambiguation mechanisms, performance is compromised due to additional manual disambiguation steps or erroneous selections due to inaccurate automatic disambiguations.

Techniques based on virtual constraints like damping, snapping or trolling, which are useful in sparse scenarios [54, 44], might confuse the user in dense environments. Users tend to complain mainly about flickering effects [132]. CD ratio based techniques better adapt to dense environments, although the overall index of performance depends on the level of occlusion.

A different solution is to employ occlusion management techniques. Elmqvist and Tsigas [36] analyze a broad range of techniques for occlusion management and identify five main design patterns: Multiple Viewports (using two or more separate views of the scene), Virtual X-Ray (turn occluded objects visible), Tour Planners (a precomputed camera animation reveals the otherwise occluded geometry), Volumetric Probes (user controls an object which alters the environment in its neighborhood) and Projection Distorters (nonlinear projections integrate two or more views into a single view).

Despite having so many options to deal with occlusion, when considering direct interaction in virtual environments the alternatives are limited and there is no single solution that completely solves occlusion issues. Multiple viewports and projection distorters do not integrate well in immersive environments; we cannot modify the user perspective and having addi-

tional viewports may be a distractor factor. Tour planners involves navigation and the user has to stick to the predefined navigation paths, lacking flexibility.

On the other hand, virtual x-ray and volumetric probes allow to manually remove occluders in order to get an unoccluded view of the intended target. However, these alternatives increase the cognitive load of the user, potentially increasing selection time. Moreover removing occluders may remove useful context information. The most common solution is to employ semi-transparency [120]. However spatial relationships between semi-transparent objects may result unclear to users and access tasks can be compromised.

### 2.3.3 Input and output devices

The staggering number of devices available for use in VEs makes the development of 3DUIs significantly harder than their 2D counterparts [133, 12]. Input and output devices affect the usability of existing interaction techniques [7].

Typically, interaction techniques are designed and evaluated taking into account only one hardware configuration, due to time, availability and budget limitations. This might result in techniques usable only for that specific configuration, being the comparison with other existing techniques unfair.

#### Displays

Emerging and specialized devices like holographic devices require specific interaction techniques, as they may present unique conditions in terms of working and visual areas [48]. Available display devices range from semi-immersive displays, LCD screens and projection based systems, to fully immersive displays like head mounted displays and CAVEs systems. Every display has its own field of view and provides the user with different levels of immersion [118]. The field of view determines the amount of information that is visible at a time. The more information displayed the easier is to locate an object without head movements.

Head mounted displays (HMD), typically have greater field of regard (amount of physical space surrounding the user in which visual images are displayed), with the exception of CAVE systems. On the other hand, HMDs have reduced resolution in comparison with projection based systems.

The output device also conditions the visual feedback. When using raycasting selection in a fully immersive device, the selection ray is displayed entirely, allowing the user to easily

determine the origin and the orientation of the ray. However, in a semi-immersive display, only a fraction of the ray can be displayed, decreasing the amount of information available for the user.

Moreover, the level of immersion determines how the user sees and perceives its body. In obtrusive devices like head-mounted displays, the user cannot see its body and the application, if required, has to provide with a virtual representations of his body. If the user body is not correctly tracked, proprioceptive information will conflict with the virtual avatar hindering interaction.

Furthermore, for projection based systems, user's actions can obstruct the visualization of the virtual environment. For example, when selecting an object with the virtual hand technique, the user's hand will occlude the projection of the object during corrective movements. Another example is occlusion selection, where the user has to keep its hand roughly aligned with the eye position. In both scenarios the user's hand occludes part of the projected virtual environment, increasing the chance of erroneous selections.

### Degrees of freedom of input devices

When designing a new interaction technique, it is important to consider the matching between the DoFs required for the interaction technique and the DoFs provided by the input device [54]. Most spatial input devices return up to six DoFs, but usually, the DoFs used are less. It is recommended to minimize the number of DoFs required, as the more DoFs used the harder is the control of the selection tool [12].

Virtual hand techniques only require three DoFs for the hand position, and raycasting techniques are mainly governed by the yaw and pitch of the hand (only two DoFs). Employing six DoFs for tasks requiring less could be confusing if the input device is not well constrained [51], as changes in the unused DoFs are not visible to the user. Wingrave et al. in [134] observed that users performing with raycasting tend to move their arms forward and backward to select objects placed at different depths. This is totally unnecessary as raycasting is almost insensitive to hand position, specially for selecting distant objects. This behavior is common for novice users which unknowingly hinder their ability and thus they have to be taught not to perform in that way.

### Ergonomics

The physical device employed has to match the functionality of the interaction technique [54]. It makes no sense to employ a sphere-shaped device for virtual pointing, as the way of grasping the device should provide the pointing direction by proprioception.

Furthermore, most of the existing input devices are equipped with a number of buttons. The mapping between the buttons and the functionalities of the interaction technique is crucial for its usability. For example, a button press in a hand held device (like a wand) introduces instability when the button is pressed. This effect, nicknamed Heisenberg effect [134], may prevent successful selections when high accuracy is required. As a result, buttons introducing less hand instability should be employed.

Performance is also tightly coupled with the muscle groups involved [97, 19]. Smaller muscle groups achieve higher motor precision than bigger ones [138, 63], thus impacting on user performance. If possible, input devices employing smaller muscle groups should be employed.

### User fatigue

One of the most known issue in virtual reality applications is fatigue. The reduction of the fatigue is especially important if a hand-held device is used, as fatigue levels can raise rapidly. Interacting with our own body can raise fatigue levels and extended use may induce simulation sickness [67] and muscular strain. Selection techniques are more prone to arm and wrist strain, while for example, navigation techniques are more prone to induce simulation sickness. Knowing the requirements of arm effort, wrist effort and hand fixation of each selection technique allow to build an effort profile.

Arm and wrist effort can be extrapolated taking into account the degrees of freedom required to control the selection tool. Virtual hand techniques will require more arm effort than virtual pointing techniques, while virtual pointing techniques will require more wrist effort. Other device-ray mapping such as occlusion selection, which require to keep the arm roughly aligned with the eye, will require increased arm effort.

In the absence of input filtering (discussed below), hand fixation reduces hand trembling and stabilize the selection tool, but it requires additional user effort. Hand fixation is tightly coupled with the precision required; the greater the impact of hand trembling the higher the hand fixation should be.

Moreover, the position and orientation of the working space with respect to the user plays

an important role in the user's comfort. For example, the user can accomplish manipulation tasks with the arm lowered in a relaxed position by defining a convenient working space. Ideally, VE applications should allow users to define their working spaces according to the their own preferences, the physical condition of the user and the desired speed/accuracy balance.

#### Application performance, end-to-end latency and noise

In order to ensure smooth interaction the VR application has to keep high and constant frame rate [128], avoid high end-to-end latency [125, 77] and filter data from noisy input signals [63, 51, 44]. If the application does not ensure these requirements, it might reduce interaction performance [116], hinder high precision tasks, and also break immersion and presence.

Noisy tracking devices in combination with users' hand trembling [63] decrease the precision of the selection technique. Selection techniques have different tolerance levels to noise. Virtual hand metaphors are more tolerant to noise as they only rely on positional tracking data. Volumetric tools and CD ratio based techniques are also more tolerant to noise as they indirectly increase the effective width. On the other hand, virtual pointing techniques are less tolerant to noise, as they mainly rely on rotational data.

When high precision is required, a device with low latency and low noise should be used. If not possible, band-pass filters or Kalman filters can be applied to reduce noise of the input signal [63]. However, too much filtering increases the end-to-end latency. Pawar and Steed in [92] state that 60ms is the maximum latency that can be introduced without degrading interaction. In situations with high latency, Wingrave et al. in [134] observed that users performed steady movements and relied on proprioception rather than on the visual feedback. Obviously these behaviors trade off speed and accuracy.

Moreover, changes in latency with respect to time, referred to as temporal jitter, also hinder interaction and thus should be avoided. People can detect small fluctuations in latency likely as low as 16 milliseconds [92].

### 2.3.4 Feedback

Selection techniques involving spatial interaction require users to perform gestures to control the selection tool. The gesture can be a simple grasp operation or a pointing operation. If no feedback is provided, the user has to rely only in proprioception and depth perception to

ensure that the gesture results in the selection of the intended virtual object.

Although interaction with close objects can be achieved only by proprioception [83], several studies revealed that users without any selection feedback are unable to efficiently interact with the virtual environment [132]. Users do not have a model of interaction with the environment but a model of how to respond to the feedback the environment provides [132]. So, it is mandatory to provide feedback [44].

A selection technique has to provide, at least, visual feedback to drive user's actions. The simplest option consists in displaying a virtual representation of the user's actions in the environment, for example, drawing the user's hand position or displaying the pointing direction. The visual feedback allows users to observe how their gestures map with the virtual tool. In situations where the CD ratio differs from one or when the shape of the selection tool changes over time, a good visual representation of the selection tool is a key feature to ensure usability. But in general, proper visual feedback highly depends on the interaction technique.

Moreover, after the selection confirmation, the selected target can be highlighted [82]. Changes on its visual properties allow the user to ensure that the object selected is the right one. For example, changing its color or displaying the object in wire frame. In contrast, continuous highlighting of the object indicated by the selection tool has to be used carefully. It might cause excessive popping and a distracting effect on the user, particularly while interacting with cluttered scenes. Furthermore, object highlighting requires to check for every frame which is the object indicated by the selection tool, thus potentially increasing the application's overhead.

In general, increasing the amount of visual feedback does not always improve user performance [98] and might even reduce selection performance [131, 48]. On the other hand, providing redundant information might allow to bypass aptitude and expertise, allowing unexperienced users to perform as experienced ones.

Furthermore, we cannot forget how to determine which are the selectable areas of the virtual environment. Although the motor and the visual space are easy to determine for the user, the control space relies on the selection technique used. Selectable areas should be indicated to avoid confusion [44], as selectability may change dynamically over time [110]. Selectable areas can also be outside the viewing frustum, the user can be provided with markers to guide them towards the desired object [134].

In addition to visual feedback, introducing different feedback modalities, like haptic and acoustic feedback, can also be beneficial [51]. Again, although it is not assured that including

additional feedback results in performance improvements [14], users often prefer the addition of extra feedback [121].

Active haptic feedback can assist users during the selection process [92, 121]. Guiding the user to possible targets by introducing gravity wells (traction) or turning surfaces sticky (friction). However it requires a fine tuning of the forces applied and in dense environments it might be counterproductive; the user might be guided to the wrong object.

An easier approach is to provide passive haptic feedback (physical constraints), which can further increase interaction precision [51]. The most adopted solutions remain on using prop-based physical constraints [53] or physical surfaces [10, 72]. Both provide spatial references, which are intuitive to learn and speed up 2D pointing tasks in free space.

The user's body can be used as a frame of reference, as the user is able to determine its own body position by proprioception. One clear example is the go-go technique [96]; the user always knows the distance between its body and its hand. Using the non-dominant hand [54] can also be considered, as it provides a frame of reference for the dominant hand, and the user can employ it to perform two tasks in parallel.

Lastly, auditory feedback [121] can reinforce user's actions. For example, it can inform the user when a target has been highlighted or successfully selected. However, similar to haptic feedback, when interacting on dense environments it might produce distracting effects and playing the same sound multiple times might become annoying.

### 2.3.5 User's preferences

Users interacting with virtual environments account for different preferences. Knowing these preferences allows interface designers to determine which are their preferred interaction techniques [134]. Users have different levels of expertise and perform actions in different ways.

According to the user's method of interaction the designer may personalize the behavior of the interaction technique to behave like the user wants. As Wingrave et al. show in [132], subtle versions of the same IT can be provided, and we can let the user choose they preferred configuration. In their experiment, they employ raycasting and occlusion selection with a snapping mechanism (the selection tool bends to the closest object within a range). The user could introduce a rotational (raycasting) or a translational (occlusion selection) offset to the virtual hand with respect to the real hand, thus allowing for a more comfortable interaction, and change the threshold of the snapping mechanism. They results showed that there was not a trend when tunning the selection techniques, each user had its own preferences.

Instead of letting the user customize its interaction, we can adapt the available techniques to better suit the user. Octavia et al. in [87] explored how to choose automatically the most suitable IT for a certain situation. They gathered physiological data to measure user frustration, user experience and the mental workload. They observed that frustration measures were strongly correlated to the task completion time. Users accepted the technique adaptation; they did not bother if the system automatically chose the best suitable technique and the performance was slightly improved. However in their study they knew a priori the intended targets and only considered two selection techniques.

As stated before, manipulating input devices in free space can easily raise fatigue [54]. We can provide the user with recalibration mechanisms to allow the user to define its working space, obtaining a more comfortable positions.

### Selection Trigger

For virtual pointing selection techniques in addition to point to the desired target, the system has to be informed when to perform the selection. The most common option is to press a button conveniently placed in the pointing device "press to select". Steed in [110] considered two additional options: "hold and select" and "dwell on object". For hold and select instead of triggering the selection when pressing the button, it is triggered when the button is released, which may be less sensitive to the Heisenberg effect. In contrast, for the dwell on object approach, the selection is triggered when the user points to an object during a fixed amount of time. Dwell time thresholds introduce a fixed constant latency, being sensitive from the "Midas Touch effect" : for high precision selections fixations may occur at objects that do not interest the user, resulting in unwanted selections.

Gestures can also be used as selection triggers. The simples approach is to perform a pinch gesture [13]. Vogel and Balakrishnan in [122] exposed different alternatives for triggering selection for free-hand pointing in large displays without using any button. They proposed two different hand gestures to perform the selection, *AirTap* and *Thumb Trigger* in combination of visual and auditory feedback (see Figure 2.19).

The selection of small objects can be compromised due to hand instability during the confirmation of the selection. So, it is important to choose a selection trigger mechanism that minimizes hand instability.

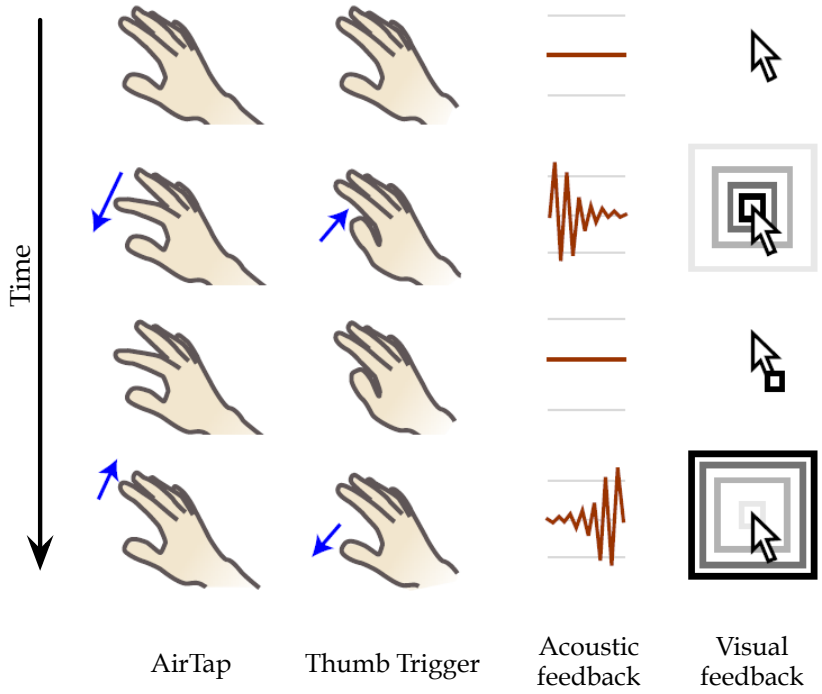



**Figure 2.19:** *The air tap (first column) mimics pressing the button of a mouse. The thumb trigger gesture (second column) consists in moving the thumb finger towards the index finger. Auditory and visual feedback is provided for both while performing the gesture (third and fourth columns). Image courtesy of Vogel et al. [122].*

## 2.4 Object selection in collaborative virtual environments

Multi-view virtual reality systems [2] enable the collaborative experience of a shared 3D space in a similar way as in real life (see Figure 2.20). Multi-view systems [103] provide tracked users with their own stereoscopic perspective-correct viewpoint. Consequently, natural forms of gestural communication, such as pointing, can immediately be used for the collaborative inspection of computer-generated 3D models.

However, real-world correspondence also results in real-world problems. One might want to show a virtual object to colleagues, which may be occluded from their respective viewpoints. To solve this problem in reality, people have to walk around the occluding objects to obtain a suitable viewing position. Often they move close to the person who is pointing to the object, sharing the viewpoint in order to see the specified object.

Eduard T. Hall in [49] explored how people deal with human interactions taking into account the space surrounding each other. Particularly, Hall showed that interactions where people are close to each other are constrained to intimate or friendship relationships. Behaviors

(a) (b)

**Figure 2.20:** *Multi-view virtual reality allows multiple users to interact with the virtual environment (a) as in real life (b). Image courtesy of Salzmann et al. [103]*

resulting in physical proximity of users do not comply with social protocols, especially for formal presentations.

### 2.4.1 Referential awareness

Meaningful exchange of information and collaborative work are only possible if we know what others are referring to. In shared physical spaces deictic gestures like pointing and tracing in combination with verbal hints generally provide sufficient cues for efficient communication. However, if the spatial layout of the explored environment becomes dense and complex, objects and features are not always visible from all the viewpoints involved. Awareness of reference gestures and selected objects can thus be compromised due to occlusion. Consider the set of visible objects for one user $V_1$ and the set of visible objects for a second user $V_2$. Objects visible for both users, $V_1 \bigcap V_2$, do not pose a communication problem. However, one user may point at an object that cannot be seen from the others's viewpoint $V_1 - V_2$ or $V_2 - V_1$.

The obvious way to alleviate this problem is changing the viewpoint, e.g., by looking over the other's shoulder. In computer-mediated collaboration multiple people may even share exactly the same viewpoint. In the context of remote collaboration this approach has been shown to be beneficial as occlusion issues are prevented and referential ambiguity can be minimized [22, 64].

In co-located multi-user VR, however, this is not an option. Aligning the virtual viewpoints of all involved users necessarily corrupts the notion of a shared 3D space and thus hinders

gestural communication [2]. Just as in reality, one must walk over to the person who indicates a point of interest to see what he/she is referring to. This however may interfere with social protocols. We assumed that being in such close proximity to one another can make the users feel uncomfortable.

For video-based tele-collaboration Chastine and Zhu suggested in [21] occlusion-compensating visualization techniques as an alternative solution. However, the suitability of special visualization techniques and the corresponding impact on mutual communication and collaboration has not been studied yet.

### 2.4.2 Proxemics

Space is one of the fundamental organizational systems of humans. The layout of cities, architecture and interiors undoubtedly affects human behavior in these environments. The extent of open space a person needs surrounding him or her depends on several factors, including physical ones such as temperature, lighting and noise. But one's culture and social relations are also encoded in terms of space. Edward T. Hall coined the term proxemics [49] for "the interrelated observations and theories of man's use of space as a specialized elaboration of culture". Proxemics deal with social and personal space and humans perception of it.

In "The Hidden Dimension" [49], Hall defines four classes of interpersonal distances and their respective meaning in regards to social relations (see Figure 2.21). All four are further subdivided into their respective close and far phases: *intimate distance* (0 cm – 15 cm – 46 cm); *personal distance* (46 cm – 76 cm – 120 cm); *social distance* (1.2 m – 2.1 m – 3.7 m); and *public distance* (3.7 m – 7.6 m – more). Hall derived these measurements from experiments and interviews he conducted with adult natives from the northeastern seaboard of the US. Since they mainly depend on culture and perceptional channels involved in the communication among one another, they also serve as rough approximations for interpersonal distances in other regions of western culture. Proxemic behaviors are also present in virtual environments when interacting with user's avatars [130] or virtual agents [5].

*Intimate distance* involves touch and olfaction. Visual perception becomes distorted. It occurs only in certain situations such as "wrestling or making love". Otherwise, people generally try to avoid such closeness. *Personal distance* is mostly observed between family members, partners and close friends. The distinction between the close and the far phase of personal distance corresponds to one's arm reach at approximately 76 cm. In more formal relations among acquaintances (e.g. in collaborative work settings), people tend to maintain

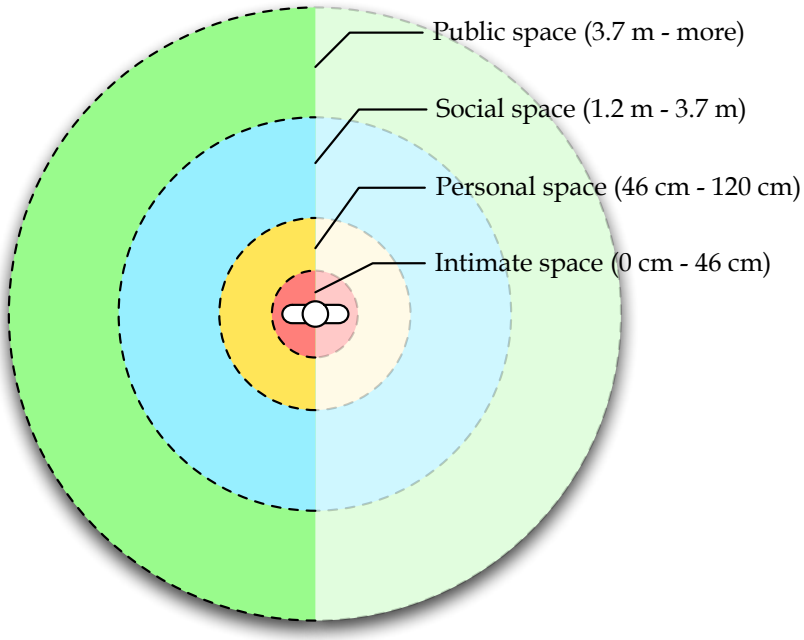



**Figure 2.21:** *Distances of men proposed by Eduard T.H its Proxemics theory [49].*

a *social distance*. Hall describes the close phase of *social distance* (up to 2.1 m) as the generally observed situation in collaborative work settings. He also notes that important people often try to keep their subordinates at this far phase. *Public distance* can be observed in settings like public speeches.

The infrastructural and technical requirements of collaborative work can have an important influence on the distances between co-workers. For instance, to see objects somebody else is referring to in dense three-dimensional scenes, it is often necessary to keep intimate distances.

## 2.5 Embedding 2D GUIs in virtual environments

One of the main tasks in virtual reality is related to application control. Application control refers to the user task of issuing commands, requesting the system to accomplish a particular function or change its internal state [12]. In 2D UIs, application control tasks are basically performed by using a graphical user interface (GUI) following a WIMP (Windows, Icons, Menus and Pointers) metaphor.

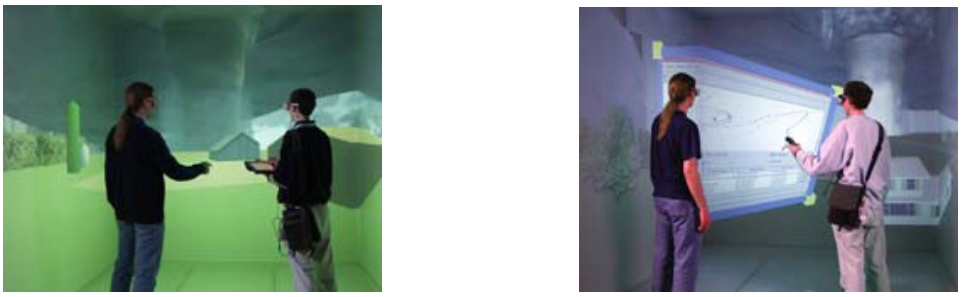
For immersive VE applications, in addition to GUIs, a greater number of interaction techniques have been proposed for accomplishing application control tasks. These interaction techniques range from interaction with 3D Menus [30] and voice-recognition systems [80] to gesture-based interaction [59] through specialized physical devices such as Tangible UIs [55].

However, the simplest and most popular GUIs are two dimensional widgets adapted to 3D space. These widgets are simple adaptations of their 2D counterparts and basically work in the same way as they do in the desktop metaphor.

Despite the broad diversity of application control techniques for 3D user interfaces, only a limited amount of attention has been paid to the development of standard APIs and reusable software components enabling 3D GUI development in a cost-effective manner and providing a wide range of 2D widgets. There have been several approaches of incorporating GUIs in virtual environments, which basically can be grouped in two different categories:

- GUI outside the virtual world: the GUI is displayed and operated in a separate device.
- GUI inside the virtual world: the GUI is displayed in 3D space as a virtual object.

On the one hand, applications which follow the GUI-outside approach have their user interface split into two parts (see Figure 2.22a). Some of the functionality is accessible through a 3D user interface (usually navigation and manipulation tasks) and application control tasks can only be controlled through a conventional GUI running on a console outside the VE. This approach simplifies UI development but presents some problems. First, it increases the cost of the system as a hand-held computer is needed [127] or specialized hardware. Second, it present obvious usability problems, the user must be aware of two different visual output devices and one hand is required to hold the device.



**(a)** **(b)**

**Figure 2.22:** *Two approaches to incorporate GUIs into virtual environments. (a) GUI outside the virtual world. The GUI is displayed in a hand-held device. (b) GUI inside the virtual world. Images courtesy of Hartling et al. [50]*

On the other hand, placing GUI components inside the virtual world offers much more possibilities and interaction styles, and thus this is often the most suitable option (see Figure 2.22b). Developers willing to put a 3D GUI inside a VE application are faced with two options. They could implement the full functionality of the 3D GUI inside the application (a non cost-effective approach) or they could adopt any of the few available APIs for 3D

GUI creation [66, 91, 50]. Specific APIs for handling 3D interaction in VE are undoubtly a reasonable solution but in their current state of development they present some limitations:

- They force developers to learn a new API. In addition, the lack of standard APIs, and mature and well established implementations, make this problem worse.
- They usually implement all the functionalities for each widget, including drawing and handling. As a consequence, available 3D toolkits offer a limited set of widgets and it is difficult to plug-in third-party components (e.g. media players and web browsers).
- A commonly accepted opinion is that GUI interfaces should be optimized for each platform. For example, a desktop interface using 2D interaction for system control is certainly appropriate for a desktop system, whereas an immersive stereoscopic-compatible interface should be used in an immersive system. Most widget libraries for VEs target only the second kind of platforms and thus two or more different versions of the applications are often required.

These problems suggest that traditional 3D GUI tools are not always appropriate for fast development of complex GUIs and particularly for fast migration of existing desktop-based 3D applications to immersive VEs. For example, an existing desktop-based 3D application can be modified to display its contents on a stereoscopic workbench with minor effort, but the migration of its GUI can be a much more difficult task.

As a solution, some authors have proposed the immersion of 2D applications into 3D worlds to make them available to VE and Augmented Reality (AR) applications. The basic idea is to project 2D image content onto texture-mapped rectangles on the 3D world, and let the users interact with them through VR input devices such as gloves and tracking systems. Current systems for launching and/or sharing existing 2D applications into VE and 3D window managers are good examples of this technique. Although application-sharing systems and 3D workspaces can be used to access 2D GUIs within VEs, they suffer from some performance and flexibility limitations mainly because they are built upon framebuffer-oriented protocols.

### 2.5.1 Software tools for 3D UI authoring

Software tools for 3D UI authoring can be broadly subdivided into three categories: general frameworks for developing VE applications, specific modules for 3D UI development, and automatic generation tools from specifications.

A number of frameworks and APIs for developing device independent VE applications have

been proposed. Some well known examples are DIVERSE [61], VRCO's CAVElib [123] and VR Juggler [25]. These tools put the emphasis on abstracting common programming tasks on VE such as window creation, viewer-centered perspective calculations, stereoscopic viewing, displaying to multiple graphics channels, cluster synchronization and accessing VR hardware such as trackers, wands and motion systems.

However, only a few provide specific modules for 3D UI creation:

- VR Juggler's Tweek [50] provide VE users with an extensible Java GUI that communicates with VR applications through CORBA (the Common Object Request Broker Architecture).
- The VEWL (Virtual Environment Windowing Library) [66] is an API providing a window manager that supports the use of menus, windows, buttons and other widgets within an immersive virtual environment. VEWL widgets use class names similar to Qt [26], although the actual rendering is done using OpenGL. VEWL provides device-independent input through DIVERSE [61]. The VEWL interface is controlled using a 3D pointer controlled via a tracked input device.
- The it3d (Interactive Toolkit Library for 3D Applications) [91] provides an input/output library for distributed devices, a 3D widget library for multimodal interaction and a gesture-recognition library.

Despite these valuable tools, there is a lack of standard APIs and well-established tools for 3D UI development. One of the few UI standardization efforts is UsiXML (USer Interface eXtensible Markup Language) [70], a XML-compliant language for describing a UI for multiple contexts. The language has been designed to support the description of interactive applications with different types of interaction techniques, modalities of use, and physical computing platforms. Although some promising tools exist for automatic generation of VRML97 or eXtensible 3D (X3D) GUIs [78, 60].

### 2.5.2 Immersing 2D applications into 3D worlds

A related and very active research topic is the immersion of 2D applications into 3D space. We distinguish two basic categories of tools: 3D window managers for desktop computers and tools for accessing remote applications from within VE and AR applications.

Tools in the first category aim at providing 3D workspaces designed for replacing 2D desktops. These tools rely on mouse-and-keyboard interaction and are intended for non-immersive

display devices. A first example is MaW [68], a prototypal 3D Window Manager which allows the application to create windows that are displayed in 3D space using OpenGL primitives. Another example is Task Gallery [102] which is a 3D prototype user interface that expands the desktop into an entire office with an unlimited number of desktops. The screen becomes a long gallery with paintings on the walls that represent different tasks.

Tools on the second category aim at providing access to remote applications from within VE and AR applications. Two different approaches can be observed [15]. Hardware oriented approaches provide access to external applications through additional display devices such as PDAs and see-through-displays [127]. Software oriented approaches directly display the UI into the virtual environment. The VE software can manage directly the actual 2D drawing (using geometric and text primitives) or it can access 2D display content generated by external applications and display them as texture-mapped rectangles.

Early work using application sharing in 3D environments is described by Dykstra [35], where texture-mapped rectangles are used to operate X applications 3D virtual spaces. A similar approach is described in [117] for immersing X window applications into a 3D scene. This idea has been adopted and extended both in VE and AR applications.

A more general approach is using VNC (Virtual Network Computing) [101] which is a remote display system which allows viewing a computing desktop running elsewhere on a network. VNC provides a distribution mechanism for desktops on the lowest level by transmitting frame buffer contents to the remote client and receiving keyboard and pointing device events, inserting these into the server-side input queue. Each time a client interacts with the shared application, the VNC server broadcasts the image of the areas affected by updates on the remote interface. VNC is the foundation of a number of systems providing immersion of 2D applications into 3D space. A number of approaches use VNC [15, 31, 108, 34] but all suffer from performance and flexibility limitations mainly because they are build upon a protocol providing little control to the VE application over the properties and behavior of GUI components.

## 2.6 Evaluation of 3D user interfaces

Focusing on usability from the beginning of the application development reduces the number of user interfaces dismissed, ensuring than the final user interface better fits the task requirements. Evaluation is critical step in all the software design paradigms, but in the development of 3D user interfaces it is even more critical. The evaluation of a 3D user interface not only covers how robust is the application, but it has to take into account the usability

of the interface. The evaluation step has to give feedback to the development step reducing usability problems and improving the overall system. It is recommended to evaluate more than once the system during its development.

Bowman et al. in [8] sumarized two evaluation methods applied to the evaluation of virtual environments: the testbed evaluation [11] and the sequential evaluation [45]. Their main difference is the scope of the evaluation: while testbed evaluation's main goal is to evaluate different interaction techniques in a generic environment, sequential evaluation focus on the evaluation of the entire application.

In this thesis we focus on the evaluation of individual interaction techniques, so we consider only the testbed evaluation method. The testbed evaluation enables to test the techniques in a generic environment and allow us to compare among other existing interaction techniques ensuring that the interaction technique will be usable in other environments.

The main steps followed by a testbed evaluation are depicted in Figure 2.23. First, an initial evaluation is done to obtain intuitive understanding of the interaction tasks involved, and which are the existing techniques providing the functionality required. Usability guidelines help to detect possible hindrances before the evaluation. This first step narrows the options considered in posterior phases.

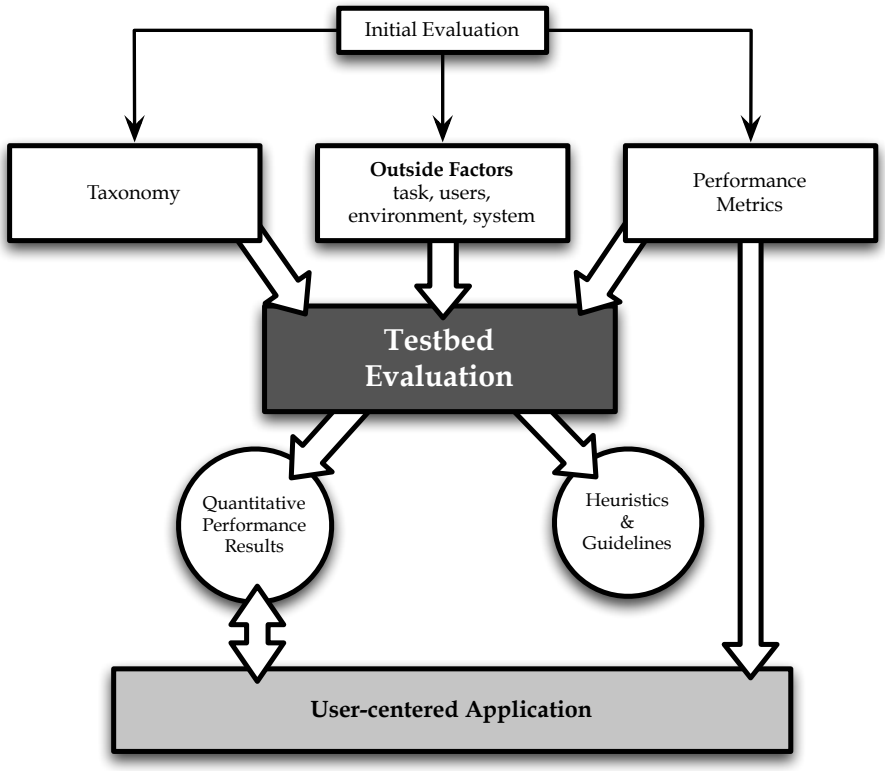



**Figure 2.23:** *Testbed evaluation flow diagram by Bowman et al. [11]*

On a second step we need to define a taxonomy of the tasks and subtasks involved in the evaluation (user task and work flow analysis), determine the possible outside factors (e.g. environment, system, users), which can potentially affect the results of the evaluation, include them in the data analysis and choose which performance metrics (e.g. accuracy, time, number of errors) should be recorded during the evaluation.

It is essential to define a complete task taxonomy. The simple task of moving one object from one position to another, can be subdivided into three different subtasks: select the object, translate the object to the final position and release it. The more complete the subdivision is the stronger is the resulting taxonomy. The taxonomy combines all options for each subtask, resulting in different designs for the same task. Some of the designs will be valid and others will be rejected for being inappropriate from a usability point of view. External evaluators may detect these inappropriate combinations thus reducing the number of combination to evaluate, which is also recommended by the sequential evaluation [45].

In VEs setups we are facing heterogeneous input and output devices, a great number of different configurations are possible which will undoubtedly impact on the evaluation. These devices will define the working position (users may be standing or seating), the working space (users can walk or they have to remain still) and physical constraints (cables, projection screen, walls). Furthermore, consider the diverse user population to run the evaluation, age, gender, expertise and so on. If detected during the design phase, all these possible outside factors have to be considered in the experimental design.

Moreover, we have to choose the performance metrics to score the interface quantitatively and qualitatively. Bowman et al. in [8] proposed the a classification of the performance metrics in three different groups:

- *Computer graphics' performance.* To provide a good level of immersion the system must have a high frame rate (over 25-30 fps) and the latency must be as low as possible. If the frame rate drops down this limit, the system would be less usable and an optimization must be done in order to achieve a higher frame rate.
- *User's performance.* The main measures are task completion time and accuracy (error rate). Due to the trade-off between speed and accuracy, in a standard scenario high accuracy rates will produce high task completion time and vice-versa.
- *User's preferences.* In contrast to the first two groups which provide objective and quantitative metrics, the user preferences are mainly subjective and qualitative. We can consider a number of different metrics, ease of use, ease of learning, sense of presence, user satisfaction and user comfort. User's preferences can also help to detect issues in

the design of the experiment (e.g. motion sickness or fatigue). Typically questionnaires are given to user before and after the experiment to gather subjective user's preferences.

All performance metrics have to be recorded during the evaluation automatically by the application, thus ensuring that all measurements are consistent among different users. Moreover, the evaluation can be monitored by one or more evaluators or even recorded to perform post-evaluation analysis. Nevertheless, videotaping in a VR setup is hard due to the impossibility of gathering the projected stereo images; we can only capture user's gestures in free space without any correspondence with the virtual environment.

A better option consists on logging all user's movements to virtually reproduce the task. By monitoring users' actions we can detect possible hardware malfunctions or even application bugs. However, it is recommended to perform pilot studies to detect these issues beforehand.

The next step is to define the experimental design. According to the taxonomy, the outside factors and the metrics, a formal, factorial experiment design can be defined. Each factor can have different levels, e.g, in a selection task we can consider "selection technique" as the first factor, with the levels "raycasting", "occlusion selection" and "virtual hand". A second factor can be the "target size", with the levels "5cm", "10cm" and "20cm", resulting in a $3 \times 3$ design. Moreover, additional external factors can be taken into account. Wingrave et al. in [134] observed differences according to the participant's experience in computers, 2D and 3D graphics and gaming experience. We can also consider if the user is right-handed or left-handed.

Typical experimental designs may range between simple experiments (one factor) to over complex (many factors) - finding the proper balance is difficult (the number of combinations increase factorially). Once the factors are defined, the designer has to choose which factors will be considered within-subjects (each user performs the evaluation for each combination) or between-groups (we only consider a subset of combinations for each user) and the order they will be presented to users. Notice than for each between-groups factor, the number of users required has to be roughly doubled to ensure validity.

Factors presenting ordering effects should be considered as between-groups factors. Ordering effects appear when the order of the factors alter the results. Nevertheless, if ordering effects are due to learning effects (users learn over trials), providing training sessions and introducing repetitions will mitigate them and will also reduce the variability of the gathered data.

Counterbalancing the order of the factors presented to users will further decrease possible interactions among them. The most common method employed is the latin squares (see

Figure 2.24) which guarantees that (1) every level appears in every position the same number of times, (1) every level is followed by each other level and (3) every level is preceded by every other level, thus counterbalancing any possible ordering effects.

$$\underset{\textbf{2 x 2}}{\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}} \quad \underset{\textbf{6 x 6}}{\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \\ 1 & 3 & 2 \\ 2 & 1 & 3 \\ 3 & 2 & 1 \end{bmatrix}} \quad \underset{\textbf{4 x 4}}{\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 4 & 1 & 3 \\ 3 & 1 & 4 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}}$$

**Figure 2.24:** *Example of the latin squares distribution for different combination of factors. Notice that no 3x3 combination meets the properties of the latin squares.*

The last step of a typical experimental design consists in defining the hypotheses. According to the theoretical basis, the experiment designer formulates the expected results of the experiment. The data analysis will determine whether these hypotheses have to be accepted or rejected.

If we are performing a Fitts' law experiment, the experimental task has to consist on simple and fast aimed movements. To ensure less variability, users' performance has to be limited by the capacity of their motor systems not by their cognitive capabilities.

In order to ensure validity, the number of errors should be less than 4% [38]. As task ID does not account for user accuracy, the number of errors has to be kept as low as possible. If we obtain a high number of errors, first, the threshold of the movement endpoint will no longer be $W$, and second, if the users notice that they are making a lot of errors, they index of performance (IP) would be affected. They will sacrifice speed for accuracy along the task until they get a good balance, which will affect the IP along the task. As noticed in [134, 132], errors will produce "bad vibe" behaviors, after a mistake users slow down for a while until they recover confidence.

Variability can also be minimized by providing training to users. Training will allow users to achieve a similar index of performance as experienced users. However, expert users can still obtain better results in some scenarios [131].

In addition, the examiner has to provide users with instructions to ensure than the speed-accuracy trade off is consistent among all users. Asking participants to perform both quickly and accurately will lead to a compromise between accuracy and speed [137], but each user

may perform in a different way. The evaluator might find the best speed-accuracy trade off given the evaluation.

Examiners have to guarantee that all users correctly understand the tasks and the work flow. A well defined work flow will ensure a fast learning step and avoid misunderstandings in the procedure. Moreover, the examiner has to ensure that conditions are the same among all users, task, objects, distance, environment and instructions. If necessary a whole system recalibration has to be performed for each participant.

It is also important to take into account how long the evaluation session will last; a great number of factors may result in longish experiments which have to be avoided. Possible solutions rely on providing breaks among trials, split the evaluation in different days or turn some within-users factors into between-group factors, reducing the overall length of the experiment, thus avoiding fatigue, sickness and discouragement.

Participants have different motivational forces acting on them which can invalidate the results. One motivation is to perform well so as to help the researcher, another might just be to perform well so as to not be embarrassed. However, as the experiment advances, the dominating motivational force is to complete the experiment so the participant can leave.

The most common approach to keep users motivated is to pay users or award the better-performing users with some reward (e.g. money, cinema tickets). Competitiveness among users can be further stimulated by displaying how they are performing [131] or showing the current *highscore* [132].

Once everything is set, the evaluation is conducted and all the user's actions are recorded: performance measures, video feed and incidences. Finally, all the gathered data has to be analyzed employing descriptive (histograms, box plots) and inferential statistics (t-tests, ANOVAs) to obtain conclusions. Before analyzing the data, a common strategy is to remove outliers (remove samples greater than $3\sigma$) and erroneous trials [48].

We have to be careful with the results of the evaluations as it is tempting to over-generalize the results to a generic context. It is desirable to include information about the environment setup and, if possible, perform the evaluation in different environments and hardware setups.

As a last remark, we have to notice that user studies are performed by human beings. As discussed by Kristensson and Blackwell in [65], ethical requirements vary among universities, research institutions, research councils or funding bodies. For example, although the Association for Computing Machinery (ACM) has a code of ethics [88], it does not consider user studies explicitly. In contrast, ethical considerations for psychological user evaluations are

well established for the American Psychological Association (APA) [3]. We now detail some guidelines extracted from [20] regarding the ethical considerations for HCI user evaluations in the APA ethical principles [3].

- Ensure that users are not harmed neither physically nor psychologically during the experiment.
- Detail all the potential risks involved during the evaluation. For example, during VR user studies users can suffer simulation sickness or get disoriented.
- The purpose of the evaluation and the data recorded during the experiment have to be detailed to the users. Exceptionally, if this information is susceptible to bias the experiment, the experimenter has to ensure that all the information is provided at the end of the experiment. Moreover, all the data gathered during the user evaluations must be confidential.
- Users have to be informed that they have the right to leave the experiment at any moment if needed.
- Experimenters have to ensure that users do not get frustrated by being unable to perform the proposed tasks. It might be desirable to state that the tasks are complicated and the tools provided are not enough to perform the task.

The standard procedure is to present, at the beginning of the experiment, a written document to the users, detailing all this information and ask for their written permission.

# Chapter 3

# Analysis of visual issues in virtual pointing tasks

Virtual pointing techniques are by far the more used 3D object selection technique in virtual environments. For most virtual pointing techniques, the underlying selection tool is a ray (or a cone), controlled through hand position and/or orientation. In comparison with virtual hand techniques, virtual pointing techniques allow users to select distant objects, and require less physical effort.

In this chapter we focus on how the visual feedback of virtual pointing techniques allows for accurate selection tasks. User's perception of the virtual environment and visual feedback are crucial for effective interaction, as users rely on visual feedback to guide the pointing gesture rather than relying only on proprioception. Furthermore, virtual pointing techniques require potential targets to be clearly visible.

These requirements mostly hold for monoscopic displays, but do not hold for immersive virtual environments. Stereoscopic virtual content does not provide the same amount of depth cues as the real world, thus decreasing the spatial understanding of the virtual environment. Moreover, occlusion is a common issue in 3D environments, and occluded objects, as we shall see, hinders selection tasks.

We first analyze the most adopted solutions for visual feedback in virtual pointing techniques, detailing their limitations. The analysis considers the amount of accuracy the visual feedback achieves, according to the adopted selection technique and the current limitations of stereoscopic vision.

Second, we present an intrinsic limitation all hand-rooted virtual pointing techniques suffers from: the eye-hand visibility mismatch. We study the potential mismatch between visible objects (those which appear unoccluded from the user's viewpoint) and selectable objects

(those which appear unoccluded from the user's hand position) and how it affects selection tasks. We show analytically and experimentally that precise selections for hand-rooted techniques can only be achieved if the object to be selected is simultaneously visible from the user's hand and the eye's position.

## 3.1 Visual feedback for pointing on stereoscopic displays

Virtual pointing techniques require the application to provide appropriate visual feedback about the pointing tool and its spatial relationship with potential targets, e.g. drawing a ray emanating from the user's hand. In an ideal scenario, the visual feedback should allow users to univocally identify the object being pointed to under all possible viewing conditions and scene layouts.

Regarding the judgment of the movement required to approach a particular target, the visual feedback (a) should give a precise indication of the direction and the amount of movement required, (b) should work under all possible viewing conditions and scene layouts, and (c) should work for all the degrees of freedom (DoFs) of the selection tool.

When designing the visual feedback for a virtual pointing technique the following factors should be considered:

- The origin of the pointing tool (eye-rooted or hand-rooted techniques).
- The tool control. We distinguish between translational control (based on hand position) and rotational control (based on hand orientation).
- The display type. Depending on provided depth cues and whether users can see their physical hand or not, we distinguish between fully-immersive displays (such as HMDs), and semi-immersive displays (such as CAVEs and other projection based stereoscopic displays).

The origin of the pointing tool is critical to choose a proper visual feedback. When the pointing direction is cast from the hand position $H$, the usual approach is to draw a ray extending out from $H$. Other visual feedback techniques such as drawing a cursor are clearly inappropriate as they would force the user to figure out how the ray extends behind the cursor.

On the other hand, when the pointing direction is cast from the user's eye $E$, its projection on the viewing plane is a single point, so we must choose another strategy for providing appropriate visual feedback.

Note that virtual pointing always requires some kind of visual feedback, as relying solely on proprioception is not sufficient for accurate selection. The only exception are image-plane techniques with translational control running on semi-immersive displays; the user's hand/finger provides a rough feedback of the pointing direction.

Our main goal is to anticipate potential usability issues for accurate pointing on stereoscopic displays by exploring how current visual feedback approaches guide the user during ballistic and corrective phases of selection tasks. The analysis distinguishes between cursor-based approaches and ray-based approaches.

### 3.1.1 Cursor-based approaches

A simple feedback option for eye-rooted techniques is to draw a cursor/avatar somewhere along the ray $E+\lambda(H-E)$. In a monoscopic display, the value chosen for $\lambda > 0$ is irrelevant, but in a stereoscopic display this value plays a key role, as the resulting cursor parallax is proportional to $\lambda$ . The analysis considers a cursor drawn at zero parallax and a cursor matching the parallax of the indicated object (see Figure 3.1).

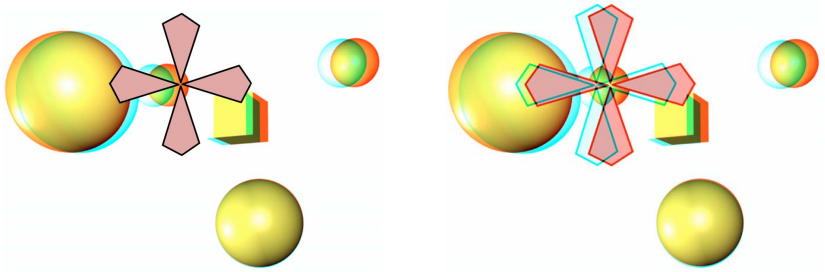



**Figure 3.1:** *Considered cursor-based visual feedback techniques in pseudo-anaglyph stereo. (a) 2D cursor, the cursor is always displayed at zero parallax. (b) 3D cursor, the cursor's parallax matches the indicated object.*

#### 2D cursors

A natural option is to draw a cursor at the depth given by the screen surface, which results in a cursor at zero parallax. Drawing an element at zero parallax condition has the unique advantage of being the only situation were the breakdown between eye accommodation and eye vergence does not occur.

Design variables for 2D cursors include cursor shape, cursor size, size of the active region [120], and visibility priority (always on top vs. normal rendering). For our analysis, we consider a cross-shaped 2D cursor covering about 5% of the screen, whose active region was defined implicitly by the selection ray (see Figure 3.1a). The cursor is drawn with depth test disabled to avoid close-up objects from occluding it, thus guaranteeing its visibility under all circumstances.

Our analysis on suitability of 2D cursors for accurate pointing on stereoscopic displays follows. We start analyzing potential depth cue conflicts. Since the 2D cursor is drawn always on top of the other objects, it will occlude all the objects whose screen projection overlaps with that of the cursor. This results in a depth cue conflict between interposition and stereopsis: objects in front of the screen will exhibit a negative parallax value, whereas they might appear partially occluded by the cursor which is perceived on the screen plane. It is well known that this disagreement hinders depth-ordering judgments and thus might have a potential negative impact on selection performance.

We now analyze the quality of the feedback provided by 2D cursors in terms of allowing users to univocally identify the object being pointed to. On a monoscopic display, a 2D cursor univocally identifies the object it is drawn on top of. However, objects displayed on stereoscopic displays have a pair of screen projections, one for each eye. The left and right eyes will be referred to as $E_L$ and $E_R$, and the left and right screen projections of an object will be referred to as $L$ and $R$, respectively. As we shall see, placing the cursor on top of only one of the two screen projections of an object does not guarantee the corresponding selection ray to intersect the object. Conversely, a cursor drawn outside the $L$ and $R$ projections can intersect the object.

Figure 3.2 illustrates when these situations might occur. For the sake of clarity, we consider the virtual object A (represented as a sphere) to be fixed, and we move the projection screen (or the virtual image plane, for the case of HMDs) towards or away from the user. The figure shows the three shadow volumes defined by joining $E_L$, $E_R$ and $E$ (the origin of the selection tool) with the object's silhouette. The selection ray intersects the virtual objects only if it is contained inside the volume defined by $E$ (dashed areas at the right of the figure). Considering intersection events among the three volumes, we can distinguish six different sections where the projection screen can be placed with respect to the virtual object.

The object would exhibit negative parallax in sections $S_1^-$, $S_2^-$, $S_3^-$ (the screen is behind the object), and the object would exhibit positive parallax in sections $S_1^+$, $S_2^+$, $S_3^+$ (the screen is in front of the object). In $S_1^+$, $S_1^-$ sections the $L$ and $R$ screen projections of the object partially overlap. This situation can be slightly confusing, since placing the cursor on top of

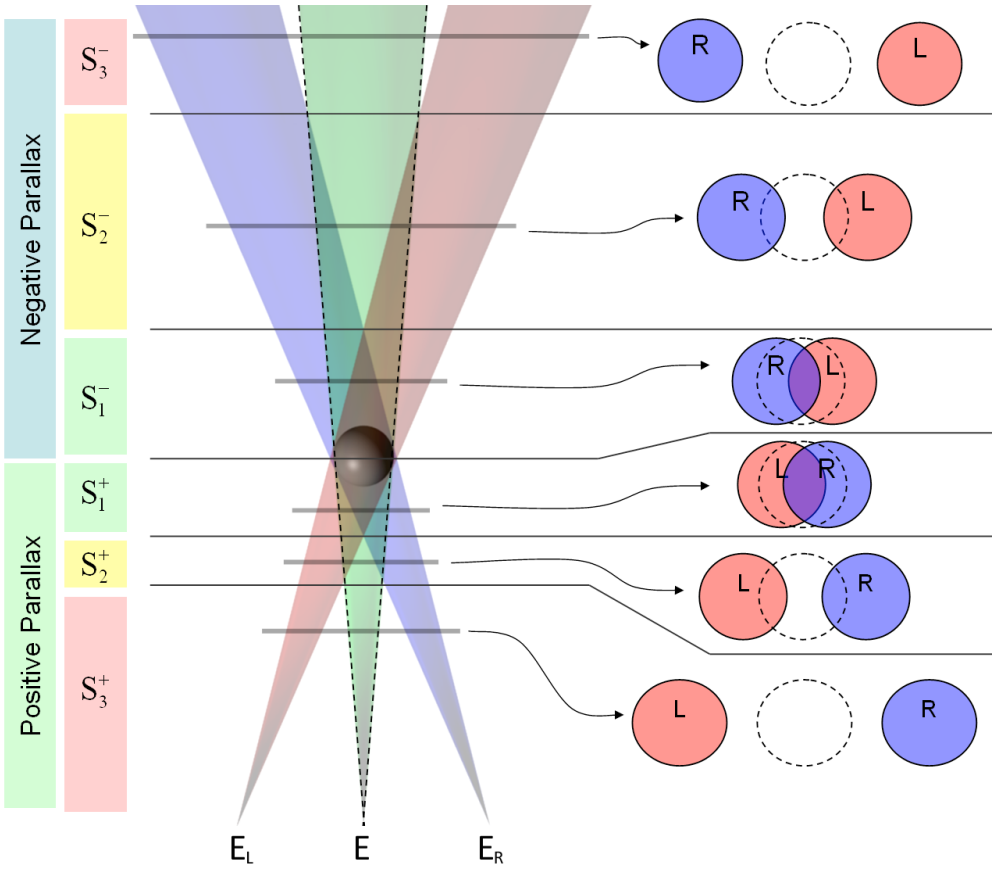



**Figure 3.2:** *Six different configurations of the L and R screen projections of an object and the selection area (dashed circle) for a 2D cursor. The position of the projection screen where these situations occur is shown on the left. The virtual object is the sphere at the intersection of the three cones.*

only one of the two screen projections of the object does not guarantee its intersection. In sections $S_2^+$ and $S_2^-$ the $L$ and $R$ screen projections are disjoint, although the selection area (i.e. the set of positions where the ray represented by a 2D cursor at that position intersects the object) partially overlaps $L$ and $R$. This situation can be more distracting than the previous one. Again, placing the cursor on top of only one of the two screen projections of the object does not guarantee its intersection. Furthermore, when the intended target is another object $B$ whose projection overlaps with the dashed circle, users might think that placing the 2D cursor at the dashed circle would allow them to select object B instead of object A. This situation is even worse in $S_3^+$ and $S_3^-$ sections, where the three projections are disjoint. In this case, there is no way to select object $A$ by placing the 2D cursor over $L$ or $R$. Instead, the user must judge the midpoint between both projections and point to that direction. Since $L$ and $R$ can be several centimeters apart, the presence of nearby objects overlapping the dashed circle might be distracting to the user.

Another key point is how a 2D cursor helps the user to judge the movement required to approach a particular target. We distinguish between the initial ballistic phase and the

corrective phase of the selection movement. Executing the ballistic movement requires the user to judge the displacement between the current location of the cursor and the screen projection of the object. A 2D cursor provides adequate feedback for this phase, as the above displacement is parallel to the viewing plane and hence it does not rely on depth cues. For the corrective phase, the separation between $L$ and $R$ comes into play and users might doubt about where to place the cursor along the segment joining $L$ and $R$ projections.

### 3D cursors

Given the poor visual feedback provided by a cursor at zero parallax condition, an alternative option is to draw the cursor at the depth defined by the first intersection point with the scene ($Q$). The user will perceive the cursor attached to the object being pointed to [113] (see Figure 3.1b). Notice that drawing the cursor at $Q$ results in varying parallax values.

In terms of application performance, drawing such a 3D cursor does not require the application to identify the object being intersected. Point $Q$ can be easily obtained by computing the pixel the selection ray projects onto and querying the associated depth value in the depth buffer.

A new design variable for 3D cursors is the projection type (perspective or orthographic). A 3D cursor rendered with perspective projection results in an arbitrarily small on-screen cursor for distant objects, making this option completely inappropriate for scenes with a large depth range. For the analysis we considered a cross-shaped orthographic 3D cursor covering about a 5% of the screen, whose active region was defined implicitly by the selection ray. The cursor has to be drawn with the depth test disabled to avoid close-up objects from occluding it, thus guaranteeing its complete visibility under all circumstances. The only difference with respect to the 2D cursor is the varying parallax value of its $L$ and $R$ screen projections.

An additional design decision is how to compute the cursor's depth when the selection ray does not intersect any object and thus point $Q$ is undefined. After some experimentation, we found the best option in case of no intersection is to preserve the last valid depth. This option minimizes parallax changes during corrective movements.

We start analyzing potential depth cue conflicts. It turns out that using a 3D cursor drawn on top of the object pointed to might result in up to three depth cues to be in disagreement (besides accommodation depth cues). On the one hand, parallax values will induce the user to perceive the cursor on top of the pointed object. On the other hand, since 3D cursor is drawn always on top of the other objects, it will occlude all the objects whose screen

projection overlaps with that of the cursor. Furthermore, since orthographic projection is used instead of perspective projection, the size of the screen projection of the cursor remains constant, thus the relative size depth cue would suggest that the cursor has not changed its depth. These depth cue conflicts, considering the frequency at which a moving 3D cursor might point to another object and hence results in a parallax change, is expected to have a potential negative impact on selection performance.

We now analyze 3D cursors in terms of allowing users to identify the object being pointed to. For a static 3D cursor, no identification problems are expected, as the cursor is drawn on the surface of the object and hence the $E_L$ and $E_R$ screen projection of the object overlaps with the $L$ and $R$ screen projection of the cursor. However, for a dynamic cursor where the object pointed to changes at high rates, the depth cue conflicts described above are also expected to distract the user.

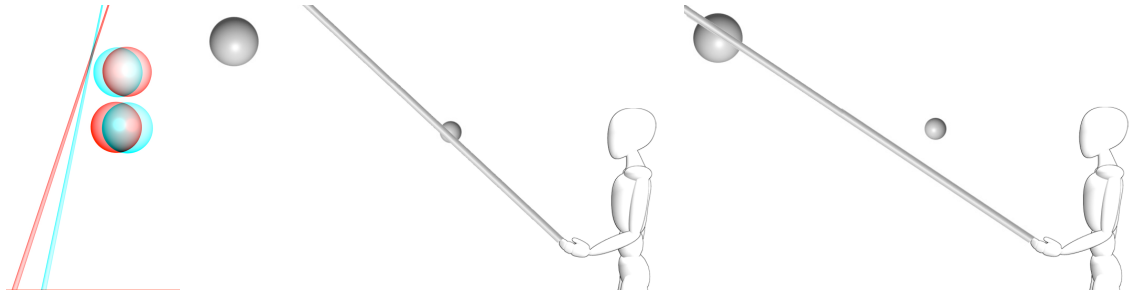
Another difference is how a 3D cursor helps the user to judge the movement required to approach a particular target. For the ballistic movement, a 3D cursor behaves like a 2D cursor. For the corrective phase, though, depth cues conflicts come into play. The worst case scenario is a small target whose screen projection is surrounded by other objects at a different depth, causing the 3D cursor to switch depth and parallax during corrective movements.

### 3.1.2 Ray-based approaches

The most used visual feedback technique for ray-based approaches is to draw a ray originating from the user's hand, following a laser beam metaphor. The first intersection point of the virtual ray with the environment determines the int object. This technique assumes a selection ray controlled by the position and the orientation of a hand-held device. The visual feedback consists simply in drawing this selection ray, which does not present any apparent depth cue conflict. Unfortunately, the misalignment between the user's viewpoint and the origin of the selection tool limits its performance during the corrective phase.

A first issue is related to the indication of the movement required to approach a particular target. In this respect, a hand-extending ray provides imprecise visual feedback to judge the vertical misalignment of the ray, as it forces the user to rely on depth cues rather than on stronger visual cues such as those provided by a cursor moving parallel to the viewing plane.

This problem is illustrated in Figure 3.3. Aiming at any of the two spheres clearly requires rotating the ray slightly to the right. However, judging which one of the spheres (if any)

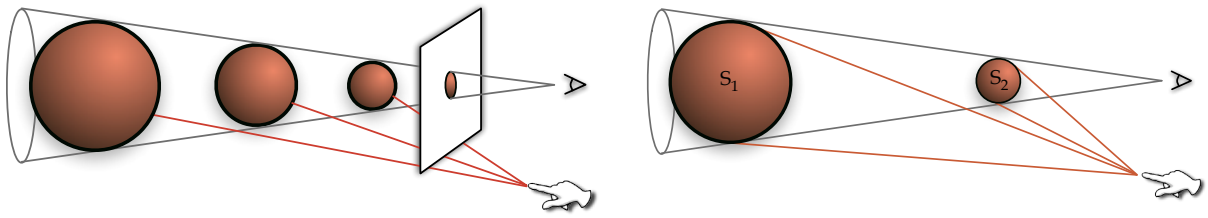


(a) (b) (c)

**Figure 3.3:** *Consider the situation depicted in (a). The judgment of the vertical alignment of the ray with respect to the targets relies solely on depth perception. Although the real layout is depicted in (b), errors on depth perception may induce the user to believe that the real layout is the one shown in (c).*

would be intersected after the rotation relies solely on depth cues. The ray seems to be closer to the sphere in the top, fact which can be misinterpreted as the situation depicted in Figure 3.3c. However, in this case the ray would intersect the sphere at the bottom (Figure 3.3b). Notice that the parallax of the sphere in the bottom matches the parallax of the ray at its closest point on the ray. As a result, factors affecting depth perception, such as ghosting effects produced by poor stereo pair separation will impact on user performance.

A second issue derives from the fact that objects are selected by pointing directly to them, which differs from cursor based techniques where objects are selected by pointing directly to their screen projections. A first consequence is that objects with exactly the same screen projection might require different ray orientations to select them (see Figure 3.4a).




(a) (b)

**Figure 3.4:** *(a) Three objects with increasing size and distance can have the same screen projection, but require notably different ray directions for their selection. (b) Objects with the same screen projection subtend different solid angles with respect to the hand position.*

Indeed, the accuracy required to select objects with raycasting is not given directly by their screen projections. Let $\Omega_E(S)$ and $\Omega_H(S)$ be the solid angle subtended by an object $S$ with respect to the user's eye and the user's hand, respectively. Considering Fitts' law, $\Omega_H(S)$ in absence of occluding objects is a good measure of the size ($W$) of $S$ and thus a measure of how much accuracy is required to select that object. $\Omega_E(S)$ is proportional to the (area of the) screen projection of $S$. Figure 3.4b shows two objects $S_1$, $S_2$ with the same screen

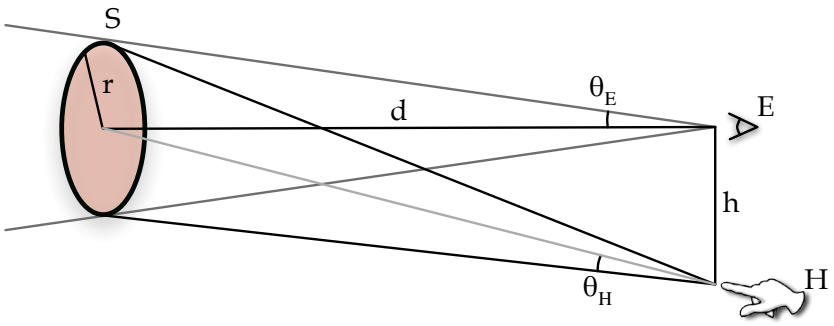



**Figure 3.5:** *Notation used in the computation of the solid angle for a 3D object*

projection but different solid angles, $\Omega_H(S_1) > \Omega_H(S_2)$.

Consider Figure 3.5, the accuracy $\Omega_H(S)$ required for selecting an object $S$ varies depending on its distance ($d$) to the eye $E$ and the distance ($h$) between the user's viewpoint $E$ and user's hand $H$ (see Figure and Equation 3.1)). Assuming that $H$ is vertically aligned with $E$ at distance $h$, if the radius $r$ of $S$ is uniformly scaled with respect to $E$, $\Omega_E(S)$ is preserved. For the sake of simplicity we consider the bounding sphere of an object $S$ located in front of $E$.

$$\begin{aligned} \Omega_H(S) &= 2\pi(1 - cos(\theta_H)) \\ \theta_H &= arcsin(\frac{r}{\sqrt{d^2 + h^2}}) \end{aligned} \tag{3.1}$$

Figure 3.6 shows a plot of the ratio $\Omega_H(S)/\Omega_E(S)$ as we increase the distance $d$ while preserving the screen projection of S (and thus $\Omega_E(S)$), for $h$ varying between 0 cm and 80 cm. This means that $r = d\sin(\theta_E)$, where $\theta_E = \arccos(1 - \Omega_E(S)/(2\pi))$ is the planar angle of the cone defined by $S$ and $E$. Obviously, $h = 0$ indicates that the position of the hand and the eye are the same thus the ratio is 1. As the eye-hand distance increases, closer objects exhibit more mismatch than distant ones, although having the same screen projection. For $h = 80$ cm, which is a typical eye-hand distance when the user is standing up, an object standing at 70 cm from the user will subtend with respect to the hand only a 50% of its solid angle with respect to the eye. Nevertheless, the ratio $\Omega_H(S)/\Omega_E(S)$ converges quite quickly to 1: objects at 2.3 m from the user expose to the hand the 90% of their solid angle with respect to the eye.

Let us now discuss the potential effects of the above fact on selection performance, assuming $h > 0$, in terms of Fitts' Law. On the one hand, if we scale down a single object with respect to the eye position (thus preserving its screen projection), we are clearly decreasing the size $W$ and indirectly the precision needed to select it increases. However, if we scale down the

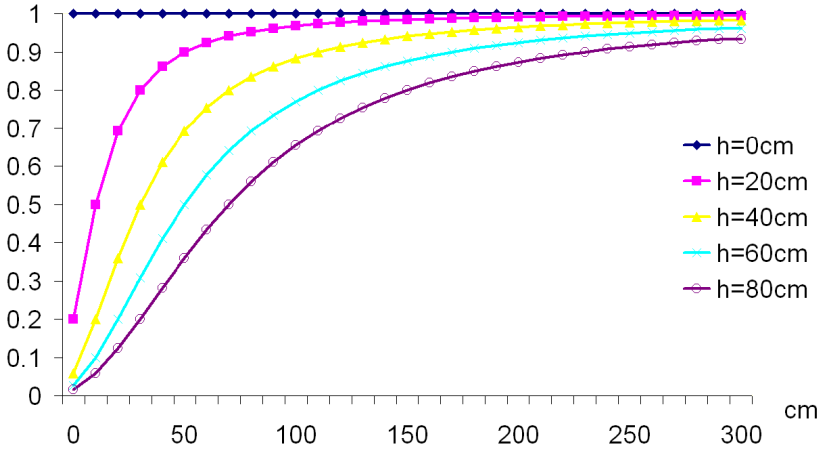



**Figure 3.6:** *Evolution of the ratio $\Omega_H(S)/\Omega_E(S)$ when the distance to $S$ increases from 10 cm to 3 m, while preserving its screen projection, for several eye-to-hand distances $h$.*

whole scene while preserving its screen projection, we are decreasing both and $W$ and $D$, which in essence means we are reducing the CD ratio while preserving the index of difficulty of the task [4], $ID = log_2(D/W + 1)$. As stated earlier, scaling down the scene means that more accuracy is required to fix the ray on top of the target, but also that the amplitude of the required movements is shorter, so at the end the effect of such a change of the CD ratio on performance is quite unpredictable.

In any case, we see this as a flaw of raycasting techniques in comparison with image-plane techniques, as the object's screen projection of a target does not convey appropriate visual feedback about the difficulty of the selection task.

### Eye-hand visibility mismatch

A fact that has been largely ignored in the literature is that whenever the selection ray originates at a point other than the user's eye, the set of visible objects which appear unoccluded from the user's eye position and selectable objects which appear unoccluded from the user's hand position, might differ.

An object $S$ is visible if at least one point in its boundary surface ($\partial S$) is visible from the user's viewpoint and is selectable if at least one point is visible from the user's hand viewpoint. Let $\nu_E$ be the set of visible objects and $\nu_H$ the set of selectable objects. The set of objects $\nu_E \cup \nu_H$ can be decomposed into three disjoint sets: $\nu_E \cap \nu_H$, $\nu_E - \nu_H$ and $\nu_H - \nu_E$.

Let us first discuss the behavior of raycasting selection for objects in $\nu_H - \nu_E$. This correspond to objects which are hidden to the user's eyes but are reachable from a ray emanating from the user's hand (Figure 3.7a illustrates this situation). Object $B$ is occluded from the eye

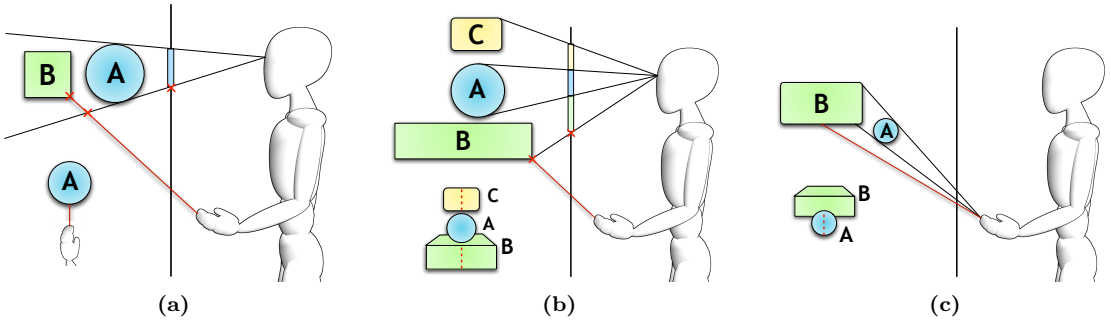



**Figure 3.7:** *Eye-hand visibility mismatch issues. (a) The user can selected an object which is hidden by another object. The last visible point on the ray, is projected over the screen projection of the occluding object, leading to misinterpretation: the ray appears to intersect object A, although the intersected object is behind. (b) A visible object cannot be selected because it cannot be reached by a ray emanating from the user's hand. The dotted line shows the path followed by the ray-scene intersection as seen on the screen, which skips object A. (c) Object A is visible from both E and H, but no point on its boundary is simultaneously visible from E and H.*

position but not from the hand position. Therefore it is currently selectable even though it is not visible. This phenomenon is particularly distracting to the user. The user might think that the currently selected object is $A$, as there is an apparent intersection of the ray with object $A$ (the screen projection of the last visible point of the ray is on the silhouette of the screen projection of $A$). In the absence of additional feedback, if the user triggers the selection confirmation at this point, the hidden object $B$ would be erroneously selected.

Let us now examine the selection behavior for objects in $\nu_E - \nu_H$. This corresponds to objects which are visible but which are not reachable from a ray emanating from the hand (Figure 3.7b illustrates this situation). Object $A$ is visible from the eye position but it is completely obscured from the hand position. Therefore, although being visible, object $A$ cannot be selected while preserving the hand position. Unlike the above case, the screen projection of endpoint of the virtual ray $Q$ appears over the projection of the currently selected object $B$, so there is no room for misinterpretation of the selected object. However, the user would be unable to select object $A$ unless he changes the origin of the pointing tool. If the user only moves the ray upwards, increasing the elevation of the ray trying to bring $Q$ to the screen projection of $A$, $Q$ seen would jump from object $B$ to object $C$. The discontinuous path followed by $Q$ on the screen is shown in Figure 3.7b bottom left. Note that $Q$ skips the regions occupied by the screen projection of $A$. Some users might perceive this unexpected effect as an anomalous behavior, once they realize that the object is not accessible, they have to move its hand to an unoccluded position.

In the last group $\nu_E \cap \nu_H$, a more accurate approach to predict potential selection problems is to consider the solid angles $\Omega_E(S)$ and $\Omega_H(S)$. An object $S$ is a potentially difficult target whenever $\Omega_E(S)$ or $\Omega_H(S)$ is below a threshold. But, as Figure 3.7c depicts, this approach

is still inaccurate, as an object $S$ with large $\Omega_E(S)$ and $\Omega_H(S)$ can still be difficult to select. Now, both $\Omega_E(A)$ and $\Omega_H(A)$ are large, but object $A$ is still difficult to select because no point in the boundary of $A$ is simultaneously visible from $E$ and $H$. As a consequence, the user can intersect object $A$ with the ray, but the intersection point is hidden by object $B$, keeping the user from having visual feedback. Therefore, a more accurate measure of the difficulty/accuracy required to select an object $S$ must be defined in terms of its simultaneous visibility. Given an object $S$, we define $\Omega_{E\times H}(S)$ as the solid angle subtended by the points of $S$ which are simultaneously visible from $E$ and $H$.

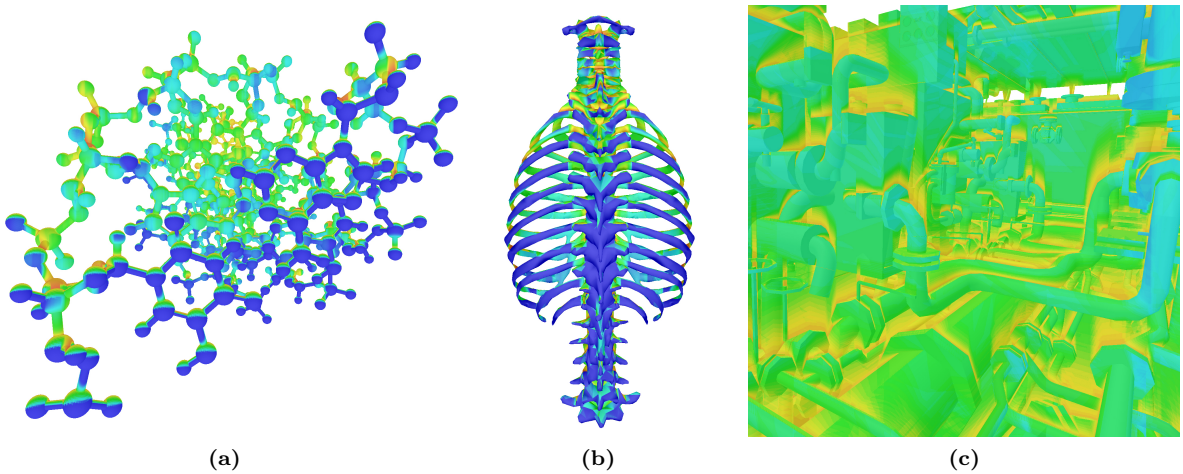
The impact on performance of the visibility mismatch depends, among other factors, on the ratio $\Omega_{E\times H}(S)/\Omega_E(S)$. In sparsely occluded scenes, we could expect this ratio to be roughly 1; the same applies when the eye-to-hand distance $h$ approaches zero. However, when $h$ increases, self-occlusion obviously increases, as more front faces from $E$ become back faces from $H$ and vice versa. But, the behavior of the inter-object occlusion is more difficult to predict, as it strongly depends on the object layout and user's position $E$. Assuming objects uniformly distributed around 3D space, one could expect $\Omega_{E\times H}(S)$ to decrease on average as $h$ increases, but many 3D scenes do not exhibit such a uniform layout.

Figures 3.8 and 3.9 show the empirical evaluation of visibility mismatch for several test scenes. For practical reasons, in Figure 3.9 we approximate $\Omega_{E\times H}(S)/\Omega_E(S)$ as the percentage of pixels in the screen projection of $S$ which are also visible from $H$. The results show that $\Omega_{E\times H}(S)$ tends to decrease as $h$ increases, although not uniformly. Note that visibility mismatch rapidly affects a large part of the scene, (around 25% of the visible pixels correspond to parts difficult to select), for typical values of the angle $\alpha = \widehat{ECH}$ with $C$ being the center of the scene.

## 3.2 Eye-Hand visibility mismatch evaluation

The theoretical study of the eye-hand visibility mismatch exposed an ignored issue which can potentially decrease selection performance when using hand-rooted techniques. In order to prove our theoretical analysis, we performed a user evaluation to explore in a controlled environment the impact of eye-hand visibility mismatch on 3D object selection tasks.

Eye-hand visibility mismatch is expected to increase the difficulty of selection tasks, but the resulting difference might be non-significant. For instance, users can develop strategies to compensate the visibility mismatch, e.g. by moving their hand quickly to a dissocluded location or anticipating occlusion.



(a) (b) (c)

**Figure 3.8:** *Simultaneous visibility on three test models. The color temperature represents the eye-to-hand distance at which each point appears occluded from $H$. Dark blue parts correspond to points whose simultaneously visibility is preserved for all $\alpha = \widehat{ECH}$ in [0, 60]. A warmer color indicates that even a small increase in $\alpha$ causes the point to become occluded from $H$.*

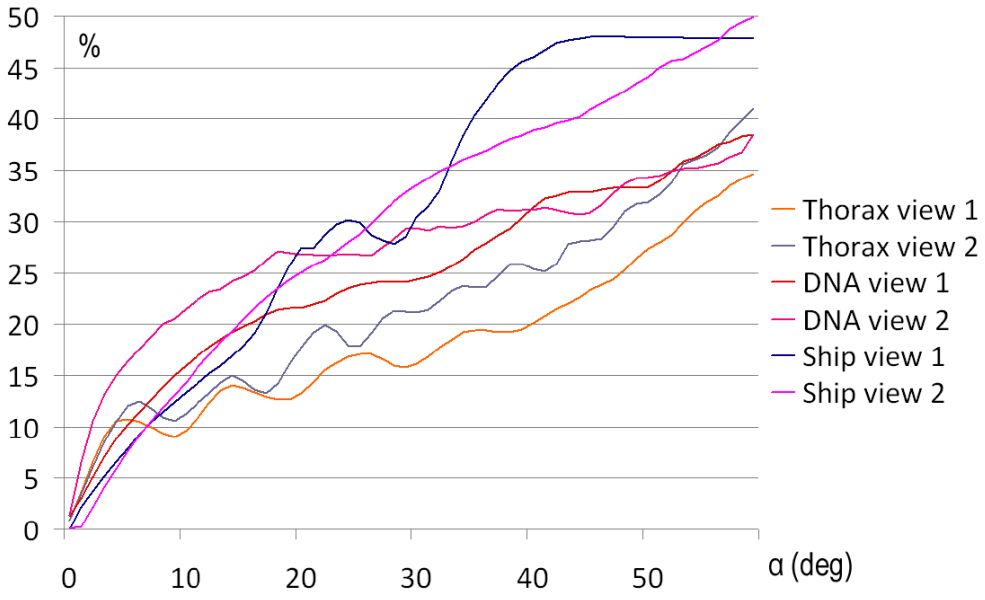



**Figure 3.9:** *Simultaneous visibility on three test models, for angle $\alpha = \widehat{ECH}$ in $[0, 60]$. The value shown is the percentage of pixels in the screen projection which are not visible from $H$.*

For completeness, we performed a Fitts' law analysis of the obtained data. We studied the correlation between selection time and the index of difficulty ($MT = a + bID$, being $ID = \log_2(A/W + 1)$). As the ray is controlled by hand rotations, considering Euclidean distances to compute $D$ and $W$ is not valid. Instead, we compute the amplitude of movement $A$ as the minimum rotation angle required for the selection ray to hit the target from the start position and the target size $W$ is computed as the apex angle of the minimum cone with apex in the user's hand containing the target (our targets were spheres). We controlled the index of difficulty and the occlusion level on a per-object basis. The correlation obtained will

determine if raycasting selection can be modeled using Fitts' law and which is the impact of the eye-hand visibility mismatch.

Twelve volunteers, aged from 22 to 35, participated in the experiment; 6 participants had no experience with VE applications; 3 had some experience and 2 were experienced users.

### Design and procedure

Users were requested to select a sequence of objects as quickly as possible, where the next object to be selected was clearly highlighted. To provide additional visual feedback, the object intersected by the selection ray was highlighted in red. Two different test model were employed, each one having seventeen spheres with varying size (from 0.2 m to 60 m) and distances (0.6 m to 70 m), placed in such a manner that all spheres had approximately the same screen projection (since users were head-tracked, actual screen projection varied according to user's head movements).

In the first test model, spheres were lay out so as to avoid inter-object occlusion from the eye and the hand position, with nearby spheres appearing above farther ones, from the user's perspective (see Figure 3.10a). We will refer to this model as the *unoccluded* model. In the second test model (see Figure 3.10b), spheres were lay out so as to create varying levels of inter-object occlusion from the user's hand position but not from the user's eye position (all spheres were clearly visible so as to keep discovery time from altering the results). The second model will be referred to as the *occluded* model.

The index of difficulty of targets ranged from 2 to 3 bits. This low range was due to the fact that all objects had the same target width. Objects located at the center had lower IDs, as in average, the amount of amplitude required to select them was lower.

To ensure that the difficulty of all trials remained constant among users, we precomputed a random sequence of objects the user had to select. The same sequence was used for all the trials, each consisting in 300 selections.

A repeated-measures within-subjects design was used. The independent variable was the test model (occluded, unoccluded). Each participant performed the experiment in one session lasting approximately 25 minutes. The experiment was divided into two blocks, one for each model. Before each block users were provided with a very short (1 min) training session. The order of the blocks was counterbalanced among users to avoid learning effects.

The dependent measures were total selection time, error rate and focus changes. The error

rate was measured by counting the number of erroneous clicks and the focus changes was the number of times the target object changed its selection status prior to confirming its selection.

### Apparatus

The experiment was conducted on a four-sided CAVE with stereo projectors at $1280 \times 1280$ resolution. The input device was a 6-DOF Wanda and a tracking system providing 60 updates/s with 4 ms latency. At the user position (90 cm from the EM emitter), position and orientation RMS errors were below 0.5 mm and 0.06 degrees, respectively. Users were provided with a wireless mouse to trigger the selection confirmation to avoid the Heisenberg effect and reduce the number of errors. The experiment was driven by a cluster of 2.66GHz QuadCore PCs with NVIDIA Quadro FX 5500 cards.

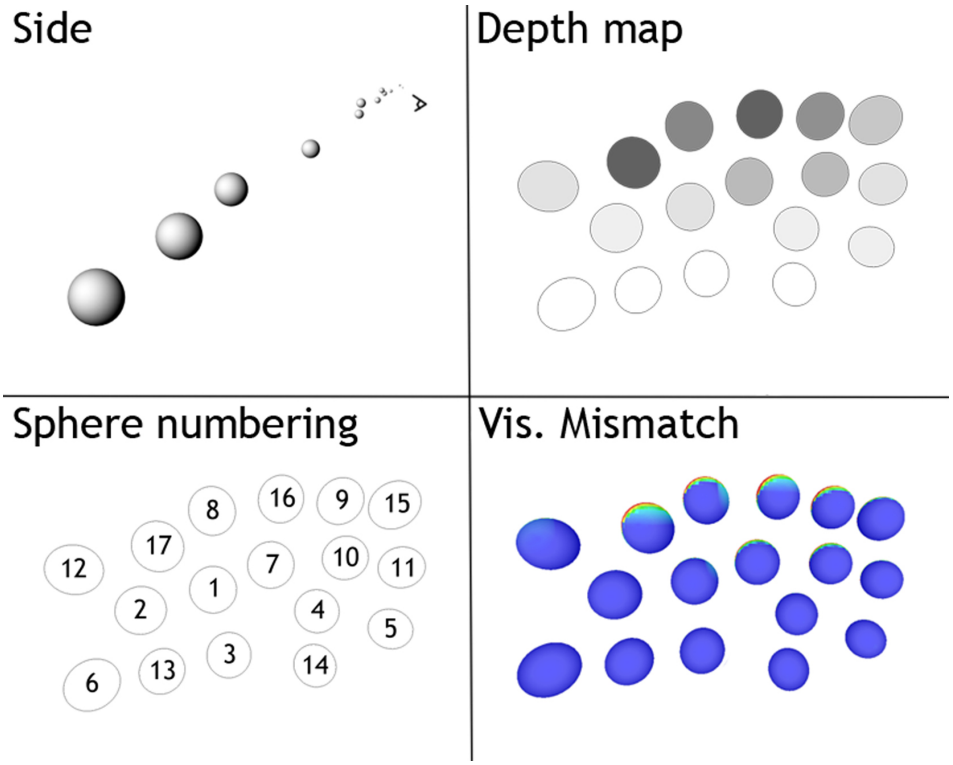
### Results

Average raycasting selection time for the unoccluded model was 264 s (0.88 s per object), whereas for the occluded model average time was 319 s (1.06 s per object) (see Figure 3.11a). The one-way ANOVA of selection time vs. test model confirmed the significant effect of the test model on selection performance ($p < 0.005; F = 10.32$). Since the screen-projection of the spheres in both models was roughly the same, this result seems to confirm our hypothesis that, for raycasting selection, the eye-hand visibility mismatch has a significant impact on selection time.

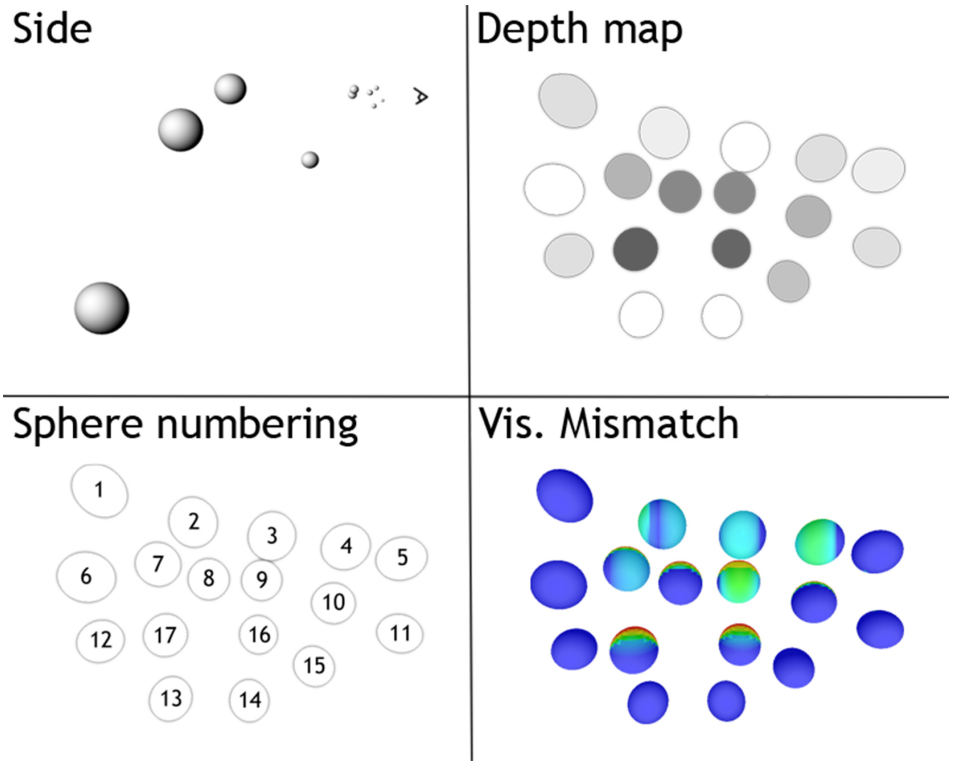
For erroneous click (see Figure 3.11b), the ANOVA of errors vs. test model did not found any significant difference, once the user has been able to point to the target, the eye-hand visibility mismatch does not pose additional hindrances.

Results on focus changes are shown in Figure 3.11c. The one-way ANOVA focus changes vs test model also found significant differences ($p < 0.05; F = 12.82$). The number of focus changes is an indication of how many times the user attempted to point at an object prior to selecting it. As expected eye-hand visibility mismatch had a negative impact.

We condensed the selection time data gathered in a per-object basis to perform a regression analysis of the selection time versus the index of difficulty. For each object we computed its mean selection time and its mean index of difficulty. The correlation of the data for the unoccluded model was high $r = 0.898$ for (see Figure 3.12a), but no correlation was found for the occluded model. A per-object study showed that objects 2, 3, 7 and 9 (see

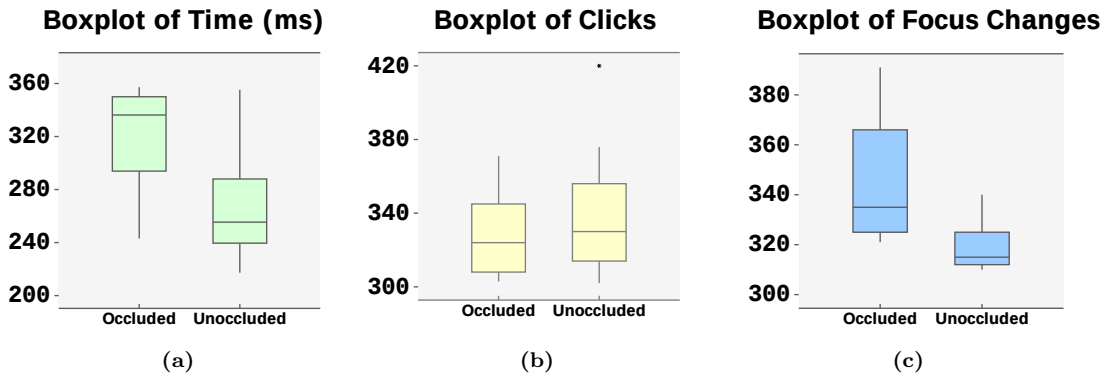



(a)




(b)

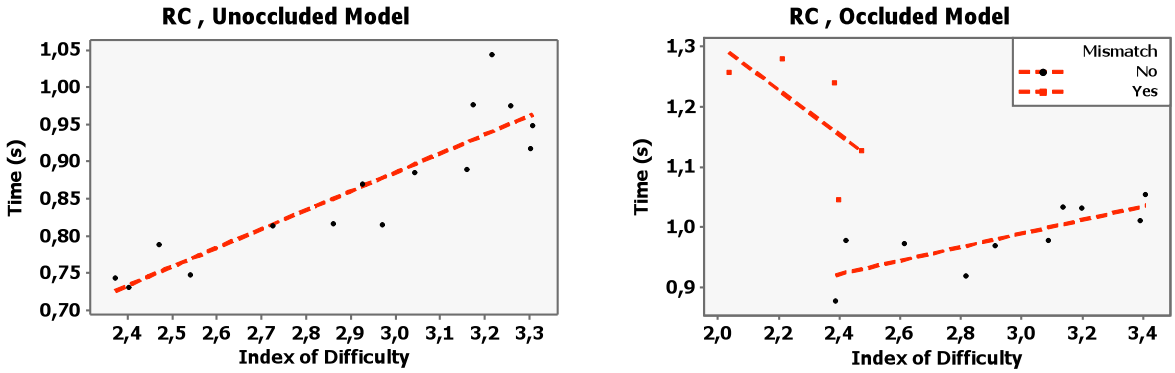
**Figure 3.10:** *Test models used in the study (a) unoccluded, (b) occluded. Color temperature encodes the potential level of mismatch. Only the spheres from 1 to 16 where potential targets; sphere 17 was introduced solely to increase mismatch. While the mismatch in (a) was due to object self-occlusion in (b) it was also due to object inter-occlusion.*




(a) (b) (c)

**Figure 3.11:** *Boxplots of selection time (s), erroneous clicks and focus changes for each test model.*

Figure 3.10b) did not adjust well to the expected positive correlation. We repeated the regression analysis ignoring these objects and then, we found a relatively high correlation ($r = 0.755$). Interestingly, when we contrasted these objects with those exhibiting a high level of eye-hand visibility mismatch, we found a perfect match (see Figure 3.12b). Selection tasks not affected by the eye-hand visibility mismatch still present a Fitts' law behavior.




**(a)** *Unoccluded Model* **(b)** *Occluded Model*

**Figure 3.12:** *Scatterplot of mean object selection time vs index of difficulty. Correlation values are: (a) 0.898 and (b) 0.755 (this last one has been computed ignoring spheres with large visibility mismatch).*

## Discussion

We have explored analytically and empirically how the eye-hand visibility mismatch hinders the selection of 3D objects. Both conclude that the eye-hand visibility mismatch has a significant impact on user's performance. However, the performance drop largely depends on the scene complexity and on the eye-to-hand distance with respect to the scene size. In addition to performance issues, the user might be forced to align its viewing and pointing direction to achieve the selection which results in uncomfortable pointing gestures.

The Fitts' law analysis showed that raycasting in the absence of the eye-hand visibility mismatch follows a Fitts' law model. Although the range of the IDs was low (2.4 to 3.4 bits), the correlation between selection time and the index of difficulty was high ($r = 0.899$). On the other hand, when the eye-hand visibility mismatch increases, the data did not longer follow the Fitts' law model, selection times were higher (see Figure 3.12b).

An open question remains though, can the eye-hand visibility mismatch be introduced into Fitts' law? Although we can estimate the amount of the eye-hand visibility mismatch, the cognitive load to overcome the mismatch cannot be ignored. Fitts' Law tasks require low cognitive load and repetitive tasks. In our experiment, users did not notice the eye-hand visibility mismatch until the ballistic movement was done. During corrective movements users realized that they could not point to the target and were forced to reposition its hand to find a suitable origin for the selection ray. In cluttered environments this might not be a trivial task. So, it is not clear whether Fitts' law can account for the eye-hand visibility mismatch for 3D object acquisition tasks.

Encouraged by these results, we proceeded to develop new selection techniques which better comply with our visual feedback analysis and avoid the eye-hand visibility mismatch. The following chapter details the achieved results in this area.

# Chapter 4

# Overcoming visual issues in virtual pointing tasks

The visual feedback analysis presented in the previous chapter concluded that visual feedback techniques have to allow users to univocally identify the object being pointed to and have to provide enough information regarding the direction and the amount of movement required to point to an object. Furthermore, we have shown that the eye-hand visibility mismatch hinders selection tasks and thus has to be avoided.

Our theoretical analysis revealed that widely employed selection techniques do not satisfy these requirements. Raycasting suffers from the eye-hand visibility mismatch and its visual feedback highly depends on the spatial understanding of the virtual environment, and visual feedback available for eye-rooted techniques are not well suited for stereoscopic vision.

The contributions presented in this chapter focus on the avoidance of the eye-hand visibility mismatch and the development of visual feedback techniques that better support plano-stereoscopic displays and collaborative referral tasks.

We propose a new selection technique where the selection tool is controlled by hand rotations as in raycasting, but emanates from the eye position like in occlusion selection. It combines the absence of eye-hand visibility mismatch of image-plane techniques with the benefits of ray control through hand rotations. Besides the new device-ray mapping, adequate visual feedback must be provided. We successfully developed two alternative visual feedback techniques which better comply with stereoscopic output devices. Our visual feedback analysis is completed by the evaluation of existing visual feedback techniques.

Moreover, in collaborative virtual environments, the viewpoint mismatch does not only affect selection tasks. Referral tasks are compromised as users do not share the same viewpoint (they can navigate freely in the environment). One user might be pointing to an object which

is occluded from the viewpoints of other users, hindering mutual exchange of information. We present a study exploring users' behavior in such situations and how existing occlusion management techniques, such as X-Ray vision, allows for improved referential awareness and better compliance with social protocols.

## 4.1 Raycasting from the eye (RCE)

Ray Casting from the eye is a new mapping for ray control which attempts to overcome the negative effects of the eye-hand visibility mismatch. The selection tool's origin in RCE is determined by the eye position ($E$), even though the orientation of the ray is still controlled by the hand orientation ($\vec{h}$). The selection ray is defined by the parametric equation $E + \lambda\vec{h}$. The first intersection of this ray with the scene objects indicates the selected object (see Figure 4.1).

In essence, RCE combines the benefits of image-plane techniques (absence of visibility mismatch and continuity of the ray movement in screen-space) with the benefits of ray control through hand rotation (requiring less physical hand movement from the user). In this sense, it can be viewed as a hybrid technique between raycasting and image-plane techniques. To the best of our knowledge, this mapping between user movements and pointing direction has never been explicitly proposed nor evaluated.

Since the selection ray is cast from the eye, the selection ray projects onto a single point in the viewing plane. As a consequence, we must choose another strategy distinct from drawing the selection ray for providing appropriate visual feedback. We discarded the 2D and 3D cursor strategies given all the limitations detailed in the previous chapter.

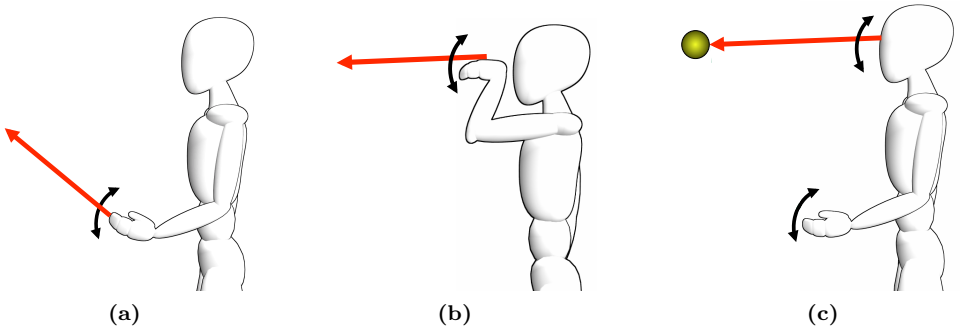



**Figure 4.1:** *In classic raycasting (a) the selection ray is cast from the user's hand, thus potentially suffering from eye-hand visibility mismatch. This problem persists unless users align their hand with the pointing direction (b), which results in an uncomfortable position. Ray casting from the eye (c) uses a selection ray cast from the eye, whose direction is controlled with the hand orientation. Since the selection ray is insensitive to hand position, users can select objects as in (b) but with less physical effort.*

### 4.1.1 Hand-to-cursor ray (HCR)

Our first solution to the visual feedback is to draw a virtual ray (the feedback ray) defined by equation $H + \lambda(Q - H)$, $Q$ being the first intersection point with the environment. More precisely, we only draw the segment $HQ$ (see Figure 4.2).

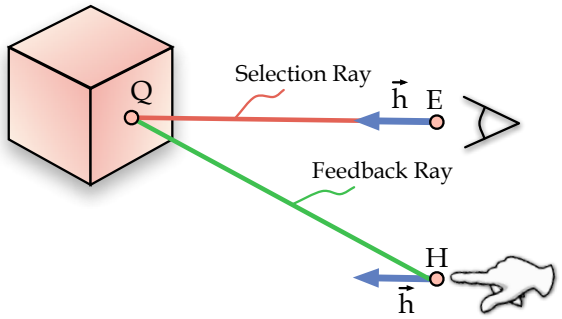



**Figure 4.2:** *As the selection ray projects into one single point, additional visual feedback is provided by drawing a ray cast from the hand to the first intersection of the selection ray with the scene.*

Similarly to 3D cursors, the parallax of the endpoint of the ray $Q$ also changes rapidly but the replacement of the 3D cursor by a ray notably alleviates convergence issues (in this respect RCE behaves like any raycasting variant). We called this visual feedback approach hand-to-cursor ray (HCR) as the virtual ray emanates from the user's hand and has its endpoint at the location which would be defined by a 3D cursor. We now state some properties of the feedback ray:

- Since the feedback ray originates at the hand position and responds to hand orientation, it feels like a normal raycasting ray. In fact, both techniques tend to be the same if the user aligns her hand with the pointing direction, as in Figure 4.1b.
- Unlike classic raycasting, the endpoint of the ray is insensitive to the hand position, and depends only on the hand orientation. When the user is standing up in a spatially-immersive display, this allows for a more comfortable pose, requiring only hand rotation gestures.
- The movement of the endpoint of the feedback ray is continuous ($Q$ behaves like a 3D cursor) and the screen projection of an object is a good measure of its effective size.

#### Limitations

In addition to the convergence issues due to fast changes of the endpoint $Q$, during pilot experiments, we noticed that some novice users had difficulties in selecting some objects due

to visual feedback conflicts. The worst case scenario for HCR seems to be the selection of close-up objects whose screen projection is surrounded by distant objects.

This situation is depicted in Figure 4.3. Suppose the user wants to select the blue sphere. Although both spheres have similar screen projections, the blue sphere is much smaller and closer to the viewer (the picture shows the two spheres with reduced depth disparity for clarity). Starting from the situation shown in the left, an untrained user might think that the action required to select the blue sphere involves intersecting it with the display ray. Since the ray is occluded by the blue sphere, the visual feedback indicates that the display ray is behind it, so the user might rotate his hand upwards (Figure 4.3, top). This movement causes the display ray to intersect the object, but this does not change the selection status for the green sphere (remember that the display ray is drawn solely for visual feedback, the pointed object being defined by the selection ray). Therefore the correct action to select the blue sphere is to rotate the hand downwards (Figure 4.3, bottom). Although this behavior may hinder selection performance during corrective phases for novice users, trained users by focusing at the endpoint of the feedback ray are able to mitigate its effects.

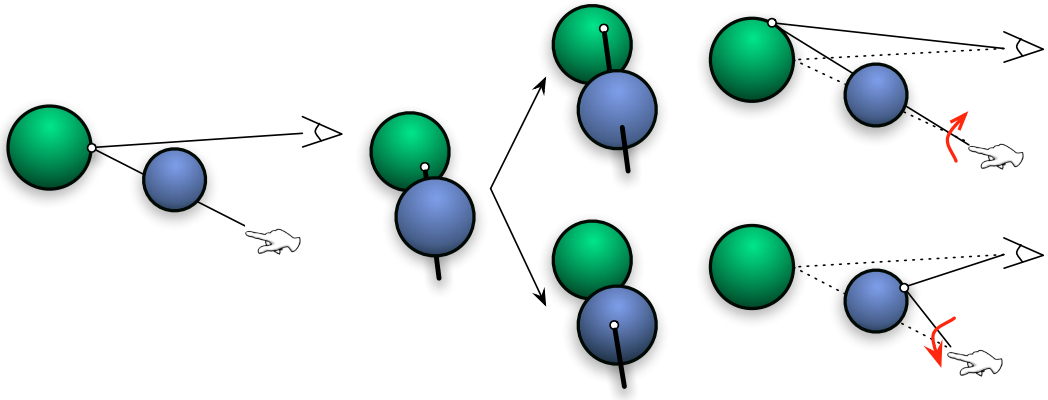


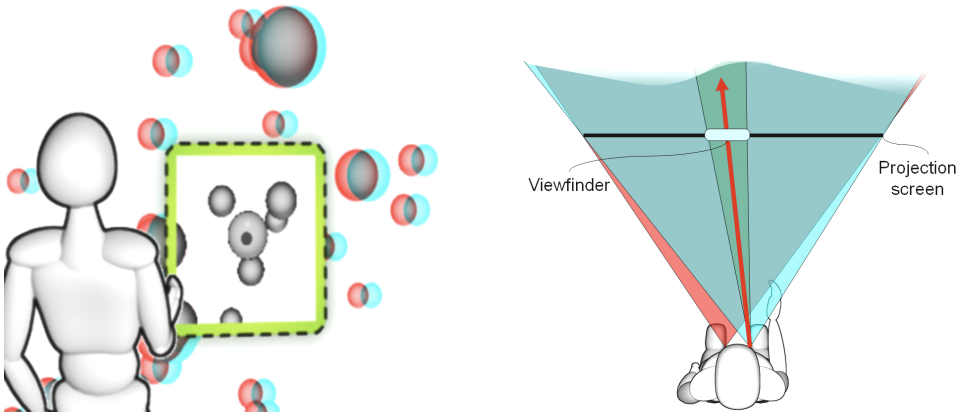
**Figure 4.3:** *Worst-case scenario for the hand-to-cursor ray.*

### 4.1.2 Viewfinder

The hand-to-cursor ray is a simple visual feedback approach but it still presents some of the limitations of 3D cursors. We now detail an improved visual feedback approach avoiding such limitations, the *viewfinder*.

The key idea is to locally flatten potential targets in the vicinity of the pointing direction by projecting them onto a small virtual screen attached to the pointing direction (see Figure 4.4a). We call this technique viewfinder because the resulting effect is similar to looking a small part of the scene through an LCD digital camera display. A 2D cursor on the middle viewfinder represents the pointing direction. Since the cursor and the objects displayed on

the viewfinder are drawn at fixed parallax, we avoid selection ambiguity problems that have discouraged the adoption of image-plane techniques in stereoscopic displays.




(a) (b)

**Figure 4.4:** *(a) User view of the viewfinder (not to scale). (b) Viewing pyramids of L/R eyes and the placement of the viewfinder.*

### Viewfinder placement

The center of the viewfinder is constrained by the pointing direction $(E+\lambda\vec{h})$. We now discuss two different options regarding the orientation of the viewfinder with respect to the pointing vector $\vec{h}$. In both cases the viewfinder is centered at the intersection of the selection ray with the projection screen. A first option is to keep the viewfinder parallel to the projection screen. This option has the advantage that the viewfinder's contents will appear exactly at zero parallax condition. A second option is to keep the viewfinder perpendicular to $\vec{h}$. In that case a part of the viewfinder's contents will appear with slightly negative parallax and another part with slightly positive parallax. In our experiments we used this second option as it imitates better the natural way of aiming at different parts of a scene with a hand-held camera, it is compatible with multi-screen systems such as CAVEs, and it preserves the solid angle subtended by the viewfinder with respect to the viewpoint, thus preserving its apparent square shape.

### Viewfinder frustum

The viewing frustum associated to the viewfinder is constructed from the dominant eye position. An additional design variable is the field-of-view (fov) of such frustum, which influences the size of the viewfinder. A small frustum has the advantage of minimizing the impact of the viewfinder on the immersive view and having a smaller application overhead. However, if too small, a potential target can enter and leave the viewfinder multiple times during the corrective phase of the selection, potentially distracting the user. Therefore the

viewfinder should be sized so as to ensure that, in most situations, it includes any given target after the initial ballistic movement to approach the target. We conducted a simple experiment to measure the deviation angle between the pointing vector and the vector joining the viewpoint with the object's center, just after the initial ballistic movement towards the target. In our particular VR system, the maximum deviation angle was found to be about four degrees. Thus in our experiments we decided to use a symmetric frustum with a fov of eight degrees and unit aspect ratio. For the near and far clipping planes we used the same values as in the immersive view.

### Drawing the viewfinder

The viewfinder itself is rendered as a textured quad perpendicular to the pointing direction, where the texture map contains the projection of the scene according to the viewing frustum defined above (Figure 4.4b). The texture map can be computed in several ways. A simple option with negligible overhead is to reuse the stereoscopic image generated for the dominant eye, by simply grabbing a rectangular region of the color buffer. OpenGL's *glCopyTexImage2D()* function can be user for that purpose. A second option is to setup the viewfinder's camera and render the scene to a texture. OpenGL's frame buffer objects can be used to minimize image data transfers. Since current scene graphs support hierarchical frustum culling, and the frustum of the viewfinder is a small fraction of the immersive viewing frustum, we expect a relatively insignificant overhead compared with that of generating the two stereoscopic images. Our prototype implementation uses the second rendering option, as it provides higher quality renderings. The same texture map is used for rendering the viewfinder both from the left and right eye positions.

Moreover, the pointing direction is indicated by drawing a small dot at the viewfinder's center (thus at zero parallax condition). The lack of depth cue conflicts in the viewfinder eliminates the need for a highly contrasted cursor. We also highlighted the boundary of the viewfinder by drawing a small frame around it, as depicted in Figure 4.4a.

### Comparison with previous approaches

Our viewfinder metaphor is related to through-the-lens approaches [114, 52]. The basic idea behind through-the-lens tools is to provide an additional view of the virtual environment, which is shown in a dedicated window, to support simultaneous exploration from two different viewpoints. Our approach differs from through-the-lens techniques proposed so far in several key points. First, the primary goal of our viewfinder is to provide accurate pointing by

avoiding depth cue conflicts and other feedback problems in stereo projections. Through-the-lens tools proposed so far focus on overcoming occlusion problems and bringing objects within arm reach for virtual hand manipulation. Second, our viewfinder is always attached to the pointing direction; as a consequence, the viewfinder's position, depth and orientation are computed automatically from the pointing direction, requiring no specific setup from the user. In through-the-lens approaches the window is either fixed with respect to the virtual environment, fixed in the image plane of the user, or mapped onto a tracked pad which can be moved independently [114]. Third, our viewfinder shows the scene from the user's viewpoint, with a viewing frustum which is a portion of the immersive frustum. This introduces no complexity on the view and leaves no room for confusion.

#### Viewfinder extensions

A simple extension of the VF metaphor to facilitate selection of small targets is to scale up the VF while preserving its fov, when the user holds the selection button for some time period (e.g. 500ms). Also the image displayed on the viewfinder can be enhanced with any post-process filter, e.g. highlighting selectable objects or applying through-the-lens approaches.

### 4.1.3 Evaluation of visual feedback techniques for virtual pointing

In the previous chapter we discussed theoretically the disadvantages of the most common visual feedback approaches for 3D object selection. In order to corroborate our findings, we conducted a user study to evaluate experimentally current device-ray mappings and visual feedback approaches for virtual pointing. Furthermore, we also evaluate the Raycasting from the Eye and the two proposed visual feedbacks.

One challenge of the experiment design was to cope with the multiple factors influencing visual feedback, including the origin of the pointing direction, its translational/rotational control, and the display device. We first conducted some pilots to choose a representative subset among all possible combinations. We choose the combinations so that pair-wise comparisons could be made in ceteris paribus condition (see Table 4.1). Regarding RCE, in addition to the two visual feedback detailed, the hand-to-ray cursor and the view finder, we also evaluated a 2D cursor. It was introduced for comparison purposes.

Other alternatives like modifying control-display ratio and volumetric tools were not considered, as they can be applied independently to all the studied techniques (although it is unclear if they provide similar improvements for all combinations).

| Mnemonic | Selection Technique | Origin | Orientation | Visual Feedback |
|---|---|---|---|---|
| RC | Raycasting | *Hand* | $\vec{h}$ | ray |
| EH2D | Occlusion Selection | *Eye* | $\overrightarrow{EH}$ | 2D Cursor |
| EH3D | Occlusion Selection | *Eye* | $\overrightarrow{EH}$ | 3D Cursor |
| HCR | RayCasting from the eye | *Eye* | $\vec{h}$ | ray |
| C2D | RayCasting from the eye | *Eye* | $\vec{h}$ | 2D Cursor |
| VF | RayCasting from the eye | *Eye* | $\vec{h}$ | ViewFinder |

**Table 4.1:** *Evaluated techniques*

### Design and procedure

Users were presented with an intensive selection task, where the next object to be selected was clearly highlighted. The virtual environment had one hundred spheres at different depths (from 1 m to 80 m), thus allowing us to evaluate the techniques on scenes with a large depth range. The size of the spheres was chosen so that all spheres had approximately the same screen projection, about $30 \times 30$ pixels, subtending about 1.5 degrees from the user's viewpoint. We did not include large targets as these are easy to select no matter which visual feedback technique is used. All targets were clearly visible from the user's viewpoint to simplify the discovery task. In order to ensure that the difficulty of all trials remained constant among users, we pre-computed a random sequence of targets. The same sequence was used in all the trials, each trial consisting in 50 selections.

A repeated-measures within-subjects design was used. The only independent variable was the selection technique (see Table 4.1). We provided a short (1 min) training session before each new technique and the order of the trials were randomized.

The dependent measures were selection time and erroneous clicks. According to the theoretical limitations of the hand-rooted techniques and the visual feedback techniques considered, we derived the following hypotheses about the results of the experiment:

- H1 : Lower mean of selection time for HCR as compared with RC.
- H2 : Lower mean of selection time for C2D as compared with EH2D.
- H3 : Lower mean of selection time for HCR as compared with C2D.
- H4 : Lower mean of selection time for VF as compared with HCR and C2D.

After each trial, users were asked to fill a short questionnaire to rate each technique using a 7-Likert scale. The questions were if they had experimented double vision, the level of

difficulty of the selection task, and if they perceived some kind of depth incoherence (depth cue conflicts).

### Apparatus

The experiment was conducted in a $2.5 \times 2$ rear-projected screen with passive stereo with $1024 \times 768$ pixel resolution. The input device was a 6-DOF Wand with an IS900 acoustic tracking system, providing 120 updates/s with $4ms$ latency. The experiment was driven by a 3.2GHz Core2Duo PC with a GF8800 GFX card.

### Participants

Twelve volunteers, aged from 21 to 35, participated in the experiment; 2 participants had no experience with VE applications; 8 had some experience and 2 were experienced users.

### Results

Figure 4.5 shows the boxplots of the data gathered during the experiment. The one-way ANOVA of selection time vs. technique showed significant differences among techniques ($p < 0.001; F = 31.25$). The following discussion is based on Tukey's pair-wise tests.

We first focus on the ray origin by comparing RC and HCR. Both techniques share rotational control and use a virtual ray emanating from the users' hand as visual feedback, they only differ in the origin of the selection tool. The Tukey's test showed that HCR was significantly faster than RC ($p < 0.001, 34\%$ faster), thus confirming H1.

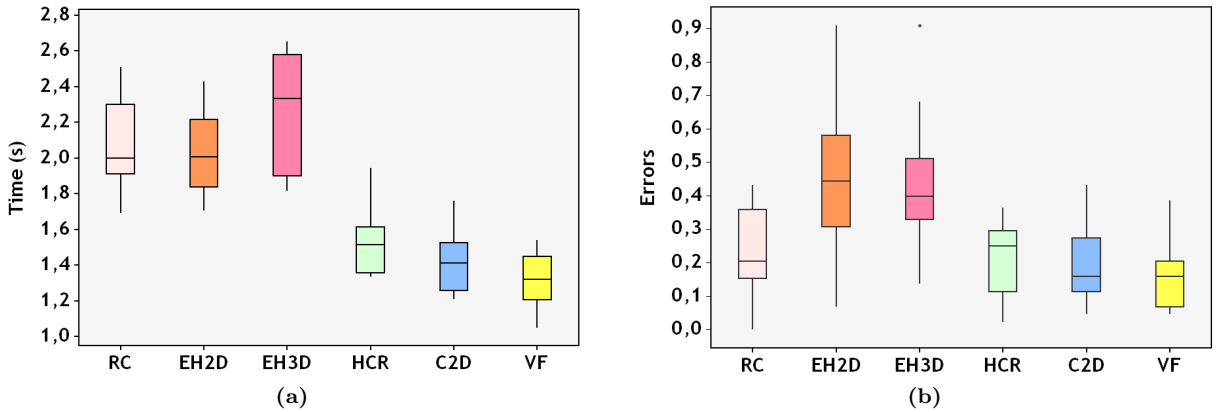



**Figure 4.5:** *Boxplots of mean selection time per object (a) and mean erroneous clicks per object (b).*

We now consider EH2D and C2D which only differ in the selection tool control: *translational control* vs. *rotational control*. The Tukey's test showed that C2D was significantly faster than EH2D ($p < 0.001, 40\%$ faster), thus confirming H2. Using the hand position to establish the pointing direction leads to poor performance and it requires more physical effort from the user.

Let us now analyze the role of visual feedback on user performance. We start by comparing EH2D and EH3D. These techniques only differ on the depth of the displayed cursor. Although the Tukey's test did not found any significant difference, a per-user analysis shows that all users performed better with EH2D, and EH3D received the lowest score in the questionnaires. Users complained about binocular fusion and depth cue conflicts when using EH3D. This confirms our hypothesis that 3D cursors drawn over the object's surface result in too many depth conflicts and too frequent parallax changes to be effective for accurate pointing. We found this limitation early in the pilot experiments and we decided not to include a 3D cursor to use in combination with the raycasting from the eye.

Finally, we now compare the three visual feedback techniques used in combination with the Raycasting from the Eye: VF, HCR and C2D. Unexpectedly, all three techniques exhibited a similar performance and the Tukey's pairwise test only showed significant differences between VF and HCR, VF being significantly faster than HCR ($p < 0.025, 15\%$ faster). We thus accept hypothesis H4 and do not accept hypothesis H3.

Regarding error rates (see Figure 4.5b), the one-way ANOVA of error rate vs. technique showed significant differences among techniques ($p < 0.001; F = 6.67$). Tukey's pairwise tests confirmed that occlusion selection based techniques (EH2D and EH3D) have a significant higher error rate than the other considered techniques, no other significant differences were found.

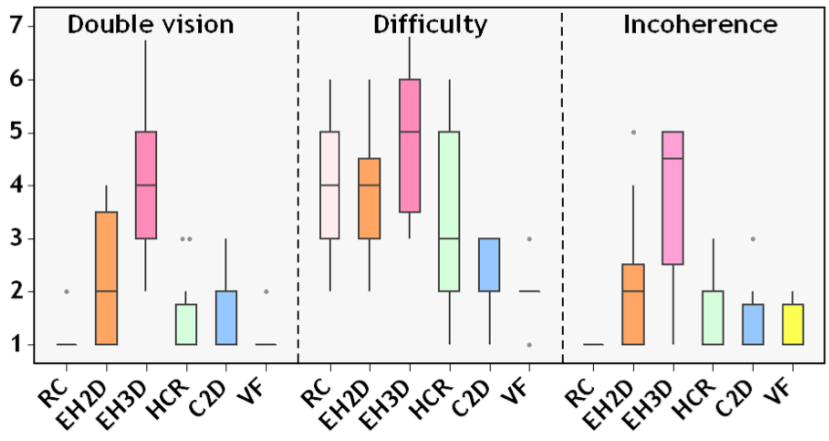



**Figure 4.6:** *Results of the questionnaires.*

### Discussion

Despite its widespread use, we have shown experimentally that raycasting provides inaccurate feedback about the alignment of the pointing tool with the target, and that it performs poorly when accurate pointing is required, even in scenes with a low level of occlusion.

In addition, RCE clearly outperformed competing device-ray mappings in terms of selection performance and error rates, no matter which visual feedback is provided. Moreover, RCE does not exhibit eye-hand visibility mismatch, which as seen in the previous chapter, significantly impacts on selection performance. Furthermore, RCE allows for more comfortable interaction than occlusion selection as the orientation of the ray is controlled through hand rotations.

Regarding the visual feedback used with RCE, all three approaches showed a similar performance. Surprisingly, we did not expect such a good result for the 2D cursor for accurate pointing. We found that users succeeded in finding the right position to place the cursor, even when the left and right projections of the target did not overlap and despite the potential binocular fusion problems we anticipated in the previous chapter. In addition, C2D was one of the best ranked techniques (see Figure 4.6).

We conducted an informal evaluation to test C2D and VF in an identical setup but in a worst case scenario. We kept the set of frontmost (1 m away) and backmost (80 m away) objects and removed the rest. In this situation the VF approach clearly outperformed C2D, both in terms of selection time, and number of errors. Users immediately complained about double vision when selecting targets with C2D. Average selection times with VF were similar in both experiments (1.3 s per object), but C2D mean selection time increased from 1.4 s to 2.1 s. Although this kind of depth disparity is not a common situation, it exposes the C2D limitations.

In summary, among the visual feedback techniques evaluated, the Viewfinder is the visual feedback technique more suitable for accurate selection on stereoscopic displays. The viewfinder does not require any spatial understanding of the environment and it better determines the amount of movement required to aim the target, both in the ballistic and the corrective phases of the movement. It can be argued that an effect similar to that produced by the viewfinder metaphor can be achieved by just asking the user to close her dominant eye during pointing tasks. Our approach provides some notable advantages tough. First, it has a much smaller impact on the immersive view, keeping stereo vision on the part of the scene outside the viewfinder. Second, it is insensitive to poor stereo separation and ghosting effects, still present in monocular vision.

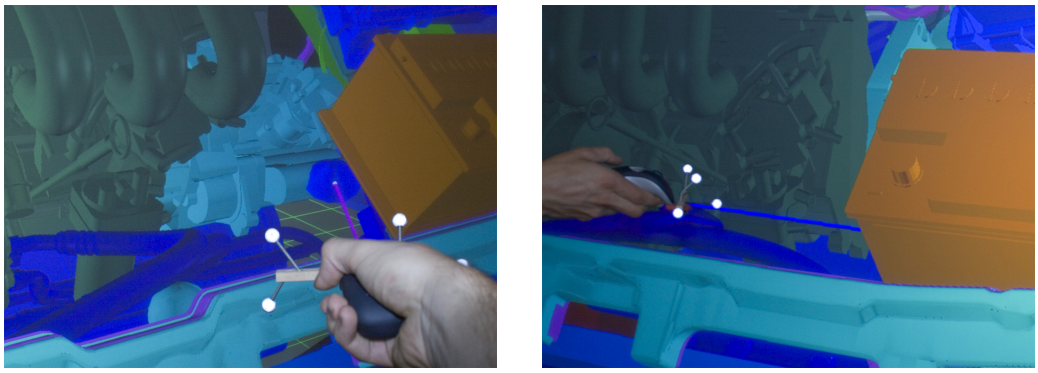
## 4.2 Supporting referential awareness in CVEs

Collaborative virtual environments (CVEs) allow users to work together in order to accomplish a particular task or, as we are now interested, in the exchange of mutual information regarding the virtual environment. We distinguish two types of CVEs: co-located and distributed. In a co-located CVE, users interact in the same VR setup (physical location) while in distributed CVEs each user can be at a different physical location. A common characteristic of CVEs is that users are provided with a unique and perspective-correct stereoscopic image that allows to explore the environment from a first person viewpoint. Thus, for each user specific visualization modes can be implemented.

Imagine a design review of a car's engine compartment where two engineers (users) are discussing the features of a new prototype. A common task is to refer to areas or pieces of the prototype detailing to others which are their defects and possible improvements. This task is commonly accomplished by directly pointing to (indication or selection) the feature (see Figure 4.7a). However in such a cluttered environment occlusion becomes an issue.

In previous chapters we have discussed the eye-hand visibility mismatch and how it can hinder selection tasks. Now, for referral tasks, there is a similar issue but the mismatch is related to the users' viewpoints. In contrast to the mismatch between the eye and the hand, which is limited, the mismatch between users' viewpoints can be extreme as users can navigate freely in the virtual environment.

We observed that during collaborative exploration of complex 3D environments the viewpoint disparity compromises referential awareness. Features which are not simultaneously visible for all users cannot be easily referred and can cause misunderstandings (see Figure 4.7b).



**(a)** **(b)**

**Figure 4.7:** *Both images illustrate the issue of interpersonal occlusion between two tracked users in a co-located CVE: an object that is fully visible to one user (a) cannot or can only partially be seen from other viewpoints (b).*

To solve this problem in reality, people have to walk around the occluding objects to obtain a suitable viewing position. A common behavior is to move close to the person who is pointing to in order to obtain a unoccluded view of the specified object (e.g. by looking over his shoulder). This behavior reduces the viewpoint disparity obtaining a similar view as the user who is pointing to.

However this behavior has two main issues. First, it discourages users to freely navigate through the environment, as they have to keep close to each other, and second, it results in physical proximity between users. Users can bump into each other and, as stated in proxemics theory [49], it does not comply with social protocols of formal presentations.

To deal with these limitations, we must provide users with mechanisms to improve referential awareness. For example, users can be provided with augmentations to easily overcome limitations we face in the real world. As each user in a CVE has its own specialized view, additional information and augmentations can be displayed only for the users requiring them; the user who initiates the referral task does not need additional information.

We believe that occlusion management techniques are the better choice. Occlusion management techniques are valuable tools which help users to better understand the spatial relationships of complex environments and allow them to see otherwise occluded objects.

In order to choose the more suitable occlusion management technique, we followed the design space proposed by Elmqvist and Tsigas [36]. They classification considered six primary dimensions:

- *Primary Purpose.* It relates to the main purpose of the technique. We might need to locate an object (discovery), obtain information encoded in the object (access) or obtain spatial information of the object and its context (relation).

- *Disambiguation Strength.* Determines the occlusion level the technique is able to handle. Four scenarios are considered: proximity, intersection, enclosement and containment (see Figure 4.8).

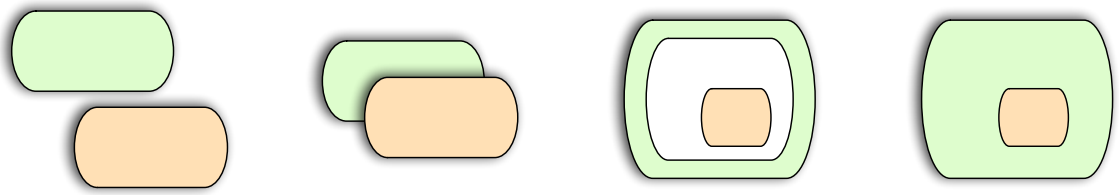


**Figure 4.8:** *Occlusion in 3D environment can be classified given the objects interactions: (a) Proximity, (b) Intersection, (c) Enclosement and (d) Containment.*

- *Depth Cues.* Some techniques alter the environment in a way that depth perception may be compromised or although we are able to see the occluded targets we cannot infer their depth. This dimension contains the amount of depth cues that are preserved.
- *View Paradigm.* Additional views can be used to provide additional information to the user. The options considered were : single view, double separate views, double integrated views, multiple integrated views and multiple separate views.
- *Interaction Model.* It determines the mechanism to trigger the augmentation technique; the user can actively enable the augmentation or it can be passively triggered. The interaction model also considers whether a change of the selected objects requires an off-line process or the changes are displayed in real time (on-line).
- *Target Invariances.* An enumeration of the properties preserved for objects affected by the technique: appearance, depth, geometry and location.

Using these six primary dimensions, Elmqvist and Tsigas [36] classified existing occlusion management techniques into five design patterns: Multiple Views (use two or more separate views of the scene), Virtual X-Ray (turn occluded objects visible automatically), Tour Planners (a precomputed camera animation reveals the otherwise occluded geometry), Volumetric probes (the user can actively alter object properties to manage occlusion) and Projection Distorters (nonlinear projections integrate two or more views into a single view).

Let us now explore which is the better suited occlusion management for our needs. In terms of their design space, our requirements are:

- As *primary purpose*, we aim to improve gestural communication between users by facilitating the discovery of referred objects. However, access and relation purposes are also desirable features.
- We require the maximum *disambiguation strength* available, in order to support occlusion handling for objects that are strongly interacting with other objects in the virtual environment.
- *Depth cues* ought to be maintained as far as possible to better support the users' spatial perception of the environment.
- Since our primary purpose is to improve interpersonal communication, the notion of a shared 3D environment must not be destroyed. A single-view paradigm must be enforced. Furthermore, we argue that target objects ought to be seen embedded in the shared 3D space to make sense of pointing gestures and to better understand the spatial layout of the scene.

- Techniques for handling interpersonal occlusions must follow a passive *interaction model*. Although the user who is performing the referral task actively selects the referred object, changes in the environment have to be passively perceived by others. In addition, the results must be computed in real time.
- Regarding *target invariances*, it is important to preserve the location, the geometry and the appearance of the objects. Otherwise, users would not obtain the same visual information as if they were looking at the objects without any augmentation.

Virtual X-Ray fulfills nearly all our requirements better than other techniques and only in terms of provided depth cues, Multiple Views, Tour Planners and Volumetric Probes may offer better performance. However, they have severe drawbacks with respect to the other dimensions in the design space.

Multiple Viewports provides additional non-occluded views which can display the referred object. However, it is difficult to determine the location of the referred objects in the virtual environment according to an auxiliary viewport, which is essential for the collaborative inspection of a 3D scene. In addition, their disambiguation strength is very low.

Tour Planner techniques rely on changing the user viewpoint to gather a non-occluded vision of selected objects. However, they require off-line computations to determine a valid unoccluded viewing position and their disambiguation strength is limited to proximity and intersect.

When using Volumetric Probes users can move occluding objects aside, but this requires active interaction and a moderate effort to obtain good results. It takes time to manipulate objects and due to the fact that the layout of the scene is changed, users might lose important context information regarding the relative placement of objects.

Finally, projection distorters are totally unsuitable as they modify the viewing projection; this will distort the user's perception of the virtual environment.

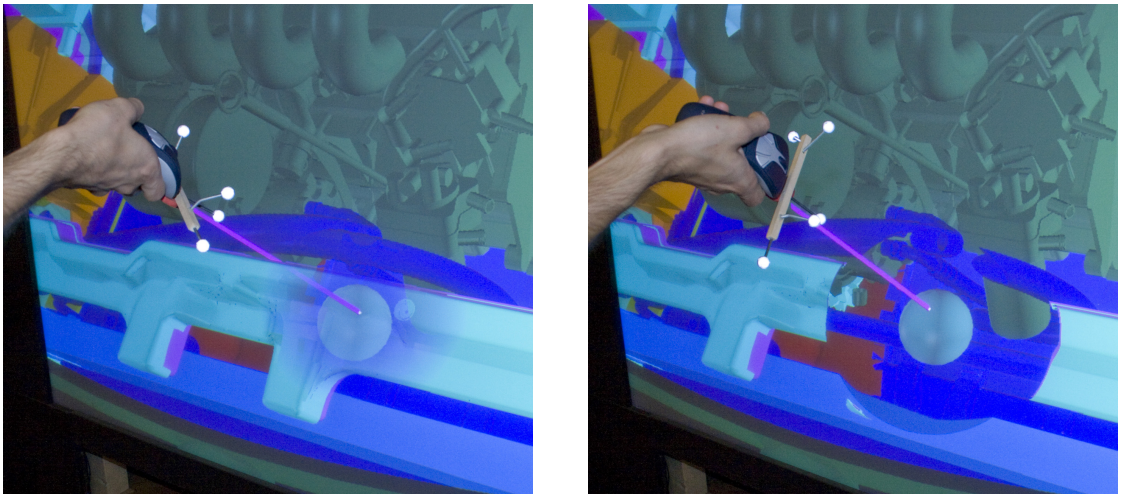
#### Show-through techniques

To facilitate collaborative inspections of a virtual 3D scene, Virtual X-Ray techniques ensure for all users that the object referenced by one user is always showing through. We therefore suggest using the term "show-through" techniques for this particular application of the Virtual X-Ray.

Virtual X-Ray techniques can solve visibility issues by removing or rendering transparently

occluding objects [16, 24, 33, 37]. In contrast to other approaches, they can solve occlusion issues while preserving the layout of the scene as well as the coherence of the virtual environment which is shared by multiple users.

We implemented two show-through techniques (see Figure 4.9) with different visualization parameters to perform a user evaluation: Cutaway and Transparency. The implementation of both show-through techniques is largely similar. First, a volume containing the selected object and both eyes of the user, is computed. For simplification, we constrained the volume to the smallest possible cylindrical shape. In our user evaluation, we were only considering the mutual presentation of simple, sphere-like objects. The axis of the cylinder is defined by the center of the object and the center between the user's eyes.



**(a)** *Transparency* **(b)** *Cutaway*

**Figure 4.9:** *Show-through techniques can improve target discovery by completely or partially removing the occluding environment.*

Once the cutting volume is defined, the fragments falling inside that area can be determined and their appearance can be transformed accordingly. For the Cutaway approach, all fragments inside the cutting volume are discarded. In contrast, for the Transparency approach, alpha values of each fragment are modified according to the distance to the axis of the cylinder. Occluding objects were only sorted in depth on a per-object basis. For our test scenario, transparency effects were correct for most viewpoints. In a different scenario, however, sorting on a per-triangle basis might be required.

The lightweight implementation of the proposed show-through techniques introduced a negligible impact on rendering performance. The most time-consuming process of our sample application was the general graphics rendering for two stereoscopic image pairs (one for each user). The application was running at around 40 fps.

### 4.2.1 Show-through techniques evaluation

We designed a two-user pointing task to analyze the usability aspects of the proposed show-through techniques in a co-located CVE setup. The experimental task was designed to investigate user performance regarding the identification of indicated targets in a dense 3D environment and the memorization of their respective locations. Both show-through techniques (Cutaway and Transparency) were compared to a baseline condition (Baseline) in which users had to obtain adequate viewpoints by walking around to see the otherwise occluded objects. In addition to these pure performance measures, we studied the impact of our visualization techniques on the users' behavior in such a collaborative work setting. We were interested to check whether users could benefit from the proposed show-through techniques in terms of maintaining more comfortable distances to each other without decreasing the efficiency of their collaborative interaction.

The experimental task we implemented for our studies refers to a collaborative design review in the automotive industry, where variations of a design are evaluated by a group of experts. Showing certain features of the model to colleagues is a frequently occurring subtask in such collaborative work settings. Co-located CVE systems are a promising technology to facilitate this as they enable immediate information exchange about features in a shared 3D environment.

Following the situation of a design review as a reference, we presented the engine compartment of a VW Golf in its original size (see Figure 4.10b) on a multi-view projection screen (see Figure 4.10a). Our experimental system supported two tracked users that could individ-

(a) (b)

**Figure 4.10:** *(a) Two-user projection-based setup employed on the evaluation. Both users are provided with correct stereo perspectives. (b) Engine model used for the user study. The black frame represents the borders of the real screen.*

ually walk around the virtual model to observe it from different viewpoints. The perceived stereoscopic view was always corresponding to the users' respective viewing position. Thus, the model was perceived to remain at a fixed location in the shared environment while users were walking around it. The 3D model was tilted about 40 degrees in order to provide users with a good overview, similar to the visual experience in the real world where people look from above at a car's engine compartment.

### Apparatus

The experiments were conducted on a projection based two-user VR setup (see Figure 4.10a). The VR system was developed by the Virtual Reality Systems Group at the Bauhaus-Universität (Weimar, Germany). The system is built on time-multiplexing of individual views using LC-shutter technology as described in [43]. The users were required to wear tracked shutter glasses which were custom made double-cell shutter glasses sizing 65 mm in width and 45 mm in height. Square 80 mm shutters were mounted in front of the projectors. While single LC-shutter elements provide a contrast ratio ranging from 1000:1 to 5000:1 between the open and the closed state, the contrast ratio achieved was of 25000:1 with their double-cell approach. As a result, absolutely no crosstalk between the views of both users was perceptible.

The physical dimensions of the projection screen were 3 m in width and 2 m in height with a resolution of $1800 \times 1200$ pixels. The tracked workspace, wherein people could walk around to examine and gather information about the scenery from different angles, covered an area of approximately 4 by 4 meters. Note that these spatial constraints of the operational environment also affect the users' behavior in terms of proxemics.

The visual stimuli were presented at a stereo depth range of $\pm 0.8$ m from the projection screen. The target objects inside the simulated engine compartment were situated in places which were hidden from most possible viewing positions (e. g. behind the engine block). Without using show-through techniques, the observer could not directly see them.

### Procedure

We assigned different tasks to the two involved users: *the experimenter* (hereinafter referred to as the presenter) was pointing to certain objects in the model that had to be identified and located by a second user, *the observer*. The role of the presenter was always assumed by a researcher. For the role of the observer, we invited volunteers to analyze how show-through techniques would affect their ability to identify, locate and memorize the indicated objects

and also their behavior in terms of proxemics.

Abstract spherical-like 3D geometries which were placed at 15 predefined positions in the model served as target objects for presentation and identification. Therefore, all targets were of the same size. We decided to use abstract target objects that had no semantic relation to the model in order to minimize the potential bias from different levels of knowledge users may have about the structure of a car's engine compartment. Only one target object was shown at a time to ensure that all experimental conditions were evaluated using the same sequences of target objects. Two different researchers were alternately running the experiments. We made sure to fulfill the role of the presenter in compliance with a strictly defined storyboard to minimize the influence of personality. Additionally, we prevented the presenter to search for the next target object in line by providing a visual localization aid. The high quality of the shutter elements we used in our setup effectively prevented these hints from being seen by the observer.

Correspondingly to the different roles of the presenter and the observer, each trial also consisted of two phases. During the *presentation phase*, the presenter was pointing to three target objects in a row, while always waiting until the observer confirmed that he had located the respective objects in the scene. In the following *retrieval phase*, the observer was asked to prove that he had memorized the three objects and their location by pointing to them in the same sequence as they were presented. In both show-through conditions, X-Ray vision facilitated the identification of presented objects for the observer, but only during the presentation phase. In the following retrieval phase, users had to rely on the information they gathered in the presentation phase to once again find the three objects in the scene.

During the experiments, different types of data were recorded. Primarily, we logged discovery time during the presentation phase and the time required for the retrieval of target objects. The logged discovery time did not include the time required by the presenter to approach the next target. Instead, we only recored the time from the moment the presenter points to the next target until the observer confirms that he has seen it. Additionally, we logged the users' head positions throughout the experiments. This enabled us to sum up the covered distance of the participants during the presentation phase as well as to track the distance both users maintained from each other during the experiment.

### Design

Each participant of our study was involved in two successive experiments. First, we compared the target retrieval performance and the learning progress of user groups that were oper-

ating with different occlusion management techniques (Baseline, Cutaway, Transparency). Thereafter, we introduced each user to the two other conditions they had not been exposed to during the first experiment. Another set of trials was performed with each technique condition in order to analyze user behavior with different techniques and to provide a basis for the users' subjective ratings. From now on, we will refer to block each set of trials performed for each condition.

We expected stronger effects of learning the task than that of techniques. Therefore, technique was compared between-subjects with repetitive blocks (five subsequent trials) as a within-subjects variable. Each user group consisted of 8 people. Before the experiment started, users were given written descriptions of the task including the advice to strive for the best possible target retrieval performance, and the hint that the sequence of target object was repeated during each block. We provided a short training session to make sure users understood the task, the respective occlusion management technique they were assigned to and also that they could effectively handle the involved interaction techniques (e.g. head tracking, ray selection). We trained the interaction procedure of our experiment using a special sequence of six target objects which did not occur during the recorded trials.

The following recorded trials also consisted of the presentation of three subsequent target objects that had to be found by the observer. Five different trials, with unique object locations, were performed during one block. The combination resulted in 15 different target locations inside the engine compartment which were repeated in three subsequent blocks. In total, 15 trials were performed by each user. Participants were encouraged to take breaks between blocks to minimize fatigue.

The dependent variables for the first experiment were: the discovery time (time during which targets were shown by the presenter until the confirmation that the observer localized them); the retrieval time (duration of the retrieval phase); and the covered distance during the discovery phase. The independent variables were the occlusion management technique and block.

We expected users to learn the task sequence rapidly, and thereby successively optimizing the effectiveness of their operations. We also assumed that they needed to move less when show-through techniques were applied. We reasoned that this could enhance the discovery phase by reducing discovery times. However, the reduced movement in the scene's surroundings and the unfamiliar visual appearance of objects that show through other geometries can also have a negative impact on the users' spatial understanding of the scene and thus affect retrieval times. From these considerations, we derived the following hypotheses about the results of the first experiment:

- H1.1 : Decreasing mean of discovery time over successive blocks.
- H1.2 : Decreasing mean of retrieval time over successive blocks.
- H1.3 : Decreasing mean of covered distance over successive blocks.
- H1.4 : Larger mean of covered distance in the Baseline condition as compared to both show-through techniques.
- H1.5 : Larger mean of discovery times in the Baseline condition as compared to both of the other techniques.
- H1.6 : Larger mean of retrieval times in both show-through conditions as compared to the Baseline condition.

During a break of approximately 15 minutes users were asked to provide information about their age and gender, as well as previous experience with VR-systems. Using a short questionnaire, we checked whether any particular problems were experienced during the experiments.

In a second experiment we wanted to compare the three occlusion management techniques within subjects. We assumed that after having gained experience during the first experiment, users would have acquired a robust cognitive model, meaning that they were well trained with the task procedure. Thus, we estimated expert performance, in that effects of further learning would be negligible. However, since users were trained with only one of the techniques, we refrained from comparing task performance within subjects. Instead, we focused on subjective preference of users and user behavior in terms of proxemics.

We logged the distance between users during the presentation phase of each trial. As in the first study, the experimenters performed the role of the presenter, making sure to follow a strict storyboard in order to avoid biasing the distance data.

We conducted three additional blocks in that second experiment conforming to those from the first one - aside from this, users were now testing other occlusion management techniques in each block. The order of the techniques was counter balanced among the 24 test subjects to avoid ordering effects.

After having completed three blocks with different technique conditions, we asked users to score techniques on a 5 point Likert-scale regarding spatial understanding (how techniques support gathering information about the position of target objects in the scene), collaboration (how techniques support interpersonal communication about the scene) and comfort (the perceived comfort while interacting in presence of different occlusion management techniques).

Without the use of visual aids, close proximity to the presenter is generally required in order to see an indicated object. Our proposed show-through techniques, instead, enable to observe indicated objects in the scene from every viewpoint that the operational environment permits. Following the findings of Hall [49], we assumed that users would tend to remain in the close phase of social distance rather than in closer proximity. The size of the operational environment, however, hardly allowed users to interact with the virtual scene, while remaining in social distance to each other. Nevertheless, we expected much shorter average distances between users in the Baseline condition as compared to both show-through techniques. Consequently, we estimated a user preference for show-through techniques with respect to comfort. Regarding spatial understanding we felt that the localization of objects in the scene was cognitively more difficult in cases in which they are only perceived as showing-through other geometries than if an appropriate viewpoint has to be obtained by walking around the virtual scene. We therefore expected the best scores for the Baseline condition in the domain of spatial understanding. On the other hand, since show-through techniques allow users to see indicated objects from other perspectives than that of the presenter, the pure amount of information that can be gathered as a group of users is considerably increased. We assumed that users would consider this advantage when scoring techniques in the domain of collaboration. In summary, we noted the following hypotheses regarding the results of the second experiment:

- H2.1 : In average, larger distances will be kept between users in both show-through techniques as compared to the Baseline condition.
- H2.2 : Strong user preference for both show-through techniques in the domain of comfort.
- H2.3 : Lower subjective ratings for both s how-through techniques as compared to the Baseline condition in the domain of spatial understanding.
- H2.4 : Stronger user preference for both show-through techniques in the domain of collaboration.

### Participants

Twenty-four paid users, aged between 19 and 31, participated in the study. All of them were students of varying disciplines ranging from engineering to computer science and to design and humanities. Seventeen participants had prior experience with VR applications while 7 did not. We organized the experiments as a competition so that the three participants with the shortest retrieval times of targets, won a ticket to the movies. Note that the

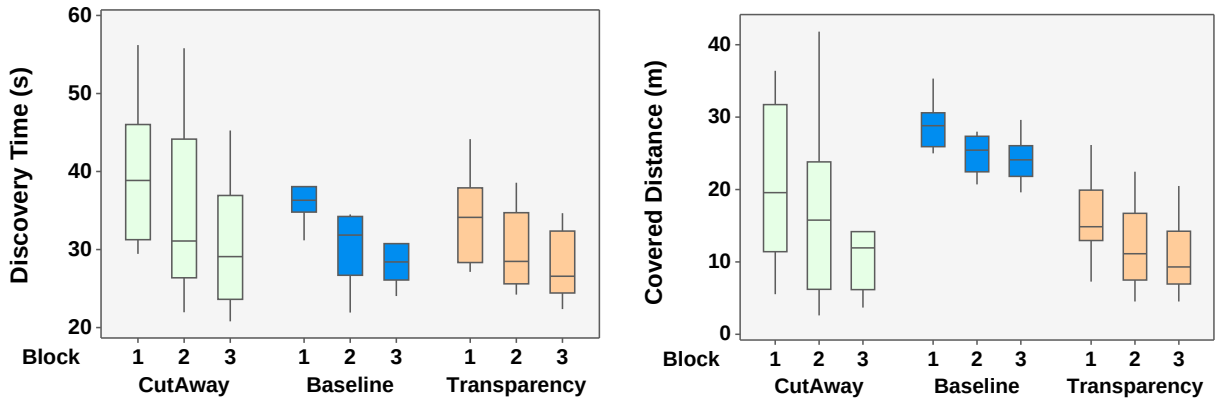
differences regarding social hierarchies between presenter (research assistant) and participant (students), were not that distinctive as they certainly are within formal presentations held for an executive committee.

### First experiment results

Performance data from the first experiment was collapsed and entered into a one-factor (technique) between subjects ANOVA considering block as a within-subjects variable. For all post-hoc comparisons, Bonferroni adjustment for $\alpha$ was applied.

Regarding discovery time (see Figure 4.11a), we found a significant effect of block ($p < 0.001; F = 29.36$). Post-hoc tests revealed that all differences between the successive three blocks were significant (all $p < 0.05$). This confirms H1.1, stating that learning would have a significant effect on the time required to find target objects. No significant differences could be found for technique. Thus, we had to reject H1.5. In contradiction to that, we were expecting longer confirmation times in the Baseline condition. During the experiment, we observed users developing appropriate strategies for efficient interaction under different technique conditions. In the Baseline condition, users generally followed the presenter to ensure a similar point of view, whereas they rarely changed their position when show-trough techniques were enabled. Therefore, users were able to localize indicated target objects in a comparable time for all techniques.

With respect to the covered distance (see Figure 4.11b), we found significant effects for block ($p < 0.001; F = 18.15$). Post-hoc comparisons revealed significant differences between




(a) (b)

**Figure 4.11:** *Boxplots of discovery time (a) and the corresponding covered distance (b) during the first experiment. We can clearly see the learning effect over blocks, and also how users were required to move more if they were not provided with show-through techniques.*

all three block conditions (all $p < 0.05$). This confirms H1.3, stating that learning the task would allow users to minimize the required physical action. Additionally, significant effects were found for technique ( $p < 0.05; F = 9.04$). Post-hoc comparisons revealed that in the Baseline condition users moved significantly more than in both other conditions ($p < 0.001$). This result confirms H1.4. Without making use of show-through techniques, people are required to adapt their viewing position more often. Between both show-through techniques, we found no significant differences.

Retrieval time was only affected by block ( $p < 0.001; F = 43.92$). Therefore, we can confirm H1.2, stating that learning the task would allow users to improve upon the rapidity of target retrieval. No significant effect of technique on retrieval time was found. We thus reject H1.6. We were assuming that the time required to retrieve target objects is strongly affected by the spatial knowledge the users were able to gather beforehand. As we cannot find an effect of technique on retrieval times, we argue that the three tested occlusion handling techniques can comparably well support the users' spatial perception of objects within the 3D scene.

**Second experiment results**

In the second experiment we analyzed the users mutual distance during the presentation phase. We compared the techniques between subjects because we expected the training during the first experiment to have an important impact at the users interaction. As such we were only considering data from these users, that were well trained with the respective technique.

Since the length of trials varied, we first normalized the distribution of distances to obtain the relative frequency of distance ranges during the experiments. Figure 4.12 shows a plot of the normalized frequencies at which distances occurred during different technique conditions. For all techniques this distribution fits accurately to a normal distribution (Anderson-Darling normality test: $p < 0.005$).

It is apparent in these graphs that show-through techniques allowed users to maintain larger distances. In the Baseline condition, instead, users were crowding each other, frequently ending up at intimate distances, sometimes even bumping into each other. Note that the position of the user was taken from the center between their eyes. So, at distances $\sim 40cm$ their shoulders were touching. Distance values under $\sim 20cm$ would mean that their heads were colliding.

We observe that in all conditions users kept about half of the time in the far phase of the personal distance. With show-through techniques applied, the other half of the time

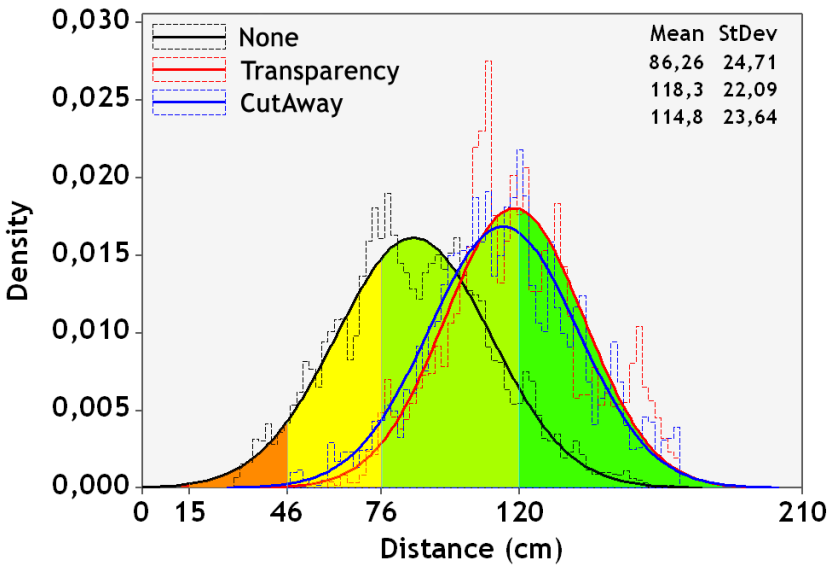



**Figure 4.12:** *The distance kept between users in relation to the applied occlusion-handling technique. Dashed outlines represents the histogram of logged distances, shaded areas represent the "distances in man" as defined by E.T.Hall [49]. 0..15cm intimate distance (close phase) 14..46cm intimate distance (far phase) 46..76cm personal distance (close phase), 76..120cm personal distance (far phase) and 120..210cm social distance (close phase).*

was spent in the close phase of the social distance. According to Hall this is generally the preferred distance in collaborative work settings as simulated in our study. Thus, we assume, that distances between users would still be larger, if not affected by the following limitations of our display setup:

- Our working space was only about 3 meter wide, which clearly limited the maximum distance among users.
- The eye-wear used in the experiment consisted of relatively large shutter glasses, but still they limited the users' field of view. Particularly, peripheral vision of users was affected and as such that the subconscious attention to changes in the surrounding environment, e.g., colleagues approaching from beside, was impaired.
- There was barely any social hierarchy present between the presenter (research assistant) and the observer (students). In a real world setting this would be different. Recall that the more important somebody is, the larger the distances kept to others generally are.

However, in absence of show-through techniques users kept more time in the close phase of the personal distance and, what is probably even more important, they could not avoid intruding frequently into the other's intimate distance. Table 4.2 shows the exact values.

For a statistical evaluation, we collapsed the data and entered it into a one-factor (tech-

| Zone | Phase | Baseline | Transparency | Cutaway |
|---|---|---|---|---|
| Intimate | Close | 0.16% | 0.00% | 0.00% |
| | Far | 5.10% | 0.05% | 0.18% |
| Personal | Close | 28.46% | 2.76% | 5.05% |
| | Far | 57.59% | 49.59% | 53.09% |
| Social | Close | 8.69% | 47.60% | 41.68% |

**Table 4.2:** *Average time spent at distinct zones of interpersonal distances when operating with each show-through technique.*

nique) between subjects ANOVA. We found a significant effect of technique on distance ($F = 21.01, p < 0.001$). This effect stems again only from differences between the Baseline condition and both show-through techniques. Between the latter two, no significant differences could be found.

We expect that if users would have had more space available, they would also have kept larger distances between the presenter and the observer, while increasing the size of the operational environment would not have had much effect on the distance observations in the Baseline condition.

### Survey

Subjective ratings (see Figure 4.13) revealed a significant user preference for the Cutaway and Transparency conditions in terms of comfort. The Friedman Rank Test showed significant differences among conditions ($p < 0.001$) with Baseline as the bottom line, which confirms H2.2. Show-through techniques allow users to keep more comfortable distances between each other. Also, less viewpoint motion is required since even otherwise occluded geometry can be seen. In the domain of spatial understanding, we could not find significant differences among techniques. Thus, we had to reject H2.3. It appears as if users were very confident about their ability to gather spatial knowledge in the virtual environment independently from the used occlusion handling technique.

In the domain of collaboration support we have found significant differences among the subjective ratings of techniques ($p < 0.001$). Again, the Baseline condition received the lowest score from users which confirms H2.4. Though our experimental task did not require much collaborative interaction but rather only fundamental information exchange, users seemed to predict further benefits for collaborative interaction from show-through techniques.

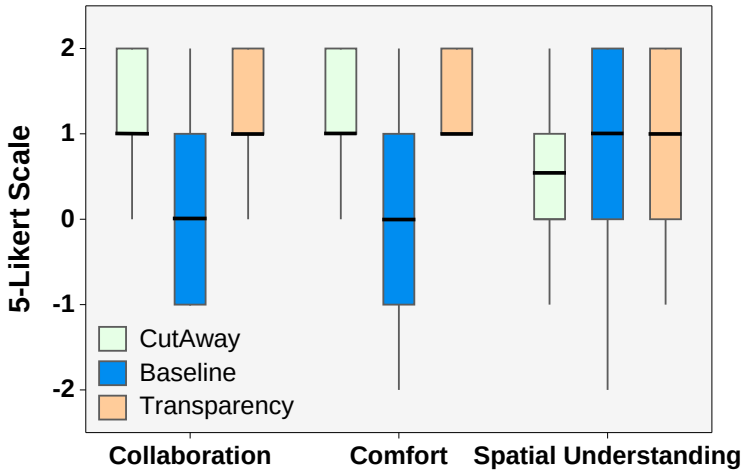



**Figure 4.13:** *Subjective comparison of the three conditions, in terms of collaboration, comfort and spatial understanding of the model.*

### Discussion

Overall, the results of the first experiment indicate that all three conditions allow for comparable user performance, not only in terms of fundamental task efficiency, but also with respect to supporting the users' learning progress. Regarding the required distance covered to localize indicated targets, we found significant benefits for both show-through techniques.

By decreasing referential ambiguity, show-through techniques were particularly beneficial for the presenter as he or she did not need to worry if an object is occluded from the observer's point of view. Showing objects is not always straightforward. One needs to consider the viewing position of the observers. In cluttered 3D environments we experience an important difference between pointing or actually showing something. Pointing does not necessarily mean that the observer is seeing the indicated feature. Instead, pointing to an object often implies for colleagues to move around in order to arrive at an appropriate viewing position. Showing, on the other hand, often involves the effort of moving something explicitly into the field of view of others.

In the second experiment, we have observed how using show-though techniques reduced the number of cases in which users needed to get very close or even bump into each other. Our findings also showed that they can maintain a socially convenient distance for most of the time while moving much less than without applying these techniques.

In our experiments, we involved only two users with distinct roles; the presenter and the observer. However, in real life these roles change and users are taking turns at showing and observing, which needs to be supported appropriately. Furthermore, with show-through

techniques, the observer sees a different perspective of the object being pointed to than the presenter, which can prove to be a limitation in certain situations. Often, for example, more than two people are involved in a collaborative discussion of a design scenario. The questions are if our techniques are similarly effective in such cases, how do they scale, and how do they handle situations wherein more than one person is pointing. One issue already became apparent: the size of the display needs to be appropriate with regards to the number of users involved. A three-meter wide display allows at most two users to maintain a comfortable distance to each other.

# Chapter 5

# Applying Fitts' Law to enhance 3D object selection

The selection of small and distant objects has been recognized as one of the major limitations of raycasting, especially when a 6-DOF sensor is used to control the selection ray. The acquisition of small targets require a high level of angular accuracy, which is difficult to achieve due to hand trembling and tracking errors. In this chapter we explore whether Fitts' law principles are able to enhance selection performance in these situations.

Fitts' law asserts that the mean selection time MT to acquire a target of size $W$ which lies at a distance $A$ is governed by the relationship $MT = a + b\log_2(A/W + 1)$; the logarithmic term is referred to as the index of difficulty of the task. Considering its formulation, three possible approaches can be considered to reduce the index of difficulty: reduce the distance $A$ towards the target, increase the size $W$ of the target, or a combination of both [4]. However, it is unclear which is the best way to optimize a task as Fitts' Law does not provide any model about how people perform acquisition tasks. For example, doubling the size of the target and halving the distance towards the target result in the same index of difficulty, but is unclear which one is the best choice.

The human movement model which better accounts for Fitts' Law is the optimized initial impulse model proposed by Meyer et al. in [81]. It states that acquisition tasks can be subdivided into a two-step movement phases: ballistic and corrective. While ballistic movements are fast movements which covers most of the distance $A$ towards the target, corrective movements are precise movements intended to correct the deviation towards the target introduced by the ballistic movement. In addition, MacKenzie et al. in [73] concluded that velocity profiles of acquisition movements depend on both $W$ and $A$. While A determines the maximum movement speed, $W$ determines the movement deceleration and the corrective movements.

In summary, techniques attempting to improve pointing performance by decreasing $A$ should concentrate on the initial large movement (ballistic phase) that covers most of the distance towards the target. On the other hand, techniques that attempt to increase $W$ should focus on the final corrective phase. The effect of changes on $W$ occurring after the ballistic movement has been initiated will be more apparent in the final corrective phase, when the user is trying to home in on the target under closed-loop feedback control [4].

The application of these principles to human computer interaction has led to a number of successful pointing facilitation techniques to improve pointing performance for WIMP interfaces [4]. However, less effort have been made to apply them to 3D object selection.

Current pointing facilitation techniques for 3D object selection are mainly based on increasing the size of the selection tool (increase $W$), techniques which statically increase the size of targets in control space, such as the sticky ray [112], and techniques which adapt the control-display ratio taking into account users' actions (increase $W$ and decrease $A$), as in PRISM [41].

Approaches focusing on decreasing the amplitude of the movement $A$ are limited to GUI elements when potential targets are known a priori. In contrast, increasing dynamically the size of objects, although it has been widely studied in 2D graphical user interfaces it has not been applied to 3D object selection.

In the 2D HCI literature there are two main *expanding targets* approaches [79]. The first approach relies on increasing the motor and the visual space of the GUI element the user is pointing to (see Figure 5.1a). This is accomplished by determining an activation area around each widget, the activation area defines the new value of $W$ for the widget. When the cursor enters into an activation area, the corresponding widget is expanded to match the activation area and vice-versa.

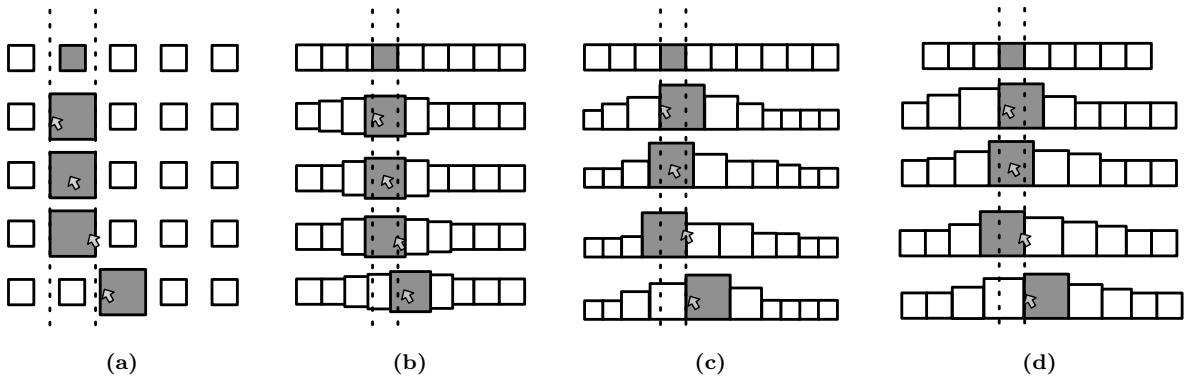



**Figure 5.1:** *Alternatives to increase the motor and/or the visual space for 2D mouse based interaction.(a) Increase the motor and the visual space. (b,c,d) Increase only the visual space. Image based on Cockburn et al. [23]*

However, this approach can only be applied if there is no overlap between activation areas. If the cursor enters into two activation areas at the same time it is not clear which GUI element has to be expanded. Moreover, the expansion of one widget might occlude neighboring widgets if not handled properly.

The second approach relies only on increasing the visual space (see Figures 5.1b-5.1d). In contrast to the previous approach, it does not alter the index of difficulty of the task, as $W$ is computed in motor space, and the visual feedback provided, in addition to scale the widget the cursor is pointing to, also scales neighboring widgets. This visual feedback misleads users and makes them believe that the motor space of the widget is scaled.

The evaluation of these expanding target techniques by Cockburn and Brock in [23], showed that they effectively provide an improvement over standard moused-based interactions. However, as expected, improvements were more apparent for targets with greater IDs. Interestingly, increasing only the visual space also results in reduced selection time. Authors stated that the increase of performance was due to the additional visual feedback provided.

In this chapter we explore how increasing dynamically the size of objects for 3D selection tasks affects selection performance and error rates. However, it turns out that the extension of 2D expanding targets techniques to 3D scenes, is not a trivial task:

- It requires a proper handling of the occlusion among potential targets. If an object is expanded, it can potentially occlude neighboring targets which were visible before the expansion. Unlike the 2D counterpart, occlusion among 3D objects is view-dependent and is a global problem, meaning that two objects arbitrarily far away in 3D can occlude each other when the user's viewpoint is aligned with them.

- Occlusion handling is especially difficult in the absence of a regular layout or with objects appearing at different scale levels. Unlike GUI elements, objects in a 3D scene exhibit no regular arrangement nor similar scale levels.

- To ensure the compatibility with VR display systems, correct parallax values on the projected stereo images are required. This means that standard techniques based on 2D image distortion cannot be used, as they completely disrupt parallax values.

We propose two orthogonal approaches: the *Dynamic Scalling* and the *Forced Dissoclusion*. The discussion of both techniques considers the use of classic raycasting selection, although both can be used in combination with other virtual pointing techniques.

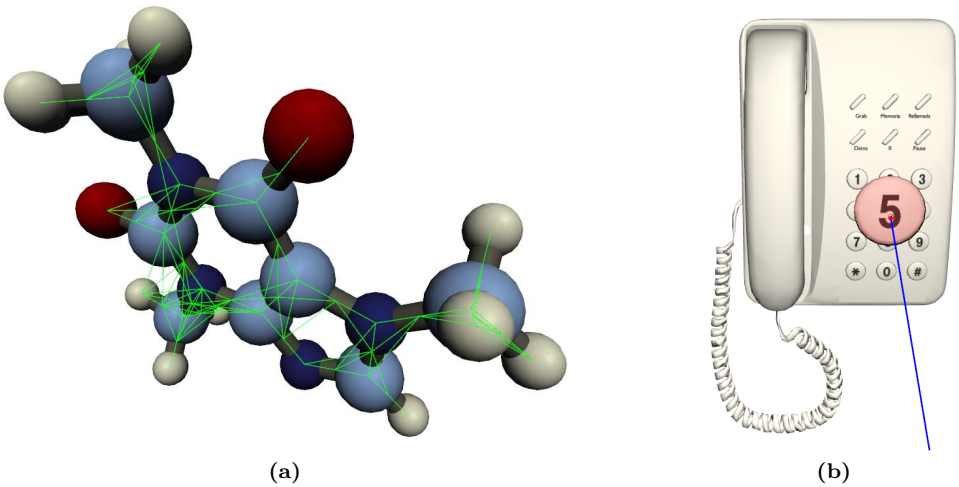
## 5.1 Dynamic Scaling

In order to increase the effective size of potential targets, objects near the selection ray can be dynamically scaled, thus increasing control and visual space. However, as noticed before, when an object is scaled it can potentially occlude neighboring objects, thus requiring some mechanism to guarantee that originally-visible objects can still be selected after the expansion. This condition must be enforced at least in the vicinity of the ray.

Although all object transformations are applied in 3D space, potential overlaps among objects must be analyzed in 2D by considering their screen projection. Potential overlaps from a given viewpoint can be encoded in a graph $G$ (see Figure 5.2a) where nodes represent objects. There is a link joining to nodes $A$ and $B$ if the screen projection of object represented by $A$ can overlap with the object represented by $B$ after a transformation. Although, the graph could have $O(n^2)$ links we have observed a linear behavior in all the tested scenes. A conservative way for computing $G$ is to consider the screen projection of the object's bounding spheres.

The graph $G$ is obviously view-dependent and must be recomputed every time the viewpoint moves far apart. Since the viewpoint can be considered to be roughly stationary during selection, $G$ must be recomputed only when the viewpoint stabilizes, indicating the potential start of a pointing act.

The dynamic scaling algorithm proceeds as follows. First, The *focus object* (the object intersected by the selection ray will be referred as the $focus$) is scaled with respect to its center. As we have considered only raycasting selection, we assume that a usable size for the focus object (in terms of the solid angle subtended from the viewpoint) has been decided




(a) (b)

**Figure 5.2:** *(a) A molecule model and its view-dependent graph. (b) The Focus object is scaled with respect to its center.*

considering the characteristics of the input device and the user preferences. The solid angle can be dynamically converted into a number of pixels $\Pi$ considering the screen resolution and the current viewpoint. The scale factor for the focus can be computed with Equation 5.1 where $P_i$ is the number of pixels in the screen projection of the (unexpanded) focus. The computed $s$ value is used to scale the focus in 3D space with respect to its center (see Figure 5.2b).

$$s_f = max(1, \sqrt[2]{\frac{\Pi}{P_i}}) \tag{5.1}$$

Once computed the scale factor for the focus object, we compute for the neighboring objects a transformation (scale and translation) which ensures that they are still selectable. This transformation is encoded as $(s, t)$ where $s$ is the scaling factor with respect to the object's center, and $t$ is a translation vector. These values can be computed by a local breadth first traversal of $G$ starting from the focus object. The traversal can be restricted to a maximum depth to limit its effect to the objects in the vicinity of the ray.

For each visited object $A$, we search the graph for the neighboring object $B$ which maximizes the overlap with $A$ (the search is limited to already visited objects for which the transformation has been already computed). If the overlap is below a certain threshold, the transformation $(s_a, t_a)$ for object $A$ is left unmodified. Above this threshold, the transformation for $A$ can be computed as depicted in Figure 5.3. Let $r_a$ and $r_b$ be the radius of the bounding circles of the screen projection of $A$ and $B$, as they appear in the original, untransformed scene. After scaling $B$ by $s_b$, the radius of $B$ is incremented by $\Delta r_b = r_b(s_b - 1)$. It seems reasonable to compute the scale factor for $A$ with the constraint $\Delta r_a = \Delta r_b$. This can be accomplished by letting $s_a = (\Delta r_b / r_a) + 1$. In our experiments we used this approach for computing $s_a$ in combination with a linear attenuation based on the distance to the focus object. The translation vector $t_a$ can be computed as $t_1 + t_2$, where $t_1 = t_b$ simply propagates to object $A$ the translation applied to object $B$, and $t_2$ is computed so that the screen projection of object $A$ is moved apart from $B$ while preserving their relative overlap in the

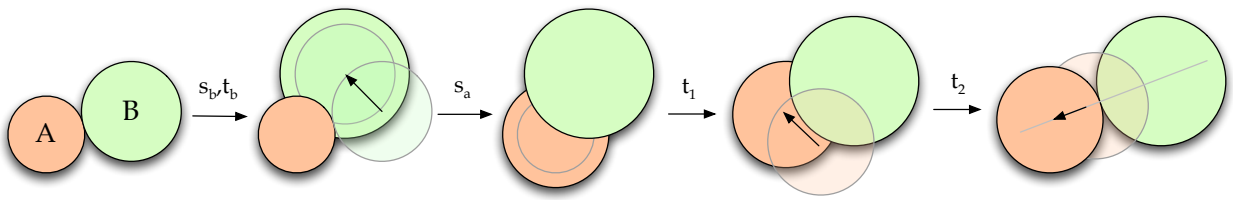



**Figure 5.3:** *Propagation of transformations for Dynamic Scaling. The transformation $(s_b, t_b)$ applied to object $B$ is used to compute the transformation $(s_a, t_a)$ for object $A$. We apply a second transformation $t_2$ to preserve the original overlap between $A$ and $B$.*

original, untransformed scene. The direction of $t_2$ is defined by the line joining the centers of their screen projections. The computation of $t_2$ can be adjusted taking into account the maximum overlap allowed, as shown in Figure 5.4. In the experiments we allowed for a 50% of overlap.

**(a)** *No overlap* **(b)** ≈ 25% *overlap* **(c)** ≈ 50% *overlap*

**Figure 5.4:** *Different overlaps can be considered when propagating transformations.*

In order to avoid abrupt changes, these transformations should be applied gradually. Every time a focus change occurs, the transformations for all objects are computed and they are smoothly interpolated across several frames. If another focus change occurs while the animation is still running, the new transformation values are computed from the interpolated values at the time of the event.

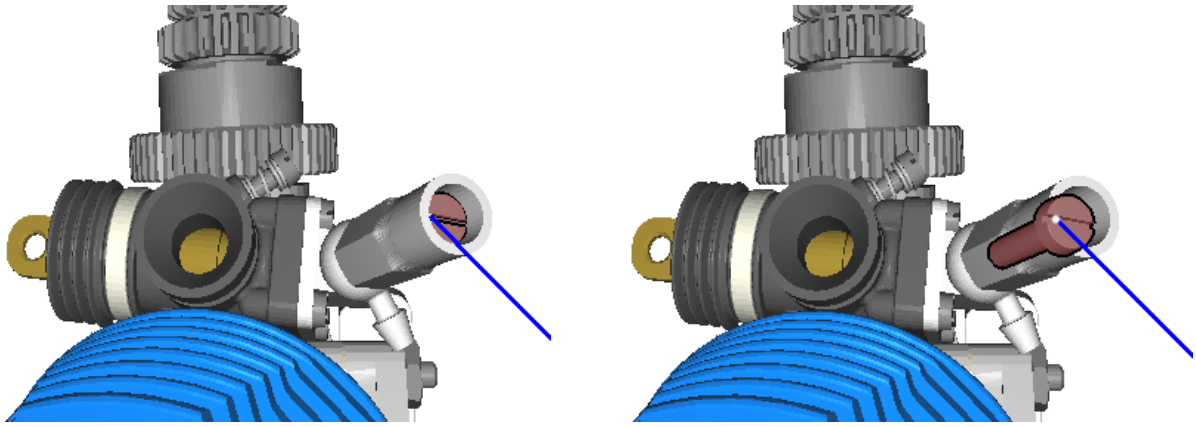
#### Technique overhead

Performance overhead is negligible with the only exception of the graph computation, which was nevertheless computed in less than 5 ms for moderately-sized scenes. Since this graph needs to be recomputed only when the viewpoint changes significantly, the impact on the frame rate can be neglected.

## 5.2 Forced Dissoclusion

An orthogonal approach to increase $W$ is to maximize the number of selectable pixels of the focus object, by forcing the object to appear completely unoccluded. This can be easily accomplished by a proper use of OpenGL's depth range functionality (see Figure 5.5).

So far, when the selection ray intersected several objects, we considered only the first intersection. Now we must guarantee that the focus object will remain selected until the ray leaves it, the ray might intersect other closer objects hidden due to the focus object. This situation

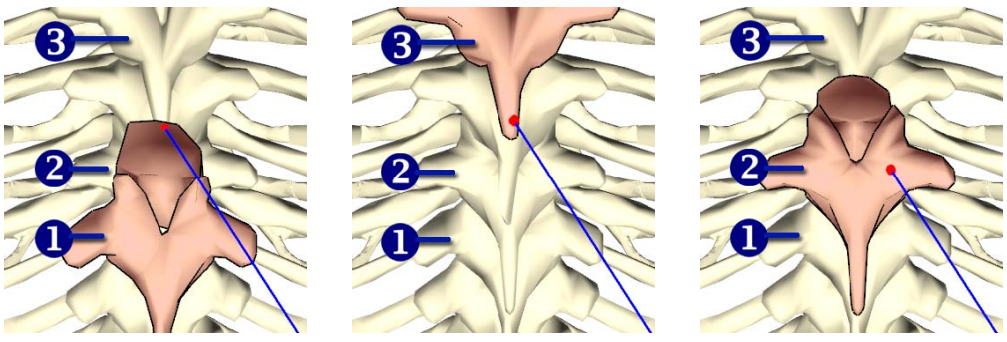


(a) (b)

**Figure 5.5:** *(a) Raycasting selection. (b) Raycasting selection where the focus object is shown completely unoccluded.*

is depicted in Figure 5.5b, where the intersection with the (hidden) envelope must be ignored until the ray leaves the selected object. This selection behavior is also straightforward using OpenGL's selection buffer with different depth ranges.

We conducted a pilot study to get a first evaluation of forced dissoclusion. Unexpectedly, most users reported an erratic behavior of the ray when interacting with densely-occluded scenes. We discovered that the reason for this behavior was the eye-hand visibility mismatch. This visibility mismatch between the user's hand and the user's eye occurs whenever the ray is controlled by the hand (see Section 3.1.2 for more details). However, these effects are much more apparent when forced disocclusion is enabled. This situation is depicted in Figure 5.6 and results in a completely counterintuitive behavior.




(a) (b) (c)

**Figure 5.6:** *Starting from situation (a) one might thought that if the user moves the ray upwards it would result in the selection of object 3 (b), but, the selected object is object 2 (c). In situation (a) the ray is intersecting first the object number 2, but due to Force Disocclusion the object 1 is still in focus. When the ray no longer intersects object 1, the intersection with object 2 is no longer ignored, and object 2 is selected. Furthermore, the ray does not even intersect object 3, it is only an illusion due to depth sorting.*

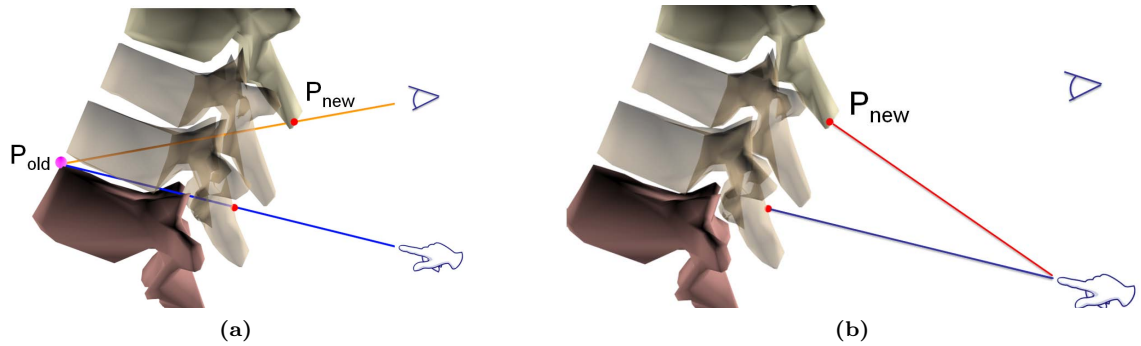



**Figure 5.7:** *Intersection algorithm for Forced Disocclusion. The ray is modified to match the expected intersection point from the users point of view.*

This issue can be ameliorated by further adapting the ray-scene intersection. First we compute the intersection of the current ray with the scene. If the ray has left the object that was selected in the previous frame, we proceed as follows. Let $P_{old}$ be the closest point in the current selection ray to the last frame intersection point with the focus object. Since we are forcing disocclusion, $P_{old}$ is not guaranteed to be the closest intersection, as shown in Figure 5.7. We compute a second ray defined by joining the user's eye position with $P_{old}$. The closest intersection $P_{new}$ of this second ray with the scene, indicates the next selected object. We introduce a slight correction in the orientation of the ray so that it intersects $P_{new}$.

This correction means that we create a subtle mismatch between the orientation of the input device and the ray orientation. The resulting offset is similar to the offset accumulated by techniques adjusting CD gain according to the speed of movement [41]. The orientation disparity occurs in the plane defined by the ray and the user's eye, and thus it is hardly visible from the user's viewpoint. The offset is accumulated until a fast movement of the user's hand is detected, we considered a threshold of 30 deg/s.

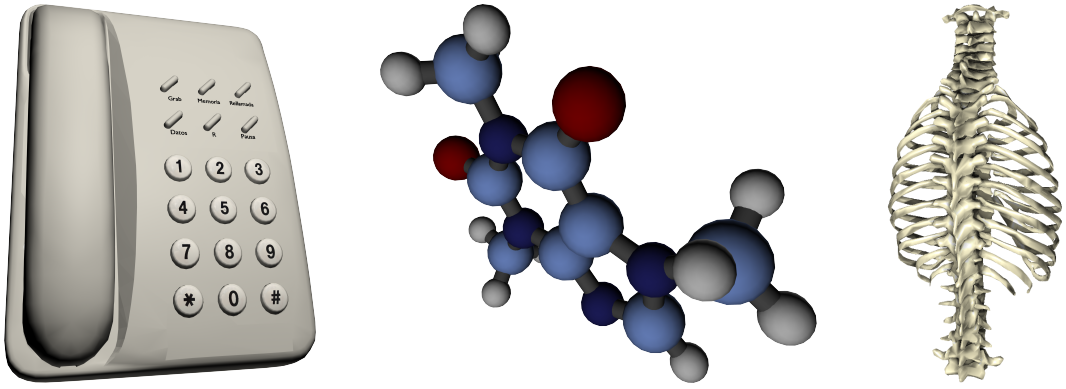
This behavior though does not appears if an eye-rooted technique is used.

## 5.3 Evaluation of expanding targets techniques

We conducted an experiment to evaluate how both expanding target techniques behave in combination with raycasting selection. Since the effort to select small and partially occluded objects in the default raycasting implementation is governed by the final corrective movements, in the best case scenario one could expect Dynamic Scaling (DS) and Forced Disocclusion (FD) to have a positive impact in selection performance.

In practice, however, this may not be the case. On the one hand, the movement of neighboring targets to avoid occlusion with expanded targets could be potentially distracting to the users and negate the benefits of DS. On the other hand, by forcing the focus object to appear unoccluded, it might occlude neighboring objects and result in poor performance. The main goal of our experiments is therefore to evaluate potential advantages of DS and FD in selection time, error rates, user discomfort and user confidence, and in scenes with multiple targets at varying densities.

To evaluate both techniques, we have designed a user evaluation where users are presented with an selection intensive tasks in three virtual environments. The environments have different object density to explore how the proposed pointing facilitation techniques behave: (a) Evaluate the effects of expanding targets in a 3D scene with low target density (see Figure 5.8a), (b) explore how behave in a representative of a scenario with multiple targets at close proximity (see Figure 5.8b) and (c) study the effects of DS and FD on a worst-case scenario with multiple potential targets with a high degree of overlap (see Figure 5.8c).



**(a)** *Telephone* **(b)** *Molecule* **(c)** *Thorax*

**Figure 5.8:** *Virtual environments employed in the evaluation of Forced Disocclusion and Dynamic Scaling.*

### Design and procedure

A repeated-measures, within-subjects design was used. The independent variables were DS (enabled, disabled), and FD (enabled, disabled). Since all targets appeared already unoccluded in the telephone environment, only DS was considered as a factor.

The dependent measures were total selection time, homing time, error rate and focus changes. For homing time we consider the time from the first intersection of the ray with the target until the selection is achieved. The focus changes were the number of times the target object changed its selection status prior to confirming the selection.

For each combination, the task was to select a sequence of objects, where the next object to be selected was clearly highlighted. The session was divided into four blocks, one for each combination of techniques (DS x FD). The order was randomized for each user. The virtual environments were always presented in increased order of density, first the telephone, then the molecule and finally the thorax.

Before each session users were provided with a short training session which required them to complete practice trials. Each participant performed the experiment in one session lasting approximately 25 minutes and were requested to complete the selection task as accurate as possible.

Additional feedback was provided by highlighting the object intersected by the selection ray, and acoustic feedback was also provided by triggering a different sound every time users hit or missed a target.

### Apparatus

The experiments were conducted on a four-sided CAVE with active-stereo projectors at $1280 \times 1280$ resolution. The input device was a 6-DOF Ascension Wanda and a tracking system with 2 receivers providing 60 updates/s with 4 ms latency. At the user position (90 cm from the EM emitter), position and orientation RMS errors were below 0.5 mm and 0.06 degrees, respectively. The experiment was driven by a cluster of 2.66GHz QuadCore PCs with GF8800 GTX cards.

### Participants

Sixteen volunteers (4 female, 12 male), aged from 24 to 41, participated in the experiment. Most participants (9) had no experience with VE applications; 5 had some experience and 2 were experienced users.

### Results

Let us first discuss the results obtained for the first environment (telephone). The one-way ANOVA showed a significant effect for DS in error rate ($p < 0.001; F = 15,50$), homing time ($p < 0.05; F = 4.37$), and focus changes ($p < 0.001; F = 26,33$). In all these measures users performed significantly better with DS enabled. This result was expected as in the telephone model potential targets are relatively far apart and benefits of enabling DS on

accuracy are more apparent. However for the total selection time no significant differences were found. One possible explanation is that users wait until the expanding animation is finished to confirm the selection.

For the second environment (molecule), the two-way ANOVA showed a significant effect for DS in error rate ($p < 0.01; F = 7.28$) and focus changes ($p < 0.01; F = 7.94$). Again users performed significantly better with DS enabled. No significant differences were found in terms of selection time and homing time for DS and FD. This suggests that DS benefits on accuracy also apply to scenes with potential targets located in close proximity. The lack of a significant effect for FD was also expected considering the relatively low level of occlusion between targets.

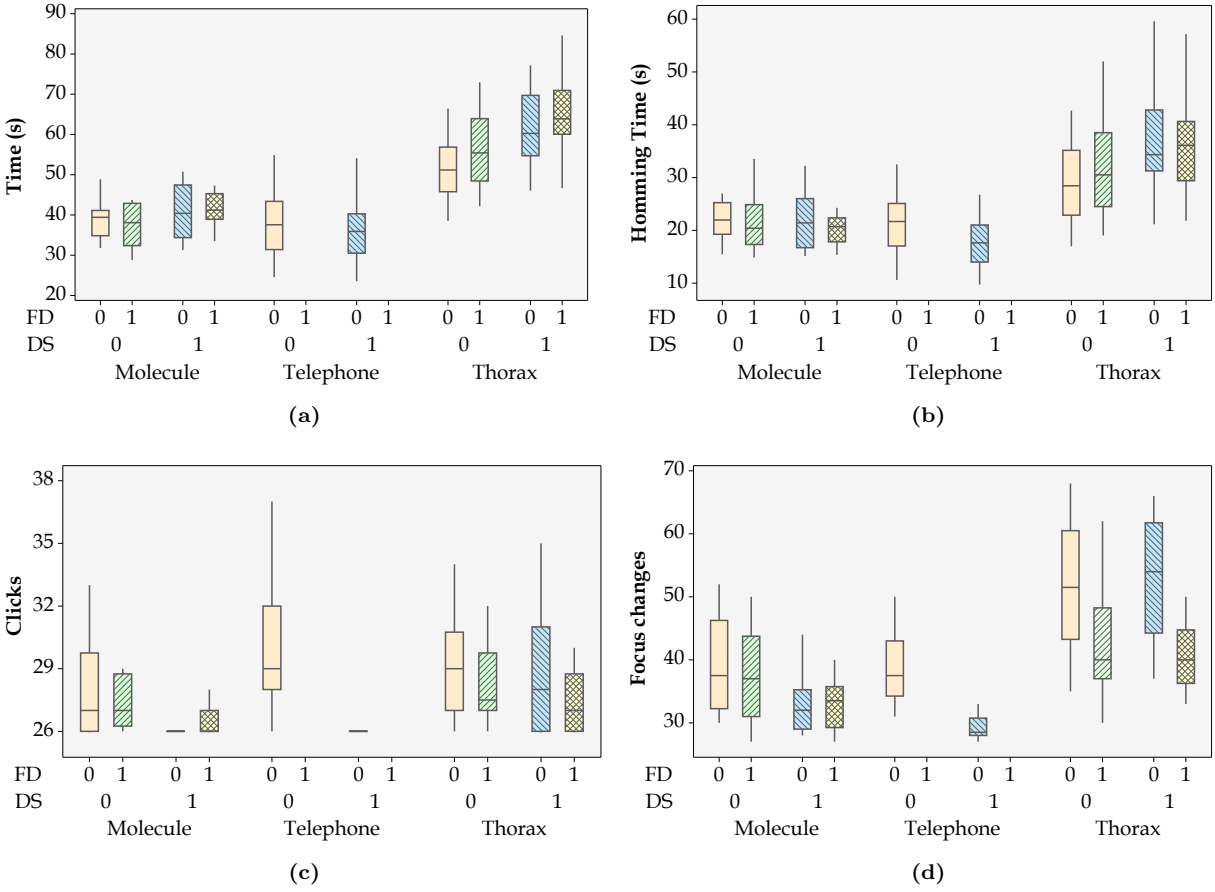



**Figure 5.9:** *Boxplots of the number of clicks (a), homming time (b), clicks (c) and focus changes (d).*

Regarding the last environment (thorax), the two-way ANOVA showed a significant negative effect for DS in selection time ($p < 0.001; F = 17.93$), and homing time ($p < 0.01; F = 7.11$), whereas we found no significant effect for DS in error rate ($p > 0.3$). Conversely, a positive effect was found for FD in error rate ($p < 0.05; F = 4.65$) and focus changes ($p < 0.001; F = 26.24$). No significant effect was found for the DSxFD.

**Survey**

After each block participants were requested to rate the easiness of the selection, the confidence on hitting the right target immediately before pressing the button, and the level of physical effort, using a 7-point Likert scale. We applied the Friedman test to rank techniques for each question. The test only showed significant differences in easiness and confidence. Users ranked the selection tasks as significantly easier when DS is enabled ($p < 0.05$), followed by FD, DS+FD and RC.

Concerning the confidence on hitting the target, the Friedman test indicated a significant difference ($p < 0.01$), ranking DS+FD in the first place, followed by DS, FD and finally RC. This seems to confirm that, since the selection of difficult targets is governed by the final corrective phase, dynamic scaling is perceived by the user as a positive facilitation technique.

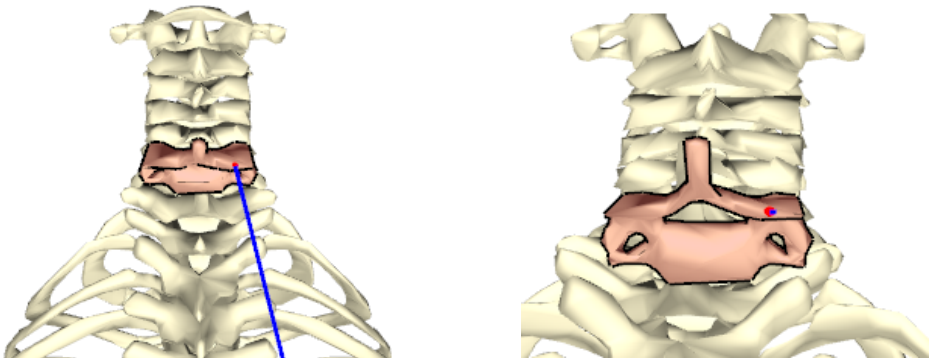
## 5.4 Discussion

The results of DS for the thorax model were worse than expected. An analysis on a per-object basis revealed that two vertebrae were particularly difficult to select with DS enabled. We discovered that the common point on these two objects was that, when expanded, their through hole was visible from the user's hand but not from the user's eye; a clear example of eye-hand visibility mismatch.

This situation is depicted in Figure 5.10. Our best explanation for the poor behavior of DS with these two objects is that the following sequence of events was repeated multiple times until the selection could be confirmed by the user:

1. The target was intersected by the ray, thus triggering its expansion.
2. The expansion caused the through object to be exposed, potentially causing the selection ray to go through the hole and thus missing the object.
3. Once the object lost the focus, the object smoothly returned to its original size.

We observed that this loop was repeated several times for both objects. The fact the hole was not visible for the eye was another major factor that came into play to make selection more difficult. This behavior seems to be confirmed by the abnormally high number of focus changes. Again this is another example of how eye-hand visibility mismatch can hinder selection tasks.



(a) (b)

**Figure 5.10:** *Difference between the scene as seen from: (a) the user's viewpoint and the user's hand viewpoint (b). Note that the hole is only visible from the hand position*

Should time performance be the primary goal, the two presented embodiments do not provide a significant advantage. As with most of Fitts' based techniques focusing on increase the size of the targets, time improvements are more apparent in situations where targets are isolated. When targets are more closely packed together, the benefit of these techniques tend to degrade, and can even be detrimental to selection time.

This does not mean though, that 3D expanding targets do not offer some advantages. In the context of real usage in a VR application, the subjective impressions of an interaction technique can play a much larger role than speed in controlled experiments. The inability to control accurately the selection ray may prove to be overly annoying to the user and thus be a source of dissatisfaction. As observed in the survey, users perceived the task easier when dynamic scaling was enabled and both approaches allowed for lower error rates.

By focusing on the final movement phase, the proposed techniques help the user to keep the selection tool over the intended object and confirm the selection with more certainty, due to increased motor space. In addition, the enlargement in visual space also facilitates the visual recognition of the object, an important aspect which is lacking in techniques manipulating the CD ratio.

On the other hand, concave objects and objects with through holes pose a problem with Dynamic Scaling, as the ray might no longer intersect the object after the expansion. An improvement will rely on modifying the intersection test to avoid such a distracting effect. Also, is still unclear how the Forced Disocclusion affects depth perception, it may be interesting to evaluate the potential conflict between occlusion and binocular depth cues.

# Chapter 6

# Interacting with 2D GUIs embedded in VEs

Application control is one of the fundamental tasks a Virtual Reality application must ensure. It refers to the user task of issuing commands, requesting the system to accomplish a particular function or changing its internal state [12]. Given its flexibility and its ease of use, WIMP user interfaces are the facto standard for PC desktop based interaction.

By providing mechanisms to efficiently deploy and interact with 2D graphical user interfaces in virtual environments, we can provide increased flexibility and functionality to VR applications. However, the usage of 2D graphical user interfaces in virtual environments present two main issues.

First, existing GUI toolkits for VEs are still too simple; they allow only for a limited number of GUI components and often lack visual authoring tools. In contrast, existing GUI toolkits for 2D desktop environments are mature, include powerful authoring tools, have a wide range of widgets and are actively maintained.

Second, efficient selection and manipulation of 2D GUI elements (widgets) are required. A common requirement for graphical user interfaces is that they have to cover a relatively small field of the user's viewport to limit the occluded content. As the GUI is placed in 3D space, it can be placed away from the user or downscaled. Nevertheless, in both cases, it will require the user to select and manipulate small widgets, potentially decreasing performance, increasing error rates and reducing comfort.

In this chapter, we address both issues by presenting a new approach for fast development of application-control graphical user interfaces in virtual environments and two orthogonal approaches to improve performance and comfort when interacting with complex GUIs.

## 6.1 A cost-effective approach for embedding 2D GUIs in virtual environments

Rather than providing a new API for defining and managing the user interface components, we aim at extending current 2D toolkits so that their full range of widgets can be displayed and manipulated either as 2D shapes on the desktop or as textured 3D objects within the virtual world. The proposed approach allows 3D GUI developers to take advantage of the increasing number of components, layout managers and graphical design tools provided by 2D GUI toolkits. Resulting programs can run on platforms ranging from fully immersive systems to generic desktop workstations with little or no modification.

The basic components of the system are depicted in Figure 6.1. The *host toolkit* is a 2D GUI extensible toolkit such as Qt [26], whose widgets are accommodated to VE applications by the *extended toolkit*. A key feature of our approach is that the additional features are provided by only subclassing the *host toolkit*. The first consequence is that existing applications with 2D GUIs already designed with the *host toolkit* can be adapted to a VE environment with minimum effort and all the application's source code for GUI creation and behavior remain unmodified.

We now introduce some notation that will be used through-out the rest of the section. The word widget is used to refer to user interface objects such as windows, buttons and sliders that are used as the basic constituents of GUIs. A widget receives mouse, keyboard and other events from the environment, and paints a representation of itself on the output device. Widgets are arranged into a hierarchical structure. A widget that is not embedded in a parent widget is called a top-level widget. Usually, top-level widgets are windows with decoration (a frame and a title bar). Non-top-level widgets are child widgets. We also

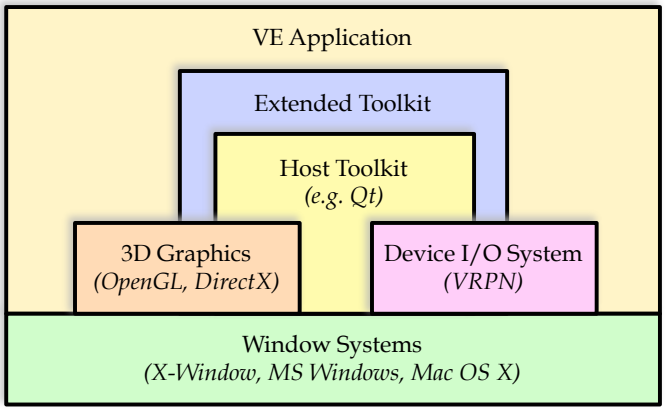



**Figure 6.1:** *System overview*

distinguish between widget objects provided by the host toolkit (called native widgets) and the objects that represent widgets as texture-mapped rectangles in 3D space (called virtual widgets).

Events of current 2D GUI toolkits can be roughly classified into two categories: application events, user events and synthetic events. *Application events* refer to GUI-related high-level events such as show, hide, close, resize and paint. On the other hand, *user events* are produced by simple user actions through input devices. According to the originating source, we also distinguish between *native events* originating from the window system or the host toolkit, and *synthetic events* initiated by the extension classes. For example, when the user presses a button on a 3D wand with the aim of selecting a menu item, a user event is created and then translated into a synthetic event (a mouse press event in this case) that is send to the native widget representing the menu item.

### 6.1.1 System components

The core of the extension toolkit consists on the components depicted in Figure 6.2. The main responsibilities of each component are described below.

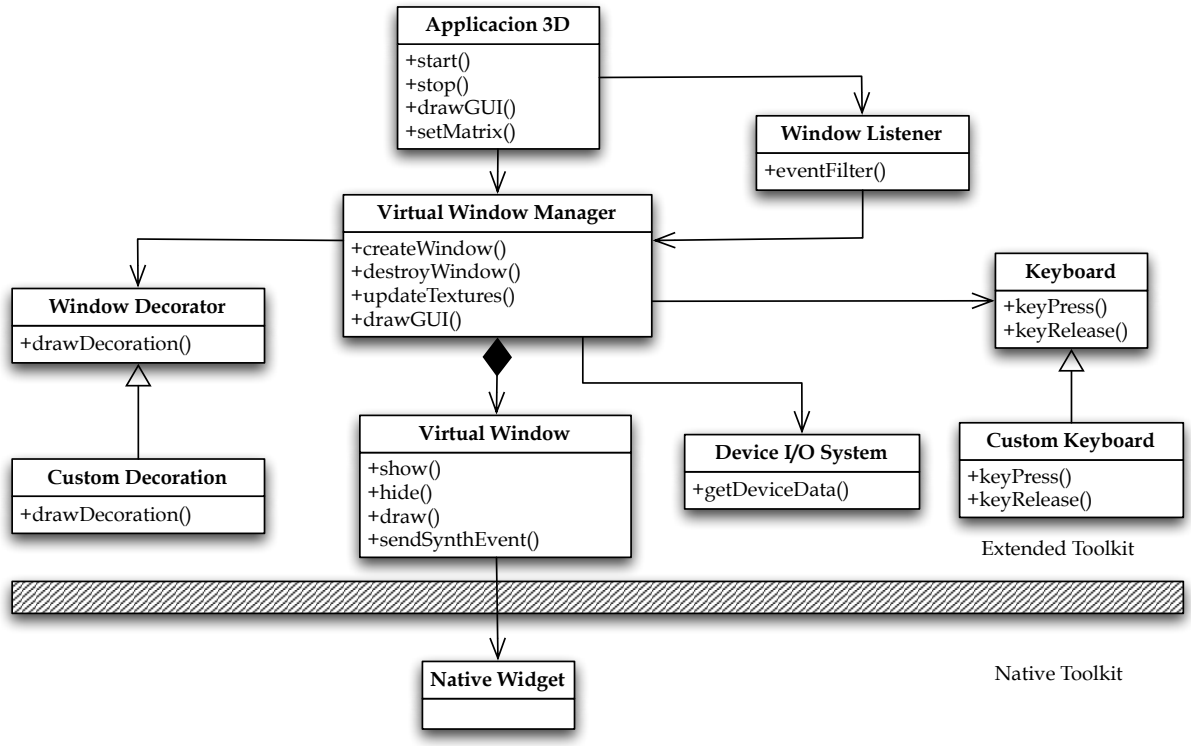



**Figure 6.2:** *UML Conceptual design of the extended toolkit. Only the most relevant components and operations are shown.*

The *Application 3D* provides a unified interface to the VE application, delegating client requests to appropriate subcomponents. This is the only component the application has to collaborate with, thus making the toolkit easier to use. The Application 3D manages the drawing of the GUI windows in 3D space as texture-mapped objects (forwarding the request to other components) and configuration options.

The *Virtual Window Manager* manages the behavior of virtual windows associated with top-level native widgets. This component handles basic window operations such as creation, destruction, show, hide and resize operations. This component also manages user events. This includes getting VR devices input data through the Device I/O System, managing interaction with the virtual window decoration through a decorator subclass, and translating user events into synthetic events that will be send to the appropriate virtual window.

*Virtual Windows* keep attributes concerning the 3D version of a top-level native widget such as rectangle size, texture size and transformation matrix. This component checks for intersection between the virtual window's plane and a selection primitive (e.g. a ray). In response to user events, Virtual Window objects send synthetic mouse events (to the child widget at a given 2D position) and keyboard events (to the child widget having the keyboard focus).

The *Window Listener* acts basically as an application-event filter. This component monitorizes the creation of new native widgets by the application and asks the Virtual Window Manager to create a new virtual window every time the application instantiates a top-level widget. Moreover, it also monitorizes application events related with a widget's life cycle: create, show, hide, paint, close and resize events originated from within the application. For each paint event (change on the image content) asks the virtual window manager to update the texture rectangle of the virtual window containing the native widget.

The rest of the components have simple responsibilities. The *Virtual Keyboard* defines an interface for a virtual keyboard that sends synthetic keyboard events to the application in response to user actions. The *Window Decorator* defines an interface for drawing the decoration of a virtual window. Decoration includes the frame, and buttons to iconify, deiconify and close the virtual window. Finally, the *Device I/O system* is used to read data from an extensible set of generic input devices.

### Widget creation and placement

Each time a native window is created, the Virtual Window Manager receives a notification. If the widget is top-level, a virtual window is created along with a texture map capturing

the contents of the widget's area. A rectangular portion of this texture is updated every time a widget of the native window changes its appearance.

An important issue is the initial window placement inside the 3D world. The placement strongly influences the user's ability and accuracy. A situation which arises in the use of multiple windows is the need to constantly arrange windows to get access to the particular window which houses the task or information needed at a given instant (window thrashing) [68]. According to the spatial reference, we can considerer windows that are world-referenced, object referenced, head-referenced and device-referenced [12]. Head-referenced and body referenced menus provide an appropriate spatial reference frame as they take the most of the user's proprioceptive sense, allowing users to accomplish some application control tasks without having to look at the menu. Since the extension toolkit knows everything about the widget hierarchy, any of these strategies can be plugged in.

### Input handling

User events such as hand movements are captured by the Device I/O System and converted into synthetic events that are sent to native widgets. We will describe the main steps involved in this process with a concrete example. Suppose a CAVE user wearing a virtual wand (the wand has a six-DOF sensor, a two-DOF joystick and three buttons). Each time the user presses the left button a new user event is generated. This event contains a ray and a button state as parameters. This event is forwarded to the Extended toolkit which searches for the nearest virtual window containing the ray intersection. Finally, the intersected virtual window creates a synthetic event (a mouse event) and posts it to the native widget. The main advantage of this solution is that the management of the GUI elements and their behavior is delegated completely to the host toolkit, thus simplifying the migration of existing GUIs. Moreover, this approach allows the system to provide several interaction techniques for object selection such as ray-casting and arm-extension.

### Host Toolkit Requirements

We now summarize the main features required for an extensible 2D GUI toolkit to be used as a host toolkit for the 3D extension:

- A hook to monitorize the creation and destruction of top-level widgets. In addition to this, the toolkit has to provide a mechanism to intercept widget-related events such as paint, show, hide, close and resize. For example, if a widget is repainted in response to

a paint event, the application has to intercept the event to know that the associated texture requires an update. This feature is required by the Event Listener component.

- An operation to send a synthetic event (keyboard/mouse) to any widget (required by the Virtual Window component).
- A function to capture the image of a native widget into a bitmap. The image will be used to update the texture of the virtual window containing the widget (required by the Event Listener). If this function is supported also on widgets hidden by other windows, then the native GUI can share the desktop space with the OpenGL window where the VE application renders the virtual world.
- Functions to traverse the widget hierarchy (access to parent and children widgets) and operations to obtain the visible child widget at a given pixel position.

### 6.1.2 Prototype

A prototype system has been implemented above Qt, a cross-platform GUI development toolkit, and evaluated. Qt fulfills all the requirements listed in the previous section. Application events can be intercepted through adding a application event-filter through *QApplications::installEventFilter()*. The filter can either stop the event or forward it to other objects. Moreover, Qt allows multiple event filters to be installed, thus avoiding conflicts with application-defined event filters.

Creation of top-level widgets can be monitorized by listening to the *QEvent::Show* event which is send each time a window becomes visible. Changes in the widget's content are monitorized by listening to the *QEvent::Paint*. The QPixmap class provides two methods to grab the contents of a widget: *grabWindow()* grabs pixels directly off the screen, whereas *grabWidget()* asks the widget to paint itself by calling *paintEvent()* with output redirected to a bitmap. Although a bit slower, the later is more suitable because it works with hidden widgets.

Synthetic events (including predefined and user-defined events) can be send to any object through *QApplicationpost::Event()* and *QApplication::sendEvent()* methods. The former adds a synthetic event to the event queue with a given object as the receiver of the event; the later sends the event directly to the object. Finally, the toolkit provides operations for traversing the object hierarchy and for returning the visible child widget at a given pixel position in the widget's own coordinate system.

An example of collaboration is shown in Figure 6.3. (1) When a widget detects that it

should repaint itself, it sends an event to the QApplication indicating which part of the widget should be repainted. (2) When appropriate, QApplication tells the widget it has to be painted. (3) This event is captured by the event filter and (4) produces a grab widget call to the QPixmap object, (5) which sends a repaint command to the QWidget. Finally, (6) the window listener asks the virtual window manager to update the texture.

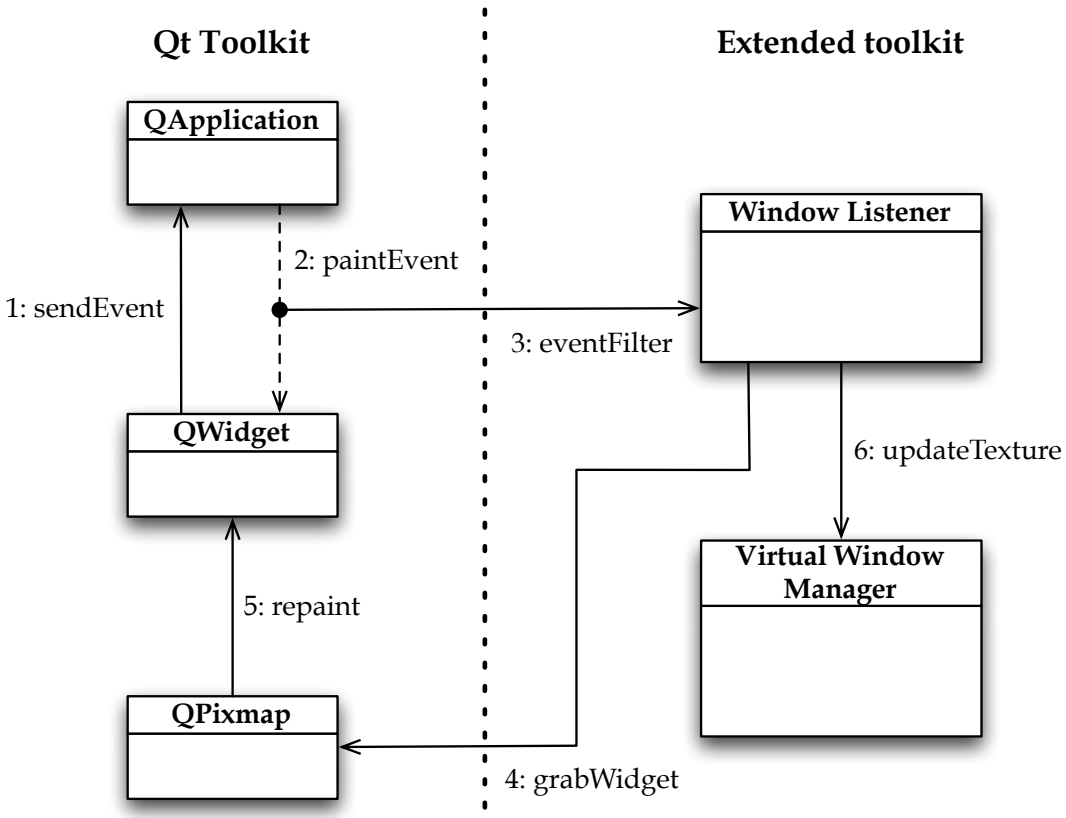



**Figure 6.3:** *Collaboration for processing a paint event between the Qt and our Extended toolkit.*

The prototype has been tested with two different applications: a scene viewer specialized for shipbuilding design (see Figure 6.4) and a volume rendering application (see Figure 6.5). Both applications had a Qt-based GUI and were extended to display stereoscopic images on either a CAVE or a stereo workbench. The source code modifications needed to accommodate the GUI to the CAVE where minimum. Moreover, it allowed the evaluation of the interaction techniques for selecting and manipulating 2D GUIs detailed in Sections 6.2 and 6.3.

The source code of the extended toolkit is available for download under the GNU GPL license at: http://www.lsi.upc.edu/∼virtual/Qt3D.

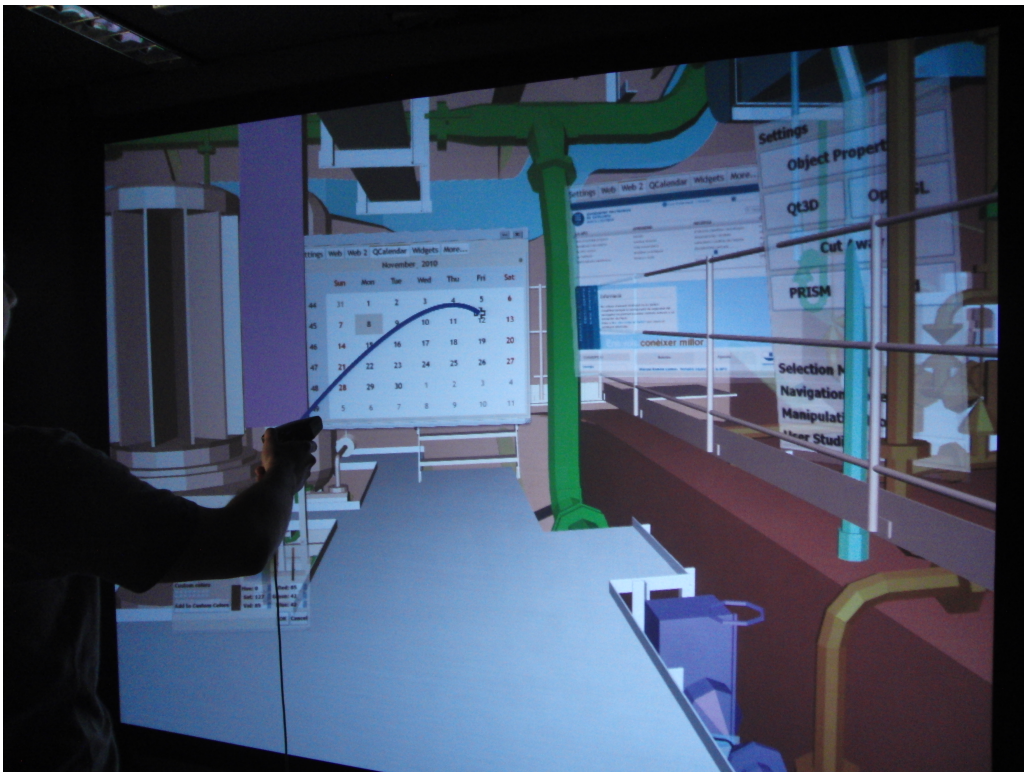



**Figure 6.4:** *Scene viewer specialized for shipbuilding using the developed GUI toolkit.*

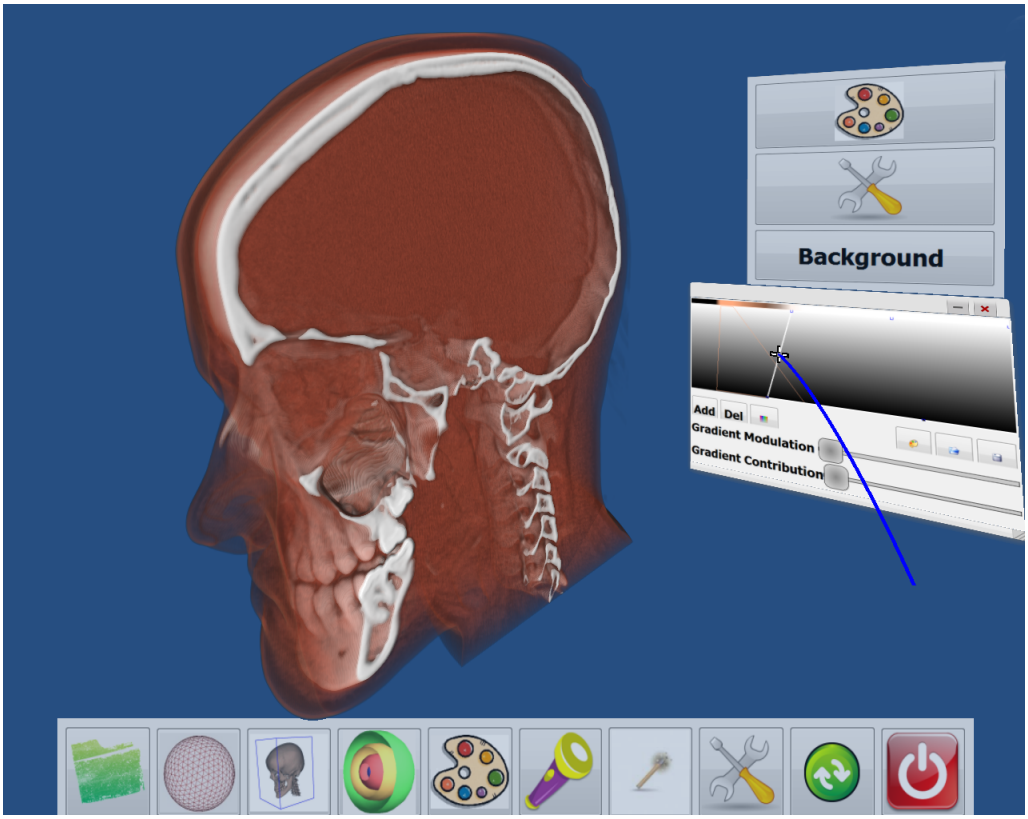



**Figure 6.5:** *Volume rendering application using the developed GUI toolkit.*

### 6.1.3 Discussion

Our approach has several advantages over previously-reported systems. Applications adopting our approach can provide an interface optimized for each platform (e.g. a desktop interface can be used on a desktop system, and an immersive interface can be used in an immersive system), without needing to modify the application. Moreover, importing 2D interfaces into the VE has some considerable advantages:

- Transparent use of all the functionalities provided by the host toolkit, enabling the use of a large number of widgets.
- Delegation of functional interface handling to an independent component.
- Users can work with the graphical user interface they are used to.
- Accommodation of different 3D selection techniques.

From the point of view of software development, the advantages of our approach are:

- An important part of the UI can be developed and tested in a desktop workstation.
- Access to UI graphical design tools (e.g. QtDesigner).
- Fast porting of existing applications to VEs with minimum changes.
- Transparent support to cluster-based systems and collaboration.

Our approach has some additional advantages over VNC-based methods for immersing 2D GUIs. The system is not framebuffer-oriented but widget-oriented. That implies that the VE application knows everything about the immersed GUI, not only framebuffer updates. That allows for much more flexibility for placing and sizing the widgets: host widgets can be automatically resized so that width and height are appropriate for fast texture conversion (power of two), widgets can be independently resized and moved from within the VE application, pop-up menus and pop-up lists can be true pop-up components, i.e. they can be drawn at a certain offset from the parent widget; the look-and-feel is completely customizable from within the VE application. Moreover, our system does not have any network nor image encoding overhead. For example, when running on a cluster, texture updates are not broadcasted to all clients because the native widgets and the virtual windows are local to each process.

## 6.2 Anisomorphic raycasting interaction

Two-dimensional windows in 3D environments can include small, nearby buttons which can be difficult to select and manipulate using standard 3D interaction techniques. For example, raycasting does not perform well when selecting small or distant objects. Small rotations of the wrist sweep out large arcs at the end of the selection ray. Therefore hand trembling and tracking errors are amplified with increasing distance, thus requiring a higher level of angular accuracy.

Accurate selection is also compromised by hand instability, amplified by the absence of constraints on the hand movements. Selections of small widgets require a considerable effort to stabilize the selection tool.

### 6.2.1 Friction Surfaces

We have conceived the Friction Surfaces metaphor to facilitate the interaction with external 2D applications being accessed from immersive VEs. The main goal is to provide accurate selection and manipulation of 2D GUIs that have not been particularly designed for VEs. To this end, we use a modified raycasting technique which adapts the CD ratio according to the size and position of the virtual window.

Friction surfaces uses two distinct modes: one which scales hand rotations when accuracy is needed (scaled mode) and one which provides direct, isomorphic interaction (normal mode). The scaled mode is activated automatically whenever the selection ray enters a virtual window. The mode is set back to normal mode when the selection ray leaves the active window.

We start by introducing some notation that will be used in the rest of the section. The *Device Coordinate System* (DCS) is an orthonormal frame centered at the position of the 6-DOF sensor attached to the user's hand. We assume the DCS is oriented as depicted in Figure 6.6a, with the negative Z axis defining the user's hand pointing direction. This pointing direction will be referred to as the *device ray*. The *Zero orientation* is an orthonormal reference basis is defined from DCS and the window's center when the scaled mode is activated and then remains unchanged until the mode is deactivated.

During the normal mode, the ray does not intersect any window, the CD ratio is always one; the device mapping is isomorphic. When the ray intersects a window, the mode switches to scaled mode and CD ratio is computed as follows. First, we compute the zero orientation reference frame $(\vec{x}, \vec{y}, \vec{z})$ given the virtual window and the DCS (see Figure 6.6b). Let $\vec{z}$ be a

unit vector in the direction of the segment joining the window center and the device position at activation time. Let $\vec{u}$ the unit vector defined by the vertical orientation of the window. We compute $\vec{x} = \frac{\vec{u}\times\vec{z}}{\|\vec{u}\times\vec{z}\|}$ and $\vec{y} = \vec{z} \times \vec{x}$.

Then, we compute the range of directions the device ray can travel before leaving the active window. Let $P_i$ be the $i$-th vertex of the virtual window. Let $\theta_i$ be the azimuthal angle (longitude) of $P_i$ in the XZ-plane, measured from the negative Z-axis of the zero orientation, with $0 \leq \theta_i < 2\pi$. Likewise, let $\phi_i$ be the zenith angle (latitude) from the XZ-plane, with $-\frac{\pi}{2} \leq \phi_i \leq \frac{\pi}{2}$.

These spherical coordinates can be computed using Equations 6.1, where $(x_i, y_i, z_i)$ are the $P_i$ coordinates relative to the zero orientation and where the inverse tangent must be suitably defined to take the correct quadrant into account:

$$(\theta_i, \phi_i) = \left( \tan^{-1}\left(\frac{-x_i}{-z_i}\right), \sin^{-1}\left(\frac{y_i}{\sqrt{x_i^2 + y_i^2 + z_i^2}}\right)\right) \tag{6.1}$$

Once computed the angles for the four window's corners, we obtain the maximum rotational angles in both directions as $\theta_{max} = \max_i \left\{|\theta_i|\right\}$ and $\phi_{max} = \max_i \left\{|\phi_i|\right\}$. Since we want to use isotropic scale on both directions, we just use the maximum of both. Therefore, the CD ratio $r$ is computed as shown in Equation 6.2, where $\psi$ is a user-defined constant. $\psi$ determines the range of directions of the input device that approximately map onto a selection ray within the virtual window. If the device ray exceeds that angle the selection ray will fall outside the active window. In our implementation we employed $\psi = \pi/4$, thus providing the user with a 90 degrees arc for interaction with the active virtual window. If the window is sufficiently close to the user ($r < 1$), we do not enable the scale mode and keep the CD ratio

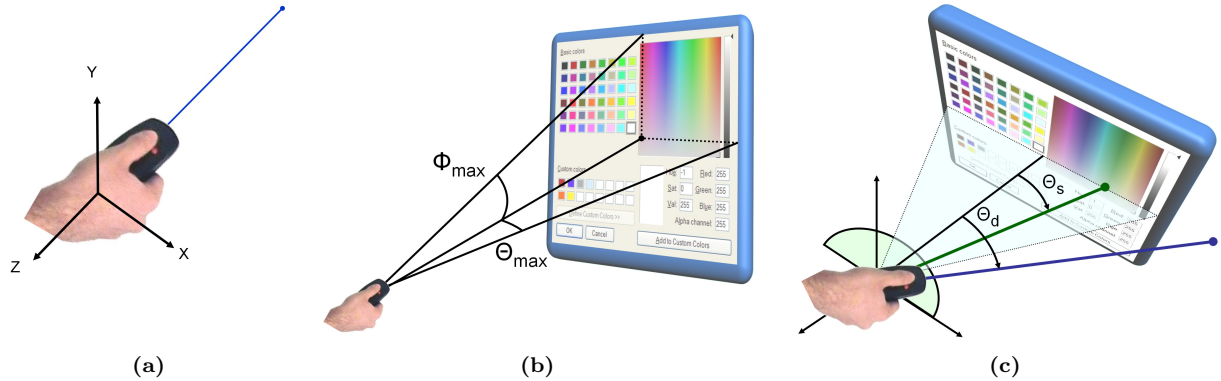



**Figure 6.6:** *Elements involved in the computation of the CD ratio during the scaled mode: (a) Device coordinate system and device ray, (b) spherical coordinates used at activation time to fix the CD ratio , (c) spherical coordinates used for computing the selection ray direction during scaled mode.*

equal to one.

$$r = \max\left(1, \frac{\psi}{\max(\theta_{max}, \phi_{max})}\right) \tag{6.2}$$

As the deactivation causes the selection ray to coincide again with the device ray, the selection ray might enter another window. We found this situation to be rare in practice as most windows are likely to be placed in front of the user. Nevertheless, unintentional activation can be easily avoided by waiting a short period of time (e.g. half a second) before the scaled mode is activated over the new window.

### Computation of the selection ray

In scaled mode, the selection ray is computed using the CD ratio defined at activation. Let $\theta_d$ and $\phi_d$ be the spherical coordinates of an arbitrary point of the device ray (distinct from its origin) with respect to the zero orientation basis. We first compute the spherical coordinates $\theta_s$ and $\phi_s$ of a target point $T$ by applying the CD ratio, $\theta_s = \theta_d/r$, $\phi_s = \phi_d/r$ (see Figure 6.6c).

The new coordinates of $T$ require a simple conversion back to cartesian coordinates (see Equation 6.3, $\rho > 0$ is an arbitrary value). The resulting selection ray passes through the current device position and point $T$.

$$\begin{aligned} x &= -\rho \sin(\theta_s) \cos(\phi_s) \\ y &= \rho \sin(\phi_s) \\ z &= -\rho \cos(\theta_s) \cos(\phi_s) \end{aligned} \tag{6.3}$$

### Feedback

Two distinct options were considered to provide visual feedback. The first option consisted in drawing both the device ray and the selection ray, using different visual attributes (such as color and thickness). However, this option appears to be quite distracting so we have opted for a single bent ray providing feedback of both the device ray and the selection ray.

Similar to IntentSelect [32] and the Flexible Pointer [90], we draw a curved line segment using a Bézier spline (see Figure 6.7). The curve originates at the user's hand and ends at the intersection $P$ of the selection ray with the virtual window's plane. These two points define the first and last control points of the Bézier curve. The second control point is

computed on the device ray so that the tangent direction at the origin is that of the device ray. Finally, the third control point is the point on the device ray closest to $P$.

When the selection ray leaves the virtual window, the scaled mode is deactivated and the displayed ray instantly goes straight. The users' feeling when scaled mode is active is that a flexible ray gets curved as it is moved over a virtual high friction surface defined by the window. Visual feedback is completed by drawing a cross-shaped cursor in the intersection point $P$.

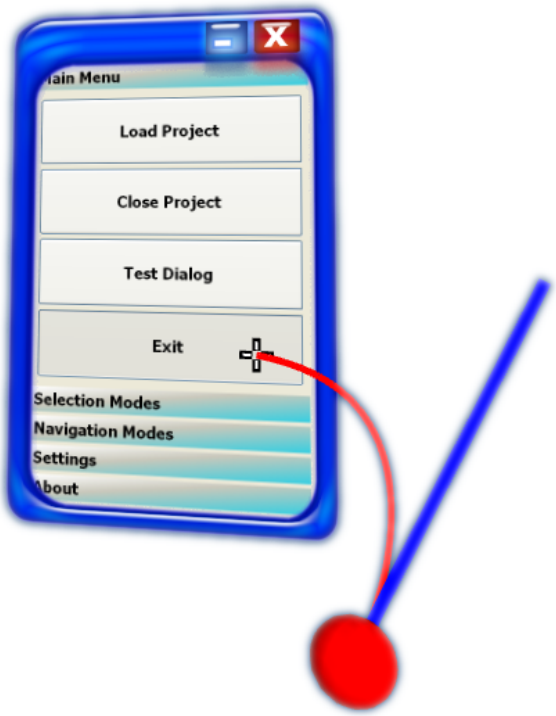



**Figure 6.7:** *The red ray corresponds to the feedback ray and the blue ray to the real device ray. The selection ray is not displayed but its defined by the hand's position and the intersection point between the feedback ray and the virtual window.*

### 6.2.2 Friction Surfaces evaluation

We conducted a usability evaluation to measure the effectiveness of the anisomorphic raycasting manipulation compared with classic isomorphic raycasting and raycasting in combination with PRISM [41]. The evaluation test was designed to evaluate the task performance in terms of time-to-complete a given task and the maximum accuracy achieved in a fixed period of time.

### Design and procedure

The test dialogs used in the experiments are shown in Figure 6.8. The first two dialogs are designed to measure task performance when selecting small/middle size objects. The first dialog contains different kinds of buttons whereas the second dialog includes basically combo boxes and selection lists. The third dialog is also designed to measure speed but putting the emphasis on manipulation rather than on selection. Finally, the fourth dialog is designed to measure the accuracy during object manipulation. In all cases the label attached to each widget indicates the requested task.

A repeated-measures, within-subject design was used. The dependent variable was the selection technique: raycasting (RC), friction surfaces (FS) and PRISM. Users performed the task once for each technique condition. The order of the conditions was randomized for each user to avoid ordering effects.

The dependent variables were selection time, error rates, the path length described by the cursor over the virtual window, and the amount of directional changes (changes of direction of the hand orientation greater than 90 degrees).

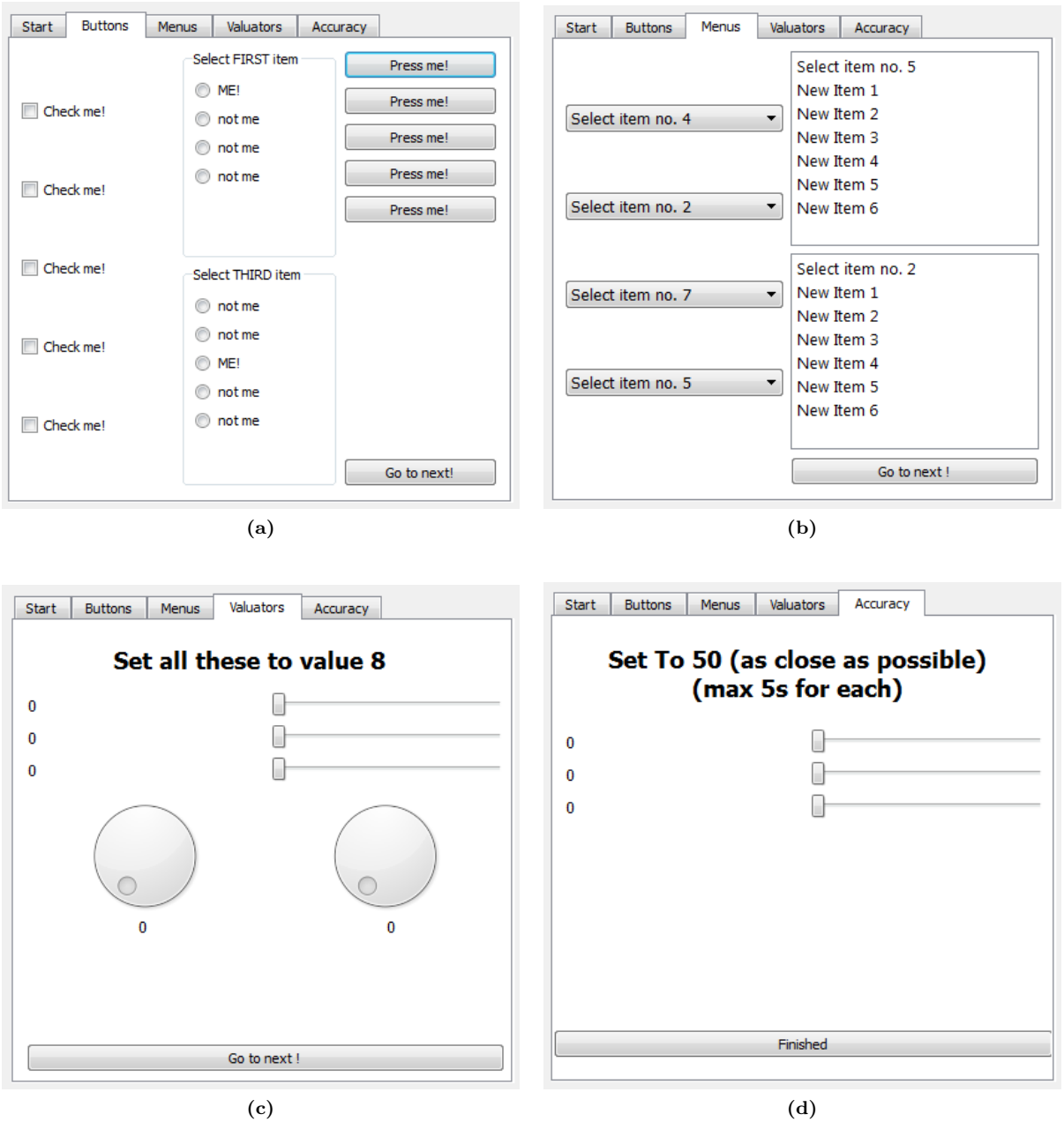
For the first three dialogs, users were requested to complete the involved tasks as quickly as possible. For the fourth dialog, users were asked to manipulate several sliders to get a certain value as accurately as possible, but giving only five seconds of time for each slider, starting from the first click on it. After that time, the slider was disabled and the user was forced to proceed with the next slider.

### Apparatus and virtual setup

All the experiments were conducted on a four-sided CAVE with a 6-DOF hand-held device and a Polhemus Fastrak tracking system with 2 receivers providing 60 updates/s with 4 ms latency. The virtual window used in the experiments was initially placed at 1.5 m from the CAVE center, covering about 20 degrees of the user's field-of-view.

### Participants

Seventeen users (undergraduate and graduate students) participated in the study, aged 22-42, 14 male and 3 female. Most participants (9) had no experience with VE applications; 5 had some experience and 3 were experienced users.




(a) (b)

(c) (d)

**Figure 6.8:** *The test dialogs used in the user evaluation.*

### Results

Figure 6.9a shows the completion time for each task. Tasks 1-4 correspond to the dialogs (a)-(d) shown in Figure 6.8. The one-way ANOVA of completion time versus interaction technique showed significant differences for task 1 ($p < 0.05; F = 4.15$) and for task 3 ($p < 0.05; F = 3.7$). Tukey pair-wise HSD tests showed only significant differences between RC and FS, ($p < 0.01$) for both tasks, users with FS performed significantly faster than users with RC.

Figure 6.9b shows the button clicks for each task. Note that in tasks emphasizing selection (tasks 1 and 2), users made less mistakes on average with FS, whereas PRISM yield better results in tasks emphasizing on manipulation (tasks 3 and 4). The one-way ANOVA showed significant differences in clicks for tasks 1 ($p < 0.001; F = 12.14$), task 3 ($p < 0.001; F = 13.62$) and task 4 ($p < 0.001; F = 5.53$). Tukey pair-wise HSD tests showed that PRISM

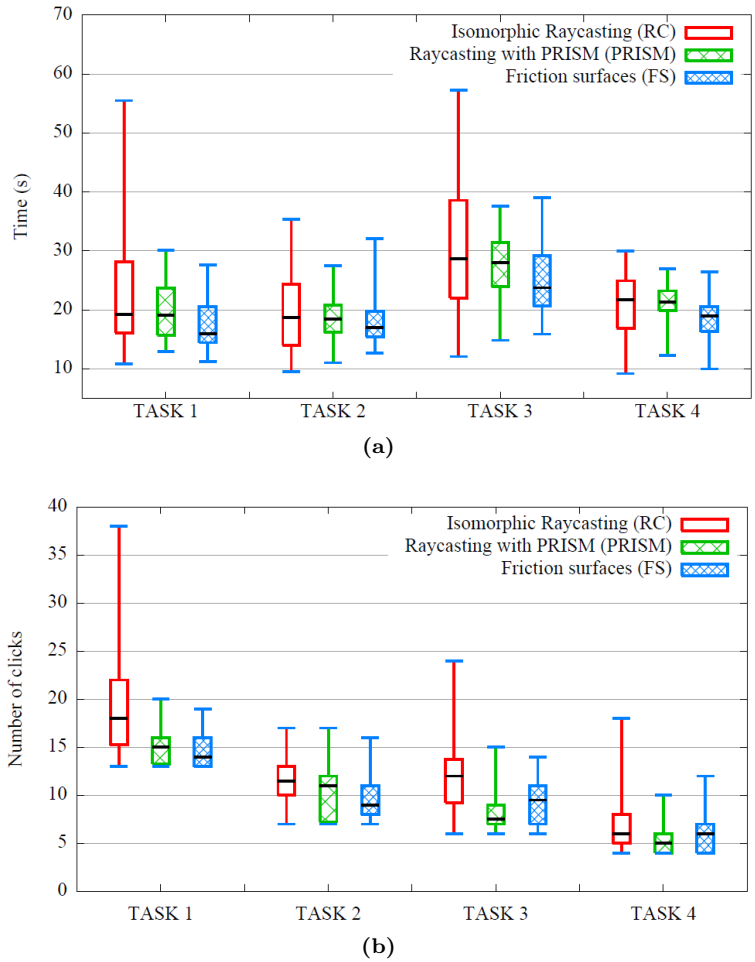



**Figure 6.9:** *(a) Time to complete the tasks involved for each dialog. (b) Number of button clicks performed during each task.*

was always significantly better than RC ($p < 0.01$), and FS was better than RC in tasks 1 and 3 ($p < 0.001$). Regarding the number of mistakes, it should be noted that PRISM incorporates a noise filter. Any motion below a given velocity is considered tracking error or inadvertent drift and the controlled ray is not moved. Note that our current implementation of Friction Surfaces does not have such a filter.

The length of the paths traced by the cursor, in native window pixel units, are shown in Figure 6.10a. The one-way ANOVA showed significant differences ($p < 0.001; F = 28.4582$). Tukey pair-wise tests confirmed that FS lead to cursor paths significantly shorter than both RC ($p < 0.01$) and PRISM ($p < 0.01$). Figure 6.10b also shows the number of times the user had to rectify their movement. Both PRISM and FS were found to produce less turns than RC ($p < 0.01$).

Figure 6.10c shows the results of the accuracy test (Task 4). The plot shows the average deviation (in slider units) from the target value when the user had only five seconds to adjust it (see Figure 6.8d). Each slider had increasing ranges and thus increasing levels of difficulty. Both PRISM and FS performed much better than RC, with nonsignificant differences between PRISM and FS.

Regarding the path traced by the 3D cursor over the virtual window, Figure 6.11 shows the paths described by users who achieved times on the first, second and third quartile values in Task 1. Color temperature represents the speed of the trace. Note that the lack of accuracy of isomorphic ray-casting forced the users to perform many attempts before the right button was selected. This is reflected by the loops around the small targets in Figure 6.11a. This contrasts with the smoother paths produced by PRISM and FS (Figure 6.11b and 6.11c).

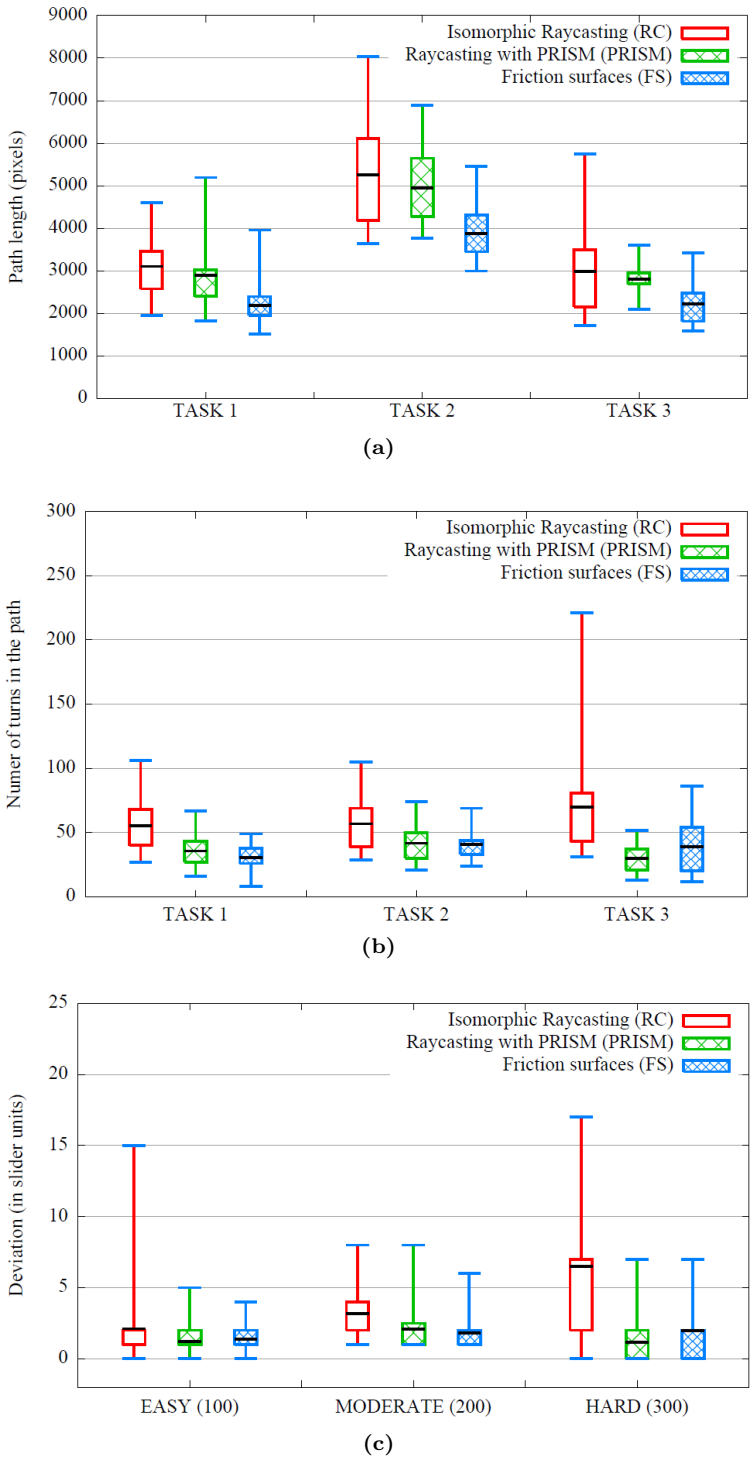
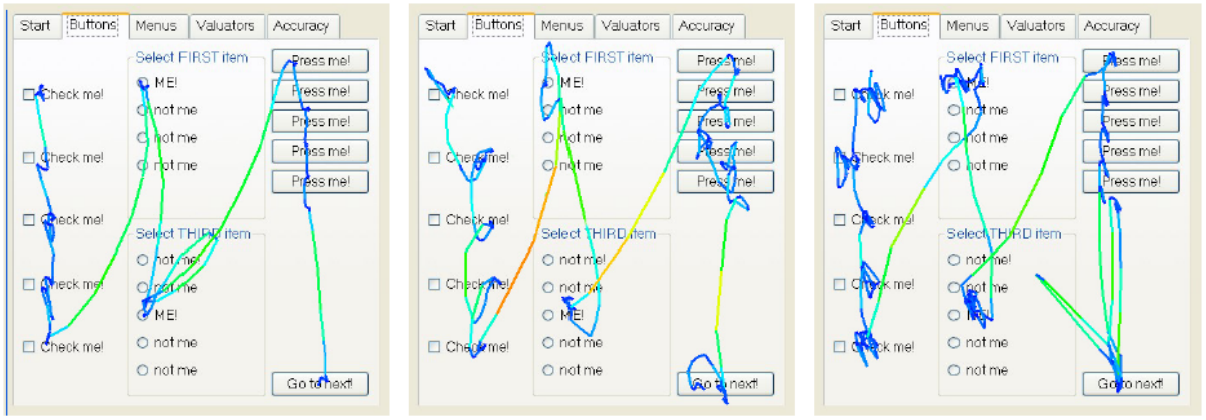
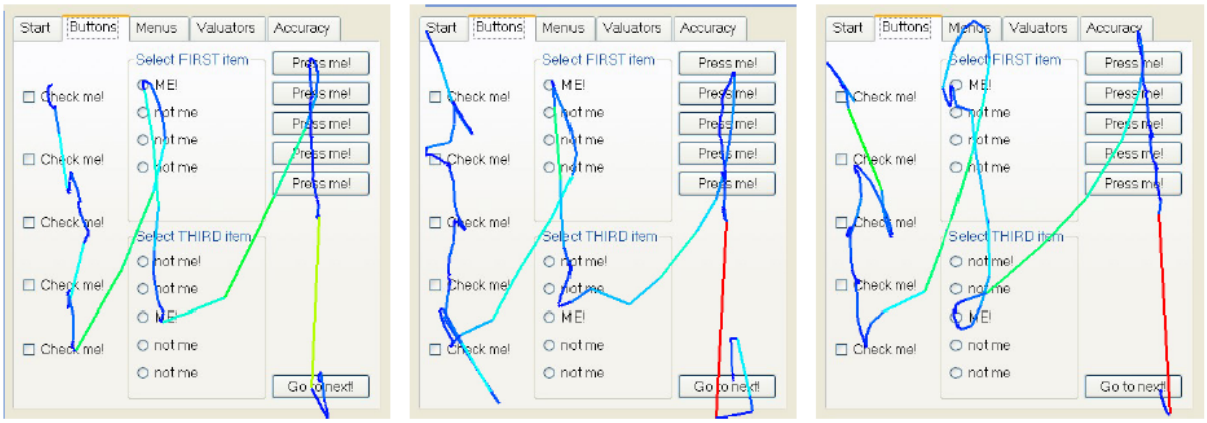


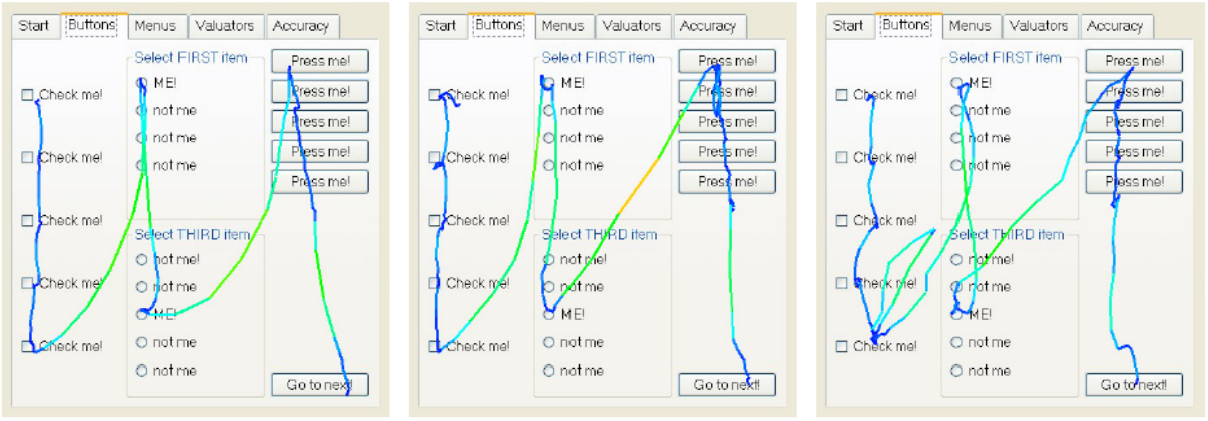

**Figure 6.10:** *(a) Length of the path traced by the cursor. (b) Number of turns in the path traced by the cursor. (c) Deviation from the target value on the accuracy test. The integer value between parentheses is the slider's range.*




(a)




(b)




(c)

**Figure 6.11:** *Path traced by the 3D cursor over the virtual window with RC (top), PRISM (middle) and FS (bottom). Paths correspond to users who achieved times on the first, second and third quartile values. Note that checkboxes can be toggled by clicking on their label.*

### Survey

After the experiments, subjects were requested to rate each interaction technique using a 7-point Likert scale. All users preferred either PRISM or FS against RC, with nonsignificant differences between PRISM and FS. Nine users preferred PRISM with average rate of 5.5; eight users preferred FS with average rate of 5.4. RC was given an average rate of 3.

We also asked subjects about what they found most difficult and what was the easiest for them. The results were similar from all users. All of them agreed on having less problems on selecting buttons and manipulating sliders with friction surfaces.

The most difficult task was to achieve a certain value on the sliders, because the free movement of the wand on their hand makes it difficult to maintain the value when the finger is moved to press or release the button. This problem was noticeably alleviated with PRISM and with our technique. Most users complained about the effort required for selecting small buttons with normal ray-casting because of the considerable effort to stabilize the wrist.

On the other hand, a few users pointed out that friction surfaces was a bit unnatural compared with isomorphic raycasting, although their performance was better using an anisomorphic mapping.

A limitation of our technique is that, for certain orientations, the curvature of the selection ray and its intersection with the virtual window is hard to perceive (when the viewpoint approaches the plane defined by the four control points of the curved ray). However, users did not find this to be a problem as the cross-shaped cursor showing the intersection of the selection ray with the virtual window was clearly visible.

### Discussion

Using an anisomorphic mapping between the user's hand orientation and the selection ray orientation, we are able to scale down hand rotations enabling accurate selection and manipulation of small GUI objects.

The user evaluation showed that PRISM and Friction Surfaces perform significantly better than classic ray-casting, with little performance differences between FS and PRISM. PRISM seems to perform better than Friction Surfaces when extreme accuracy is required (e.g. adjusting a slider with pixel accuracy) whereas Friction Surfaces is particularly suitable for fast selection of small targets. Both cannot be achieved with classical raycasting as they require the user a great effort to stabilize the pointing device before each selection, the

Heisenberg effect will further hinder precise selections.

A key feature of Friction Surfaces is that maintains both directional and nulling compliances [99] as it simulates approximately the interaction with a large spherical window (see Figure 6.12). Note that PRISM does not preserve nulling compliance and requires offset recovery techniques to reduce the accumulation of an offset value representing the angular difference between the hand and the ray being manipulated. Besides these aspects, an important difference is that Friction Surfaces does not force users to slow down their movements to gain precision. Our approach uses a larger range of movements for controlling the ray in a more reduced region.

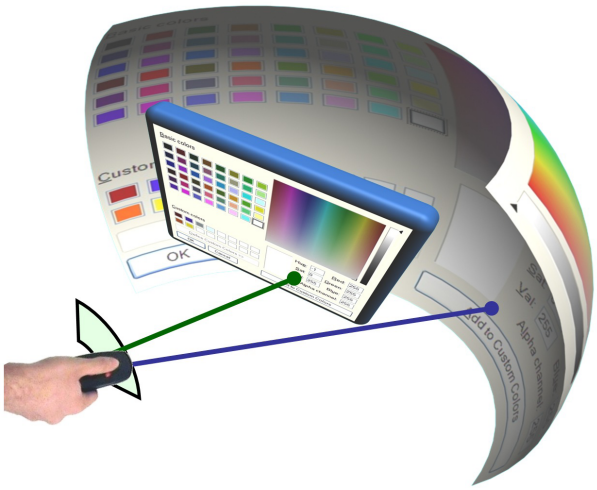



**Figure 6.12:** *Manipulation with friction surfaces simulates approximately the interaction with a large spherical window. Note that rendering this simulated window would be unusable as it would occlude most of the scene.*

## 6.3 Decoupling motor space and visual space

In a typical desktop HCI setup, the motor space is decoupled from the display space (e.g. the movement of a mouse on a horizontal plane is transformed to the movement of a cursor on a vertical screen). We propose a *Virtual Pad* metaphor exploiting how a similar decouple is beneficial for the interaction with 2D virtual windows embedded in a VE. The decouple is accomplished through a virtual pad which receives user actions (motor space) and maps them into cursor movements on the active virtual window (visual space).

By decoupling the motor and the visual space, virtual windows can be manipulated within a user-defined working volume (the virtual pad), whose location and size is completely independent from the application's visual representation. In addition, the user is able to adjust

the control space (virtual pad dimensions) allowing to seamlessly balance speed and accuracy without affecting the visual representation of the application's GUI.

The Virtual Pad metaphor has been conceived to facilitate the interaction with external 2D applications being accessed from immersive VEs that have not been particularly designed for VEs while maximizing user's comfort.

### 6.3.1 The Virtual Pad

The virtual pad (see Figure 6.13) is a rectangular region that defines the control space, i.e., the region the user has to touch or point to when interacting with a virtual window. This tool establishes the link between the working space and the visual space.

In our implementation the virtual pad is rendered as a wireframe rectangle providing the user with a reference frame with minimal visual obtrusion, the user is supposed to be looking at the virtual pad only from time to time or through peripheral vision.

The virtual pad is user-adjustable, it allows the user to change its size and location. Allowing to seamlessly balance speed and accuracy without affecting the visual representation of the application's GUI, through increasing or decreasing its area.

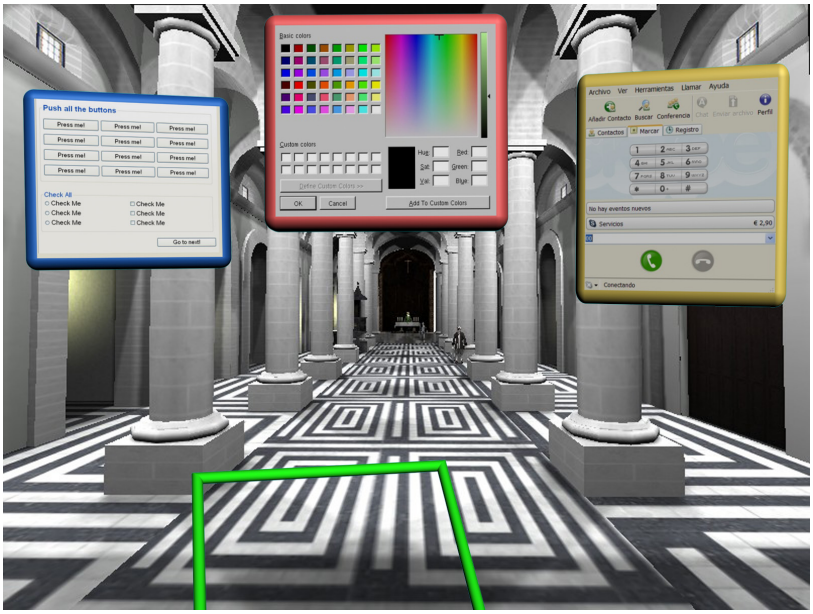



**Figure 6.13:** *Virtual windows on VE. The green rectangle represents the frame of the virtual pad.*

### Motor-to-visual space mapping

When using our virtual pad metaphor in combination with pointing techniques like ray-casting, the mapping from control space to visual space is straightforward:

- Compute the intersection of the ray/selection volume with the virtual pad's plane.
- Compute the new cursor location in the virtual window by transforming linearly the pad's coordinates into window coordinates.

In contrast, when using our virtual pad metaphor in combination with the virtual hand technique, a projection method must be adopted. Due to the absence of constraints on the hand movements, the user hand will move close to but not exactly over the virtual pad. Therefore, the hand position must be projected into the virtual pad before mapping its position to a new cursor position. Several projections can be adopted. Note that the projection type actually defines the shape of the working space. We have considered two different projections:

- *Parallel projection*: the normal vector of the virtual pad is used to project the hand position into the virtual pad. The resulting working volume is a rectangular prism.
- *Viewpoint projection*: the hand position is projected to the virtual pad from the user's viewpoint. Note that this is equivalent to using the ray-casting variant where the ray's direction is defined by the segment joining the user's viewpoint with the hand position. The resulting working volume is a pyramidal frustum.

### Activation/deactivation

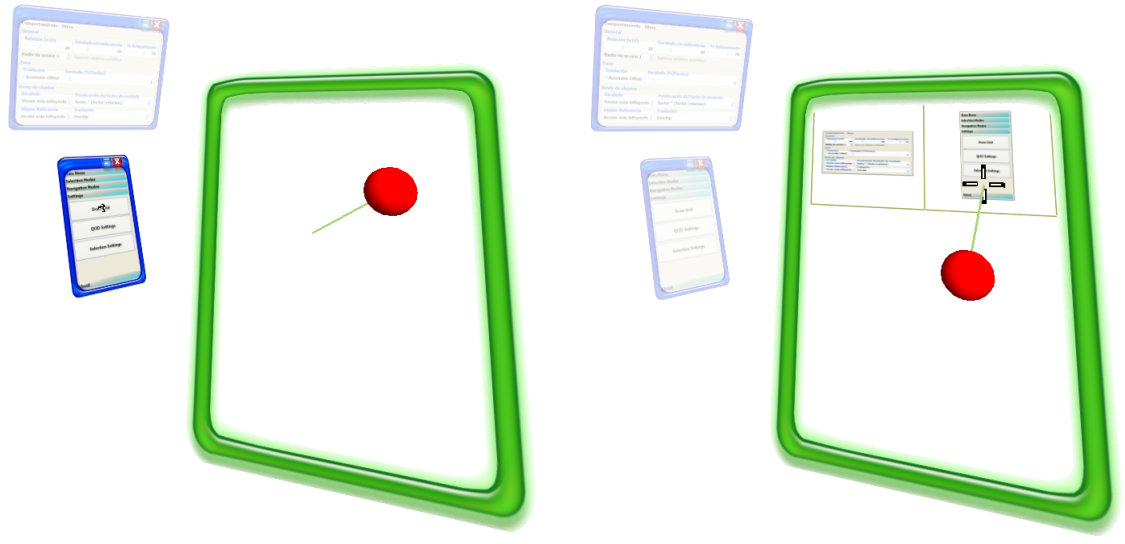
The proposed metaphor has been conceived to manipulate 2D GUIs (or in general, constraint 2D manipulation) inside 3D worlds. Therefore, in a practical situation, this approach has to co-exist with the selection and manipulation techniques used for interacting with the rest of 3D objects. The activation of the virtual pad should therefore be in accordance with these techniques. In our implementation, raycasting is used to both select 3D objects and activate the external application the user wants to interact with.

Activation of a virtual window causes the virtual pad to be adjusted so that its aspect ratio matches the aspect ratio of the virtual window, as described above. Once the window becomes active all the actions over the virtual pad are mapped to it.

Several activation strategies can be adopted. If we want the user not to leave the virtual pad every time he wants to interact with another window, we need to provide mechanisms to choose the active window automatically. For selecting the active window we can use hints given by window events and changes in the window hierarchy. In our implementation a window becomes active automatically when (a) it is shown as the result of a user action (e.g. the user clicks a menu to open a file dialog) or (b) its child window is closed or hidden (e.g. after closing a modal dialog, the focus is assigned to the window where the action was initiated).

This behavior is particularly convenient with pop-up menus and pop-up windows. Manual activation of any window is also provided by simply clicking over the chosen window, although this forces the user to move the ray outside the virtual pad.

As discussed above, when there is an active window all the actions over the virtual pad are mapped to it. When no window is active, the virtual pad has to provide convenient feedback of this fact. We have decided to let the virtual pad to behave as a virtual desktop when no window is active (see Figure 6.14b). In this case the pad shows an icon for every application window (including maximized, minimized and hidden windows). We also use a button trigger to manually deactivate the active window and enable the virtual desktop behavior of the pad. This allows the user to operate different windows with minimum effort and without leaving the virtual pad.



(a) (b)

**Figure 6.14:** *(a) Interaction using the Virtual Pad. (b) If no window is selected, the Virtual Pad works as a window selector.*

### 6.3.2 Virtual Pad evaluation

Since the virtual pad can be freely adjusted to suit user preferences, we can expect a higher degree of comfort on positions reducing fatigue in wrist, arm, shoulder and neck, plus the additional benefit of dynamic adjustment of the speed/accuracy tradeoff.

Therefore we have designed an experiment to measure the price we have to pay in user performance to achieve this higher degree of comfort and flexibility. More precisely, we want to measure the impact in performance of the decoupling between control and visual spaces, choosing ray-casting as the interaction technique and 2D GUI manipulation as the reference task.

When motor and visual spaces are not superimposed, target acquisition requires a different cognitive strategy to use vision to control the cursor rather than a more direct visuomotor mechanism [107]. Although this decoupling is ubiquitous in desktop HCI setups, very few studies have analyzed its impact in user performance.

#### Design and procedure

A repeated-measures, within-subjects design was used. The dependent variable was the interaction technique used, (a) Direct Manipulation of the virtual window (Direct, see Figure 6.15a), (b) interaction through a Virtual Pad (VPad, see Figure 6.15b) with exactly the same size of the virtual window but located in a more comfortable position and (c) interaction through a 50% enlarged pad (VPad+) in a coplanar position with the VPad condition (see Figure 6.15c). The three conditions were randomized for each user. Before each ex-

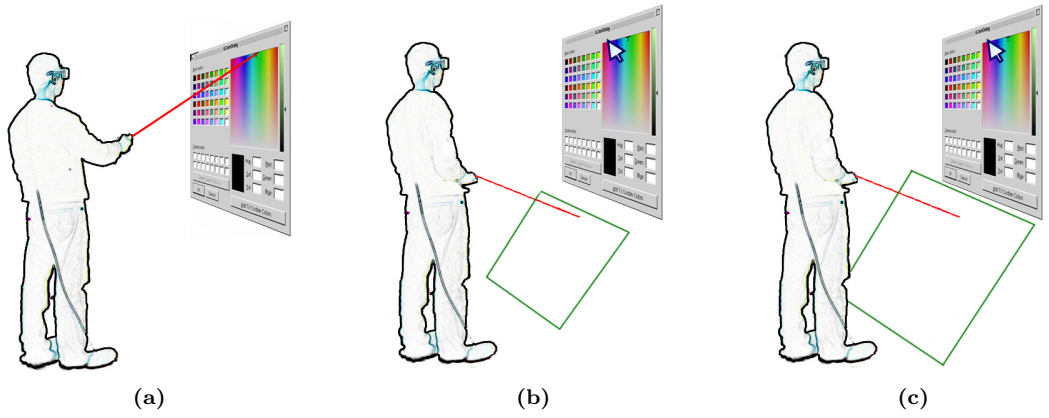
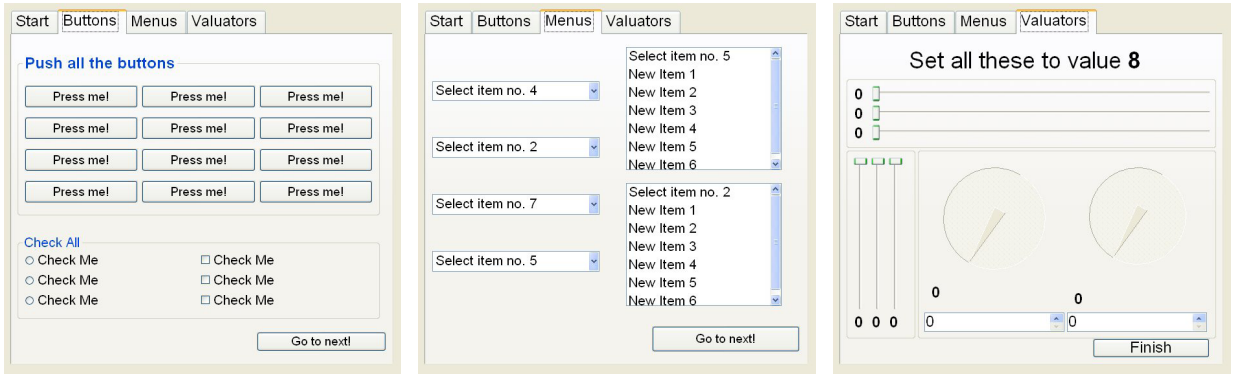



**Figure 6.15:** *Scenarios used in our experiment: (a) direct manipulation of a window through raycasting, (b) manipulation through a virtual pad rotated 45 degrees with respect to the window and (c) manipulation through an 50% scaled virtual pad.*




**(a)** **(b)** **(c)**

**Figure 6.16:** *The test dialogs used for the evaluation of the Virtual Pad metaphor*

periment users were provided with a short training session (5 min) which required them to complete a few practice trials.

As a dependent variables we recorded the time-to-complete the task and the number of clicks. Also, at the end of the experiment, users were provided with a short questionnaire.

The dialogs used in the experiments are shown in Figure 6.16. The first two dialogs are designed to measure task performance on selecting small/middle size objects. The first dialog contains different kinds of buttons whereas the second dialog includes combo boxes and selection lists. The third dialog is designed also to measure speed but putting the emphasis on manipulation rather than on selection. In all cases the label attached to each widget indicates the requested task, so users can be more focused at purely interaction tasks. For all dialogs, users were requested to complete the involved tasks as quickly as possible

### Apparatus

All the experiments were conducted on a four-sided CAVE with a 6-DOF wanda and a Polhemus Fastrak tracking system with 2 receivers providing 60 updates/s with 4 ms latency. The virtual window used in the experiments was initially placed at 1.5 m from the CAVE center, covering about 20 degrees of the user's field-of-view.

### Participants

Thirteen users (undergraduate and graduate students) participated in the study, aged 22-38, 11 male and 2 female. Most participants (6) had some experience with VE applications; 5 had no experience and 2 were experienced users.

### Results

Figure 6.17a shows the completion time for each task. Tasks 1-3 correspond to the dialogs (a)-(c) shown in Figure 6.16. Note that on average the completion times are quite similar on tasks 1 and 2 which emphasize on selection. We performed a correlated samples one-way ANOVA on the data with completion time as the dependent variable and the decoupling (Direct, VPad or VPad+) as the independent variable. We found no significant differences in tasks 1 and 3, but it showed a significant difference for task 2 ($F = 6.29; p < 0.01$). Tukey pair-wise HSD tests revealed a significant difference between Direct vs VPad ($p < 0.05$) and between Direct vs VPad+ ($p < 0.01$).

Regarding button clicks, Figure 6.17b shows the button clicks for each task. Note that in tasks emphasizing on selection (1 and 2), the number of clicks is a good measure of the number of mistakes. This does not applies to task 3, because sliders and spin boxes support different interaction modalities. For example, the value of the spin box can be modified

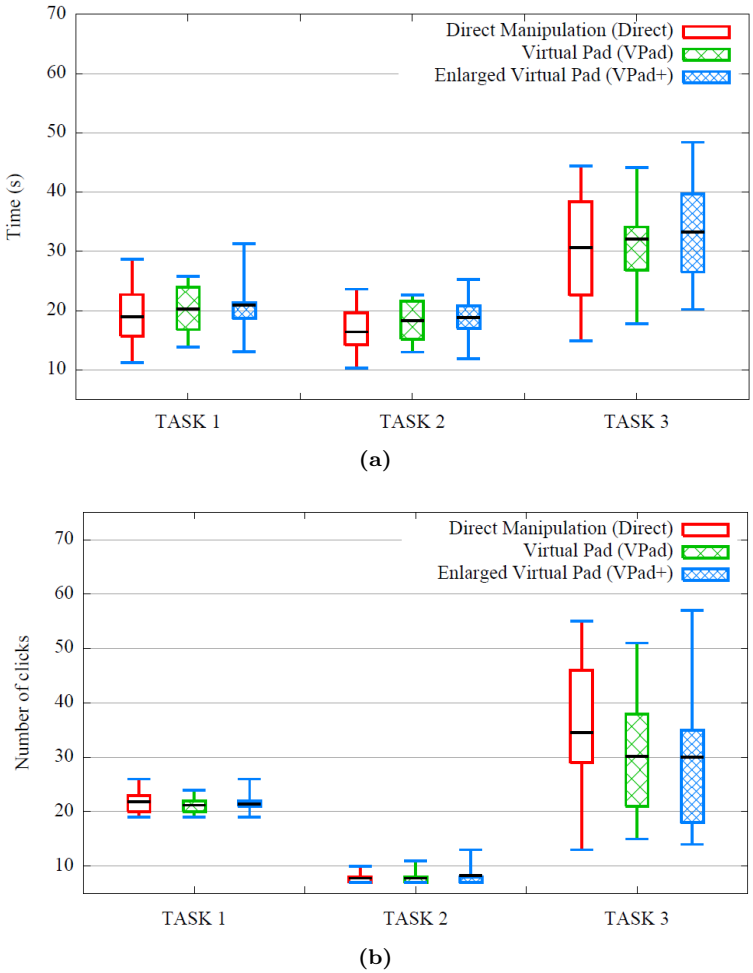



**Figure 6.17:** *(a) Time to complete the tasks involved for each dialog. (b) Number of button clicks performed during each task.*

with repeated clicks or by holding down the wanda button. A correlated samples one-way ANOVA on the data with button clicks as the dependent variable revealed no significant differences.

### Survey

After the experiments, subjects were requested to rate the arm fatigue with VPad and Direct manipulation. As expected, all users reported less muscular fatigue when using VPad. Some users even find it comfortable to slightly rest the wanda against the body. Virtual pads in a completely horizontal orientation are also perfectly possible and greatly reduce arm fatigue. Note that direct manipulation with a virtual window in this position is unfeasible on most projection-based VR systems such as the CAVE using front projection on the floor screen.

### Discussion

The Virtual Pad metaphor has many points in common with pen-and-tablet interfaces [55] and transparent props [104], but also important differences. Besides the fact that it does not require the user to hold any physical prop, the basic difference is that the Virtual Pad decouples the motor and the visual space. It allows to scale virtual windows avoiding visual obtrusion and readability issues while the motor space can be sized according to user preferences and the desired speed/accuracy trade-off.

From the user evaluation, we can state that decoupling the motor space from the visual space does not introduce significant differences in interaction time and erroneous selections. This is also supported by the work of Wang and MacKenzie in [124] which revealed that moderate disparity between motor and visual spaces does not have significant effects on interaction performance.

Interestingly, although we expected differences between VPad the VPad+ conditions, it did not happen. The results somehow match the experiments with 2D mouse based interfaces where constant CD gain conditions do not have significant differences in user performance [4]. Furthermore, note that the corresponding dialog on Figure 6.16b includes several combo boxes. A click on a combo box caused a new pop-up window to appear, which automatically received the focus. In our experiment, focus changes did not affect the pad size. Therefore movements on the pad were mapped to the smaller pop-up window (about 33% of the parent's width), resulting in a much lower CD ratio. We believe that the unexpected CD ratio change (twice per pop-up, eight times per trial) were particularly distracting, since no significant performance differences were found between VPad and VPad+ for Tasks 1 and 3.

In addition, we also expected improved accuracy rates, buy it neither happened. As observed by Wingrave et al. in [132], decreasing the degree of accuracy required may induce users to decrease their accuracy; they no longer need to be precise. This might also happened for the VPad+ condition.

In summary, the Virtual Pad metaphor is recommended when the footprint is a major requirement or when the GUI has not been designed specifically for a VE. By allowing to readjust the control space, user can interact with virtual windows with a more comfortable position. Although statistically significant, the average time for VPad on task 2 was only 11% higher than Direct, so this overhead can be easily accepted in exchange for a more comfortable and flexible interaction.

Moreover, the virtual pad can be used to define the area of the motor space allowed for interaction with virtual windows, thus allowing to seamlessly decouple the interaction between the virtual environment and the virtual windows. The virtual pad can be invisible, but reachable due to proprioception.

# Conclusions

The thesis this PhD dissertation is defending can be summarized as follows:

> *Although 3D interaction techniques for object selection have been used for many years, they still exhibit major limitations regarding effective, accurate selection of targets in real-world VR applications. Some of these limitations are concerned with visual feedback issues (occlusion, visibility mismatch, depth perception in stereoscopic displays) and the inherent features of the human motor system (instability when interacting in free space, speed-accuracy trade-off, neuromotor noise). More efficient 3D interaction techniques can be designed by devising new strategies for controlling the selection tool and for providing appropriate visual feedback, drawing the inspiration from Fitts' law, occlusion management literature and depth perception studies.*

We have provided theoretical and empirical evidence about important limitations of major virtual pointing techniques. Although virtual pointing techniques typically deliver superior performance than their virtual hand counterparts, they require appropriate continuous visual feedback about the scene and the selection tool to complement the information derived from proprioception.

Providing precise visual feedback about the selection tool on stereoscopic displays is a challenging problem. Visual feedback solutions should try to encompass the opposite goals of guaranteeing a clear, unoccluded indication of the selection tool while minimizing the disagreement among parallax, interposition and perspective depth cues. We have shown that drawing a ray extending out from the user's hand, which is the most typical representation of the selection tool, hinders selection tasks as the perception of the tool's tilt angle relies on depth perception.

Occlusion is a major source of difficulties when indicating objects in complex 3D scenes. We have shown the negative effects of occlusion and visibility mismatch (both within users and among users) in selection and referral tasks.

Eye-hand visibility mismatch has been proven to be a factor leading to a significant loss of performance in all hand-rooted pointing techniques. Our analysis suggests that for these techniques the effective width of an object should be computed in terms of the part of the object simultatenously unoccluded from the user's hand and eye locations. This results in a paradoxical situation where some objects might exhibit a null effective width despite being visible on the screen.

In order to overcome these visual-feedback limitations, we have proposed a new strategy for controlling the selection tool dubbed Ray Casting from the Eye (RCE). RCE encompasses the efficient rotational control of hand-rooted techniques (thus minimizing the physical effort) with the mismatch-free feature of eye-rooted techniques. RCE clearly outperforms classic ray casting, particularly in real-world, complex 3D scenes. RCE is especially effective when used in combination with the newly introduced viewfinder metaphor. By flattening the surroundings of the pointing direction, the viewfinder locally turns a 3D object selection task into a simpler 2D selection task where depth perception is no longer an issue.

RCE control can be freely combined with further facilitation techniques for object selection such as speed-based adaptation of the CD-ratio. RCE combined with PRISM-based CD-ratio adjustment offers excellent performance even in cluttered scenes with objects spanning a large range of depth values.

The analysis of major limitations of existing selection techniques and the proposed solutions were published in:

- Ferran Argelaguet and Carlos Andujar. **Efficient 3d pointing selection in cluttered virtual environments**. IEEE Computer Graphics and Applications, 29(6):34-43, 2009.
- Ferran Argelaguet, Carlos Andujar and Ramon Trueba. **Overcoming eye-hand visibility mismatch in 3D pointing selection**. In Proceedings of the 15th ACM Symposium on Virtual Reality Software and Technology, VRST '08, pages 43-46, 2008.
- Ferran Argelaguet and Carlos Andujar. **Visual feedback techniques for virtual pointing on stereoscopic displays**. In Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology, VRST '09, pages 163-170, 2009.

Visibility mismatch among users in collaborative virtual environments is a common hindrance during referral tasks, where one user is willing to show some object or feature to another user. Typically, referral tasks are accomplished using selection techniques, with one user (the presenter) selecting an object and asking the other users (observers) to look at it. Viewpoint

mismatch often results in the selected object to be visible only for the presenter. According to proxemics theory, users carrying out a collaborative tasks should stay at the close phase of the social distance (about 1.2 to 2.1 m in Western culture). However, our experiment performed in a co-located CVE showed that the viewpoint mismatch forces observers to stay close to the presenter. Such a close distance between users does not agree with social protocols and can cause users to bump accidentally into other users. Furthermore, this discourages other users to freely navigate through the virtual environment.

In order to better support referral tasks, we explored which occlusion management techniques ameliorate this situation and whether they hinder or not the spatial understanding of the virtual environment. X-Ray vision techniques have been found to provide the best solution for inter-personal occlusion. We have shown that semi-transparency and cut-away views are suitable for guaranteeing the visibility of the objects being referred. The empirical evaluation of these techniques showed that observers are able to gather the same amount of spatial understanding with these techniques than when following the presenter, with additional advantages related to freedom of movement and social protocols.

The evaluation of show-through techniques was published in:

- Ferran Argelaguet, André Kunert, Alexander Kulik, Carlos Andujar, and Bernd Froehlich. **See-through techniques for referential awareness in collaborative virtual reality**. International Journal of Human-Computer Studies. Volume 69, Issue 6, June 2011, Pages 387-400.
- Ferran Argelaguet, André Kunert, Alexander Kulik and Bernd Froehlich. **Improving co-located collaboration with show-through techniques**. In IEEE Symposium on 3D User Interfaces 2010, pages 55-92, 2010.

The review of Fitts' law and existing pointing facilitation techniques for 3D object selection, revealed that although expanding targets techniques have been used to improve 2D target acquisition tasks, they were never applied for 3D object selection tasks.

Following Fitts' law guidelines, we proposed two orthogonal techniques for improving 3D target acquisition tasks: Dynamic Scaling and Forced Disoclussion. Both techniques increase the effective size of potential targets, thus reducing the accuracy needed during corrective movements. In an ideal scenario, increasing the size of potential targets should result in improvements on selection performance and reduced error rates. However, the user study showed that their drawbacks exceeded their benefits. As with most of Fitts' law based techniques which increase the size of potential targets, time and error rate improvements

are more apparent in situations where targets are isolated. In cluttered environments, the benefits of these techniques tend to degrade, and can even be detrimental to selection time. First, unwanted expansions of objects surrounding the target might discourage its selection and second, they might break the object layout.

Both approaches presented additional limitations related to the eye-hand visibility mismatch, as they were evaluated in combination with classic ray casting selection. While Dynamic Scaling computes the potential occlusions among objects in image space, Forced Disocclusion required an alternative ray-scene intersection. Nevertheless, these limitations are avoided when used in combination with Ray Casting from the Eye.

The Dynamic Scaling and the Forced Disocclussion approaches were presented in:

- Ferran Argelaguet and Carlos Andujar. **Improving 3D selection in immersive environments through expanding targets**. SG '08 Proceedings of the 9th international symposium on Smart Graphics, 5166/2008, pages 45-57, 2008.

The operation of 2D widgets using 3D input devices relies also on selection techniques. The last contribution of this dissertation (although the first contribution chronologically) is concerned about improving the deployment and the interaction of 2D graphical user interfaces embedded in virtual environments. First, we proposed a cost-effective approach to embed 2D GUIs into virtual environments. Rather than providing new tools, we focused on how to adapt well established 2D GUI toolkits into VEs. This greatly simplifies the GUI design and prototyping steps as developers and designers benefit from the widget functionality and the design tools provided by existing 2D toolkits.

When 2D GUI toolkits designed for desktop computers are deployed to virtual environments, they might include small, nearby buttons which can be difficult to select and manipulate using standard 3D interaction techniques. Although 2D toolkits support widget customization, complex 2D GUIs will require a complete redesign increasing the development step.

Instead of developing methods to automatically redesigning existing 2D GUIs, we developed new strategies for controlling the selection tool for the particular case of selecting 2D widgets embedded in 3D space. We explored whether decreasing the CD ratio according to the size of virtual windows (Friction Surfaces) and decoupling the motor and the visual space (Virtual Pad) allow for increased precision and increased comfort when interacting with 2D GUIs embedded in VEs.

The evaluation of Friction Surfaces showed that selection of small widgets, in comparison with standard raycasting selection, is enhanced both in terms of user performance and error

rate. The reduction of the CD ratio increased the accuracy of selection and manipulation tasks without decreasing selection time.

Regarding the Virtual Pad, its evaluation showed that the decoupling between the motor and the visual space results in no significant performance penalty with respect to direct interaction. When the reduction of the visual footprint of the GUI is a strong requirement, the virtual pad allows the user to adjust its working space to maximize comfort without decreasing performance.

Contributions regarding 2D GUI interaction were published in:

- Carlos Andujar and Ferran Argelaguet. **Anisomorphic ray-casting manipulation for interacting with 2D GUIs**. Computer & Graphics, 31(1):15-25, 2007.
- Carlos Andujar, Marta Fairen and Ferran Argelaguet. **A cost-effective approach for developing application-control guis for virtual environments**. In IEEE Symposium on 3D User Interfaces (3DUI 2006), pages 45-52, 2006.
- Carlos Andujar and Ferran Argelaguet. **Virtual pads: Decoupling motor space and visual space for flexible manipulation of 2D windows within VEs**. In IEEE Symposium on 3D User Interfaces (3DUI 2007), pages 99-106. 2007.
- Carlos Andujar and Ferran Argelaguet. **Friction surfaces: Scaled ray-casting manipulation for interacting with 2D GUIs**. 12th Eurographics Symposium on Virtual Environments, pages 101-108, 2006.

As a concluding remark, we would like to note that the interaction techniques proposed in this dissertation do not consider any specific input and output device, although they are obviously constrained by the limitations of current input and output devices. Improvements in tracking technology will allow for a more accurate tracking of the user's actions, and better displays will enhance the user's perception of the virtual environment. We believe though that the major contributions of this dissertation will still be valid despite forthcoming advances in VR technology. We can provide the user with extremely realistic volumetric displays and perfectly accurate tracking systems, but pointing gestures will be still limited by the human motor system, which is unlikely to improve in the near future.

Although new and better interaction techniques will arise, or in a mid-term future, brain-computer interfaces might partially replace traditional gesture-based interfaces, we believe that the presented techniques are a good choice for both current and upcoming VR applications.

# Bibliography


[1] Johnny Accot and Shumin Zhai. Refining Fitts' law models for bivariate pointing. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 193–200. ACM, 2003.

[2] Maneesh Agrawala, Andrew C. Beers, Ian McDowall, Bernd Fröhlich, Mark Bolas, and Pat Hanrahan. The two-user responsive workbench: support for collaboration through individual views of a shared space. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 327–332. ACM Press/Addison-Wesley Publishing Co., 1997.

[3] American Psychological Association. Certification of compliance with ethical principles. http://www.apa.org/ethics/code/index.aspx.

[4] Ravin Balakrishnan. "Beating" Fitts' Law: Virtual enhancements for pointing facilitation. *International Journal of Human-Computer Studies*, pages 857–874, 2004.

[5] Jim Blascovich, Andrew C. Beall, Jack M. Loomis, Jeremy N. Bailenson, Jeremy N. Bailenson, and In Neuromancer. Interpersonal distance in immersive virtual environments. *Personality and Social Psychology Bulletin*, 29:1–15, 2003.

[6] Joan De Boeck, Tom De Weyer, Chris Raymaekers, and Karin Coninx. Using the Non-Dominant Hand for Selection in 3D. *3DUI '06: Proceedings of the 3D User Interfaces*, pages 53–58, 2006.

[7] Doug A. Bowman, Brian Badillo, and Dhruv Manek. Evaluating the need for display-specific and device-specific 3D interaction techniques. In *ICVR'07: Proceedings of the 2nd international conference on Virtual reality*, pages 195–204, Berlin, Heidelberg, 2007. Springer-Verlag.

[8] Doug A. Bowman, Joseph L. Gabbard, and Deborah Hix. A survey of usability evaluation in virtual environments: classification and comparison of methods. *Presence: Teleoperators and Virtual Environments*, 11(4):404–424, 2002.

[9] Doug A. Bowman and Larry F. Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. *SI3D '97: Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 35–ff, 1997.

[10] Doug A. Bowman, Larry F. Hodges, and Jay Bolter. The virtual venue: User-computer interaction in information-rich virtual environments. *Presence: Teleoperators and Virtual Environments*, 7(5):478–493, 1998.

[11] Doug A. Bowman, Donald B. Johnson, and Larry F. Hodges. Testbed evaluation of virtual environment interaction techniques. In *VRST '99: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 26–33. ACM, 1999.

[12] Doug A. Bowman, Ernest Kruijff, Joseph J. LaViola, and Ivan Poupyrev. *3D User Interfaces: Theory and Practice.* Addison Wesley, 2004.

[13] Doug A. Bowman, Chadwick A. Wingrave, and J Campbell. Using pinch gloves for both natural and abstract interaction techniques in virtual environments. *HCI International*, pages 629–633, 2001.

[14] Stephen Brewster. Multimodal feedback for the acquisition of small targets. *Ergonomics*, 48:1129–1150(22), 2005.

[15] Matthias Bues, Roland Blach, and Frank Haselberger. Sensing Surfaces: bringing the desktop into virtual environments. In *EGVE '03: Proceedings of the workshop on Virtual Environments 2003*, pages 303–304. ACM, 2003.

[16] Michael Burns and Adam Finkelstein. Adaptive cutaways for comprehensible rendering of polygonal scenes. *ACM Transactions on Graphics*, 27(5):1–7, 2008.

[17] S. K. Card, W. K. English, and B.J. Burr. Evaluation of mouse, rate-controlled isometric joystick, step keys, and text keys for text selection on a crt. *Ergonomics*, 2:601–613, 1978.

[18] Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. A morphological analysis of the design space of input devices. *ACM Transactions on Information Systems*, 9(2):99–122, 1991.

[19] Géry Casiez, Daniel Vogel, Ravin Balakrishnan, and Andy Cockburn. The impact of control-display gain on user performance in pointing tasks. In *Human-Computer Interaction*, volume 23, pages 215–250. Taylor & Francis, 2009.

[20] Pedro Concejero Cerezo. Código ético de la investigación en usabilidad e Interacción Persona-Ordenador. Pruebas con usuarios. Asociación Interacción Persona-Ordenador, 2006.

[21] Jeff Chastine and Ying Zhu. The cost of supporting references in collaborative augmented reality. In *Proceedings of graphics interface 2008*, GI '08, pages 275–282. Canadian Information Processing Society, 2008.

[22] Jeffrey W. Chastine, Kristine Nagel, Ying Zhu, and Luca Yearsovich. Understanding the design space of referencing in collaborative augmented reality environments. In *Proceedings of Graphics Interface 2007*, GI '07, pages 207–214. ACM, 2007.

[23] Andy Cockburn and Philip Brock. Human on-line response to visual and motor target expansion. *GI '06: Proceedings of Graphics Interface 2006*, pages 81–87, 2006.

[24] Chris Coffin and Tobias Hollerer. Interactive perspective cut-away views for general 3d scenes. In *Proceedings of the IEEE 3D User Interfaces*, 3DUI '06, pages 25–28, 2006.

[25] Infiscape Corporation. VRJuggler - http://www.vrjuggler.org.

[26] Nokia Corporation. Qt - cross platform application and UI framework. http://qt.nokia.com.

[27] E. Crossman. The measurement of perceptual load in manual operations. Master's thesis, PhD thesis, University of Birmingham, 1956.

[28] P. Crossman, E. Goodeve. Feedback control of hand-movement and Fitts' law. *The Quarterly Journal of Experimental Psychology Section A: Human Experimental Psychology*, 35:251–278, 1983.

[29] Raimund Dachselt and Anett Hübner. A survey and taxonomy of 3D menu techniques. *EGVE 06: Proceedings of the 12th Eurographics Symposium*, pages 89–99, 2006.

[30] Raimund Dachselt and Anett Hübner. Three-dimensional menus: A survey and taxonomy. *Computers & Graphics*, 31(1):53–65, 2007.

[31] Philippe Dax. VREng - virtual reality engine. http://sourceforge.net/projects/vreng/.

[32] Gerwin de Haan, Michal Koutek, and Frits H. Post. IntenSelect: Using dynamic object rating for assisting 3D object selection. *Eurographics Symposium on Virtual Environments*, pages 201–209, 2005.

[33] J. Diepstraten, D. Weiskopf, and T. Ertl. Transparency in interactive technical illustrations. *Eurographics*, 21(3):317–325, 2002.

[34] Stephen DiVerdi, Daniel Nurmi, and Tobias Höllerer. ARWin-a desktop Augmented Reality Window manager. In *ISMAR '03: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, page 298, 2003.

[35] Phillip Dykstra. X11 in virtual environments: combining computer interaction methodologies. *X Resour.*, (9):195–204, 1994.

[36] N. Elmqvist and P. Tsigas. A taxonomy of 3D occlusion management for visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(5):1095 –1109, 2008.

[37] Niklas Elmqvist, Ulf Assarsson, and Philippas Tsigas. Employing dynamic transparency for 3D occlusion management: Design issues and evaluation. *Human-Computer Interaction INTERACT 2007*, pages 532–545, 2008.

[38] P. Fitts. The information capacity of the human motor system is controlled by the amplitude of movement. *Journal of Experimental Psychology*, 6(47):381–391, 1954.

[39] P. Fitts and J Peterson. Information capacity of discrete motor response. *Journal of Experimental Psychology*, 2(67):103–112, 1964.

[40] Andrew Forsberg, Kenneth Herndon, and Robert Zeleznik. Aperture based selection for immersive virtual environments. *UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 95–96, 1996.

[41] Scott Frees and Drew Kessler. Precise and Rapid Interaction throught Scaled Manipulation in immersive virtual environments. *Proc. of IEEE Virtual Reality 2005*, pages 99–106, 2005.

[42] Scott Frees, G. Drew Kessler, and Edwin Kay. PRISM interaction for enhancing control in immersive virtual environments. *ACM Transactions on Computer-Human Interaction*, 14(1):2, 2007.

[43] Bernd Fröhlich, Roland Blach, Oliver Stefani, Jan Hochstrate, Jörg Hoffmann, Karsten Klüger, and Matthias Bues. Implementing multi-viewer stereo displays. In *WSCG (Full Papers)*, pages 139–146, 2005.

[44] Joseph L. Gabbard. A taxonomy of usability characteristics for virtual environmnets. Master's thesis, Departament of Computer Science, Virginia Tech, 1997.

[45] Joseph L. Gabbard, Deborah Hix, and J. Edward Swan. User-centered design and evaluation of virtual environments. *IEEE Computer Graphics and Applications*, 19(6):51–59, 1999.

[46] T Germer, T Götzelmann, M Spindler, and Thomas Strothotte. Springlens : Distributed nonlinear magnifications. *Eurographics'06 Short Papers*, pages 123–126, 2006.

[47] Tovi Grossman and Ravin Balakrishnan. Pointing at trivariate targets in 3D environments. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 447–454. ACM, 2004.

[48] Tovi Grossman and Ravin Balakrishnan. The design and evaluation of selection techniques for 3D volumetric displays. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 3–12. ACM, 2006.

[49] Edward T. Hall. *The Hidden Dimension.* Doubleday, 1966.

[50] Patrick Hartling and Carolina Cruz-Neira. Tweek: A framework for cross-display graphical user interfaces. In *Computational Science and Its Applications - ICCSA 2005*, volume 3482 of *Lecture Notes in Computer Science*, pages 317–329. Springer Berlin / Heidelberg, 2005.

[51] Kenneth P. Herndon, Andries van Dam, and Michael Gleicher. The challenges of 3D interaction: a CHI '94 workshop. *SIGCHI Bull.*, 26(4):36–43, 1994.

[52] A. Hill and A. Johnson. Withindows: A framework for transitional desktop and immersive user interfaces. In *IEEE Symposium on 3D User Interfaces 2008*, pages 3–10, March 2008.

[53] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. Passive real-world interface props for neurosurgical visualization. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 452–458. ACM, 1994.

[54] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. A survey of design issues in spatial input. In *UIST '94: Proceedings of the 7th annual ACM symposium on User interface software and technology*, pages 213–222. ACM, 1994.

[55] Hiroshi Ishii and Brygg Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '97, pages 234–241. ACM, 1997.

[56] Liang J. and Green M. JDcad: A highly interactive 3D modeling system. *Computers & Graphics 18*, 4:499–506, 1994.

[57] Robert J.K. Jacob, Audrey Girouard, Leanne M. Hirshfield, Michael S. Horn, Orit Shaer, Erin Treacy Solovey, and Jamie Zigelbaum. Reality-based interaction: a framework for post-WIMP interfaces. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 201–210. ACM, 2008.

[58] R. J. Jagacinski and D. L. Monk. Fitts' law in two dimensions with hand and head movements. *Journal of Motor Behavior*, 17:77–95, 1985.

[59] Joseph J. LaViola Jr. A survey of hand posture and gesture recognition techniques and technology. Technical Report CS-99-11, Brown University. Computer Science Department,, 1999.

[60] Y. Jung, S. Webel, M. Olbrich, T. Drevensek, T. Franke, M. Roth, and D. Fellner. Interactive textures as spatial user interfaces in x3d. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 147–150. ACM, 2010.

[61] John Kelso, Steven G. Satterfield, Lance E. Arsenault, Peter M. Ketchan, and Ronald D. Kriz. DIVERSE: a framework for building extensible and reconfigurable device-independent virtual environments and distributed asynchronous simulations. *Presence: Teleoperators and Virtual Environments*, 12(1):19–36, 2003.

[62] Stuart T. Klapp. Feedback versus motor programming in the control of aimed movements. *Journal of Experimental Psychology: Human Perception and Performance*, 1:147–153, 1975.

[63] Werner König, Jens Gerken, Stefan Dierdorf, and Harald Reiterer. Adaptive pointing - design and evaluation of a precision enhancing technique for absolute pointing devices. In *Human-Computer Interaction, INTERACT 2009*, volume 5726 of *Lecture Notes in Computer Science*, pages 658–671. Springer Berlin / Heidelberg, 2009.

[64] Robert E. Kraut, Mark D. Miller, and Jane Siegel. Collaboration in performance of physical tasks: effects on outcomes and communication. In *Proceedings of the 1996 ACM conference on Computer Supported Cooperative Work*, CSCW '96, pages 57–66, 1996.

[65] Per Ola Kristensson and Alan Blackwell. Ethics in large-scale user studies: Guidelines vs. practice. In *CHI 2011 Workshop on Ethics in Large Scale Trials & User Generated Content*, 2011.

[66] Daniel Larimer and Doug A. Bowman. VEWL: A framework for building a windowing interface in a virtual environment. *in Proceedings of IFIP TC13 International Conference on Human-Computer Interaction, Interact'2003*, pages 1–5, 2003.

[67] Joseph J. LaViola, Jr. A discussion of cybersickness in virtual environments. *SIGCHI Bulletin*, 32:47–56, January 2000.

[68] Geoff Leach, Ghassan al Qaimari, Mark Grieve, Noel Jinks, and Cameron McKay. Elements of a three-dimensional graphical user interface. *INTERACT '97: Proceedings of the IFIP TC13 Interantional Conference on Human-Computer Interaction*, pages 69–76, 1997.

[69] SangYoon Lee, Jinseok Seo, Gerard J. Kim, and Chan-Mo Park. Evaluation of pointing techniques for ray casting selection in virtual environments. *Third International Conference on Virtual Reality and Its Application in Industry*, 4756(1):38–44, 2003.

[70] Q. Limbourg, J. Vanderdonckt, L. Bouillon B. Michotte, and V. Lopez. UsiXML: A language supporting multi-path development of user interfaces. *In 9th IFIP Working Conference on Engineering for Human-Computer Interaction jointly with 11th Int. Workshop on Design, Specification, and Verification of Interactive Systems EHCIDSVIS*, 2004.

[71] R.W. Lindeman, J.L. Sibert, and J.K. Hahn. Hand-held windows: towards effective 2D interaction in immersive virtual environments. In *Virtual Reality, 1999. Proceedings., IEEE*, pages 205 –212, 1999.

[72] R. Linderman, J. Sibert, and J. Hahn. Towards usable VR: an empirical study of user interfaces for immersive virtual environments. *Proceedings of the SIHCHI conference on Human factors in computing systems.*, pages 64–71, 1999.

[73] C. L. MacKenzie, R. G. Marteniuk, C. Dugas, D. Liske, and B. Eickmeier. Three-dimensional movement trajectories in Fitts' task: Implications for control. *The Quarterly Journal of Experimental Psychology Section A: Human Experimental Psychology*, 39(4):629–647, 1987.

[74] I. Scott MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7:91–139, 1992.

[75] I. Scott MacKenzie and William Buxton. Extending Fitts' law to two-dimensional tasks. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 219–226. ACM, 1992.

[76] I. Scott MacKenzie, Tatu Kauppinen, and Miika Silfverberg. Accuracy measures for evaluating computer pointing devices. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '01, pages 9–16. ACM, 2001.

[77] I. Scott MacKenzie and Colin Ware. Lag as a determinant of human performance in interactive systems. In *CHI '93: Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, pages 488–493. ACM, 1993.

[78] José Pascual Molina Massó, Jean Vanderdonckt, Francisco Montero Simarro, and Pascual González López. Towards virtualization of user interfaces based on UsiXML. *Web3D '05: Proceedings of the tenth international conference on 3D Web technology*, pages 169–178, 2005.

[79] Michael McGuffin and Ravin Balakrishnan. Acquisition of expanding targets. *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 57–64, 2002.

[80] Michael F. McTear. Spoken dialogue technology: enabling the conversational user interface. *ACM Comput. Surv.*, 34(1):90–169, 2002.

[81] David E. Meyer, Richard A. Abrams, Sylvan Kornblum, Charles E. Wright, and J. E. Keith Smith. Optimality in human motor performance: ideal control of rapid aimed movements. *Psychological Review*, 95:340–370, 1988.

[82] Mark R. Mine. Virtual environments interaction techniques. Technical Report TR94-018, Dept. of Computer Science, University of North Carolina at Chapel Hill, 1995.

[83] Mark R. Mine, Frederick P. Brooks, Jr., and Carlo Sequin. Moving objects in space: Exploiting proprioception in virtual-environment interaction. *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 19–26, 1997.

[84] Atsuo Murata. Extending effective target width in Fitt's law to two-dimensional pointing task. In *International Journal of Human-Computer Interaction*, volume 11(2), pages 137–152. L. Erlbaum Associates Inc., 1999.

[85] Atsuo Murata and Hirokazu Iwase. Extending Fitts' law to a three-dimensional pointing task. *Human Movement Science*, 20(6):791 – 805, 2001.

[86] K. Nieuwenhuizen, J.-B. Martens, Lei Liu, and R. van Liere. Insights from dividing 3D goal-directed movements into meaningful phases. *Computer Graphics and Applications, IEEE*, 29(6):44 –53, nov. 2009.

[87] Johanna Octavia, Chris Raymaekers, and Karin Coninx. Adaptation in virtual environments: conceptual framework and user models. *Multimedia Tools and Applications*, pages 1–22, 2010.

[88] Association of Computing Machinery. Code of ethics. http://www.acm.org/about/code-of-ethics.

[89] Alex Olwal, Hrvoje Benko, and Steven Feiner. SenseShapes: Using statistical geometry for object selection in a multimodal augmented reality system. In *ISMAR '03: Proceedings of the 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality*, page 300, 2003.

[90] Alex Olwal and Steven Feiner. The flexible pointer: An interaction technique for selection in augmented and virtual reality. *In Conference Supplement of UIST '03 (ACM Symposium on User Interface Software and Technology)*, pages 81–82, 2003.

[91] Noritaka Osawa, Kikuo Asai, and Fumihiko Saito. An interactive toolkit library for 3D applications: it3d. *EGVE '02: Proceedings of the workshop on Virtual environments 2002*, pages 149–157, 2002.

[92] Andriy Pavlovych and Wolfgang Stuerzlinger. The tradeoff between spatial jitter and latency in pointing tasks. In *EICS '09: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, pages 187–196. ACM, 2009.

[93] Hans Peter, Wyss Roland, and Blach Matthias Bues. ISith – intersection-based spatial interaction for two hands. *IEEE Symposium on 3D User Interfaces 2006*, pages 59–61, 2006.

[94] Jeffrey S. Pierce, Andrew Forsberg, Matthew J. Conway, Seung Hong, Robert Zeleznik, and Mark R. Mine. Image plane interaction techniques in 3D immersive environments. *Symposium on Interactive 3D Graphics*, page 5, 1997.

[95] I. Poupyrev and T. Ichikawa. Manipulating objects in virtual worlds: Categorization and empirical evaluation of interaction. *Journal of Visual Languages & Computing*, 10:19–35, 1999.

[96] Ivan Poupyrev, Mark Billinghutst, Suzanne Weghorst, and Tadao Ichikawa. The go-go interaction technique: Non-linear mapping for direct manipulation in VR. *ACM Symposium on User Interface Software and Technology*, pages 79–80, 1996.

[97] Ivan Poupyrev, Suzanne Weghorst, Mark Billinghurst, and Tadao Ichikawa. A framework and testbed for studying manipulation techniques for immersive vr. In *VRST '97: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 21–28. ACM, 1997.

[98] Ivan Poupyrev, Suzanne Weghorst, Mark Billinghutst, and Tadao Ichikawa. Egocentric object manipulation in virtual environments: Empirical evaluation of interaction techniques. *Computer Graphics Forum*, 3(17):41–52, 1998.

[99] Ivan Poupyrev, Suzanne Weghorst, and Sidney Fels. Non-isomorphic 3d rotational techniques. *CHI '00: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 540–547, 2000.

[100] Schmidt R.A., H.N. Zalaznik, B. Hawkins, J.S. Frank, and J.T. Quinn. Motor-output variability: A theory for the accuracy of rapid motor acts. *Psychological Review*, 86(5):415 – 451, 1979.

[101] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, and Andy Hopper. Virtual Network Computing. *IEEE Internet Computing*, 2(1):33–38, 1998.

[102] G. Robertson, M. van Dantzich, D. Robbins, M. Czerwinski, K. Hinckley, K. Risden, D. Thiel, and V. Gorokhovsky. The task gallery: a 3D window manager. *Human Factors in Computing Systems*, pages 494–501, 2000.

[103] Holger Salzmann, Mathias Moehring, and Bernd Froehlich. Virtual vs. real-world pointing in two-user scenarios. *IEEE Virtual Reality Conference*, pages 127–130, 2009.

[104] Dieter Schmalstieg, L. Miguel Encarnação, and Zsolt Szalavári. Using transparent props for interaction with the virtual table. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, I3D '99, pages 147–153. ACM, 1999.

[105] Greg Schmidt, Yohan Baillot, Dennis G. Brown, Erik B. Tomlin, and J. Edward II Swan. Toward disambiguating multiple selections for frustum-based pointing. *3D User Interfaces*, pages 87–94, 2006.

[106] John B. Smelcer. Spatial and temporal characteristics of rapid cursor-positioning movements with electromechanical mice in human-computer interaction. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 35:431–458(28), 1993.

[107] Barton A. Smith, Janet Ho, Wendy Ark, and Shumin Zhai. Hand eye coordination patterns in target selection. In *Proceedings of the 2000 symposium on Eye tracking research & applications*, ETRA '00, pages 117–122. ACM, 2000.

[108] J. Marques Soares, P. Horain, and A. Bideau. Sharing and immersing applications in a 3D virtual inhabited world. In *5th virtual reality international conference (VRIC 2003)*, pages 27–31, 2003.

[109] Kay M. Stanney. *Handbook of virtual environments: design, implementation, and applications*. Lawrence Erlbaum Associates, 2002.

[110] Anthony Steed. Selection/towards a general model for selection in virtual environments. *IEEE Symposium on 3D User Interfaces 2006*, pages 103–110, 2006.

[111] Anthony Steed and C Parker. 3D selection strategies for head tracked and non-head tracked operation of spatially immersive displays. *8th International Immersive Projection Technology Workshop*, pages 163–170, 2004.

[112] Frank Steinicke, Timo Ropinski, and Klaus Hinrichs. Object selection in virtual environments with an improved virtual pointer metaphor. *Computer Vision and Graphics, International Conference, ICCVG 2004*, 32:320–326, 2004.

[113] Frank Stenicke, Timo Ropinski, Gerd Bruder, and Klaus Hinrichs. Interscopic user interface concepts for fish tank virtual reality systems. *Virtual Reality Conference, IEEE*, pages 27–34, 2007.

[114] Stanislav L. Stoev and Dieter Schmalstieg. Application and taxonomy of through-the-lens techniques. *VRST '02: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 57–64, 2002.

[115] Vildan Tanriverdi and Robert J. K. Jacob. Interacting with eye movements in virtual environments. In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 265–272. ACM, 2000.

[116] R.J. Teather, A. Pavlovych, W. Stuerzlinger, and I.S. MacKenzie. Effects of tracking technology, latency, and spatial jitter on object movement. In *IEEE Symposium on 3D User Interfaces 2009*, pages 43–50, 2009.

[117] Alexandre Topol. Immersion of Xwindow applications into a 3D workbench. In *CHI '00: extended abstracts on Human factors in computing systems*, pages 355–356. ACM, 2000.

[118] F. Tyndiuk, G. Thomas, V. Lespinet-Najib, and C. Schlick. Cognitive comparison of 3D interaction in front of large vs. small displays. *VRST '05: Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pages 117–123, 2005.

[119] Gerard P. van Galen and Willem P. de Jong. Fitts' law as the outcome of a dynamic noise filtering model of motor control. *Human Movement Science*, 14(4-5):539 – 571, 1995.

[120] Lode Vanacken, Tovi Grossman, and Karin Coninx. Exploring the effects of environment density and target visibility on object selection in 3D virtual environments. *IEEE Symposium on 3D User Interfaces, 3DUI '07.*, pages 115–122, 2007.

[121] Lode Vanacken, Chris Raymaekers, and Karin Coninx. Evaluating the influence of multimodal feedback on egocentric selection metaphors in virtual environments. *Haptic and Audio Interaction Design*, pages 12–23, 2006.

[122] Daniel Vogel and Ravin Balakrishnan. Distant freehand pointing and clicking on very large, high resolution displays. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 33–42. ACM Press, 2005.

[123] Vrco. Cavelib - http://www.vrco.com.

[124] Y. Wang and C. MacKenzie. Effects of orientation disparity between haptic and graphic displays of objects in virtual environments. In *INTERACT '99*, pages 391–398, 1999.

[125] Colin Ware and Ravin Balakrishnan. Reaching for objects in VR displays: lag and frame rate. *ACM Transactions on Computer-Human Interaction*, 1(4):331–356, 1994.

[126] Colin Ware and Kathy Lowther. Selection using a one-eyed cursor in a fish tank VR environment. *ACM Trans. Comput.-Hum. Interact.*, 4(4):309–322, 1997.

[127] Kent Watsen and Rudolph P. Darken. A handheld computer as an interaction device to a virtual environment. *3rd International Immersive Projection Technology Workshop (IPTW'99)*, 1999.

[128] B. Watson, V. Spaulding, N. Walker, and W. Ribarsky. Evaluation of the effects of frame time variation on VR task performance. In *Virtual Reality Annual International Symposium*, pages 38–44. IEEE, 1997.

[129] W. Weaver and C. E. Shannon. *The Mathematical Theory of Communication*. Urbana, Illinois: University of Illinois Press, 1949.

[130] Laurie M. Wilcox, Robert S. Allison, Samuel Elfassy, and Cynthia Grelik. Personal space in virtual reality. *ACM Trans. Appl. Percept.*, 3:412–428, 2006.

[131] Chadwick A. Wingrave and Doug A. Bowman. Baseline factors for raycasting selection. In *Proceedings of Virtual Reality International*, pages 10 (CD–ROM proceedings), 2005.

[132] Chadwick A. Wingrave, Doug A. Bowman, and Naren Ramakrishnan. Towards preferences in virtual environment interfaces. In *EGVE '02: Proceedings of the workshop on Virtual environments 2002*, pages 63–72. Eurographics Association, 2002.

[133] Chadwick A. Wingrave, Joseph J. Laviola, Jr., and Doug A. Bowman. A natural, tiered and executable UIDL for 3D user interfaces based on concept-oriented design. *ACM Transactions on Computer-Human Interaction*, 16(4):1–36, 2009.

[134] Chadwick A. Wingrave, Ryan Tintner, Bruce N. Walker, Doug A. Bowman, and Larry F. Hodges. Exploring individual differences in raybased selection: Strategies and traits. *IEEE Virtual Reality 2005*, pages 163–170, 2005.

[135] R.S Woodworth. The accuracy of voluntary movements [monograph supplement]. *The psychological review*, 1899.

[136] Shumin Zhai, William Buxton, and Paul Milgram. The "silk cursor": investigating transparency for 3D target acquisition. *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 459–464, 1994.

[137] Shumin Zhai, Jing Kong, and Xiangshi Ren. Speed-accuracy tradeoff in Fitts' law tasks–on the equivalency of actual and nominal pointing precision". *International Journal of Human-Computer Studies*, 61(6):823 – 856, 2004.

[138] Shumin Zhai, Paul Milgram, and William Buxton. The influence of muscle groups on performance of multiple degree-of-freedom input. In *Proceedings of the SIGCHI conference on Human factors in computing systems: common ground*, CHI '96, pages 308–315. ACM, 1996.