Design and Evaluation of 3D Selection Techniques based on Progressive Refinement

Felipe Bacim\textsuperscript{a,}*, Regis Kopper\textsuperscript{b}, Doug Bowman\textsuperscript{a}

\textsuperscript{a}Dept. of Computer Science and Center for Human-Computer Interaction
Virginia Tech, 2202 Kraft Dr., Blacksburg, VA 24060, US
\textsuperscript{b}Dept. of Computer and Information Science and Engineering
University of Florida, E405 CSE Building, Gainesville, FL 32611, US

Abstract

Issues such as hand and tracker jitter negatively affect user performance with 3D selection techniques based on the ray-casting metaphor. This makes it difficult for users to select objects that have a small visible area, since small targets require high levels of precision. We introduce an approach to address this issue that uses progressive refinement of the set of selectable objects to reduce the required precision of the task. We present three exemplar techniques (Sphere-casting refined by QUAD menu (SQUAD), Discrete zoom, and Continuous zoom) and derive a design space for progressive refinement from their characteristics. We explore the tradeoffs between progressive refinement and immediate selection techniques in two studies: first comparing SQUAD to ray-casting; and second comparing the zooming techniques to ray-casting. In both studies, an analytical evaluation based on a distal pointing model and an empirical evaluation demonstrate that progressive refinement selection can provide significant benefits compared to immediate techniques. In the first study, SQUAD was much more accurate than ray-casting, and SQUAD was faster than ray-casting with small targets and less cluttered environments. The issue with SQUAD, however, is that it requires all selectable objects to be visually distinct. The zooming techniques address this issue by exploring other areas of the progressive refinement design space. They allow users to use the spatial relationships among objects as criteria.

*Corresponding author. Tel.: +1 540 235 6969; fax: +1 540 231 9218
Email addresses: fbacim@vt.edu (Felipe Bacim), kopper@vt.edu (Regis Kopper), bowman@vt.edu (Doug Bowman)
for selection and to increase precision without requiring precision in pointing. The results of the second study show that while the zooming techniques were significantly slower than ray-casting, accuracy was much higher. Additionally, depending on the size of the target, users chose not to use zoom and, therefore, performed as fast as with ray-casting.

Keywords: 3D interaction, Selection, Performance models, User studies

1. Introduction

Selection, which involves the specification of one or more objects by the user, is one of the fundamental 3D interaction tasks (Bowman et al., 2004). Although various metaphors for selection of single objects have been developed, such as virtual-hand (Stoakley et al., 1995) and image-plane techniques (Pierce et al., 1997), ray-casting (Mine, 1995) is perhaps the most popular selection style in virtual environments (VEs) due to its simplicity and generality. Ray-casting requires only two degrees of freedom and works at any distance, while virtual hand techniques require at least three degrees of freedom and are often limited to a certain distance from the user. Even though ray-casting provides better performance than virtual hand techniques in many applications, it also has limitations. When the visual size of the target is small, due to object size, occlusion, or distance from the user, ray-casting is slow and error-prone (Steed and Parker, 2004), because it does not provide high-precision pointing at a distance. Small movements of the user’s hand result in larger and larger movements of the ray as the distance along the ray increases.

Several examples of applications for which these issues make ray-casting unsuitable can be found in the literature. For instance, in the supermarket used in the first IEEE 3DUI Grand Prize (Figueroa et al., 2010), users had the task of selecting small, partially occluded objects in a highly cluttered environment. Basic ray-casting would not allow the user to select these objects from a distance; in general, high error rates would result. To make ray-casting practical for this task would require the user to remove occluding objects to increase the visual size of the targets or to spend time traveling close to the targets to make them easier to select. Either of these options would result in long selection times.

Another example is the selection of 2D targets on large displays using distal pointing, which allows users to point at the screen from a distance
Applications that use large high-resolution displays can benefit from this style of interaction, since the traditional mouse and keyboard are not practical while standing up and moving around, but basic ray-casting does not provide enough accuracy to select smaller objects when users are not close to the display. Therefore, more physical navigation is required. If multiple objects need to be selected, users are required to constantly go through the process of deciding when and where to move to be able to select the objects.

We find situations where these problems occur in other applications, such as astrophysical or atomic datasets, large document collections, authoring of crowd simulation (Ulicny et al., 2004), and detailed information visualizations. These applications could benefit from techniques that allow accurate selection in cluttered environments without requiring users to be precise.

A number of techniques have been proposed to deal with the precision limitations of ray-casting. Examples include snapping (Haan et al., 2005), cone-casting (Liang and Green, 1993), 3D bubble cursor (Vanacken et al., 2007) and PRISM-style pointing (Frees et al., 2007). However, in highly cluttered environments these techniques require users to interact very carefully to accomplish a single precise selection, and may actually result in worse performance than standard ray-casting in some situations.

To address this challenge, we propose selection methods that use progressive refinement of the set of selectable objects, a concept we initially presented by Kopper et al. (2011). The main idea is to gradually reduce the set of selectable objects in such a way that the task requires less precision. Progressive refinement is an example of “effortless” interaction, in which a series of rough or imprecise actions can be used to accomplish a precise result. This is in contrast to traditional techniques, which use immediate selection and require precision. There is an inherent tradeoff between these two categories of techniques. Progressive refinement requires a process of selection, often using multiple steps, although each step can be very fast and accurate. Immediate techniques, on the other hand, involve a single high-precision spatial selection at the expense of being slow and having a higher error probability. The primary goal of the work presented in this paper is to explore this tradeoff. In other words, we want to know when it makes sense to sacrifice the simplicity of immediate selection in order to improve speed and/or accuracy.

In this paper, we describe our efforts to design and evaluate 3D selection techniques based on the concept of progressive refinement, and to explore the
related tradeoffs in speed and accuracy. This article is an extended version of a paper from the 2011 IEEE Symposium on 3D User Interfaces (Kopper et al., 2011), in which we described the concept of progressive refinement, presented an initial design space for progressive refinement techniques, and reported on the initial design and evaluation of a technique called SQUAD, demonstrating its benefits compared to ray-casting. In this article, we additionally describe the design and evaluation of new progressive refinement techniques, complementing SQUAD, from other parts of the design space. Therefore, in addition to the contributions presented previously (Kopper et al., 2011), the work presented in this paper describes new techniques, improves our understanding of the progressive refinement design space, and sheds further light on the tradeoffs related to progressive refinement and immediate 3D selection techniques.

2. Related Work

Bowman et al. (2004) divided selection techniques into four main categories: selection by pointing (e.g., ray-casting (Mine, 1995)), selection by touching (e.g., virtual-hand (Mine et al., 1997)), selection by occlusion (e.g., image-plane techniques (Pierce et al., 1997)) and indirect selection (e.g., selection by attributes (Bowman et al., 2004)). All these techniques can be classified as immediate selection, since they only require a single high-precision selection without refinement.

Ray-casting (Mine, 1995) is a widely used pointing-based technique, in which the user points with a virtual ray extending from the hand or input device to specify an object in the scene. Although it is very simple, this technique in its pure form suffers from a number of issues, mostly because of natural hand tremor and tracker jitter, which make it difficult for the user to control the origin and orientation of the ray. This is a bigger issue with ray-casting than with other techniques because the small hand movements are amplified at the end of long rays, causing ray-casting to be less precise as target objects get farther from the user. These issues make ray-casting difficult to use when the objects have a small visual size (Steed and Parker, 2004), as selecting such objects by pointing requires high levels of precision.

In order to address these issues, a number of improvements have been proposed. Even though such techniques improve selection performance in general, they can have a negative effect in very cluttered environments. Cone-casting (Liang and Green, 1993), for example, extends ray-casting by adding
a cone-shaped volume to the ray to make it easier to select objects that are distant. In cluttered environments, however, many objects will fall inside the cone, so that the user still has to point precisely to select the desired object. The snapping technique presented by Haan et al. (2005) uses a selection volume to calculate and accumulate scores over time for each object. This way, it can estimate which object the user wants. The bubble-cursor (Grossman and Balakrishnan, 2005) is a 2D technique that dynamically resizes a circular cursor so that it only contains one object. A 3D extension of the bubble-cursor, which uses a sphere instead of a circle, was presented by Vanacken et al. (2007). Both of these techniques may actually perform worse in cluttered environments, since even small movements will cause the ray or the cursor to constantly snap or resize to select new targets.

Other techniques improve ray-casting accuracy by changing the control-display ratio, either automatically (e.g., PRISM (Frees et al., 2007), when moving slowly) or manually (e.g., ARM (Kopper et al., 2010), by pressing a button). While these techniques can achieve very high levels of precision, all of them have limitations. PRISM and ARM cause a significant mismatch of the physical pointing direction to the perceived pointing position, and the mapping is nonlinear. Finally, these techniques require the user to interact very carefully and with full attention. Our proposed approach of selection by progressive refinement aims to allow “lazy” interaction with high accuracy.

There are existing progressive refinement techniques in the literature. For example, the shadow cone-casting technique (Steed and Parker, 2005) uses continuous movement with cone-casting to disambiguate selection. The depth-ray technique (Grossman and Balakrishnan, 2006), which adds depth control to the classic ray-casting technique to select occluded objects, requires two actions to specify the target. PORT (Lucas, 2005) allows the selection of multiple objects and uses a series of movement and resizing actions to define the set of targets.

The flower ray technique (Grossman and Balakrishnan, 2006), in which occluding targets are concurrently selected by ray-casting and disambiguated in a second phase by a marking menu is perhaps the closest existing technique to our approach. Although this technique is suited for highly cluttered environments, it requires high precision for the ray selection and does not scale well to a large number of objects, since the marking menu object specification is done in a single phase.

The navigation techniques for multiscale virtual environments presented by Bacim et al. (2009) also use progressive refinement selection. Users have
to navigate the hierarchy of the virtual environment using widgets based on spatial (MSWIM) or hierarchical (HiSMap) information, and then select a small representation of the part of the virtual environment they want to navigate to. Even though these techniques are not affected by the size of the object, HiSMap takes objects out of their spatial context, which means it has the same limitations as SQUAD, and MSWIM can be slow since users have to navigate the hierarchy and search for the objects in a miniature of the world.

Several different techniques for zooming in 2D user interfaces have been developed in the past, some of which may be characterized as progressive refinement. For instance, Pad++ (Bederson and Hollan, 1994) divides data hierarchically and allow users to zoom in and out in the hierarchy by clicking in parts of the structure. Schaffer et al. (1993) developed a variable zoom algorithm to generate fish-eye views of hierarchically structured clustered networks. These two differ from our methods mainly because they propose techniques that use the data and the data organization to perform zoom. The zoom bar (Jog and Shneiderman, 1997) introduced the idea of controlling the zoom window directly by resizing bars that represent width and height, and moving them to change the position. This idea is similar to the discrete zoom technique, but it requires users to finely control the position and size of the zoom window.

Zoom-and-pick (Forlines et al., 2005) is a selection technique in which a square fish-eye lens is used to magnify the pixels around the cursor. This technique is similar to both of our zooming techniques, but it has two disadvantages. First this technique magnifies the pixels around the cursor and does not change the resolution of the magnified area, while our techniques increase the resolution of the magnified area by changing rendering parameters, thus displaying a detailed view of the area. In addition, the control-display ratio in zoom-and-pick remains the same as the original zoom level when the user zooms, while our techniques modify the control-display ratio relative to the current zoom level. This allows our techniques to decrease the required precision.

In 3D environments, Hansen et al. (2008) developed StarGazer, a 3D user interface for selection of targets using continuous zoom and pan based on gaze. This technique is an example of progressive refinement and is similar to our continuous zoom, but instead of zooming alone, first the user gazes at the object or region of interest, and the system increasingly pans and zoom to bring objects in that region to the center of the screen, repositioning
the objects and requiring users to continuously gaze at the area of interest. This technique may not be suited for cluttered environments, since it would require several refinement steps. 3D zooming can also be found in map visualization applications such as Google Earth (Google, 2011), which allow the user to move the camera and zoom in and out based on an object of reference (the Earth) to view and select different items. By double-clicking a point on Earth, the system automatically creates an animation and zooms into that point. This method is similar to the one used for the discrete zoom technique, but requires more precision when selecting the region of zoom since magnification is maximized.

To the best of our knowledge, with the exception of our prior work (Kopper et al., 2011), there has been no prior generalization of the progressive refinement concept, and no comparison of progressive refinement techniques to immediate techniques.

3. Selection by Progressive Refinement

The concept of selection by progressive refinement is to gradually reduce the set of selectable objects in such a way that the overall selection task requires less precision. Based on this concept, we designed three techniques and developed a design space.

3.1. SQUAD Selection

The Sphere-casting refined by QUAD menu (SQUAD) selection technique (Kopper et al., 2011) uses two distinct refinement phases. In the first phase, the user specifies a volume containing the target object. The user then refines the initial selection progressively by selecting the subset of objects containing the target from a four-item menu displaying all the remaining objects, until the target is finally selected. SQUAD makes it possible to accomplish precise selection without requiring the user to use precise actions at any moment during the selection task.

We designed SQUAD as part of our entry to the 3DUI Grand Prize contest (Figueroa et al., 2010). The main challenge proposed by the contest was to design techniques that support interaction in a highly cluttered environment. In a virtual supermarket, users had to select specific objects identified by textures with unique characteristics. To achieve rapid yet precise selection, we designed SQUAD as a progressive refinement technique that divides
selection into two discrete steps, the first being spatial and in-context and the second being out-of-context.

The first step uses a modified version of ray-casting that casts a sphere onto the nearest intersecting surface to determine which objects will be selectable. We call this subtask sphere-casting. The user simply has to ensure that the desired object is inside or touching the sphere, so that it can be picked from among the other objects in the next phase. Items that will be made selectable are highlighted. In order to improve confidence that the desired object will be available, the sphere’s radius increases the farther the user is from the nearest intersecting surface, thus increasing the overall number of objects available in the second phase. Figure 1 illustrates this selection phase. (Note that in the study described in section 4, however, the sphere size is fixed since all objects are placed at the same distance from the user.) Sphere-casting avoids the precision issues of ray-casting, and also allows selection of occluded objects.

We considered two ways of implementing sphere-casting for our design: to shoot a ray (like ray-casting), and create a sphere at the first intersection of the ray with the environment, or to use a cone (like cone-casting), and create the sphere at the position of the nearest object that is in the cone. Since the latter would cause issues with cluttered environments, for example, when trying to select a visible object behind groups of other objects, we decided to implement the first option.

Upon completion of the first phase, all objects that are inside or touching the sphere are evenly distributed among four quadrants on the screen, without regard for the spatial locations of the objects in the 3D environment. We call this the quad menu, and note its similarity to marking menus. Contrary to zone menus (Zhao et al., 2006), where breadth of selection is achieved by relative position of multiple marking gestures, in the quad menu phase users refine the selection by repeatedly pointing anywhere in the quadrant that contains the item they are looking for, each time reducing the number of objects per quadrant until the desired object is the only one left. This process is illustrated in Figure 2. The maximum number of selections necessary in the quad menu is \(\lceil \log_4 n \rceil\), where \(n\) is the initial number of items. For example, if the sphere has between 17 and 64 objects inside it, our technique would require at most four clicks to select the target (one click for sphere-casting and three clicks for the quad menu).

In SQUAD, the objects are distributed evenly and randomly across the four quadrants. Although not relevant to the study we performed, there are
many criteria that can be used for distribution. For instance, to preserve spatial context in the first refinement of the quad menu, one could distribute the items according to their location in the sphere. Other examples include distribute the objects using color, shape, or any other data related to them that may be relevant to the application at hand.

SQUAD is an example of a progressive refinement technique that works well in environments where there are many objects that are arranged along a surface, and where the desired object is visually distinct from the rest. For other selection tasks or environments, however, different design choices (e.g., using a cone as the selection volume or distributing items in the menu based on spatial location) might be preferred.

3.2. Progressive Zoom Techniques

The primary limitation of SQUAD is that it requires target objects to be visually distinct. This happens because users have to perform the refinement
phase of selection outside of the objects’ spatial context. One way to increase accuracy in selection without taking objects out of their spatial context is to increase the target size. In order to do this, users can change the viewpoint and move closer to the object, or zoom into the region of the screen containing the target. For the techniques presented in this paper, we chose zooming for two reasons: it is based on screen-space, so it is much simpler than navigating closer to the object; and once the object is selected, the view frustum can return to its original state without requiring the user to perform navigation.

Based on these ideas, we designed two progressive refinement techniques: discrete and continuous zoom. These techniques reduce the level of precision required to select the target by using zoom to magnify it. Both techniques are examples of progressive refinement techniques that allow users to increase precision without requiring them to refine selection out of the spatial context of the target objects.
The next subsections describe the two zoom techniques we designed and the different ways of implementing zoom in virtual environments.

3.2.1. Discrete Zoom

The discrete zoom technique uses 2D menus for specification of the zooming area, discrete actions to define when to zoom, and ray-casting. The user can choose to point directly to the object and select it, or to point to the region of the screen that contains it and perform zoom to increase the target size. Once the object is selected, the user goes back to the initial unzoomed view, and is able to perform other actions.

While a traditional ray-casting implementation is used to point and select objects at any zoom level, a menu is used to quickly determine the region of zoom. The menu is an overlay on the screen and contains four quadrants, similar to the quad menu in SQUAD, that are always visible. However, differently than SQUAD, the menu is translucent and the objects in the environment are visible all the time. The quadrant containing the cursor is highlighted, and it is used to determine the region of the screen that will be magnified (Figure 3).

Figure 3: Three refinement steps of the discrete zoom technique.

Once the quadrant is selected, the system performs a quick (0.25s) animation from the current frustum to the magnified version of the selected quadrant. This animation helps users to keep track of the context and also the location of the object they want to select. The user then refines selection by repeatedly selecting the quadrants to zoom and decrease the precision needed. However, instead of refining until there is only one object in the quadrant like in SQUAD, users have the ability to decide how much magnification is enough, and then select the object directly using ray-casting. This technique also allows zooming out to the previous set of quadrants if the wrong quadrant is selected.
3.2.2. Continuous Zoom

The continuous zoom technique works in a similar way to discrete zoom, but instead of using quadrants for zooming, it zooms in the direction of the cursor continuously. This way, users can simply point roughly toward the object and zoom in until the target is large enough for selection.

Just like with the discrete zoom technique, traditional ray-casting is used for selecting targets, and the zoom feature is used to quickly zoom in toward the target object. Differently than the discrete zoom technique, which has discrete levels of zoom, the continuous zoom can be stopped at any level so that the user can use ray-casting. To zoom, users first have to roughly position the cursor over the region they intend to magnify (Figure 4).

Figure 4: Three snapshots of different zoom levels achieved with the continuous zoom technique.

Once the zoom in button is pressed, the system starts to magnify the region of the screen around the cursor such that objects under the cursor remain in that position independently of the zoom level. While zooming, users can adjust the position of the cursor to change the region of the screen that they are zooming into. In order to minimize precision issues while zooming, this technique doubles magnification in each dimension every second the zoom button is pressed, up to 256x magnification. As with discrete zoom, users can choose to zoom out, and they have to decide how much magnification is needed, if any.

3.2.3. Zoom Implementation

Zooming in computer graphics is usually done either by applying a scale to the modelview matrix or by scaling the width and height of the view volume (Shreiner and Group, 2009). These techniques zoom in to the center of the current view, however, whereas our techniques need to be able to
zoom into specific areas of the screen. This section describes four different methods of implementing zoom that we considered, discussing the strengths and weaknesses of each one.

*Render to Texture & Texture Magnification.* The idea of the first method is simple: render the scene to a texture and magnify this texture (Figure 5). The issue is that magnifying a texture will magnify the pixels, and therefore decrease visual detail. This can be seen in the last magnification in Figure 5, where the pixels are visible and it is hard to identify the characters.

Figure 5: Issue of magnifying a texture for zooming.

*Alternative to Zoom: Move the Camera.* The second method is actually not zooming at all; rather, it moves the camera and keeps the view frustum unchanged (Figure 6.c.1). However, there are a few issues with this method. First, by displacing the camera, occlusion issues may be generated. The new view has to include everything that was visible in the selected area of the
screen. In order to do this, the new camera position may be inside or in front of an object in the environment which will occlude everything else (see the case of the red object to the left in Figure 6.c.1). Second, new objects that were not in the area selected may appear, confusing the user (see the case of the red object to the right in Figure 6.c.1).

Figure 6: Illustration and snapshots of zooming into a quadrant using the discrete zoom technique, with two initial steps in selection of a screen region to zoom (a and b), then three methods for zooming: (c.1) move the camera, (c.2) rotating the camera & frustum manipulation, and (c.3) frustum manipulation & off-axis projection.

*Rotating Camera & Frustum Manipulation.* To solve the issues with moving the camera to zoom, we designed the method of rotating the camera and adjusting the frustum, or reducing the field of view (Figure 6.c.2). This alternative generates similar results to the magnified texture, without the issues of that technique. However, there is an issue with this method too. Since there is a change in the camera, and perspective tends to distort objects as farther they are from the screen, the image generated is not exactly the same as the one selected initially.

*Frustum Manipulation & Off-Axis Projection.* Addressing the issues of the previous method, the final alternative is to perform zoom without rotating the camera and instead adjusting the frustum to do off-axis projection while also reducing the field of view (Figure 6.c.3). This generates the exact same image as the one shown in the selected region of the screen initially, and zoom is correctly applied in the image plane without distortions or spatial aliasing. We used this method to implement our zooming techniques.
3.3. Design Space

Based on existing techniques and the ones we designed, we identified four dimensions of the design space of progressive refinement selection techniques, which are shown in Figure 7.

- **Type of progression**
  - discrete
  - continuous

- **Refinement criteria**
  - in the original spatial context
  - out of the original spatial context
  - by object attributes

- **Display of selectable objects**
  - in context
  - out of the original spatial context

- **Strategy**
  - enforced by the technique
  - determined by the user

Figure 7: Design space of selection by progressive refinement.

First, progressive refinement can be done either through several discrete steps, as in SQUAD, or with a continuous process, as in shadow cone-casting (Steed and Parker, 2005) and our continuous zoom technique.

The method of refinement defines another dimension of the design space. This refers to the criteria that are used to reduce the set of selectable objects. Refinement can be specified spatially within the environment context, for example through the use of a volume or area in the image plane, limiting the region of the environment where the target can be. Both zoom techniques are examples of in-context refinement, in which all refinement is performed in the target’s spatial context. Refinement can also be done through “out-of-context” subset specification, which involves picking a subset of objects from a list or menu instead of from the environment. The quad menu refinement in SQUAD is an example. Finally, refinement can be done by the specification of attributes of the desired object, such as color, size or shape.

The design space is further defined by the method used to display the current set of selectable objects. Subsets of selectable objects can be displayed in context, for example through zooming, visual explosion, highlighting, moving the viewpoint closer to the subset or through the removal or dimming of non-selectable objects. The subset of selectable objects can also be dis-
played out-of-context (as in SQUAD), through the use of menus, which may be sorted in some way or arranged randomly.

Finally, the strategy for selection of targets can be either enforced by the technique or determined by the user. For instance, with SQUAD, users have to reduce the set of selectable objects by selecting the quadrants until there is only one object in the quadrant they select, and they do not have the choice of selecting individual targets once they get to the refinement phase. On the other hand, the zooming techniques were designed such that users can determine what level of zoom is enough for selecting a target and can perform selection at any zoom level.

We can further characterize progressive refinement selection techniques along a continuum based on the gradualness of refinement (Figure 8). At one end of the spectrum we have the immediate techniques, which directly specify the target object. This can be thought of as a “refinement” from the entire set of selectable objects in the environment to one or zero (in case of a failed selection) in a single step. At this end of the continuum, too much precision may be required, as an exact element needs to be specified immediately. At the other end of the continuum we can imagine a technique that has many refinement steps, with an extreme case being a technique where each refinement simply excludes one object from the set of selectable objects. Here precision is also required, and in fact such a technique requires many high-precision selections. In the middle of the continuum are the techniques of interest, where the reduction in the set of selectable objects is rapid and accurate.

4. Evaluation 1 - SQUAD

We conducted an experiment comparing SQUAD to standard ray-casting. We evaluated the task of pointing at circular targets that varied in radius, on a screen that was filled with distractor objects varying in number and density.

4.1. Goals and Hypotheses

The overall goal of the experiment was to explore the tradeoff between ray-casting and SQUAD. While ray-casting is an immediate technique that requires only one click, it requires precision with visually small targets. SQUAD, on the other hand, requires very little precision from the user, at the expense of multiple steps until the desired target is selected.
With this tradeoff in mind, we expected there to be an interaction between technique and target size. We hypothesized that SQUAD would take constant time with respect to target size, while ray-casting would be slow with small targets and fast with large targets. We were unsure how the constant SQUAD times would compare to the times for ray-casting with the various target sizes, but expected that SQUAD would be faster in at least some target size conditions.

We also hypothesized that the number of distractor objects around the target would have a significant effect on time to select with SQUAD, but that the number of distractors would have no effect on ray-casting. We expected that SQUAD would outperform ray-casting when the number of distractors was small.

With respect to accuracy, we hypothesized that SQUAD would yield virtually no errors, due to its low required precision, whereas ray-casting would have more errors as the target sizes decreased.

Finally, we hypothesized that situations in which the tracking has more jitter would result in more errors and slower time for ray-casting, but would not impact SQUAD, as all the steps of the technique require very low pointing precision.

4.2. Design

We used a factorial within-subject design with repeated measures. There were four independent variables: technique (ray-casting, SQUAD), tracking...
(normal, jittery), target size (radii 0.01m or 0.26°, 0.015m or 0.40°, 0.04m or 1.06°), and the number of distractors inside the selection sphere (referred to as distractor density) (16, 64, 256). Thus, the design was 2x2x3x3.

The order of presentation of technique and tracking was counterbalanced, blocked by technique, such that each participant performed both tracking conditions within the same technique before moving to the next one. Within the combinations of technique and tracking, each of the nine conditions of target size vs. distractor density was repeated eight times and presented in random order.

4.3. Analytic Evaluation

Before running an empirical study (section 4.4), we analytically evaluated performance in our experimental conditions based on predictive models.

The tradeoff between speed and accuracy described in Fitts’ law is well known for pointing tasks (MacKenzie, 1992; Zhai et al., 2004). Recently, a similar model was shown to apply for distal pointing tasks. In distal pointing, the input device is remotely located with respect to the display area and the pointing is done in a direct fashion, as opposed to indirectly, for example, through the use of a mouse (Kopper et al., 2010). SQUAD and ray-casting both use distal pointing, making this model relevant to our study.

Kopper et al.’s predictive model of distal pointing states that the time to acquire a distal target through direct pointing depends strongly on the angular width of the target and, to a lesser degree, on the angular amplitude of the wrist/arm movement required to complete the task. The difficulty of the task is expressed as

\[ ID_{DP} = \left[ \log_2 \left( \frac{\alpha}{\omega^k} + 1 \right) \right]^2, \tag{1} \]

where \( ID_{DP} \) is the index of difficulty, \( \alpha \) is the angular amplitude of the movement and \( \omega \) is the angular width of the target. The constant \( k \) is a power factor greater than one that expresses the greater importance of the target width relative to movement amplitude. The value of \( k \) was shown to be around three in the experimental setting used by Kopper et al. While our study used a different environment, we believe that it was similar and the value of \( k \) should be approximately the same. The predicted time for selection with ray-casting based on the index of difficulty is defined as

\[ MT = 1.091 + 0.028 \times ID_{DP}, \tag{2} \]
expressed in seconds.

The goal of SQUAD is to reduce the index of difficulty of an individual pointing action to a minimum at the expense of increasing the number of actions needed to achieve the goal of selecting a single unique object in a highly cluttered environment. In order to reduce $ID_{DP}$ to a minimum in our study, we set the diameter of the selection sphere to $26.3^\circ$. The targets were chosen within a constant distance range from the starting point, so that the movement amplitude was selected randomly between $10.0^\circ$ and $17.9^\circ$, with an average $\alpha$ of $14.0^\circ$. This yields an $ID_{DP}$ of

$$ID_{DP} = \left[ \log_2 \left( \frac{14.0}{26.3^3} + 1 \right) \right]^2 \approx 1.23 \times 10^{-6}. \quad (3)$$

Thus, the index of difficulty of the task of selecting the target region becomes virtually zero, and the expected time to select the target is very small. Similarly, the difficulty of selecting a quadrant in the quad menu is minimal, as the angular width of each of the quadrants is $45^\circ$, yielding an $ID_{DP}$ very near zero. According to Kopper et al.’s model, the intercept of the regression line for predicted selection times (when $ID_{DP}$ tends to zero) is $1.091s$. However, values of $ID_{DP}$ this close to zero have not been tested experimentally. With $\omega$ higher than $\alpha$, we anecdotally observed that selection time is typically under the lower limit of $1s$ set in Kopper et al.’s model.

During the quad menu phase of selection, the user needs to first find the quadrant containing the intended target, then point and click to select it. Although the target stands out and is easily distinguishable from the distractors, in theory the time for visual search will increase with the number of distractors, since more distractors reduces target size, which in turn diminishes the perceived contrast (Jr and Fullenkamp, 1988). Thus, we hypothesize that the time it takes to select a target using SQUAD selection is

$$MT_{SQUAD} = c + \sum_{i=1}^{R} (c + v_i), \quad (4)$$

expressed in seconds, where $R$ is the number of refinement iterations required during the quad menu phase of the technique, $c$ is an empirically determined constant related to the time it takes to point at a target whose difficulty tends to zero, and $v_i$ is the visual search time to find the target in the quad menu before movement starts. We expect $v_1$ to take the longest time, because there
is a switch in interaction mode, from sphere-casting to quad menu selection, and a change in the visual environment. Also, the number of distractors is at its maximum, and it decreases as refinements are made, reducing the target search space and time. Due to the visual distinctness of the target in relation to the distractors in our experimental setting, we expect $v_i$ to be low in all phases of refinement and to not affect selection time by a large amount.

Here, we note some interesting characteristics of SQUAD selection as compared to ray-casting. First, our model predicts that target size plays no role in the time it takes to complete a selection. We acknowledge that there may be a longer search time for visual segmentation in highly dense environments with small and occluded targets, but the motor movement time is constant once the target has been found. Second, the time it takes to select a target with SQUAD selection is directly proportional to the amount of clutter – or the number of distractor objects that exist in the region of the desired target. While the time it takes to select a target grows exponentially (see eq. 5) with the increased number of iterations, the growth in the number of iterations is rather slow, on the order of $[\log_4(n)]$, where $n$ is the number of objects inside the sphere (Figueroa et al., 2010).

In order to compare ray-casting with SQUAD, we decided to vary both the target size and the number of distractor objects that fall inside the selection sphere at any given time. We defined target $\omega_s$ as 0.53°, 0.80° and 2.12°, yielding for ray-casting an $ID_{DP}$ of 42.9, 23.4 and 1.67, respectively. Thus, the predicted time to complete the ray-casting tasks for each of the respective target sizes was 2.29s, 1.74s and 1.14s, respectively. We set the number of distractor objects inside the sphere to be 16, 64 and 256, yielding a total of 3, 4, and 5 clicks to select the target with SQUAD in each distractor density condition, or 2, 3 and 4 refinements. This leads to a theorized $3c + v_{l_2}$, $4c + v_{l_3}$ and $5c + v_{l_4}$ seconds to select a target, where $v_{l_R}$ is the total visual search time across all $R$ refinement phases in each condition. The value of $c$ needs to be empirically determined, but we expect it to be less than one. We believe that $v_{l_R}$ will be low and will not affect movement time by a large amount.

Table 1 shows the predicted times for ray-casting and SQUAD for all target sizes and densities, considering $c = 0.5$, $v_0 = 0$, and for $i > 0$, $v_i = 0.2 + ((R - i) \times 0.1)$. We estimate $c = 0.5$, which is smaller than the minimum time predicted in Kopper’s model, because the index of difficulty for selecting the quadrants is much smaller than what was tested in the model validation. We estimate the visual search time to increase slowly for each refinement step as the number of refinements increases, since the red target gets smaller
and the pop-out effect decreases. Thus, we hypothesize that the final visual search time for our tasks can be defined as

\[ v_{t_R} = \sum_{i=1}^{R} (0.2 + ((R - i) \times 0.1)), \]

expressed in seconds. The growth in the visual search time in each iteration indicates an exponential relationship between the selection time and the number of refinements. The table demonstrates how we expect ray-casting performance to be affected by the target size, but not by the number of distractors, and SQUAD performance to be affected by the number of distractors but not by the target size. Additionally, it also reflects the exponential relationship with a larger difference between “Medium density” and “High density” (1.0s) when compared to the difference between “Low density” and “Medium density” (0.9s).

Table 1: Predicted times for ray-casting using Kopper et al.’s predictive model and for SQUAD, with different densities and target sizes.

<table>
<thead>
<tr>
<th>Density</th>
<th>Target Size</th>
<th>Ray-casting</th>
<th>SQUAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Small</td>
<td>2.29s</td>
<td>2.0s</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>1.47s</td>
<td>2.0s</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>1.14s</td>
<td>2.0s</td>
</tr>
<tr>
<td>Medium</td>
<td>Small</td>
<td>2.29s</td>
<td>2.9s</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>1.47s</td>
<td>2.9s</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>1.14s</td>
<td>2.9s</td>
</tr>
<tr>
<td>High</td>
<td>Small</td>
<td>2.29s</td>
<td>3.9s</td>
</tr>
<tr>
<td></td>
<td>Medium</td>
<td>1.47s</td>
<td>3.9s</td>
</tr>
<tr>
<td></td>
<td>Large</td>
<td>1.14s</td>
<td>3.9s</td>
</tr>
</tbody>
</table>

According to our analytical evaluation, ray-casting should be faster than SQUAD for almost all conditions, except for small targets in low density environments, in which SQUAD should be 0.29s faster. However, these results assume values for both \(c\) and \(v\), and assume that Kopper et al.’s model can be applied directly to the ray-casting condition in our experiment. Thus, these predictions may not be a good representation of the final results. The predictions were tested in our empirical evaluation.
4.4. Empirical Evaluation

In order to empirically validate the results from our analytic evaluation, we performed a comparative study of SQUAD and standard ray-casting.

4.4.1. Apparatus

We used a back-projected VisBox-SX system, with only one projector (monoscopic) to display the experimental environment on a 2.29m x 3.05m screen. The resolution of the graphics was 1400px x 1050px. A wireless Intersense IS-900 Wand was used for controlling the cursor on the screen.

The experimental software was written using the Vizard Virtual Reality Toolkit by WorldViz. It ran under Microsoft Windows XP on a workstation with an Intel Core2 Duo 6600 CPU at 2.40GHz and 2GB of RAM. The frame rate was fixed at 55 frames per second for all conditions except with the high-density distractor conditions, in which it went down to around 15 frames per second in the sphere-casting phase only, because many collision tests with the selection sphere were necessary. We were comfortable with the drop in frame rate for that one condition because the sphere-casting selection was very easy to perform.

The environment consisted of circular objects as shown in Figure 9. The user stood at the center of an invisible sphere of 2.155m radius, at an orthogonal distance of 1.52m to the display surface. The red target and the gray distractors were evenly distributed on the surface of the sphere. There was no head tracking or any virtual navigation of the environment and the viewpoint remained at a fixed location. The perspective projection of the objects caused them to have the correct visual size from the user’s point of view at the center of the sphere. We made the decision to render the circles on the surface of a virtual sphere, as opposed to on the flat screen plane, because the effective angular width of objects displayed far from the center of a flat screen decreases (Kopper et al., 2010). The perspective rendering of the circles near the edges of the display compensated for this effect in our environment, such that all objects had the same angular width from the user’s point of view.

The target position was randomly selected from a list of candidate targets that fell inside a torus-shaped section on the surface of the display sphere, and was limited by a small radius of 0.52m and a large radius of 0.77m. This ensured that targets were presented in all directions from the center of the screen.
The cursor position was determined by a function of the yaw and pitch of the IS-900 wand and the display’s field of view. With the user standing at a fixed position in front of the display, the position of the cursor closely matched the ray extending from the wand. We decided to rely only on the angular readings of the wand, rather than implementing 3D ray-casting based on the combination of position and orientation information, because we wanted to keep the motor difficulty to complete the task constant. Participants were told to keep their hand position within a small range over a mark on the floor that determined the center of the virtual sphere, and not to reach out with their arms. With the cursor position dependent solely on the wand’s yaw and pitch, we were able to keep the motor behavior identical to that of ray-casting from the sweet spot at the center of the virtual sphere. There was then, of course, a mismatch between the position of the displayed cursor.
and that of the position of the 3D ray extending from the user hand with the screen. This offset was, however, minimal and no participant seemed to mind, or even notice, the difference.

Each task began with only two objects on the screen: a large yellow circle in the center, and the red target. Once the user clicked the large yellow object, it disappeared and the rest of the screen was filled with distractor objects. We did this for two reasons. First, clicking at a pre-determined spot before the start of a task meant that the angular amplitude of the movement was kept in a controlled range. Second, by not showing the distractors in the beginning, the user could find the target location before starting the task, reducing any cognitive time to segment the target from the distractors to a minimum.

For the ray-casting condition, a crosshair represented the cursor and the task was finished when the user clicked the trigger button on the IS-900 wand. When the cursor intersected with an object, either the target or a distractor, the object was highlighted with a yellow border.

In the SQUAD condition, after the user clicked on the yellow object in the center of the display to begin the task, the cursor changed to a sphere (Figure 9, left). All objects that were inside or intersecting with the surface of the sphere were rendered with a highlight, indicating that they were active for selection. The sphere-casting action was committed by a click with the trigger button, and the display changed to the quad menu (Figure 9, right). In order to maintain experimental control, for each distractor density, the quad menu contained the same number of elements, even if the sphere did not have exactly that number of objects inside. These numbers were close enough that no participant ever noticed a mismatch between the objects inside the sphere and the objects displayed in the quad menu. In the quad menu, we decided to limit the display of the objects to approximately 50° of the view-field, as opposed to the full 90° of the projection screen. We made this decision to minimize the potential visual search time after the menu was displayed, and we found that 50° was enough to display a large number of objects, while still allowing the user to spot the target without any head movement. To refine the quad menu selection, the user only needed to point anywhere in the quadrant that contained the target and click the trigger button.

We applied a dynamic recursive low-pass filter (Vogel and Balakrishnan, 2005) to the raw pitch and yaw data from the IS-900 wand. This filter provided a rapid response time while reducing tracking jitter to a minimum.
(the Kalman filters provided by the IS-900 system had a significant lag when the cursor was moving precisely, causing strange cursor “stickiness” effects).

For the jittery conditions, we applied a random offset between $-0.21^\circ$ and $0.21^\circ$ at each frame to the filtered yaw and pitch readings of the wand. This resulted in a maximum error of 80% of the smallest target width, such that participants had a reasonable chance of successful selection in the hardest ray-casting condition.

4.4.2. Participants

We recruited 16 voluntary unpaid participants from the campus community to perform the study. Participants’ ages ranged from 20 to 31 years old, with a median age of 22.5. Nine of the participants were female.

4.4.3. Procedure

Upon arrival, participants were greeted by the experimenter and given an informed consent form to read and sign. They were then given a colorblindness screening test and proceeded to complete a background questionnaire. After that, they were shown the experimental setting and started learning the first technique/tracking combination. The learning was done with an easy condition so they could understand the technique without making errors. They were then given a practice session, in which they had to practice all nine target size/distractor density combinations in the current condition for at least 90s.

After practicing, they were reminded that they had to perform the trials as quickly as possible while trying not to make errors, and then performed eight sets of each of the nine combinations. When errors were made, the application displayed a message (“Not quite!”) for 0.7s and the next task was displayed. The erroneous trial was then put into an array of trials that was presented in a new random order after the end of the set of trials for the current technique/tracking combination. This process was repeated to a maximum of five attempts per trial. If the user made five errors on a trial, it was deemed failed and was not presented again. The target position was the same for all attempts of a given trial.

After the end of each technique/tracking session, the participant completed a set of rating-scale questions and rested for up to two minutes. They then moved on to the next condition, following the same protocol, until all four technique/tracking combinations were completed.
Finally, the participant filled out a post-experiment questionnaire, comparing both techniques overall and in light of the other variables.

4.4.4. Results

We performed a factorial ANOVA with repeated measures on both dependent variables: time to complete a task and mean number of errors per trial.

Time. Overall, ray-casting was significantly faster than SQUAD ($F_{1,15} = 4.92, p < .05$), but only by two-tenths of a second. Interestingly, there was no main effect of tracking ($F_{1,15} = 0.001, p = 0.979$).

There were main effects of both distractor density ($F_{1,15} = 398.6, p < .0001$) and target size ($F_{1,15} = 153.4, p < .0001$). This indicates that the effects of distractor density on SQUAD and target size on ray-casting were so large that the variables were significant overall, but when we examine the interactions, we get a clearer picture of the effects.

The tradeoff of a single precise selection compared to multiple coarse selections can be clearly seen in the interactions of technique with target size and distractor density. Figure 10 shows the significant interaction between technique and distractor density ($F_{2,30} = 290.51, p < .0001$). The 95% confidence interval, at each density level, showed that SQUAD was significantly faster in the low density, that there was no significant difference for the medium density, and that ray-casting was significantly faster with high density.

As expected, all densities were significantly different from each other with SQUAD, while there was no statistical evidence for a difference for ray-casting between any distractor density pairs. There was, however, a slight increase in the mean task completion time for ray-casting as the distractor density increased. We believe that this may have been caused by an increase in visual processing time, as the distractors were highlighted as the cursor intersected with them. However, further studies should be done to verify this effect.

The other interaction that evidences the tradeoff is that of technique with target size. There was a highly significant interaction of these factors ($F_{2,30} = 135.17, p < .0001$), illustrated by Figure 11. As expected and predicted by the distal pointing model, pairwise comparisons showed that the smallest targets took significantly longer to select with ray-casting, while there were no significant differences among target sizes for SQUAD.
Looking at the interaction between technique and tracking, we expected to see that ray-casting would be slower with jittery tracking, while SQUAD would not be affected by tracking jitter. However, this interaction was not significant ($F_{1,15} = 3.93, p = 0.066$). Despite near-significance, the mean difference in time with ray-casting was only about 0.1s, and more errors were made with bad tracking, which could indicate that participants favored speed over accuracy, even if instructed otherwise.

No other significant interactions were found, which is consistent with our hypotheses.

Figure 12 shows the mean results for all technique-density-target size combinations (the three densities for ray-casting are averaged in this graph, since density had no effect on ray-casting performance). It is clear from this graph that SQUAD was significantly faster than ray-casting with low density and either small or medium size targets, and with medium density and small targets. In two other conditions, there was no significant difference between the two techniques. Finally, there are four conditions where ray-casting was significantly faster than SQUAD.
Errors. As expected, there was a significant main effect of technique with respect to errors ($F_{1,15} = 56.86, p < .0001$), with more errors being made with ray-casting. In fact, virtually no errors were made with SQUAD; the overall error rate with this technique was 0.007 errors per trial.

The lack of errors with SQUAD makes it interesting to look at the effects of tracking, target size and distractor density on ray-casting. Thus, we performed a new repeated measures ANOVA removing all the SQUAD conditions.

As expected, there was a significant main effect of target size on the number of errors per trial with ray-casting ($F_{2,30} = 46.21, p < .0001$), with more errors made with the smallest targets.

Although the average number of errors was higher for the jittery conditions, we found no statistical evidence of this difference ($F_{1,15} = 1.12, p < .31$). We believe that, since the amount of jitter was controlled, users were able to learn and compensate for it, since the low pass filter applied before the jitter allowed most participants to keep the cursor fixed on an exact pixel when needed. That, combined with the facts that the maximum jitter was 80% of the minimum target width and the cursor intersections with
objects were continuously highlighted may have allowed users to adapt and learn to select accurately with ray-casting despite the jittery cursor.

**User Preference.** SQUAD was largely preferred by the participants. When asked which technique they favored overall, when the cursor was jittery and when the targets were small, all 16 participants answered SQUAD. When asked about which technique they preferred when there were many distractors in the scene, the majority (nine) still preferred SQUAD, suggesting that the increased number of steps did not outweigh the overall preference of the technique; two participants were undecided, and the remaining five preferred ray-casting when many distractors were present.

It is also interesting to look at subjective ratings the participants gave for various aspects of both techniques. Participants were asked to fill out a survey immediately after completing each of the techniques, and to rate the techniques on a seven-point scale for ease of learning, ease of use, and how hard the techniques were in various conditions (when the cursor contained artificial jitter, the targets were small, and there were many distractors). Also on a seven-point scale, participants were asked to rate their wrist, leg and back fatigue. Before answering the survey after the last technique, participants were instructed to respond independently of the answers to the first one.
We performed Wilcoxon Signed Rank tests on each of the questions. There was no significant difference in the reported ease of learning between the techniques \((n = 7, W = 20, \text{insignificant})\). For ease of use, ray-casting \((\text{mdn} = 4.5)\) was ranked significantly more difficult than SQUAD \((\text{mdn} = 1)\) \((z = 3.16, p < .001)\). Participants found ray-casting significantly more difficult when the cursor was jittery \((z = 3.24, p < .001)\) and when the targets were small \((z = 3.5, p < .001)\). There was no significant difference with respect to task difficulty when many distractors were present \((\text{mdn}_{\text{ray-casting}} = 2, \text{mdn}_{\text{SQUAD}} = 3.5, z = -0.18, p = 0.19)\).

Participants reported significantly more arm fatigue with ray-casting \((\text{mdn} = 5)\) than with SQUAD \((\text{mdn} = 3.5)\) \((z = 2.65, p < .05)\). No significant difference was found between the two techniques for leg \((n = 5, W = 5, \text{insignificant})\) and back fatigue \((\text{mdn}_{\text{ray-casting}} = 3, \text{mdn}_{\text{SQUAD}} = 2, z = 1.52, p = 0.064)\).

### 4.5. Model Validation

Based on the analytic evaluation and the empirical results of both techniques, we can validate the predictive models for the ray-casting and SQUAD pointing tasks.

Figure 13 left shows the close linear fit of the \(ID_{DP}\)'s of each ray-casting conditions based on the distal pointing model to the actual task performance time. However, we note that the intercept of the regression line is quite a bit higher than that predicted by the original model proposed by Kopper et al. (2010). We believe that this is due to the nature of how errors were considered in the two experiments. In Kopper's experiment, errors did not invalidate a trial, so participants could be more careless when trying to select a target, as they could click multiple times to achieve the selection. In our case, on the other hand, the whole trial had to be attempted again, so we believe participants were more careful and certain that the cursor was inside the target area before they clicked. This resulted in a higher minimum time to complete a trial. The slope of the regression line is quite similar (0.028 in the original model and 0.037 in our experiment), and the correlation coefficient \((R^2)\) is as high as 97.5%, which provides evidence that the distal pointing model was valid in our experimental environment.

We can analyze the SQUAD pointing trials based on the time it took for each of the phases, which consisted of sphere-casting followed by two, three or four refinements. Overall, as we predicted, the selection time is consistent with the exponential model proposed for the relationship with the number of refinements, as shown in Figure 14. This is different than the
linear relationship presented by Kopper et al. (2011), since the visual search time is revised in this paper. Notice that the growth is exponential (although the exponential growth is quite gradual) and the intercept is very close to one, which would indicate the time for the first phase of SQUAD and the increase in the visual search time as more refinements are needed.

Interestingly, there was a significantly longer time for the quad menu selection in the first refinement step of the high-density distractors condition, in which there were a total of 256 objects in the quad menu. The difference was on the order of 0.2s longer than in any other refinement phase, which were all within 0.05s. The conclusion we derive from this is that visual search time was only meaningful when there were a very large number of objects in the quad menu, while in all other conditions, this time was negligible. Because of this, we believe that when there are three or fewer refinements, the relationship between selection time and number of refinements should be linear. Additionally, the average initial selection in the empirical study was $c \approx 0.51s$, which means that the intercept of the regression represented in Figure 14 is not satisfying. To correct these issues, Figure 15 shows a model

$MT = 1.7443 + 0.037ID_{DP}$

$R^2 = 0.97507$
that combines both linear and exponential relationships between the selection time and the number of refinements. The intercept, in this case, is closer to the results of the study, and allows us to compare them with the predicted times. While our assumption that $c = 0.5$ was coincidently close to what $c$ is, the total visual search time is smaller than we predicted. For two refinements, $v_R$ was approximately $1.94 - 3 \times 0.51 \approx 0.41$ s, compared to the predicted 0.5 s. For three refinements, $v_R$ is approximately $2.66 - 4 \times 0.51 \approx 0.62$ s, compared to the predicted 0.9 s. Finally, for four refinements, $v_R$ is approximately $3.74 - 5 \times 0.51 \approx 1.19$ s, compared to the predicted 1.4 s.

5. Evaluation 2 - Zoom Techniques

We conducted a second experiment comparing the zoom techniques to standard ray-casting. We evaluated the task of pointing at circular targets that varied in radius, but without distractors on the screen, since object density should not have any effect on zoom technique performance.
5.1. Goals and Hypotheses

The main goal of the second experiment was to explore the trade-off between ray-casting and both zoom techniques. Similar to SQUAD, the zoom techniques divide the task of selection into multiple steps to reduce the required precision. A secondary goal was to evaluate the effect of user strategy on performance with the zoom techniques, since the amount of zoom used could have a major effect on both speed and accuracy. A small amount of zooming might reduce task completion time, but might also result in more errors if the zoomed-in target size is not large enough. A large amount of zooming would eliminate errors due to the large target size, but would also increase task completion time. We believed that the optimal strategy for the zoom techniques was to zoom in only until the target size was just large enough to make the distal pointing task trivial (i.e., reduce IDDP to a value fairly close to zero).

With respect to speed, we expected both zoom techniques to have performance similar to ray-casting when the user’s strategy was close to optimal. We also hypothesized that users who zoomed in farther would have fewer er-

\[ MT_{R=3} = 0.5102 + 0.7157R \]
\[ MT_{R\geq 3} = 1.002e^{0.3283R} \]
\[ R^2 = 0.99935 \]
rors but would also perform slower, and that users who zoomed in less would have more errors.

We expected the zoom techniques to result in virtually no errors, just as SQUAD did, since they both significantly decrease required pointing precision if the user chooses to zoom in far enough. If they choose to not take advantage of zoom, or use it very little, we would expect a large number of errors.

5.2. Design

Similar to the first experiment, we used a factorial within-subject design with repeated measures. There were three independent variables: technique (ray-casting, discrete zoom or continuous zoom), target size (small, with radius 0.01 m or 0.26°; medium, with radius 0.02 m or 0.53°; and large, with radius 0.04 m or 1.06°), and the strategy, used only with the zoom techniques (small amount of zoom, where users zoom until the target has radius of 0.04 m; large amount of zoom, where users zoom until the target has a radius of 0.16 m, and free zoom). The first two levels of the strategy variable represent “forced” strategies, allowing us to evaluate the impact of particular zoom strategies in a controlled way, while the third level of this variable allowed users to zoom by any amount, or not at all, allowing us to observe what strategies participants used instinctively. Thus, the design was (2x3 + 1)x3.

The order of presentation of technique and strategy was counterbalanced, blocked by technique, such that each participant performed all strategy conditions within the same technique before moving to the next one. The free strategy condition was always the last one used for each technique, since we wanted to first teach and enforce the strategies to the users and then observe which one was used when they had the choice. Within the combinations of technique and strategy, each of the three conditions of target size was repeated eight times and presented in random order.

5.3. Analytic Evaluation

We used the same concepts and models explained in section 4.3 to analytically evaluate performance in our experimental conditions. The first step was to determine the index of difficulty based on the different conditions described in the previous section.

The goal of the zoom techniques and progressive refinement in general is to reduce the index of difficulty (described in section 4.3) so that accuracy can be increased. To achieve this goal, the zoom techniques increase the angular
width of the target ($\omega$) by expanding the region of the screen containing it. To illustrate this, we demonstrate the index of difficulty for the final selection of the small target with three different values of $\omega$: no zoom at all (equivalent to ray-casting), 4x zoom (equivalent to the “small amount of zoom” strategy), and 16x zoom (equivalent to the “large amount of zoom” strategy).

Since the targets were placed within a constant range from the starting point, the movement amplitude was selected randomly between 10.0° and 17.9°, with an average $\alpha$ of 14.0°. With the zoom techniques, however, this value depends on where the user zooms into and how distant the target ends up from the cursor. For this example, we will use a constant value of 14.0° to be consistent with the previous experiment. For the condition where no zoom is applied to select the small size target, $\omega$ is 0.53°. This yields an $ID_{DP}$ of

$$ID_{DP} = \left[ \log_2 \left( \frac{14.0}{0.53} + 1 \right) \right]^2 \approx 42.9.$$  \hfill (6)

Thus, the predicted time for selection using MT defined by Kopper et al. (2010) is approximately 2.29s. For the 4x zoom condition, $\omega$ is 2.12°, and the $ID_{DP}$ is

$$ID_{DP} = \left[ \log_2 \left( \frac{14.0}{2.12} + 1 \right) \right]^2 \approx 1.7.$$  \hfill (7)

Since the index of difficulty is much smaller, the predicted time is reduced to approximately 1.14s. For the 16x zoom condition, $\omega$ is 8.48°, and the $ID_{DP}$ is

$$ID_{DP} = \left[ \log_2 \left( \frac{14.0}{8.48} + 1 \right) \right]^2 \approx 0.001.$$  \hfill (8)

Therefore, the predicted time for selection is approximately 1.09s.

In the case of continuous zoom, the index of difficulty for the final selection is typically smaller, since users first point roughly to the area containing the target and continuously adjust the position of the cursor to match the position of the target. This would yield a similar index of difficulty to the index achieved in the first phase of SQUAD. When users finally reach the desired zoom level, 16x for example, $\omega$ will still be 8.48°, but $\alpha$ will be reduced since the cursor will closer to the target.
In the case of refinement steps with the discrete zoom technique, \( \omega \) will be 45° until the user reaches the desired zoom level (similar to the quad menu phase with SQUAD). This should make the index of difficulty for selection very close to zero, and the expected time for selection to reduce.

Since the zoom techniques increase the number of steps needed for selection, we need to sum the predicted times for each step to predict the total time to select a target using each one of the techniques. For the discrete zoom technique, each time the user zooms in, he must point at a quadrant of the display, wait 0.25 seconds for the zooming animation, and visually search for the target after the zooming is complete. After the final zooming step, the user must use ray-casting to point to the target. The time to point to a screen quadrant is \( c \), which is an empirically determined constant (difficulty tends to zero since \( \omega \) is 45°). We refer to the visual search time as \( v_i \), suggesting that the amount of visual search time after each refinement may decrease since the target is larger and the user should be able to predict its location. The final ray-casting step is denoted as \( MT_{RC} \), and depends on the zoom level, the original target size, and the final distance from the cursor to the target. Thus, if the number of refinements is denoted as \( R \), we hypothesize that the time it takes for selection of a target is:

\[
MT_{DISCRETE} = \sum_{i=1}^{R} (0.25 + c + v_i) + MT_{RC}(z \cdot \omega, \alpha),
\]

expressed in seconds. Different parameters are used for the calculation of the index of difficulty in \( MT_{RC} \); \( \omega \) is multiplied by the zoom factor, and \( \alpha \) is determined by the distance of the cursor to the target.

To calculate the number of refinements \( R \), we need to take into consideration the amount of zoom used. Since it doubles each time the user selects a quadrant, \( R \) can be defined as

\[
R = \log_2 (z),
\]

where \( z \) is the final zoom factor used.

For the continuous zoom technique, we take into consideration the following steps: pointing at the general region containing the target, zooming in, and making the final selection of the target using ray-casting. The time for the first step is \( c \), which is an empirically determined constant related to the time it takes to point at a target whose difficulty tends to zero. The second step should take \( R \) seconds, where \( R \) is also defined as the logarithm
(base two) of the zoom level (it takes one second to double the zoom level; we refer to this as R to allow comparison with the number of refinements in the discrete zoom technique). The time for the final step is \( MT_{RC} \), which is the time it takes to do the final adjustment to the cursor and select the target with ray-casting, based on the zoom level, the original target size, and the distance from the cursor to the target after zooming. The final equation for predicting the time it takes to select a target is:

\[
MT_{CONTINUOUS} = c + R + MT_{RC}(z \cdot \omega, \alpha),
\]

expressed in seconds. To calculate \( MT_{RC} \), we also multiply \( \omega \) by the zoom factor (since the target size will increase by this factor), and include a smaller \( \alpha \) (since the cursor will be very close to the target) for the calculation of the index of difficulty used in \( MT_{RC} \).

It is interesting to note here that, differently than SQUAD, the performance of both zoom techniques depends on target size. Not only will target size determine the amount of zoom users choose to employ and therefore increase the time spent zooming, but ray-casting is also used for the final selection. As opposed to pointing at a large region of the screen to select, users have to point to the target directly. This means that, even though it is possible to make the target as big as the screen with these techniques, doing so would greatly increase the time spent zooming. This is the main trade-off of these techniques, and we created the two enforced strategies for the empirical evaluation in order to find an appropriate balance between accuracy and time.

Based on the \( ID_{DP} \) calculations above, the predicted time to complete the ray-casting tasks for each of the respective target sizes was 2.29s, 1.47s and 1.14s, respectively. The predicted performance with the zoom techniques depends on the target size and zoom level (which is determined by the target size and the strategy). For example, with small targets, 4x zoom, \( \alpha = 14 \) and \( \omega = 2.12 \), the continuous zoom technique would be predicted to take \( c + \log_2 4 + 1.14 \) seconds, and the discrete zoom technique would be predicted to take \( (0.25 + c + v_1) + (0.25 + c + v_2) + 1.14 \) seconds. As in the evaluation of SQUAD, the value of \( c \) needs to be empirically determined, but we expect it to be less than one. Table 2 presents the predicted times assuming \( c = 0.5 \) for both techniques, and constant \( v_n = 0.5 \) for the discrete zoom. We expect the \( MT_{RC} \) time to be very close to zero for the continuous zoom technique, since the cursor will already be positioned on the target. However, we use
the smallest amount of time predicted by $MT_{RC}$ (1.09s).

Table 2: Predicted times for ray-casting using Kopper et al.’s predictive model and for the zoom techniques, with different target sizes and zoom strategies.

<table>
<thead>
<tr>
<th>Target Size</th>
<th>Ray-casting</th>
<th>Strategy (zoom factor)</th>
<th>Continuous</th>
<th>Discrete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>2.29s</td>
<td>Small (4x)</td>
<td>3.59s</td>
<td>3.64s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Large (16x)</td>
<td>5.59s</td>
<td>6.14s</td>
</tr>
<tr>
<td>Medium</td>
<td>1.47s</td>
<td>Small (2x)</td>
<td>2.59s</td>
<td>3.39s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Large (8x)</td>
<td>4.59s</td>
<td>4.89s</td>
</tr>
<tr>
<td>Large</td>
<td>1.14s</td>
<td>Small (1x)</td>
<td>1.14s</td>
<td>1.14s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Large (4x)</td>
<td>3.59s</td>
<td>3.64s</td>
</tr>
</tbody>
</table>

In contrast to the SQUAD study, all predicted times here indicate that the zoom techniques will be slower than or the same speed as ray-casting. They also indicate that our hypothesis that continuous zoom is faster than discrete zoom may be correct. We also believe $MT_{RC}$ to be smaller in the case of continuous zoom, since users will already have the cursor near the target once they reach the desired zoom level, and this is not reflected in the predicted times. To validate these predictions, we performed an empirical evaluation.

5.4. Empirical Evaluation

In order to empirically validate the results from our analytic evaluation, we performed a comparative study of the discrete zoom, continuous zoom and standard ray-casting techniques.

5.4.1. Apparatus

We used a back-projected VisBox-SX system, with only one projector (monoscopic) to display the experimental environment on a 2.29m x 3.05m screen. The resolution of the graphics was 1400px x 1050px. A wireless Intersense IS-900 Wand was used for controlling the cursor on the screen. This was the same hardware setup used in the first experiment.

The experimental software was written using the Vizard Virtual Reality Toolkit by WorldViz. It ran under Microsoft Windows XP on a workstation.
with an Intel Core2 Duo 6600 CPU at 2.40GHz and 2GB of RAM. The frame rate was fixed at 60 frames per second for all conditions.

The environment consisted of a circular target on top of a regular grid as shown in Figure 16. The user stood at the center of an invisible sphere of 2.155m radius, at an orthogonal distance of 1.52m to the display surface. The red target was placed on the surface of the sphere to keep the angular width of objects displayed far from the center of a flat screen constant (Kopper et al., 2010). As in the first experiment, no head tracking or virtual navigation was used and the viewpoint remained at a fixed location. The grid in the background was used to create a context for selection and increase awareness of the zoom level. We did not include distractors in this experiment, since our previous study showed that distractors do not affect ray-casting performance.

![Figure 16: Experimental setup with the discrete zoom technique.](image)

Both target position and cursor position were determined in the same
way as in the first experiment. The target position was randomly selected from a list of candidate targets that fell inside a torus-shaped section of the display sphere, and was limited by a small radius of 0.52\textit{m} and a large radius of 0.77\textit{m}. The cursor position was determined by a function of the yaw and pitch of the IS-900 wand and the display’s field of view.

Similar to the first experiment, each task began with the background and only two objects on the screen: a large yellow circle in the center, and the red target. Once the user clicked the large yellow object, it disappeared and only the background and the target remained. Users could only use the techniques after selecting the yellow circle and starting the task.

For all conditions in this experiment, a crosshair represented the cursor and the task was finished when the user clicked the trigger button on the IS-900 wand after meeting the task requirements for selection. When the cursor intersected with an object, that object was highlighted with a yellow border.

In the discrete zoom condition, after the user clicked on the yellow object in the center of the display to begin the task, the quadrants used for zooming were shown. The quadrant containing the cursor was highlighted, indicating that it would be the one used if zoom was activated. Moving the wand’s joystick forward and then releasing it to its neutral position caused a zoom action, in which the system expanded the highlighted quadrant with an animation (that lasted for 0.25s) until it filled the entire screen, using the approach described in Section 3.2.1. Moving the joystick backward and releasing it caused the view to zoom out to the previous view.

In the continuous zoom condition, after clicking on the yellow object, the user could hold the wand’s joystick forward in order to zoom in at a fixed rate in the direction of the cursor, doubling the amount of zoom for every second the joystick was pressed. Holding the joystick backward would result in zooming out at the same rate towards the original view frustum.

For both zoom techniques, in the enforced strategy conditions, the user was required to zoom in toward the target until the required zoom level was achieved, at which time the system would play a sound. Users were not able to select the target and the target was not highlighted until the required level of zoom was achieved. On the other hand, in the free strategy condition users were able to select the target at any zoom level, and were able to zoom in until the smallest target was as large as the screen.

We applied the same dynamic recursive low-pass filter (Vogel and Balakrishnan, 2005) to the raw pitch and yaw data from the IS-900 wand as in the
first experiment. This filter provided a rapid response time while reducing tracking jitter to a minimum.

5.4.2. Participants

We recruited 24 voluntary unpaid participants from the campus community to perform the study and achieve a fully counterbalanced design. Participants’ ages ranged from 19 to 31 years old, with a median age of 24. Eight of the participants were female.

5.4.3. Procedure

Upon arrival, participants were greeted by the experimenter and given an informed consent form to read and sign. They were then given a colorblindness screening test and proceeded to complete a background questionnaire. After that, they were shown the experimental setting and started learning the first technique/strategy combination. The learning was done with an easy condition so they could understand the technique without making errors. They were then given a practice session, in which they had to practice all three target sizes in the current condition, with eight trials per target size.

After practicing, they were reminded that they had to perform the trials as quickly as possible while trying not to make errors, and then performed another eight sets of each of the three target sizes. When errors were made, the application displayed a message (“Not quite!”) for 0.7 s and the next task was displayed. The erroneous trial was then put into an array of trials that was presented in a new random order after the end of the set of trials for the current technique/strategy combination. This process was repeated to a maximum of five attempts per trial. If the user made five errors on a trial, it was deemed failed and was not presented again. The target position was the same for all attempts of a given trial.

After the end of each technique/strategy session, participants rested for up to two minutes if necessary, then moved on to the next condition. At the end of each technique session, participants completed a set of rating-scale questions and were required to rest for two minutes. They then moved on to the next set of conditions for another technique, following the same protocol, until all three techniques were completed.

Finally, the participant filled out a post-hoc questionnaire, comparing both techniques overall and in light of the other variables.
5.4.4. Results

We performed factorial ANOVAs with repeated measures on the dependent variables: time to complete a task, mean number of errors per trial and zoom strategy in the free zoom condition for the zoom techniques. For time and errors, we first analyzed the independent variables technique and target size, with seven levels of technique (ray-casting, continuous-small \(^1\), continuous-large \(^2\), continuous-free \(^3\), discrete-small \(^4\), discrete-large \(^5\), and discrete-free \(^6\)) and three levels of target size (small, medium, and large).

To analyze the zoom techniques and the different strategies we analyzed the independent variables technique, target size and strategy, with two levels for technique (discrete zoom, and continuous zoom), three levels of target size (small, medium, and large), and three levels of strategy (small amount of zoom, large amount of zoom, and free zoom). The reason for dividing the analysis of time and errors into two parts is that ray-casting only has one strategy, and therefore should not be part of the analysis of strategy.

Finally, to analyze the amount of zoom used in the free zoom strategy, we looked at the independent variables technique and target size, with two levels for technique (continuous-free, and discrete-free) and three levels of target size (small, medium, and large).

**Time.** The first analysis of time compares ray-casting to the zoom techniques, and considers each combination of technique and strategy as a separate technique. Overall, there was a main effect of technique \((F_{6,138} = 106.206, p < .001)\). Ray-casting was significantly faster than discrete-small \((F_{6,18} = 101.2, p < .001)\), discrete-large \((F_{6,18} = 101.2, p < .001)\), discrete-free \((F_{6,18} = 101.2, p = 0.001)\), continuous-large \((F_{6,18} = 101.2, p < .001)\), and continuous-free \((F_{6,18} = 101.2, p = 0.021)\). Continuous zoom was significantly faster than discrete (for continuous-small and discrete-small, \(F_{2,22} = 64.276, p = 0.006\); for continuous-large and discrete-large, \(F_{6,18} = 101.2, p < .001\); and for continuous-free and discrete-free, \(F_{6,18} = 101.2, p = 0.024\)). This supports our prediction that ray-casting is generally faster than the

---

\(^1\) continuous zoom with the small amount of zoom strategy
\(^2\) continuous zoom with the large amount of zoom strategy
\(^3\) continuous zoom with the free zoom strategy
\(^4\) discrete zoom with the small amount of zoom strategy
\(^5\) discrete zoom with the large amount of zoom strategy
\(^6\) discrete zoom with the free zoom strategy
zoom techniques, and confirms our hypothesis that the continuous zoom technique is faster than the discrete technique.

There was also a main effect of target size ($F_{2.46} = 136.075, p < .0001$). As expected and predicted by the distal pointing model, pairwise comparisons showed that participants took significantly longer to select the small targets when compared to medium ($F_{2.22} = 952.6, p < .001$) and large targets ($F_{2.22} = 952.6, p < .001$), and medium targets when compared to large targets ($F_{2.22} = 952.6, p < .001$). By examining the interactions between all these variables, we are able to better understand their effects.

There was a significant interaction between technique and target size ($F_{12.276} = 15.12, p < .001$). As indicated previously, for all techniques, participants performed significantly slower as the targets got smaller. Interestingly, the smaller the targets are, the greater are the differences between the mean times of the techniques, as can be seen in Figure 17. With small targets, the differences between ray-casting, continuous-free and discrete-free are 0.577s and 1.093s respectively, while with large targets, the differences are 0.045s and 0.396s. This is expected, since users spend more time zooming with smaller targets.

![Figure 17: Interaction between technique and target size. The error bars represent standard error.](image)

Finally, it is clear from the graph in Figure 17 that ray-casting was significantly faster than both zoom techniques, and that continuous zoom was faster than discrete zoom with all target sizes. This confirms our hypothesis that the continuous zoom technique is faster than the discrete zoom technique. However, it also shows that ray-casting is faster for all conditions.
Despite both zoom techniques being slower than ray-casting, they are significantly more accurate, as shown in the next section.

In order to understand the performance of the zoom techniques, we analyzed the effects of strategy on selection time. We found a main effect of zoom strategy ($F_{2,46} = 213.326, p < .001$), with significant differences between the large amount of zoom strategy and both the small amount of zoom ($F_{2,22} = 316.8, p < .001$) and free zoom strategies ($F_{2,22} = 316.8, p < .001$). There were no significant differences between the small amount of zoom and free zoom strategies. This indicates that the effects of using the large amount of zoom strategy on both zoom techniques were large enough to create significant differences overall.

Additionally, there were significant interactions between strategy and target size ($F_{4,92} = 19.728, p < .001$), and between strategy, technique and target size ($F_{4,92} = 3.045, p < .021$). Figure 18 illustrates how different strategies affect performance for different target sizes. Similar to the interaction between technique and target size, the difference between the large amount of zoom strategy and the other strategies is smaller with small targets, and there are no differences between the small amount of zoom and free zoom strategies. Figure 19 shows how strategy affected performance of both techniques, and how more zoom results in more time. In addition, continuous zoom is constantly faster than discrete zoom, even when more zoom is used. No other significant interactions were found, which is consistent with our hypotheses.

**Errors.** Similar to the analysis performed for time, the first analysis of errors compares ray-casting to the zoom techniques, and considers each strategy as a separate technique. Target size ($F_{2,46} = 16.84, p < .001$) and technique ($F_{6,138} = 22.204, p < .001$) both showed significant effects for the number of errors. Small targets induced more errors than both medium ($F_{2,22} = 10.873, p = 0.001$) and large targets ($F_{2,22} = 10.873, p < .001$), but there were no significant differences in the number of errors between medium and large targets. Both zoom techniques performed significantly better than ray-casting with all strategies, but there was no significant difference between the zoom techniques. The analysis of the interactions between the variables will allow us to understand better the effects of each of them on the number of errors.

There was a significant interaction between target size and technique ($F_{12,276} = 14.841, p < .001$), and pairwise comparisons show how the tech-
Figure 18: Interaction between strategy and target size. The error bars represent standard error.

Technique affected performance in relation to each target size. In this case, the only differences that are significant are within ray-casting, with mean errors of 0.521 per trial with small targets, 0.11 with medium targets and 0.052 with large targets. No significant differences were found among the zoom technique conditions, and the mean error per trial is virtually the same for all sizes of targets and techniques (around 0.028 errors per trial). The number of errors with ray-casting alone, then, was large enough to cause the main effects of technique and target size. The data also confirm the hypothesis that the zoom techniques would present virtually no errors, as can be seen in Figure 20.

It is clear from this graph that ray-casting caused significantly more errors than zoom techniques for medium and small targets, and that there are no differences between the zoom techniques. This confirms our hypothesis that the zoom techniques are more accurate than ray-casting, and that the zoom techniques are precise enough that target size does not significantly affect performance. However, with this analysis it is not possible to say whether strategy affects the number of errors. To evaluate that, we performed the second analysis without ray-casting.
Interestingly, this second analysis indicated a main effect of strategy \( (F_{2,46} = 14.676, p < .001) \). As can be seen in Figure 21, the large amount of zoom strategy provided more accuracy than the others for all sizes of targets. The small amount of zoom and free zoom strategies provided the same accuracy as ray-casting when used with a large target, but for all target sizes.

There were no other significant interactions for the number of errors.

**User Preference.** Participants were asked to rank the techniques based on preference, both overall and when the targets were small, and were asked to rank the strategies overall. The continuous zoom technique was preferred for both conditions by 17 participants. The other 7 participants preferred the discrete zoom technique for both conditions. For second choice, zoom techniques were still preferred both overall (6 participants ranked continuous second; 12 participants ranked discrete second; and 6 participants ranked ray-casting second) and when targets are small (7 participants ranked continuous second; 16 participants ranked discrete second; and 1 participant ranked ray-casting second). Thus, just as in the first experiment, not a single participant preferred ray-casting over the zoom techniques, even when
taking into consideration larger targets. For strategy, 17 participants preferred the use of a small amount of zoom, and 7 preferred the use of a large amount of zoom.

For the subjective ratings, participants were instructed to fill out a survey immediately after completing each of the techniques, and to rate the techniques on a seven-point scale for ease of learning, ease of use, the difficulty of using each the techniques when the targets were small, and the preferred strategy for the zoom techniques. Participants were always instructed to respond to questions about a technique independently of the answers of the others.

We performed Wilcoxon Signed Rank tests on each of the questions. For ease of learning, discrete zoom \((mdn = 1.5)\) was ranked significantly more difficult than continuous zoom \((mdn = 1)\) \((z = 2.78, p = 0.0054)\). There were no significant differences for ray-casting \((mdn = 1)\) compared to discrete zoom \((n = 14, W = 61, p = 0.0578)\) and to continuous zoom \((n = 9, W = 18, insignificant)\). For ease of use overall, no significant differences were found between the zoom techniques \((n = 17, W = 33, p = 0.44)\),

Figure 20: Interaction between technique and target size. The error bars represent standard error.
but both discrete \((mdn = 2) (z = 3.92, p = 0.0001)\) and continuous zoom \((mdn = 2) (z = 3.64, p = 0.0003)\) were considered significantly easier than ray-casting \((mdn = 4)\). Similarly, for ease of use with small targets, there were no significant differences found between the zoom techniques \((n = 18, W = 13, p = 0.78)\), but both discrete \((mdn = 4) (z = 3.98, p = 0.0001)\) and continuous zoom \((mdn = 4) (z = 4.05, p = 0.0001)\) were considered significantly easier than ray-casting \((mdn = 6.5)\). Finally, there was no significant difference in the preferred strategy between the zoom techniques \((n = 7, W = 4, insignificant)\), and most users preferred the small amount of zoom strategy.

Strategy. We analyzed the amount of zoom participants used to perform selection with the zoom techniques in the free zoom condition. Overall, both target size \((F_{2,46} = 119.6, p < .001)\) and technique \((F_{1,23} = 4.767, p = 0.039)\) had a significant effect on the amount of zoom used. As can be seen in Figure 22, smaller targets require more magnification, as expected. However, when using the continuous zoom technique, users use significantly less zoom than with the discrete zoom technique. Even though this means that our
hypothesis about consistent strategies when using both techniques was not supported, it is an interesting result by itself since the different designs of the two zoom techniques influenced how much zoom participants used. This could also help explain the difference in the time taken when using the zoom techniques. To understand this issue, we looked at the pairwise comparisons of the different conditions.

For target size, there were significant differences between all sizes ($p < .001$), with more zoom being used for the small target, with a mean zoom factor of 5.08$x$, a little less for medium targets, with a mean zoom factor of 2.572, and almost no zoom for large targets, with a mean zoom factor of 1.509. Analysis per technique showed that participants used less zoom with the continuous zoom technique than with discrete zoom, with mean zoom factors of 2.512 and 3.716.

We found a significant interaction between target size and technique ($F_{2,46} = 119.6, p < .001$). Even though users used more zoom for smaller targets with both techniques, the slope for the continuous zoom is smaller (Figure 22).

Figure 22: Interaction between technique and target size. The error bars represent standard error.
5.5. Model Validation

As we did for experiment 1, we used information from both analytical and empirical evaluations to validate the predictive models we developed for the techniques. We first performed an analysis of the ray-casting data. Figure 23 illustrates the linear regression of each ray-casting condition and their corresponding $ID_{DP}$ based on the distal pointing model to the actual task performance time. The intercept of the regression line is also higher than the original model by Kopper et al. (2010), but slightly smaller than the one in experiment 1. The slope of the regression line is quite similar (0.028 in the original model, 0.037 in experiment 1, and 0.0319 in this experiment), and the correlation coefficient ($R^2$) is as high as 97.7%, just as in experiment 1. Thus, the distal pointing model was also valid for ray-casting in this experiment.

![Figure 23: Scatter plot and regression line for the ray-casting conditions.](image)

To validate the predictive models for the zoom techniques, we analyzed the strategies “small amount of zoom” and “large amount of zoom”. These conditions ensured that size of the target after zooming is always the same. To achieve this, different amounts of zoom had to be applied depending on the initial size of the target. This is very similar to the concept of the number of refinements for the SQUAD technique.
As can be seen when comparing Figures 24 (small amount of zoom) and 25 (large amount of zoom), the slope of both techniques is smaller when the goal is to have a larger target. With the discrete zoom, we believe this happens because the visual search time reduces since the target gets larger. With the continuous zoom, we hypothesize that users tend to perform zoom all the time, since it takes a long time to get to the maximum zoom. Additionally, the time it takes for users to perform two refinement steps changes between the two strategy conditions. While using the small amount of zoom strategy, users took 3.43s and 4.039s with the continuous and discrete techniques, respectively. On the other hand, with the large amount of zoom strategy they took 3.048s and 3.374s. The reason for that is the reduced IDDP when the goal is to make the targets larger, which is consistent with our prediction and again shows that the final step must be faster.

Another interesting observation is while both techniques have a linear increase in time as the number of refinements grows, the slope for continuous zoom is smaller than for discrete zoom. This was predicted by our model, and is due to the fact that, with the continuous zoom, users point roughly...
to the position of the target and the final step is only an adjustment in the
cursor position, with a much smaller $ID_{DP}$. This means that the user takes
more time to zoom and less time to select the target the larger it gets.

Finally, even though we only estimated the parameters for predicting the
time each technique would take, the predictions are close to the final results.
The slopes are similar to what we predicted, and as we suspected, the time
for final selection with the continuous zoom technique tends to zero. As the
number of refinements grows, the smaller the $c$ and $MT_{RC}$ parameters are.
As can be seen in Figure 25, with 2 refinements, users have to spend 2s
zooming and, therefore, spend 1s performing the other steps. With the 4
refinements condition, users have to spend 4s zooming, and therefore around
0.25s selecting the target. Since users could move the zoom window while
zooming, part of the final pointing step was performed as they zoomed, and
since the target was much bigger, they could simply click once they got to
they maximum zoom level, without the need for cursor adjustments.
6. Discussion

Our study of selection techniques based on progressive refinement highlights an important tradeoff. Immediate techniques offer rapid selection at the expense of high required precision, while progressive refinement techniques offer minimal precision requirements at the expense of gradual refinement. Our analytic and empirical evaluations have verified this tradeoff with respect to selection time, although our progressive refinement techniques were faster than or equivalent to ray-casting for very difficult selection tasks. In terms of accuracy, however, progressive refinement techniques were clearly superior. Interfaces based on techniques (such as our zooming techniques) that allow the user to choose immediate selection for easy tasks and progressive refinement for difficult tasks should provide near-optimal speed and accuracy.

The analytical evaluation of SQUAD selection was backed up by the results of an empirical study comparing it to standard ray-casting. We verified that the there is, indeed, a performance tradeoff between immediate techniques that use one precise action to select an object and progressive refinement techniques that require very low precision at the expense of multiple steps. The use of SQUAD, and, by extension, other progressive refinement selection techniques, should be based on a consideration of this tradeoff. We found that SQUAD is significantly faster for selection of small objects and selection in low-density environments. When errors are considered, the case for SQUAD is even stronger, as it achieved near-perfect accuracy. A positive aspect of our approach is that the increase in the number of refinements needed grows very slowly, such that the task is not likely to take many refinement phases. Another interesting aspect of SQUAD is that the time to complete a task grows linearly as the number of objects increases exponentially up to 3 refinements, whereas with standard ray-casting, the time increase is always exponential as targets undergo a linear decrease in size.

While SQUAD was highly efficient with near-zero error rates and better time than standard ray-casting with small targets, it has some limitations that need to be acknowledged. SQUAD was designed with a particular application setting in mind, and the nature of the task, which involved distant objects roughly arranged on a surface, influenced the design of SQUAD. Its sphere-casting component is not well-suited for selecting from among items distributed in depth. However, SQUAD can be adapted to work well in such situations. For example, instead of a selection sphere, a cone or cylinder could
be used to specify a deeper region of initial selection for further refinement in the quad menu phase.

In addition, small improvements can be made to increase performance in terms of time by combining other enhancement techniques with the progressive refinement techniques. Changing SQUAD, for example, to allow users to select objects directly in the quad menu phase instead of requiring them to keep selecting quadrants could reduce the time to complete the task without affecting accuracy. Additionally, as seen in the analytical evaluation of SQUAD, the number of refinement steps determines the time it takes to select an object. Finding a way to reduce the number of objects in the sphere-casting phase of SQUAD without affecting the accuracy or the time to perform sphere-casting would be ideal. We think that the use of a dynamically sized sphere (such as the bubble cursor) to include fewer objects in the initial selection would achieve this goal.

SQUAD works well for tasks that require the selection of objects that are visually distinct from other possible objects in the vicinity, and that do not depend on the spatial context. However, SQUAD is not appropriate for selection tasks that depend on object location rather than visual features. In order to solve this issue, we designed two techniques based on the idea of zooming. In this case, the refinement consists in the specification of an area in the view that zooms to fill the display, therefore decreasing the number of selectable objects and increasing the angular width of the target. As shown in Section 3.2, this is not equivalent to navigating close to objects to select them, since all items within the visual region remain in view while zooming, and after the selection task the view frustum returns to its original size and shape.

The analytical evaluation of the zoom techniques showed how each of the refinement steps affects performance, and the empirical evaluation results supported these claims. Just as with SQUAD, we found that there is a trade-off between the number of refinements and the time to complete the tasks. Making the target larger, for example, with the “large amount of zoom” strategy, does increase accuracy, but it also increases the time to perform selection considerably. Users need to keep this in mind when using these zoom techniques; if they zoom in too far, they will end up taking a lot more time. The “small amount of zoom” strategy seemed to provide the best balance between time and accuracy, and it is similar to the one used by the majority of users when they were allowed to choose their own strategy.

Overall, we demonstrated that the continuous zoom technique provides
better performance in terms of time, and as we predicted, accuracy with the zoom techniques is always better than with ray-casting, no matter the strategy used. We found that target size affects the time to complete selection with both zoom techniques; the smaller the target, the more time it takes since the user has to perform more zooming. However, unlike SQUAD, the number of distractors does not affect the zoom techniques. Therefore, while SQUAD can be recommended for situations that require selection of very small targets, the zoom techniques may be a better fit for highly cluttered environments.

Interestingly, some users commented that they preferred the conditions in which they were forced to follow a specific strategy to the ones in which they were free to choose the amount of zoom they needed. The reasoning behind this was that they would not have to spend time trying to determine if the zoom level was sufficient for selection of a specific target. This is one of the differences between the refinement steps with SQUAD and the zoom techniques. While the user is forced to refine until there is only one object in the quadrant with SQUAD, with both zoom techniques the user can choose to not zoom at all or make the target as large as the screen.

Another comment made by some users about the continuous zoom technique is that, even though they tried to point to the target as they were zooming, they could not keep the zoom window in the position that they wanted. Because of this, some users preferred not to point directly at the target, and instead focus on controlling the zoom window. This demonstrates that the continuous zoom technique is affected by similar issues as ray-casting, but in a different way. For instance, while jitter causes ray-casting to be less accurate, it only affects the time for the continuous zoom, since the cursor may not be on top of the target once the desired zoom level is reached.

Improvements can be made to the zoom techniques such that performance in terms of time is improved and the issues described by the users can be solved. One possible design change is to remove the screen quadrants from the discrete zoom technique, and instead use a zoom window that follows the cursor. This would remove the visual search time from each iteration of the discrete zoom technique and the final adjustment of the cursor would take the same time as with continuous zoom. Compared to the continuous zoom, this would greatly reduce the time it takes to zoom and would also eliminate the precision issues related to controlling the zoom window while zooming.

The main advantage of using progressive refinement techniques is accu-
racy, and such techniques should be used when accuracy is critical. With all three techniques, we achieved near-zero error rates in conditions where ray-casting could not. We also showed that there are certain conditions in which progressive refinement techniques can be faster than regular ray-casting. However, this only happened with SQUAD when there were smaller targets and fewer distractors. An important goal of our future work, therefore, is to modify these techniques, or design new ones, to provide the accuracy benefits without negatively affecting speed.

Finally, we found that the distal pointing model proposed by Kopper et al. (2010) accurately predicted performance with ray-casting (in both evaluations). We also were able to find evidence that the performance of discrete progressive refinement selection techniques can be modeled as a function of the number of refinements necessary for completing a task. The use of analytical models in the evaluation of such techniques can provide a benefit in more realistic settings, in which control can be traded off for ecological validity. In such situations, many aspects, such as user strategy, confound the experimental control, and using reliable analytical models in the performance assessment may be the best choice.

7. Conclusions and Future Work

In this paper, extending the work by Kopper et al. (2011), we have proposed the concept of selection by progressive refinement and described a design space for such techniques. We have designed three progressive refinement techniques, SQUAD, Discrete Zoom, and Continuous Zoom, which allow users to select targets accurately without requiring precise pointing. We evaluated these techniques against standard ray-casting. The results for SQUAD indicate that there is a tradeoff between the number of refinements and the required pointing accuracy that must be taken into account for the design of 3D selection techniques. When the visual size of objects is too small and the density of the environment is not too high, selection can be achieved more efficiently by progressive refinement. The results for both zoom-based techniques indicate that performance can be similar in relation to time with virtually no errors. More importantly, these techniques provide an alternative to SQUAD when the selection task is dependent on the spatial context. Both studies showed that progressive refinement techniques can ensure very high levels of accuracy because they do not require precise pointing.
We plan to continue this research by designing and evaluating additional progressive refinement techniques. There are still several unexplored areas in the design space, and our findings about strategy indicate that it would be interesting to compare how users perform with techniques that enforce strategies, such as SQUAD, and others that allow users to choose their own strategies, such as the zoom techniques in the free zoom conditions. Another planned evaluation will compare the usability of immediate techniques and progressive refinement techniques when navigation of the environment is allowed. In addition, we plan to implement the proposed changes to SQUAD and the zoom techniques, and design new progressive refinement techniques. For instance, by using sphere-casting followed by selection in a call-out menu or an explosion technique, we could provide alternatives to selection and refinement in the spatial context of the target. Another example would be to use focus+context techniques, such as the fish-eye view (Furnas, 1986), to zoom into a region of the screen without eliminating the overall context of the environment. An example that is more distinct from the existing techniques would be to navigate the hierarchical structure of the environment to select objects, instead of selecting them directly in the environment, similar to the navigation technique HiSMAP (Bacim et al., 2009). Finally, we plan to expand our study of progressive refinement techniques to other 3D interaction tasks, such as navigation and manipulation.

References


Figueroa, P., Kitamura, Y., Kuntz, S., Vanacken, L., Maesen, S., De Weyer, T., Notelaers, S., Octavia, J.R., Beznosyk, A., Coninx, K., Bacim, F.,


