

Increasing Selection Accuracy and Speed through Progressive Refinement

Felipe Bacim de Araujo e Silva

A dissertation submitted to the faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
In
Computer Science and Applications

Doug A. Bowman, Chair
Ravin Balakrishnan
Yong Cao
Christopher L. North
Nicholas F. Polys

June 22, 2015
Blacksburg, Virginia

Keywords: 3D selection, 3D user interfaces, progressive refinement

Copyright © 2015 Felipe Bacim de Araujo e Silva
All Rights Reserved

Increasing Selection Accuracy and Speed through Progressive Refinement

Felipe Bacim de Araujo e Silva

Abstract

Although many selection techniques have been proposed and developed over the years, selection by pointing is perhaps the most popular approach for selection. In 3D interfaces, the laser-pointer metaphor is commonly used, since users only have to point to their target from a distance. However, the task of selecting objects that have a small visible area or that are in highly cluttered environments is hard when using pointing techniques. With both indirect and direct pointing techniques in 3D interfaces, smaller targets require higher levels of pointing precision from the user. In addition, issues such as target occlusion as well as hand and tracker jitter negatively affect user performance. Therefore, requiring the user to perform selection in a single precise step may result in users spending more time to select targets so that they can be more accurate (effect known as the speed-accuracy trade-off).

We describe an approach to address this issue, called Progressive Refinement. Instead of performing a single precise selection, users gradually reduce the set of selectable objects to reduce the required precision of the task. This approach, however, has an inherent trade-off when compared to immediate selection techniques. Progressive refinement requires a gradual process of selection, often using multiple steps, although each step can be fast, accurate, and nearly effortless. Immediate techniques, on the other hand, involve a single-step selection that requires effort and may be slower and more error-prone. Therefore, the goal of this work was to explore this trade-off. The research includes the design and evaluation of progressive refinement techniques for 3D interfaces, using both pointing- and gesture-based interfaces for single-object selection and volume selection.

Our technique designs and other existing selection techniques that can be classified as progressive refinement were used to create a design space. We designed eight progressive refinement techniques and compared them to the most commonly used techniques (for a baseline comparison) and to other state-of-the-art selection techniques in a total of four empirical studies. Based on the results of the studies, we developed a set of design guidelines that will help other researchers design and use progressive refinement techniques.

Table of Contents

List of Figures.....	v
List of Tables.....	viii
1 Introduction.....	1
1.1 Motivation.....	1
1.2 Progressive Refinement.....	2
1.3 Definitions.....	4
1.4 Research Questions.....	5
1.4.1 Design.....	5
1.4.2 Theory.....	6
1.4.3 Evaluation.....	6
1.5 Approach Overview.....	7
1.6 Contributions.....	9
2 Related Work.....	11
2.1 Single-Object Selection.....	11
2.1.1 Immediate Selection.....	11
2.1.2 Selection by Progressive Refinement.....	14
2.2 Volume Selection.....	18
2.2.1 Immediate Selection.....	18
2.2.2 Progressive Refinement.....	19
2.3 Discussion.....	20
3 Design and Evaluation of SQUAD: A Single-Object Selection Technique based on Progressive Refinement.....	21
3.1 Design of SQUAD Selection.....	21
3.2 Evaluation 1 – SQUAD vs. Ray-Casting.....	23
3.2.1 Goals and Hypotheses.....	23
3.2.2 Experimental Design.....	24
3.2.3 Analytic Evaluation.....	24
3.2.4 Empirical Evaluation.....	28
3.2.5 Model Validation.....	35
3.3 Discussion.....	37
4 Design and Evaluation of Single-Object, Progressive Refinement Selection through Zooming.....	40
4.1 Design of Zoom Techniques.....	40
4.1.1 Discrete Zoom.....	40
4.1.2 Continuous zoom.....	41
4.1.3 Design Considerations for Different Zoom Implementations.....	42
4.2 Evaluation 2 - Zoom Techniques.....	44
4.2.1 Goals and Hypotheses.....	44
4.2.2 Experimental Design.....	44
4.2.3 Analytic Evaluation.....	45
4.2.4 Empirical Evaluation.....	47
4.2.5 Model Validation.....	57
4.3 Discussion.....	59
5 Improvement and Evaluation of Progressive Refinement Techniques for Single-Object Selection.....	61
5.1 Improved Design of Single-Object Selection Techniques.....	61

5.1.1	Double Bubble.....	61
5.1.2	Flexible Rapid Incremental Zoom (FRIZ).....	65
5.2	Evaluation of Improved Technique Designs	66
5.2.1	Goals and Hypotheses.....	67
5.2.2	Experimental Design	68
5.2.3	Apparatus.....	68
5.2.4	Participants.....	69
5.2.5	Procedure.....	69
5.2.6	Results.....	70
5.3	Discussion	83
6	Design and Evaluation of Gesture-Based Progressive Refinement Techniques for Volume Selection	87
6.1	Leap Motion Controller.....	88
6.2	Designing Point Cloud Selection	89
6.2.1	Techniques for Dividing the Point Cloud.....	89
6.2.2	Technique for Selection of a Point Cloud Subset.....	92
6.3	Evaluation of Point Cloud Selection Technique Designs.....	93
6.3.1	Goals and Hypotheses.....	94
6.3.2	Experimental Design	94
6.3.3	Apparatus.....	95
6.3.4	Participants.....	98
6.3.5	Procedure.....	98
6.3.6	Results.....	98
6.4	Discussion	106
7	Design Space and Guidelines.....	110
8	Conclusions	114
8.1	Contributions and Limitations.....	115
8.2	Future Work.....	116
9	References	120

List of Figures

Figure 1 – Gradualness continuum.	3
Figure 2 – Sphere-casting.....	22
Figure 3 – Quad menu. Note that the target object needs to be visually distinct for the selection to be feasible.	23
Figure 4 – Left: Experimental setup with sphere-casting. Right: quad menu stage of SQUAD in the experiment	29
Figure 5 – Interaction between technique and distractor density. The error bars represent standard error.....	32
Figure 6 – Interaction between technique and target size. The error bars represent standard error.	33
Figure 7 – Mean results for all technique-density-target size combinations. Note that all ray-casting densities are averaged and displayed in a single line, since there was no significant difference among them. The error bars represent standard error.....	34
Figure 8 – Scatter plot and regression line for the ray-casting pointing condition.	36
Figure 9 – Scatter plot and exponential regression line for the SQUAD pointing condition.....	37
Figure 10 – Scatter plot and combination of linear and exponential regression lines for the SQUAD pointing condition.....	38
Figure 11 – Three refinement steps of the discrete zoom technique.	41
Figure 12 – Three snapshots of different zoom levels achieved with the continuous zoom technique.	41
Figure 13 – Issue of magnifying a texture for zooming.	42
Figure 14 – 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.	43
Figure 15 – Experimental setup with the discrete zoom technique.....	48
Figure 16 – Interaction between technique and target size. The error bars represent standard error.	52
Figure 17 – Interaction between strategy and target size. The error bars represent standard error.	52
Figure 18 – Interaction between technique and strategy. The error bars represent standard error.	53
Figure 19 – Interaction between technique and target size. The error bars represent standard error.	54
Figure 20 – Average number of errors per trial with each strategy. The error bars represent standard error.....	55
Figure 21 – Interaction between technique and target size. The error bars represent standard error.	56

Figure 22 – Scatter plot and regression line for the ray-casting conditions.....	57
Figure 23 – Scatter plot and regression line for the continuous zoom and discrete zoom conditions with the small amount of zoom strategy. For the continuous zoom technique, refinement is defined by $\log_2 z$, where z is the final zoom level.....	58
Figure 24 – Scatter plot and regression line for the continuous zoom and discrete zoom conditions with the large amount of zoom strategy. For the continuous zoom technique, refinement is defined by $\log_2 z$, where z is the final zoom level.....	58
Figure 25 – Two selection phases of Double Bubble: Bubble Cursor using screen-space for initial in-context selection (left), and Bubble Cursor used to enhance selection of object in the menu.....	62
Figure 26 – Illustration of the selection depth issue of sphere-casting with first person (left) and top-down view perspectives (right). Multiple objects are occluded by the sphere, as seen on the left image and marked by the white arrows on the right image, but are not selectable because they are not inside the sphere.....	63
Figure 27 – Illustration of the 2D Bubble Cursor implementation in its minimum bubble size, with first person (left) and top-down view perspectives (right). It allows the selection of all visible objects inside the bubble, independent of depth (highlighted on the right).....	64
Figure 28 – Initial implementation of the Double Bubble technique.....	65
Figure 29 – Two refinement steps of the improved zoom technique.....	66
Figure 30 – Interaction between target size and technique for time in abstract-only.....	72
Figure 31 – Interaction between distractor density and technique for time in abstract-only.....	72
Figure 32 – Interaction between target size and technique for errors in abstract-only.....	73
Figure 33 – Interaction between distractor density and technique for errors in abstract-only.....	74
Figure 34 – Average time for completion of tasks in the Abstract-only and Supermarket conditions.....	75
Figure 35 – Interactions between technique and target size in the supermarket conditions.....	76
Figure 36 – Interactions between technique and distractor density in the supermarket conditions.....	76
Figure 37 – Overall mean errors per trial for each technique in the abstract-only and supermarket conditions.....	78
Figure 38 – Interaction between target size and technique for errors in the supermarket conditions.....	78
Figure 39 – Interaction between distractor density and technique for errors in the supermarket conditions.....	79
Figure 40 – Average time to select a target in the supermarket-free condition.....	80
Figure 41 – Average number of errors per trial in the supermarket-free condition.....	81
Figure 42 – Slice technique for dividing point clouds into two half spaces.....	90
Figure 43 – Lasso technique for dividing point clouds into two half spaces.....	91
Figure 44 – Volume Brush technique.....	92
Figure 45 – Swipe to select one of the two point cloud subsets.....	93

Figure 46 – Complete setup used in the evaluation of gesture-based progressive refinement techniques for point cloud selection.	95
Figure 47 – All environments and tasks used in the evaluation of point cloud selection techniques, with target point clouds highlighted in yellow.	96
Figure 48 – Task progression bars and countdown timer during the completion of the tail task.	97
Figure 49 – Interaction between technique and feedback type for time.	99
Figure 50 – Time for completion of different tasks for each technique.	100
Figure 51 – Percentage of extra points left (left) and percentage of goal points left (right) for different feedback types.	101
Figure 52 – Interaction between technique and feedback type for the number of refinements.	103
Figure 53 – Interaction between technique and feedback type for number of mistakes.	104
Figure 54 – Interaction between technique and chosen feedback.	105
Figure 55 – Refinement profile for the Slice technique, stairs task, and closed-loop feedback.	107
Figure 56 – Trend lines of all techniques for the stairs task and closed-loop feedback. Green line represents the trend line for Slice, purple line for Lasso, and yellow line for Volume Brush.	108
Figure 57 – Trend lines of all techniques and feedback types for the stairs task. Green lines represent the trend line for Slice, purple lines for Lasso, and yellow lines for Volume Brush. Solid lines show closed-loop feedback, and dotted lines show open-loop.	108
Figure 58 – Design space of selection by progressive refinement.	110
Figure 59 – Sequence of actions for the two-finger tapping technique. Yellow circles represent fingers touching the screen, and the blue square represents the zoom window preview.	117
Figure 60 – Sequence of actions for the discrete zoom technique in a tablet application.	117
Figure 61 – Setup used for the switch interface and sequence of actions demonstrating the use of a progressive refinement technique combined with the switch interface in a tablet application.	118
Figure 62 – SQUAD used on a tablet.	119
Figure 63 – Example of a focus+context technique gradually increasing the size of a target.	119

List of Tables

Table 1 – Predicted times for ray-casting using Kopper et al.’s predictive model and for SQUAD, with different densities and target sizes.	28
---	----

1 Introduction

1.1 Motivation

Selection, which involves the specification of one or more objects by the user, is one of the fundamental tasks in 3D environments (Doug A. Bowman, Kruijff, LaViola, & Poupyrev, 2004). Selection by pointing is perhaps the most popular selection style in both 2D (interfaces that use input devices with up to two degrees-of-freedom) and 3D user interfaces (interfaces that use input devices with more than two degrees-of-freedom), either by direct or indirect pointing. More specifically, in 3D user interfaces, distant laser-pointer-style pointing is commonly used for direct pointing for the same sorts of tasks. Indirect pointing techniques, such as L-DOP (Debarba, Nedel, & Maciel, 2012) which uses pointing to specify an area of the display and touch to move a precision cursor inside that area, can be used to increase precision.

However, the task of selecting objects that have a small visible area or that are in highly cluttered environments is hard when using pointing techniques. With both indirect and direct pointing techniques in 3D interfaces, smaller targets require higher levels of pointing precision from the user. This usually results in users spending more time to select targets so that they can be more accurate (effect known as the speed-accuracy trade-off). In addition, precision issues can be even more aggravated for selection techniques based on the laser-pointer metaphor like ray-casting. While all pointing techniques are negatively affected by hand and tracking jitter, it is worse for ray-casting since jitter gets amplified at the end of the ray. While users can overcome this issue by navigating closer to the target, either virtually or physically, this approach requires them to leave the initial selection context and spend time navigating until the target is easier to select, and, depending on the task, navigating back to the initial position.

Since selecting objects is usually not the user's primary task, decreasing the precision needed for selection when targets are small is fundamental to improve the overall usability of user interfaces.

Several examples of applications in which hand and tracking jitter make ray-casting unsuitable can be found in the literature. For instance, in the supermarket used in the first IEEE 3DUI Grand Prize (Bacim, Kopper, Leal, Ni, & Bowman, 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 (Kopper, 2011). 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.

Finally, these issues may affect the usability of several other types of applications that can use both 2D and 3D interaction, such as the visualization of and interaction with astrophysical or atomic datasets (Fu, Goh, & Ng, 2010), interaction with large document collections (Bragdon, DeLine, Hinckley, & Morris, 2011), authoring of crowd simulations (Ulicny, Ciechowski, & Thalmann, 2004), detailed information visualizations (Prachyabrued, Ducrest, & Borst, 2011), and interaction with 3D point clouds (Burgess et al., 2015). 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 pointing techniques. Interaction techniques that use area or volume of selection can be significantly more accurate and faster than direct selection when targets are small. However, in highly cluttered environments these techniques still require users to interact very carefully to accomplish a single precise selection, and may actually result in worse performance in some situations. The same applies for techniques that use snapping to objects to assist in the selection task. If there are too many objects, users will have to point carefully to select the target. Other techniques address precision issues by changing the control-display ratio. While these techniques can achieve very high levels of precision, they cannot be used in every situation and can cause a significant mismatch of the physical pointing direction to the perceived pointing position. Finally, these techniques require the user to interact very carefully and with full attention.

While there has been a good amount of previous work that try to provide more accuracy in selection, there is no general solution for easy 3D selection when the target is small or in cluttered environments.

1.2 Progressive Refinement

We describe an approach called Progressive Refinement to address the issues described in the previous section. The idea is to gradually refine the set of selectable objects to reduce the required

precision of the task. In many progressive refinement techniques, a series of rough or imprecise actions can be used to accomplish a precise result, making the overall interaction take less effort by the user. This is in contrast to traditional techniques, which use immediate selection and require great care and precision by the user. The goal of progressive refinement technique design is to make the multiple selection steps as simple and fast as possible such that the techniques can be as fast as the baseline technique independent of target size, while being much more accurate when the target is small. This can be thought of as “beating” the speed-accuracy tradeoff. While some existing selection techniques can be characterized as progressive refinement techniques, this work is the first to describe the approach formally and to analyze the inherent tradeoffs deeply.

We can further characterize progressive refinement selection techniques along a continuum based on the gradualness of refinement and exemplar techniques for different parts of the continuum (Figure 1). 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.

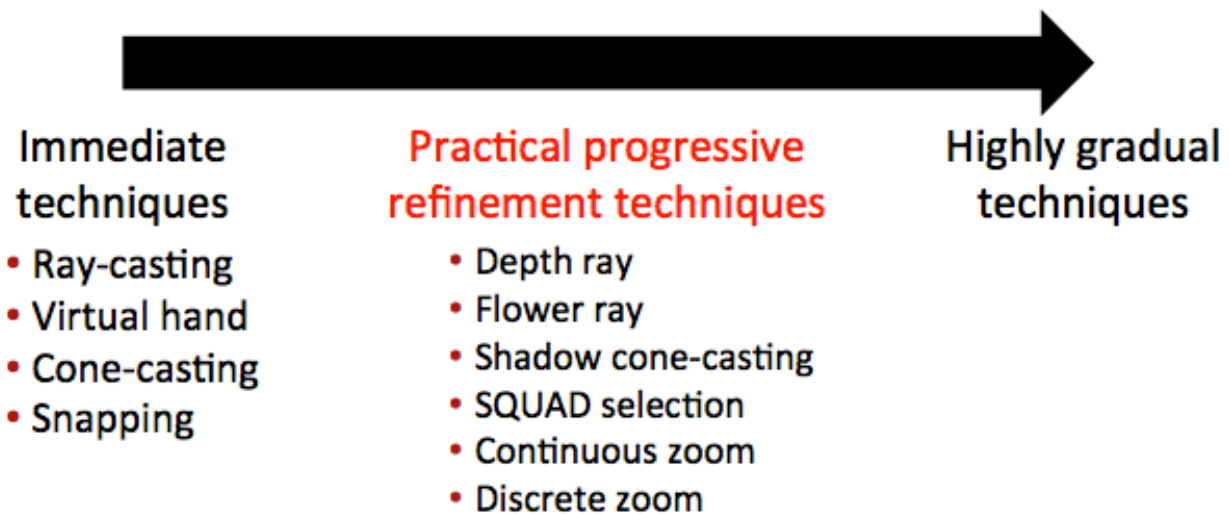


Figure 1 – Gradualness continuum.

There is an inherent tradeoff between immediate techniques and progressive refinement 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 for difficult selection tasks. The primary goal of the work presented in this dissertation 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 through progressive refinement.

By using the concept of progressive refinement, we believe it is possible to provide highly accurate selection techniques that require significantly less precision from the user when compared to immediate techniques. By dividing the selection task into much simpler tasks, we can achieve such levels of precision. Since these tasks can be performed much faster than the original tasks, our goal also includes making these steps as fast as possible so that progressive refinement techniques can be at least as fast as immediate techniques.

1.3 Definitions

This section presents the definitions for several terms used throughout this document that are important to the discussion of our research.

Selection: specification of one or more objects by the user.

3D Selection: selection in which the user input is performed in a physical 3D spatial context (e.g., with gestures in space using a Kinect) (Doug A Bowman, Kruijff, LaViola Jr, & Poupyrev, 2004).

Immediate Techniques: interaction techniques that require users to perform selection in a single precise action.

Selection by Progressive Refinement: interaction techniques that allow users to gradually decrease the set of selectable objects in such a way that the overall selection task requires less precision.

Ray-casting: selection technique based on the laser-pointing metaphor, in which the user casts a ray toward the screen and the object that first intersects with the ray will be selectable.

Refinement Step: one step from the process of selection by progressive refinement.

Distal Pointing: to interact while standing or walking by pointing at the display from a distance.

Gradualness: a characteristic of progressive refinement techniques that is related to the number of refinement steps required for selection.

Spatial Context: the spatial region of the environment in which the target lies.

Control-Display (C-D) Ratio: ratio between the interaction space and the visual space (Blanch, Guiard, & Beaudouin-Lafon, 2004).

Zoom: magnification of the visual space by readjusting the size of the view frustum.

1.4 Research Questions

The research questions presented here are divided into three categories: design, theory, and evaluation. The work on the design question provided an understanding of how progressive refinement can be used to achieve rapid and accurate selection in 3D user interfaces. The theory question helps to guide the design of new techniques based on progressive refinement. Finally, the evaluation questions help us analyze the performance of the techniques we have designed.

1.4.1 Design

The question related to the design of progressive refinement techniques allowed us to determine the best design characteristics for each type of technique. These characteristics are defined in the design space, and their combination produces unique techniques.

Question D1: How can we effectively use the concept of progressive refinement to design pointing-based techniques for complex single-object selection tasks?

We hypothesized that we would be able to design effective progressive refinement techniques that require less precision but do not affect speed in selection, by making each refinement step easy to perform but limiting the gradualness (i.e., keeping the number of refinement steps low).

We designed a total of five progressive refinement techniques for single-object selection based on the pointing metaphor. The goal was to try as many combinations of design characteristics as possible to determine the trade-offs of different choices for different selection tasks.

Question D2: How can we use the concepts of selection by progressive refinement for multiple-object selection tasks and other interaction metaphors?

Our main hypothesis was that we would be able to highlight the benefits of progressive refinement by designing new techniques for multiple object selection tasks, since immediate selection of multiple objects at once can be quite challenging. We also hypothesized that progressive refinement could be used to improve a different interaction method other than pointing-based techniques: gesture-based input.

We specifically explored the use of progressive refinement for a different task scenario: multiple object selection through the selection of a 3D spatial region that may contain one or more objects, which we refer to as **volume selection**. We focused on creating these techniques using different interaction metaphors based on in-air gestures. We designed a total of three selection techniques based on progressive refinement, with the main goal of validating the concept of progressive refinement for other tasks and using different metaphors.

1.4.2 Theory

The question related to theory had the goal of generalizing and formalizing the design of progressive refinement techniques.

Question T1: How can we categorize existing progressive refinement interaction techniques to facilitate the design of new techniques?

Our main hypothesis related to this question was that progressive refinement techniques can be characterized by a small set of independent design dimensions.

We provided an answer to this question by creating the design space for selection by progressive refinement. The main objectives of this design space are to describe the similarities and differences among techniques, to help in the design of new techniques and the improvement of existing ones, and to provide a framework for a set of guidelines that provide insights about trade-offs so that other researchers will know how various design choices affect performance and usability.

1.4.3 Evaluation

The goal of our evaluations was to understand the trade-offs of the selection techniques that use progressive refinement when compared to commonly used immediate selection techniques. There is an inherent trade-off 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 evaluations was to explore this trade-off. 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.

Question E1: What are the trade-offs of selection techniques based on progressive refinement when compared to existing state-of-the-art immediate selection techniques...

Question E1.1: ...in a controlled environment?

Question E1.2: ...in real-world applications?

The main hypothesis for this question was that the evaluation of progressive refinement would highlight the trade-off between a single precise selection and multiple imprecise selections. The secondary hypothesis was that we would see similar performance in both controlled and real-world environments, and performance benefits for using progressive refinement would be similar with increased levels of ecological validity.

We first used controlled environments to determine the best designs in terms of speed and accuracy. We also evaluated our techniques and state-of-the-art immediate selection in real-world applications to validate the results obtained in the controlled environments and to determine how performance is affected when users have to perform other tasks, such as navigation.

Question E2: How can we compare different options of the design space? More specifically, what are the trade-offs between...

Question E2.1: ... *discrete* and *continuous* types of progression?

Question E2.2: ... all the different refinement criteria?

Question E2.3: ... *in-context* and *out-of-context* display of selectable objects?

Question E2.4: ... *technique-enforced* and *user controlled* refinement control?

Question E2.5: ... *closed-loop* and *open-loop* feedback?

Our final hypothesis was that we would be able to identify the trade-offs between all design space dimensions to provide design guidelines and recommendations.

This question was answered through the evaluation of all technique designs. The investigation of the different trade-offs allowed us to find the advantages and disadvantages of each technique and choices in the design space.

1.5 Approach Overview

This section provides an overview of the steps taken towards the completion of this work.

1) Design of pointing-based progressive refinement selection techniques for single-object selection

This step consisted of the design of our pointing-based techniques: Sphere-casting refined by QUAD menu (SQUAD) (described in section 3.1) and both Discrete and Continuous Zoom techniques (described in section 4.1).

2) Creation of initial design space/taxonomy

Based on the first set of techniques we implemented, we created an initial design space for progressive refinement techniques.

3) Evaluation of initial technique designs

This step consisted of two studies that evaluated different approaches for progressive refinement. In these studies, we compared our techniques to ray-casting in controlled environments where the only task was to select targets as quickly and as accurately as possible. Our goal was to find the trade-offs between the different design choices. These studies are described in detail in section 3.2 and chapter 4.

3.1) Empirical evaluation 1: SQUAD vs. Ray-casting

The first study compared the use of our SQUAD technique to regular ray-casting in a controlled environment. We compared their uses for different target sizes and number of distractors. We evaluated the trade-offs of using SQUAD, and showed that it is always more accurate than ray-casting, independent of target size, and that it can be faster than ray-casting for certain situations. This evaluation is described in section 3.2.

3.2) Empirical evaluation 2: zoom techniques vs. Ray-casting

This study compared two different zoom techniques (Continuous Zoom and Discrete Zoom) to ray-casting. Different target sizes and enforced number of refinements for zoom were compared, and the progressive refinement techniques proved to be much more accurate than ray-casting, while not resulting in a big speed penalty. This evaluation is described in chapter 4.

4) Final design of pointing-based progressive refinement techniques

In this iteration of design of pointing-based 3D selection techniques, we improved the design of the techniques we had developed initially to create new techniques: Double Bubble, based on SQUAD, Expand and Bubble Cursor, and FRIZ, based on the strengths of both discrete and continuous zoom. This design is presented in section 5.1.

5) Comparison of final technique designs to state-of-the-art immediate selection techniques

This step consisted of the formal evaluation of the new technique designs and comparison to state-of-the-art immediate selection techniques. This evaluation showed how different target sizes and number of distractors around the target affect selection in both highly-controlled environments (as in the previous studies) and real-world environments, with varying degrees of task validity. While

there was no significant difference between the different progressive refinement techniques and immediate techniques in task speed, progressive refinement techniques were more accurate in the real-world environment task scenarios. More importantly, there was a difference in the performance when techniques were tested in real-world scenarios, which shows the importance of verification of external validity when testing selection in 3D UIs. The complete evaluation is shown in section 5.2.

6) Design of gesture-based progressive refinement techniques for volume selection

Previous efforts focused on the design of pointing-based progressive-refinement techniques for single-target object selection, and their evaluations showed that progressive refinement is a viable alternative to immediate selection. In order to validate the concept of progressive refinement for other tasks and interaction metaphors, we designed three selection techniques that use bare-hand gestures and progressive refinement to support the task of point cloud selection, where the user has to select a volume of space containing many target points (presented in section 6.2).

7) Evaluation of volume selection techniques

This step consisted of the formal evaluation of the gesture-based techniques for point cloud selection. Techniques were compared with different point cloud selection tasks and two types of refinement feedback were tested (open- and closed-loop). Closed-loop was largely preferred over open-loop. These results are shown in section 6.3.

8) Final taxonomy development

After completing the design and evaluation of all progressive refinement techniques, we compiled the trade-offs of each design into a final taxonomy. The goal was to help designers decide when it is best to use each one of the design alternatives we evaluated. The final design space is shown in chapter 7.

9) Development of design guidelines

This step involved the formulation of design guidelines for progressive refinement selection techniques. Guidelines and recommendations are presented in chapter 7.

1.6 Contributions

Our contributions to the Human-Computer Interaction community can be divided into three main parts:

- 1) Description and formal analysis of the concept of progressive refinement for selection techniques. With the introduction of this concept, we also provide a taxonomy that organizes

the design space to help interaction designers decide the best categories of techniques for their needs.

- 2) Empirically validated design guidelines for progressive refinement-based selection techniques, which will help in the design of new innovative techniques that can be both fast and accurate even when the targets have small visible areas.
- 3) A large set of new interaction techniques that explore different parts of the design space and can be used in a variety of situations in which fast and accurate selection is needed.

With these contributions, we hope to inspire innovation in 3D UIs. The techniques designed for this project already proved to improve performance and reduce effort in selection. They can be used to improve usability and have a positive impact on multiple application areas, including gaming/entertainment interfaces that use natural interaction and gestures (e.g., Nintendo Wii, Playstation Move, Microsoft Kinect, and Leap motion). In addition, we believe that progressive refinement can be used to improve selection in other user interfaces, such as touch-based UIs.

2 Related Work

This chapter presents relevant prior work on the design and evaluation of selection techniques for both single target selection and volume selection. Each category is subdivided into two main types of techniques: Immediate Selection (selecting targets in a single precise selection); and Selection by Progressive Refinement (using multiple discrete steps or continuous refinement, without precision requirements, to perform selection).

2.1 Single-Object Selection

Single-object selection refers to the task of selecting a single object at a time. This is the most common selection task, and many techniques focus on improving precision and reducing the time it takes to complete selection (Argelaguet & Andujar, 2013).

2.1.1 Immediate Selection

Doug A. Bowman et al. (2004) divided 3D selection techniques into four main categories: selection by pointing (e.g., ray-casting (Mine, 1995)), selection by touching (e.g., virtual-hand (Mine, Brooks Jr., & Sequin, 1997)), selection by occlusion (e.g., image-plane techniques (Pierce et al., 1997)) and indirect selection (e.g., selection by attributes (Doug A. 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 & 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.

The next sections present an overview of different approaches to improve precision in immediate selection.

Selection by Spatial Region

Area cursors are a commonly used solution for precision issues with pointing (Kabbash & Buxton, 1995). However, as noted by Worden, Walker, Bharat, and Hudson (1997), their original implementation is not suited for applications that require discrimination between two targets that are too close together.

To solve this, Hertzum and Hornbæk (2007) presented two techniques that dynamically change their area based on the number of objects that are close to the cursor. The Box cursor creates boxes that represent different objects close to the cursor position and maps the position of the boxes to a trackpad. By simply tapping on the box area, users select the target directly. The Pie cursor uses a similar approach, but divides the trackpad into “pie slices” and uses arrows to indicate which area of the trackpad needs to be touched to select the object. The disadvantage, however, is that users perform worse with these methods as the number of objects in the scene increases, since it either makes the cursor too small or requires too much precision in selecting with the touchpad.

In 3D environments, Cone-casting (Liang & Green, 1993) extends ray-casting by adding a cone-shaped volume to the ray to make it easier to select objects that are distant. Disambiguation, in this case, is done by selecting the object that is closest to the center of the cone. Similarly, Forsberg, Herndon, and Zeleznik (1996) presented the aperture-based selection technique, in which the user controlled both the cone direction and aperture size. 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.

Snapping

Snapping techniques work by picking the best object that is near the cursor, and snapping the cursor to it. The simplest example of a snapping technique is to pick the object that is closest to the ray, and move the ray to point to it (Wingrave, 2009). The technique presented by Haan, Koutek, and Post (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 & 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, Grossman, and Coninx (2007). 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.

C-D Ratio Manipulation

Other techniques improve ray-casting accuracy by changing the control-display ratio. These techniques work in a similar way as common mouse acceleration does, in which the scale of the cursor movements depends on the speed in which the mouse moves (i.e., the greater the speed, the greater the scale). For instance, while ARM (Kopper, Bowman, Silva, & McMahan, 2010) requires users to manually reduce the control display ratio by pressing a button, PRISM (Frees, Kessler, & Kay, 2007) adjusts the cursor speed based on how fast or slow the user moves it. DyCoDiR (Kopper, 2011) dynamically changes the cursor speed based on movement speed, like PRISM, and also based on the distance from the screen. While these techniques can achieve very high levels of precision, they cause a significant mismatch of the physical pointing direction to the perceived pointing position. Adaptive Pointing (König, Gerken, Dierdorf, & Reiterer, 2009), similar to DyCoDiR, changes the control-display ratio dynamically based on the cursor speed, but it also takes into account the offset between the motor-space position and the screen-space position to provide a smoother transition between relative and absolute pointing. These techniques, however, require the user to interact very carefully and with precision. Our proposed approach of selection by progressive refinement aims to allow “lazy” interaction with high accuracy.

A solution for improving precision of distal pointing for large displays, presented by Peck (Peck, North, & Bowman, 2009), is to use physical navigation to increase or decrease the scale of interaction. If the user is visualizing hierarchically structured data (e.g., a map) far from the display, the cursor grows in size and allows users to select objects in higher levels of the hierarchy (e.g., a country). If the user is closer to the display, lower level objects would be selectable (e.g., a city). This does, however, require the user to move physically, and may not be desirable when selection is not the main task.

Benko, Wilson, and Baudisch (2006) presented five different techniques for improving precision of multi-touch selection called Dual Finger Selections: Dual Finger Offset reduces the control-display ratio whenever a secondary finger touches the surface; Dual Finger Midpoint creates a cursor in the midpoint of two fingers touching the surface; Dual Finger Stretch allows users to simultaneously change the cd ratio and zoom level by using a secondary touch to manipulate the size of a window; the Dual Finger X-Menu provides a menu with control over cursor speed; and Dual Finger Slider provides the same functionality but uses the distance between fingers to determine cursor speed. While the techniques provide improved precision, they may require more cognitive load and are not suited for handheld devices, since they require interaction with two hands.

2.1.2 Selection by Progressive Refinement

There are several selection techniques in the literature that can be classified as progressive refinement.

Menu Disambiguation

In 2D environments, enhanced area cursors can be used to disambiguate selection (Findlater et al., 2010). The Click-and-Cross technique allows users to select an area of the screen, and expand the items contained in that area into crossing arcs that do not require direct selection, but only that the cursor crosses that region. This technique is similar to the approach used in SQUAD, but requires more precision from the user when too many objects are presented on the screen. The other technique presented by Findlater et al., the Visual-Motor-Magnifier creates a small magnified window that also increases precision. This approach is similar to our zoom techniques, but it only provides one preset magnification level.

Ramos et al. (2006) presented two techniques that aim to solve selection of occluded objects and could also be classified as progressive refinement. Tumbler uses a stack representation to show occlusion with different layers, and splatter uses object proxies to create a new view where all objects are accessible, and moving the cursor over the proxies show the original objects without occlusion. Handle Flags (Grossman, Baudisch, & Hinckley, 2009) is similar to splatter, but used in the context of pen interaction. This approach is similar to the one used in SQUAD, but it does not require objects to be taken out of context and has only two phases for selection. The main difference, however, is that users use direct pointing for selection of the proxy objects, and SQUAD uses quadrants on the screen to reduce the required precision.

The Flower Ray technique (Grossman & 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. Ren and O'Neill (2013) used a similar approach in their Menu Cone technique, in which a cone is used in the first phase and objects inside that cone are distributed in a circular marking menu. These techniques may not be suited for highly cluttered environments, since the marking menu object specification is done in a single phase.

J. Cashion, Wingrave, and LaViola (2012) designed Expand, which works similarly to SQUAD in that it involves an area selection and a refinement phase. The main difference is that instead of distributing objects in quadrants, the objects are distributed in a grid and users have to select the

objects directly. While this approach works well for most cases, it may require too much precision from the user when disambiguation includes a large number of objects.

Baudisch, Zotov, Cutrell, and Hinckley (2008) presented Starburst, which is a target expansion technique to reduce precision required for selection. It does so by creating selectable tiles for each individual target that expand to the edges of the screen. This approach, however, does not scale well as it provides a tile for every object on the screen and tiles would become too narrow in cluttered environments.

The motion-pointing technique (Fekete, Elmqvist, & Guiard, 2009) allows users to select individual objects without pointing at them, by assigning different types of elliptical motions to each object. It also reduces the number of selectable objects by selecting the top four motion matches and distributing them in a pie menu for direct selection. This technique, however, may not be suited for interfaces with many objects, as the required precision would increase.

Continuous Disambiguation

A number of interaction techniques developed for 3D virtual environments can also be included in the progressive refinement definition. For example, the depth-ray technique (Grossman & Balakrishnan, 2006), which adds depth control to the classic ray-casting technique to select occluded objects, requires two actions to specify the target. Ren and O'Neill (2013) extended this idea in Marker Cone, in which regular cone-casting is combined with depth-ray. Similarly, Benko and Feiner (2007) showed the balloon selection technique, which allows users to control depth of selection over a 2D touch surface by using the distance between two hands.

The navigation techniques for multiscale virtual environments presented by Bacim, Bowman, and Pinho (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.

Zooming

Zooming for selection can be considered a form of progressive refinement. 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 & 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 & 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. Focusing on pointing, Guiard and Beaudouin-Lafon (2004) presented the concept of multiscale pointing, which refers to the combination of moving the view to make the target visible and pointing at the target. Bourgeois and Guiard (2002) studied the use of one-handed and two-handed multiscale pointing, and showed that two-handed input and constant speed improve task completion time. Our continuous zoom technique uses constant navigation speed. However, since zoom is only used for selection and navigation is only temporary, we decided to use one hand to control both zoom and selection.

Zoom-and-pick (Forlines, Balakrishnan, Beardsley, van Baar, & Raskar, 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.

Another example is the High-Precision Magnification Lenses (Appert, Chapuis, & Pietriga, 2010), which focuses on solving the mismatch between the visual representation and motor precision in focus+context techniques. Three different techniques were developed: speed, which maps the precision of the pointer to the speed of input device with a continuous function; key, which gives the user two speed modes (focus speed and context speed); and ring, which decouples the cursor from the center of the focus region, gives more precision by using focus speed when the user has the cursor inside it, and allows user to move the focus area quickly by using context speed when the cursor is at the border of the area. Our zoom techniques use constant speed, independent of the amount of zoom used, so that the motor precision is matched to the current view.

In 3D environments, Hansen, Skovsgaard, Hansen, and Møllenbach (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, 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.

J. Cashion et al. (2012) compared SQUAD to a zoom technique that can be classified as progressive refinement, although selection is always limited to two steps independent of target size. The first technique, called zoom, could be described as a combination of our continuous and discrete zoom techniques: the user first selects a region of the screen to zoom (with a fixed amount of magnification), and then selects the object directly. This gives less control to the user, and the technique is again limited by the target size.

Albinsson and Zhai (2003) implemented and evaluated Zoom-Pointing for touch interfaces, in which the user enters a zoom mode, draws a rectangle to determine a zoom area, and then performs selection. Based on the analysis of this technique, two new techniques were proposed: Cross-Keys, which uses buttons to adjust the position of a crosshair; and Precision-Handle, which works by creating a pivot point when the user first tries to touch the target and manipulating a lever to move a crosshair. The zooming technique proved to be the fastest and most precise technique of the techniques evaluated. Roudaut, Huot, and Lecolinet (2008) presented two techniques that could be characterized as progressive refinement: TapTap, in which users first select an area of the screen to create a pop-up magnification and then select the target; and MagStick, which is similar to the Precision-Handle, but uses snapping to facilitate selection. Olwal, Feiner, and Heyman (2008) introduced three selection techniques based on rubbing and tapping: with Rub-Pointing, users rub the screen diagonally to zoom and lift their finger to select the target; with Zoom-Tapping, users tap with the non-dominant hand to zoom while adjusting position with their dominant hand; and with Rub-Tapping, which uses Rub-Pointing with the dominant hand and tapping with the non-dominant hand for confirming selection.

2.2 Volume Selection

Volume Selection refers to the task of selecting one or more objects by specifying a volume inside the 3D environment instead of specifying discrete objects. This is often used in applications which require selection of a large number of objects at one time.

2.2.1 Immediate Selection

There are a number of examples of immediate volume selection in 3D user interfaces. For instance, Lucas (2005) created a number of multiple-object selection techniques for immersive environments, including a resizable 3D box and a 2D lasso drawn with a pen and tablet interface. Similarly, the bimanual selection techniques presented by Ulinski, Zanbaka, Wartell, Goolkasian, and Hodges (2007) allow the specification of a selection volume by moving, resizing and skewing a selection cube by using free-hand gestures. The biggest disadvantage of these techniques is that they require careful choice of the point of view used when performing the selection action, a large amount of selection volume manipulation, and high-level of input precision.

Dynamic Dragging, presented by Keefe, Zeleznik, and Laidlaw (2008), is a free-hand 3D lasso selection technique used in the selection of streamlines in volumetric data. Users draw the selection lasso by dragging a 3D pen in space using a line with dynamic size based on input properties such as dragging direction as speed. While this approach smooths out the output of the generated lasso, performing it in 3D space can still be quite challenging and requires high amount of effort from the user.

In order to solve these issues, several different selection techniques have been developed to assist the selection task using dataset features of their specific application. For instance, Wiebel, Vos, Foerster, and Hege (2012) created What You See Is What You Pick (WYSIWYP) for selection of pre-segmented structures in volumetric data visualization. This technique uses what is currently visible on the screen to determine what will be selected when users perform the selection action, which is a simple click. Stoppel, Hege, and Wiebel (2014) also created a visibility-driven technique for volumetric data selection, but focused on automatic detection of surface patches instead of pre-segmenting data to generalize the selection method.

Another example application is presented by Burgess et al. (2015): a visual analytics tool for galaxy point cloud visualization which uses the Leap Motion Controller for input (a consumer-level hand-tracking device). In order to reduce the precision required for selection, the application pre-calculates point cloud clusters based on the input datasets to reduce the selectable objects to spatial

regions-of-interest. When users perform the pinching gesture used to select, the hand snaps to the closest cluster of points and all those points are selected.

Yu, Efsthathiou, Isenberg, and Isenberg (2012) presented two selection techniques for particle cluster visualizations: Cloud Lasso and Teddy Selection. Both techniques use different particle density algorithms in combinations with two-dimensional lasso to make it easier to select clusters of particles that are within a user-defined density threshold. Shan, Xie, Gao, and Chi (2014) presented the same idea, but also used a circle shape for selection for a large scale cosmology simulation application.

Owada, Nielsen, and Igarashi (2005) focused on the selection of 3D regions for segmentation of volumetric datasets by drawing the contour of the region of interest. The technique then uses the density of the matched contour in projected volume data to perform segmentation.

Even though there are many examples of immediate selection techniques for the volume selection task, they may not very practical for applications that contain large numbers of unstructured targets. Additionally, while domain-specific techniques can improve selection performance, they are often not generalizable since they depend on specific dataset features that can only be applied to that domain.

2.2.2 Progressive Refinement

Only a few existing techniques use progressive refinement to deal with volume selection. For example, the shadow cone-casting technique (Steed & Parker, 2005) uses continuous movement with cone-casting and intersection operations to create a selection volume and disambiguate selection (i.e., only objects that were inside the selection cone at all times are selected).

Zhou, Correia, and Laidlaw (2008) presented a selection technique for streamlines in volumetric data that uses progressive refinement. Users can draw multiple 3D lassos using the Dynamic Dragging technique (Keefe et al., 2008) using a haptic interface that constrains drawing to planes perpendicular to the streamlines to reduce the required input precision. Selection is then refined by using Boolean operations between the lassos. Our lasso approach (presented in section 6.2.1) uses the progressive refinement of 2D screen-space lasso selections, further reducing the amount of precision required at the cost of increasing the amount of navigation necessary to perform 3D selections.

Another example is Force Brushes (Jackson, Coffey, & Keefe, 2012), in which haptic feedback is used to provide cues in streamline visualizations in addition to a progressive refinement approach.

Selection is done in two phases: selection of an initial region of interest in the data; and a progressive growth of the initial selection volume. This second phase is controlled by dragging the initial selection perpendicular to the streamline, but is limited to the context of the initial selection by haptic constraints generated based on the data being visualized.

All techniques we present in section 6.2 use progressive refinement for volume selection in a point cloud application. We use two distinct phases to reduce the set of selectable points: specification of two point-cloud subsets and removal of the unwanted subset. These two phases can be done iteratively until the only points left are the selection target.

2.3 Discussion

This chapter presented existing work on two types of selection techniques (immediate selection, and selection by progressive refinement) for two types of selection tasks (single-object selection and volume selection). While immediate selection allows users to perform selection in a single step, it requires users to be precise. Selection techniques that can be classified as progressive refinement, on the other hand, require less precision from the users by gradually reducing the set of selectable objects at the cost of using a process with multiple steps. The two selection tasks present different sets of challenges, and investigating how the concept of progressive refinement can be used to improve selection for each task helped us generalize and expand its design space (presented in chapter 7).

Therefore, the main goal of this research was the investigation of the progressive refinement trade-off through the design and evaluation of progressive refinement techniques for these different selection tasks. To the best of our knowledge, there had been no prior generalization of the progressive refinement concept, and no comparison of progressive refinement techniques to immediate techniques. Finally, while there are a number of selection techniques that can be categorized as progressive refinement techniques, very few explore its design space for 3D selection, distal pointing, or gesture-based selection.

3 Design and Evaluation of SQUAD: A Single-Object Selection Technique based on 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 and effort (Kopper, Bacim, & Bowman, 2011). Based on this concept, we designed SQUAD, our first progressive refinement technique. Our primary motivations were to start to answer our design question **D1**, which asked how we can effectively design interaction techniques for complex single-object selection tasks. We evaluated our design and compared SQUAD to Ray-casting in an effort to start to answer our evaluation question **E1.1**, which was about the investigation of the trade-offs between progressive refinement and immediate selection techniques.

3.1 Design of 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, described by Bacim 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 2 illustrates this selection phase. (Note that in the study described here 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.



Figure 2 – Sphere-casting.

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, Agrawala, & Hinckley, 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 3. 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.

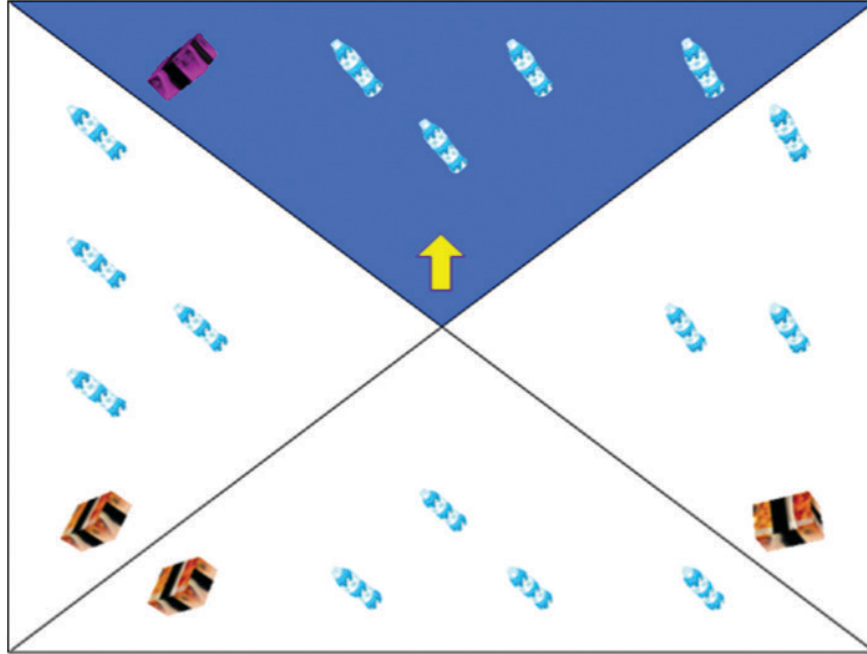


Figure 3 – Quad menu. Note that the target object needs to be visually distinct for the selection to be feasible.

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 Evaluation 1 – SQUAD vs. Ray-Casting

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.

3.2.1 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.

3.2.2 Experimental 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.

3.2.3 Analytic Evaluation

Before running an empirical study, 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, Kong, & Ren, 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

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

where ID_{DP} is the index of difficulty, α is the angular amplitude of the movement and ω 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. (2010). 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

$$MT = 1.091 + 0.028 * ID_{DP}, \quad (2)$$

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° . The targets were chosen within a constant distance range from the starting point, so that the movement amplitude was selected randomly between 10.0° and 17.9° , with an average α of 14.0° . 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° , 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 ω higher than α , 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 $\lceil \log_4(n) \rceil$, 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 ω 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 $3c + v_{t_2}$, $4c + v_{t_3}$ and $5c + v_{t_4}$ seconds to select a target, where v_{t_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_{t_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 $i>0$, $v_i = 0.2 + ((R - i) * 0.1)$. 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

$$v_{t_R} = \sum_{i=1}^R (0.2 + ((R-i)*0.1)), \quad (5)$$

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).

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.

Density	Target Size	Ray-casting	SQUAD
Low	Small	2.29s	2.0s
	Medium	1.47s	2.0s
	Large	1.14s	2.0s
Medium	Small	2.29s	2.9s
	Medium	1.47s	2.9s
	Large	1.14s	2.9s
High	Small	2.29s	3.9s
	Medium	1.47s	3.9s
	Large	1.14s	3.9s

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

3.2.4 Empirical Evaluation

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

Apparatus

We used a back-projected VisBox-SX system, with only one projector (monoscopic) to display the experimental environment on a $2.29m \times 3.05m$ screen. The resolution of the graphics was $1400px \times 1050px$. 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 4. 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.

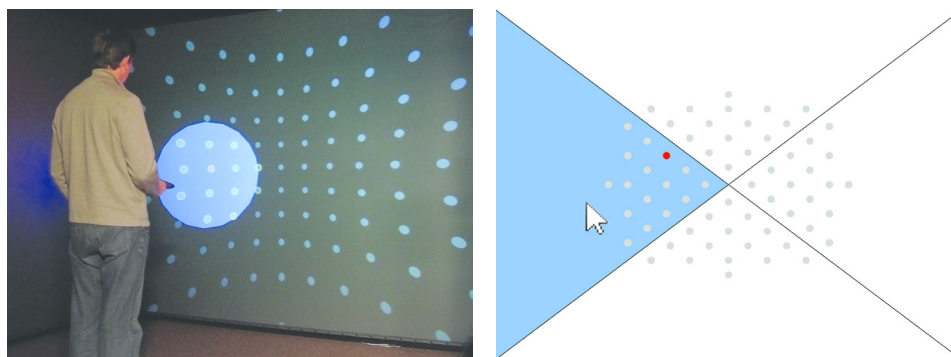


Figure 4 – Left: Experimental setup with sphere-casting. Right: quad menu stage of SQUAD in the experiment.

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 4, 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 4, 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° and 0.21° 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.

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.

Procedure

Upon arrival, participants were greeted by the experimenter and given an informed consent form to read and sign. They were then given a color-blindness 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.

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 5 shows the significant interaction between technique and distractor density ($F_{2,30} = 290.51, p < .0001$).

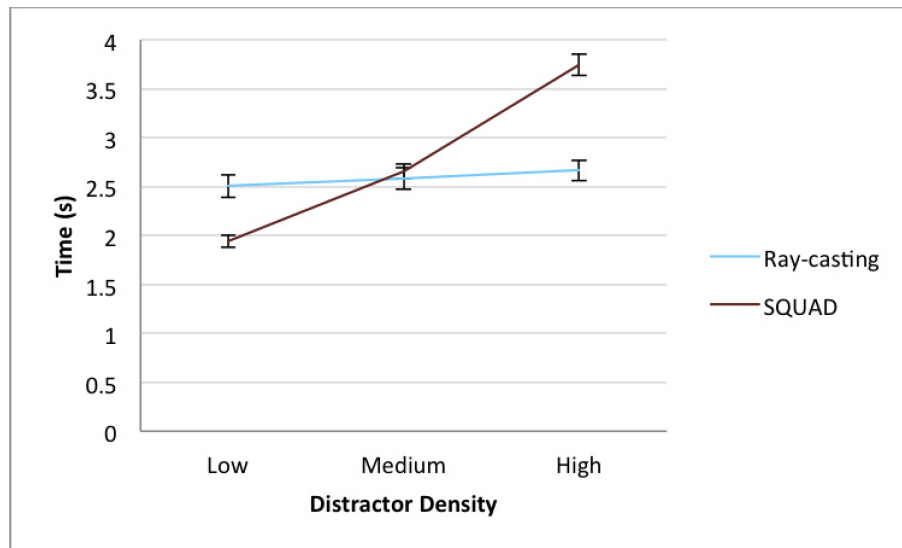


Figure 5 – Interaction between technique and distractor density. The error bars represent standard error.

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 6. 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.

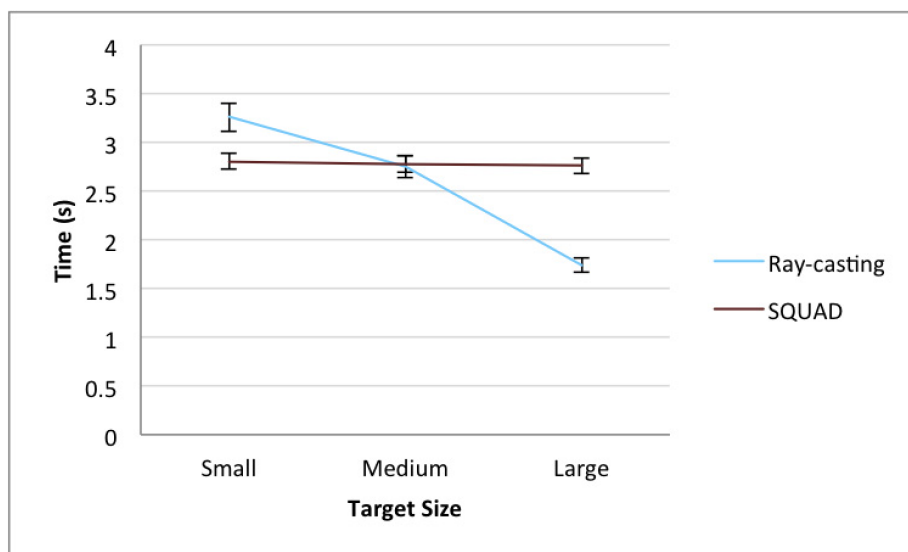


Figure 6 – Interaction between technique and target size. The error bars represent standard error.

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 7 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.

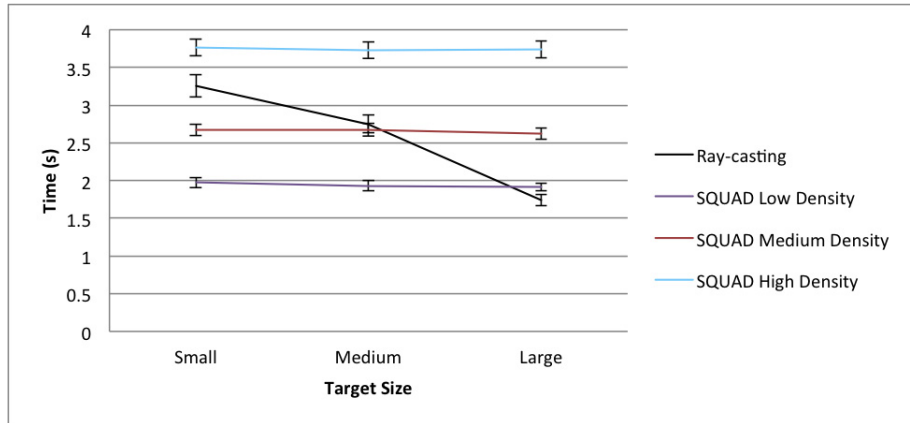


Figure 7 – Mean results for all technique-density-target size combinations. Note that all ray-casting densities are averaged and displayed in a single line, since there was no significant difference among them. The error bars represent standard error.

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 ($n=7, W=20, insignificant$). For ease of use, ray-casting ($mdn=4.5$) was ranked significantly more difficult than SQUAD ($mdn=1$) ($z=3.16, p<.001$). 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 ($mdn_{raycasting} = 2, mdn_{SQUAD} = -3.5, z = 0.18, p = 0.19$).

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

3.2.5 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 8 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.

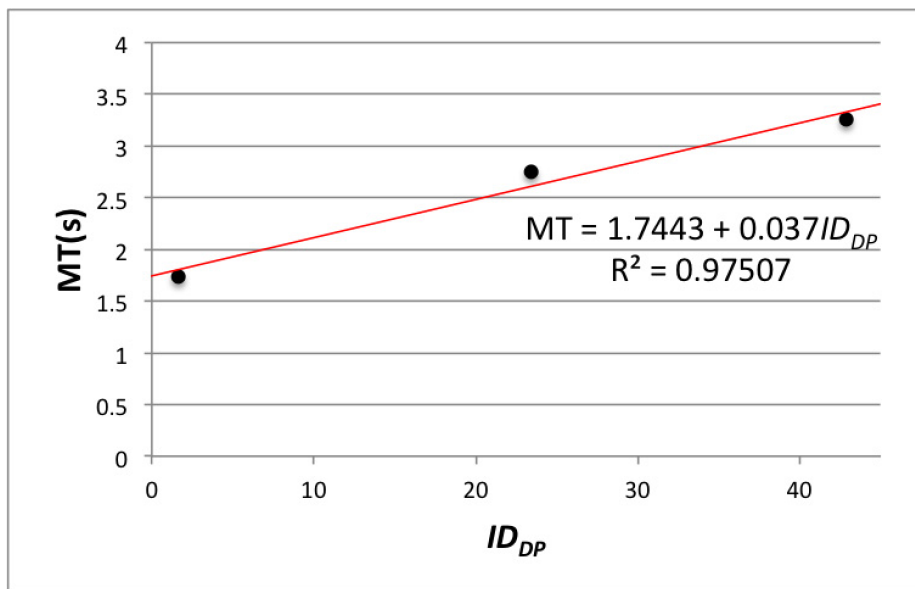


Figure 8 – Scatter plot and regression line for the ray-casting pointing condition.

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 9. 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.

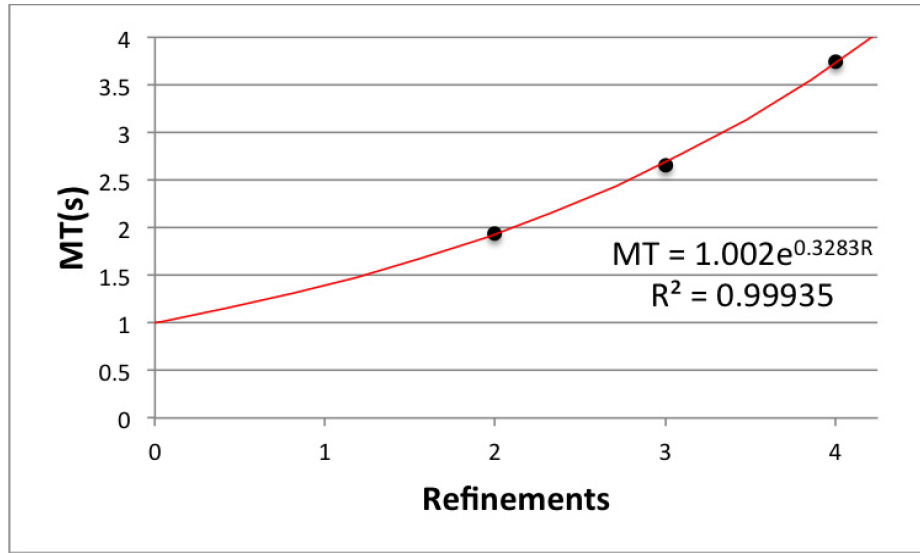


Figure 9 – Scatter plot and exponential regression line for the SQUAD pointing condition.

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 9 is not satisfying. To correct these issues, Figure 10 shows a model 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 coincidentally close to what c is, the total visual search time is smaller than we predicted. For two refinements, v_R is approximately $1.94 - 3 \cdot 0.51 \approx 0.41s$, compared to the predicted 0.5s. For three refinements, v_R is approximately $2.66 - 4 \cdot 0.51 \approx 0.62s$, compared to the predicted 0.9s. Finally, for four refinements, v_R is approximately $3.74 - 5 \cdot 0.51 \approx 1.19s$, compared to the predicted 1.4s.

3.3 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 (and thus higher error rates), while progressive refinement techniques offer minimal precision requirements (and

near-zero error rates) at the expense of more time-consuming gradual refinement. Our analytic and empirical evaluations have verified this tradeoff with respect to selection time, although SQUAD was faster than or equivalent to ray-casting for very difficult selection tasks. This result indicates that multiple easy selection steps can indeed be faster than a single difficult selection. In terms of accuracy, SQUAD was clearly superior.

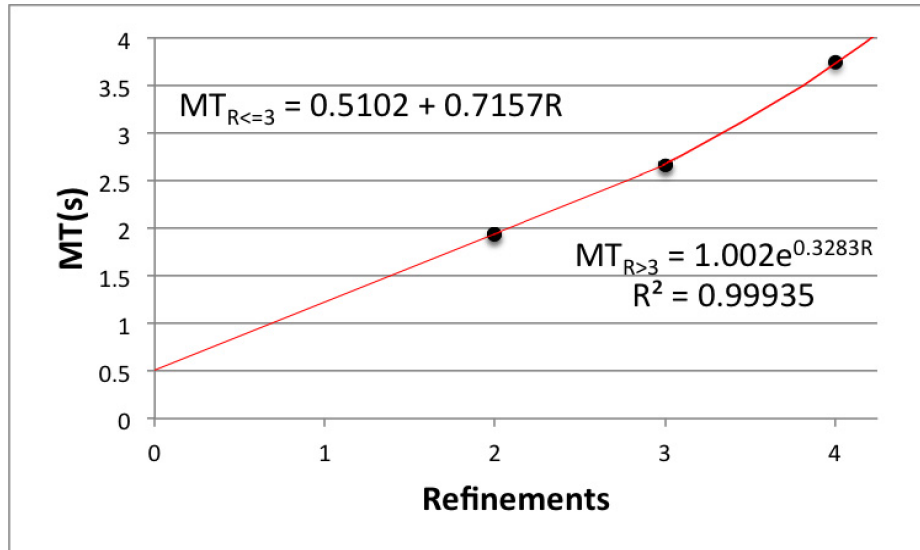


Figure 10 – Scatter plot and combination of linear and exponential regression lines for the SQUAD pointing condition.

These results partially answer our research question **E1.1**, as they highlight some possible trade-offs between a progressive refinement technique (SQUAD) and a commonly used immediate selection technique (Ray-casting).

A positive aspect of the SQUAD 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, whereas with standard ray-casting, the time increase is exponential as targets undergo a linear decrease in size.

The design of SQUAD and its evaluation also helped us begin to answer our design question **D1**, which refers to the design of efficient progressive refinement techniques for single-object selection. However, 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. It 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. Additionally, 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 SQUAD to improve efficiency by combining other enhancement techniques with the progressive refinement techniques. For example, allowing 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, 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 near ideal. The use of a dynamically sized sphere (such as the bubble cursor) to include fewer objects in the initial selection can achieve this goal, as shown in chapter 5.

Finally, we found that the distal pointing model proposed by Kopper et al. (2010) accurately predicted performance with ray-casting. 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.

4 Design and Evaluation of Single-Object, Progressive Refinement Selection through Zooming

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 chapter, 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. This chapter focuses further on the investigation of our research questions D1 (how to design effective progressive refinement techniques for single-object selection) and E1.1 (the trade-offs between progressive refinement and immediate selection). Additionally, in this chapter, we begin the evaluation of different design space dimensions (question E3) with the design of two different zoom techniques and different strategies for refinement control.

4.1 Design of Zoom Techniques

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.

4.1.1 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 11).

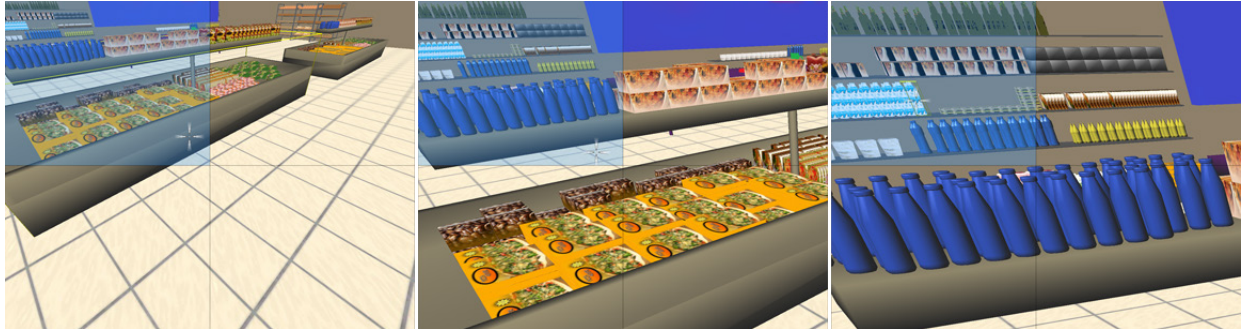


Figure 11 – Three refinement steps of the discrete zoom technique.

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.

4.1.2 Continuous zoom

The continuous zoom technique works in a similar way to discrete zoom, but it zooms in the direction of the cursor continuously instead of using quadrants (Figure 12). This way, users can simply point roughly toward the object and zoom in until the target is large enough for selection.



Figure 12 – Three snapshots of different zoom levels achieved with the continuous zoom technique.

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.

4.1.3 Design Considerations for Different Zoom Implementations

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 & 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 13). 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 13, where the pixels are visible and it is hard to identify the characters.

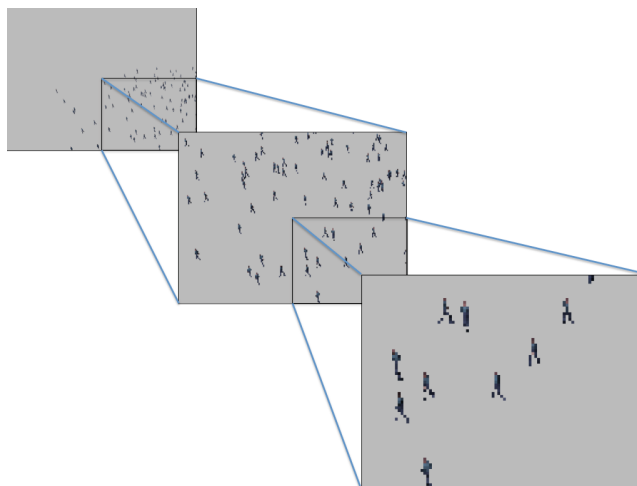


Figure 13 – Issue of magnifying a texture for zooming.

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 14.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 that will occlude everything else (see the case of the red object to the left in Figure 14.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 14.c.1).

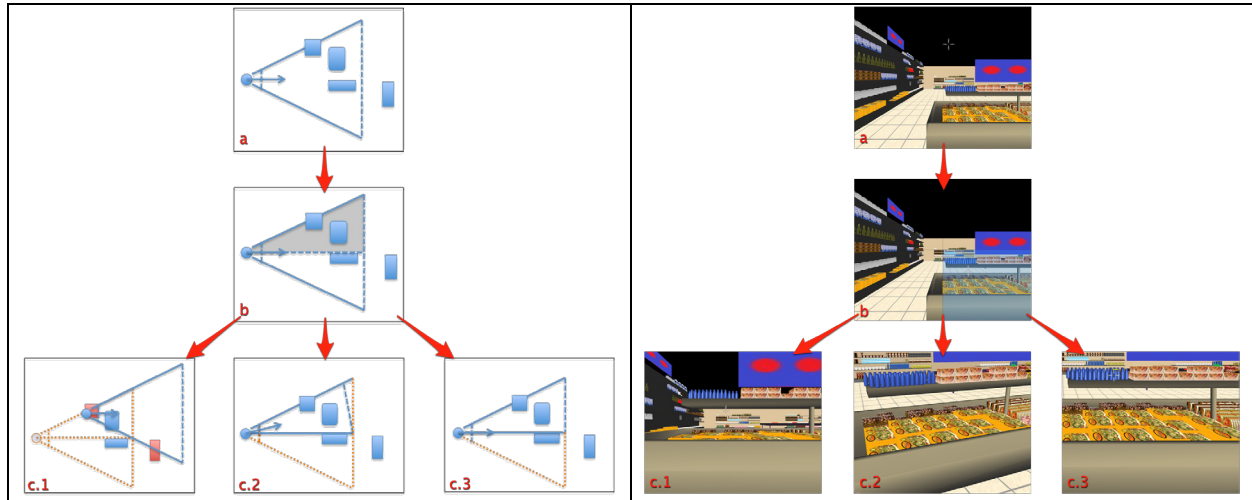


Figure 14 – Illustration and snapshots of zooming into a quadrant using the discrete zoom technique, with two initial steps in selection of a screen region to zoom (a and b), then three methods for zooming: (c.1) move the camera, (c.2) rotating the camera & frustum manipulation, and (c.3) frustum manipulation & 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 14.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 14.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.

4.2 Evaluation 2 - Zoom Techniques

We conducted a second experiment comparing the zoom techniques to standard ray-casting. We evaluated the same task as in the SQUAD study—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.

4.2.1 Goals and Hypotheses

The main goal of the second experiment was to explore the trade-offs between ray-casting and both zoom techniques. 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 the index of difficulty 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 errors 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 users choose to not take advantage of zoom, or use it very little, we would expect the number of errors to be similar to ray-casting.

4.2.2 Experimental Design

Similar to the SQUAD 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 0.01m or 0.26°; medium, with radius 0.02m or 0.53°; and large, with radius 0.04m or 1.06°), and strategy, used only with the zoom techniques (small amount of zoom, where users must zoom until the target has radius of 0.04m; large amount of zoom, where users must zoom until the target has a radius of 0.16m, and free zoom, where users can choose how much to zoom). The first two levels of the strategy variable represent enforced 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 3 (target sizes) x (1 (ray-casting) + (2 (zoom techniques) x 3 (strategies))), for a total of 21 conditions.

The order of presentation of technique and strategy was counterbalanced, blocked by technique, such that each participant used all strategy levels 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 levels of target size was repeated eight times and presented in random order.

4.2.3 Analytic Evaluation

We used the same concepts and models explained by Kopper et al. (2010) to analytically evaluate performance in our experimental conditions.

The goal of the zoom techniques and progressive refinement in general is to reduce the index of difficulty so that accuracy can be increased. To achieve this goal, the zoom techniques increase the angular width of the target (ω) 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 ω : 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 α 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, ω is 0.53°. This yields an ID_{DP} of

$$ID_{DP} = \left[\log_2 \left(\frac{14.0}{0.53^3} + 1 \right) \right]^2 \approx 42.9. \quad (5)$$

Thus, the predicted time for selection based on equation 2 is approximately 2.29s. For the 4x zoom condition, ω is 2.12°, and the ID_{DP} is

$$ID_{DP} = \left[\log_2 \left(\frac{14.0}{2.12^3 + 1} \right) \right]^2 \approx 1.7. \quad (6)$$

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

$$ID_{DP} = \left[\log_2 \left(\frac{14.0}{8.48^3 + 1} \right) \right]^2 \approx 0.001. \quad (7)$$

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. When users finally reach the desired zoom level, 16x for example, ω will still be 8.48° , but α will be reduced since the cursor will be closer to the target.

We need to sum the predicted times for each step to predict the total time to select a target. 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 ω 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 z , the original target size ω , and the final distance from the cursor to the target α . Thus, if the number of refinements is denoted as R , we hypothesize that the time it takes for selection of a target is:

$$MT_{DISCRETE} = \sum_{i=1}^R (0.25 + c + v_i) + MT_{RC}(z^* \omega, \alpha), \quad (8)$$

expressed in seconds.

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

$$R = \log_2(z), \quad (9)$$

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 defined as in equation 9 (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:

$$MT_{CONTINUOUS} = c + R + MT_{RC}(z^* \omega, \alpha), \quad (10)$$

expressed in seconds. To calculate MT_{RC} , we include a smaller α (since the cursor will be very close to the target).

In contrast to 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. 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.

4.2.4 Empirical Evaluation

In order to validate the results from our analytic evaluation, we performed an empirical study using the design described in section 5.2.

Apparatus

We used a rear-projected VisBox-SX system, with only one projector (monoscopic) to display the experimental environment on a $2.29m \times 3.05m$ screen. The resolution of the graphics was $1400px \times 1050px$. A wireless Intersense IS-900 Wand, tracked with six degrees-of-freedom, was used for controlling the cursor on the screen. The wand includes five buttons (four activated by the thumb and a trigger activated by the index finger) and a joystick controlled by the thumb.

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 15. 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). 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.

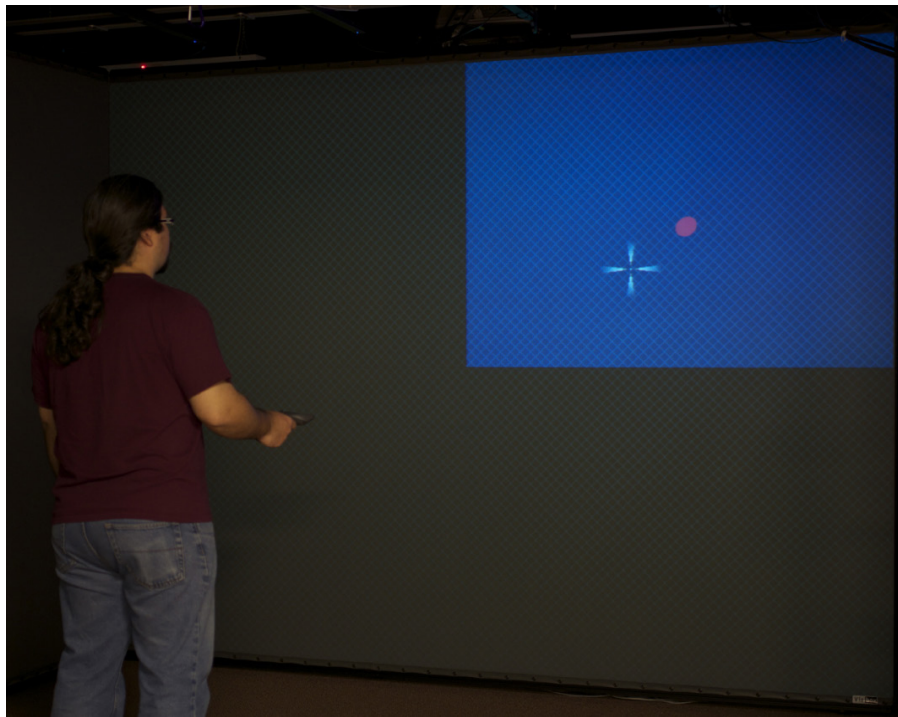


Figure 15 – Experimental setup with the discrete zoom technique.

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 $0.52m$ and a large radius of $0.77m$. 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.

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 select the target 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 with their index finger 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 a 0.25s animation until it filled the entire screen. 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 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 cursor "stickiness" effects).

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.

Procedure

Upon arrival, participants were greeted by the experimenter and given an informed consent form to read and sign. They were then given a color-blindness 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 0.7s 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/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, participants filled out a post-hoc questionnaire, comparing both techniques overall and in light of the other variables.

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 (i.e., continuous zoom with the small amount of zoom strategy), continuous-large, continuous-free, discrete-small, discrete-large, and discrete-free) 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 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 16. 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.

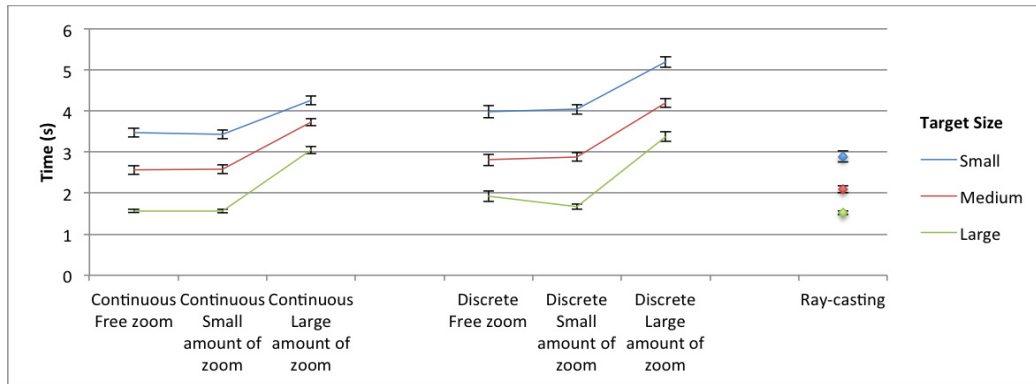


Figure 16 – Interaction between technique and target size. The error bars represent standard error.

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.

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 17 illustrates how different strategies affect performance for different target sizes.

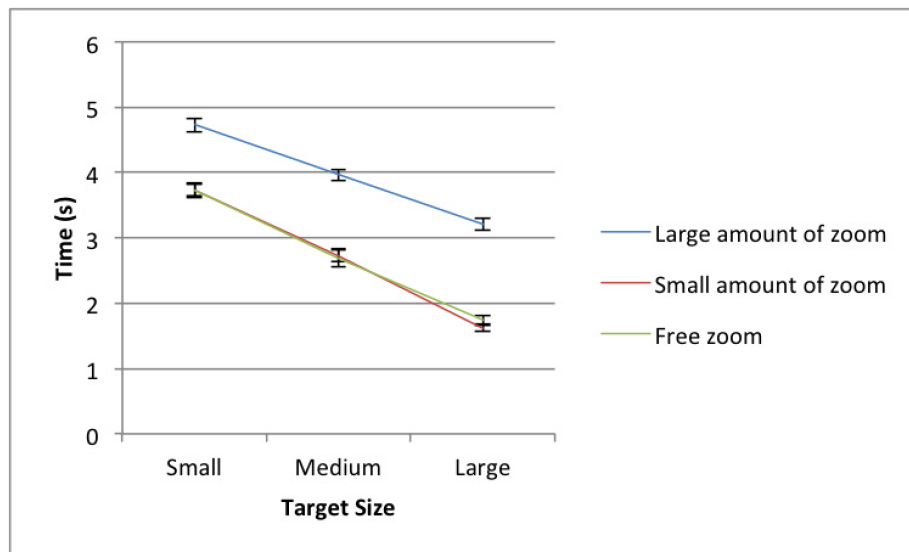


Figure 17 – Interaction between strategy and target size. The error bars represent standard error.

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 18 shows how strategy affected

performance of both techniques, and how more zoom results in more time. In addition, continuous zoom was constantly faster than discrete zoom, even when more zoom was used. No other significant interactions were found, which is consistent with our hypotheses.

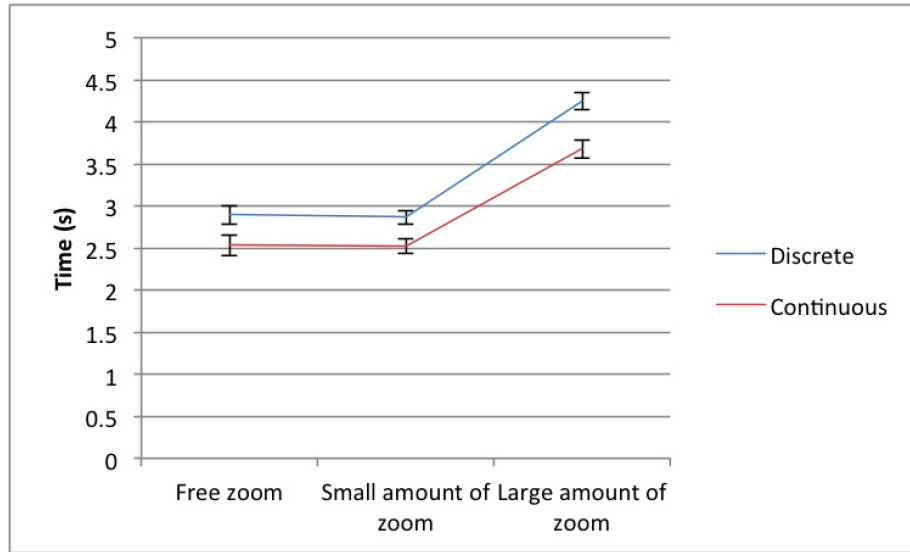


Figure 18 – Interaction between technique and strategy. The error bars represent standard error.

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 (discrete zoom: $F_{2,22} = 15.005$, $p < .001$; continuous zoom: $F_{2,22} = 15.005$, $p < .001$), but there was no significant difference between the zoom techniques.

There was a significant interaction between target size and technique ($F_{12,276} = 14.841$, $p < .001$), and pairwise comparisons show how each 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 confirm the hypothesis that the zoom techniques would result in virtually no errors, as can be seen in Figure 19. This also 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.

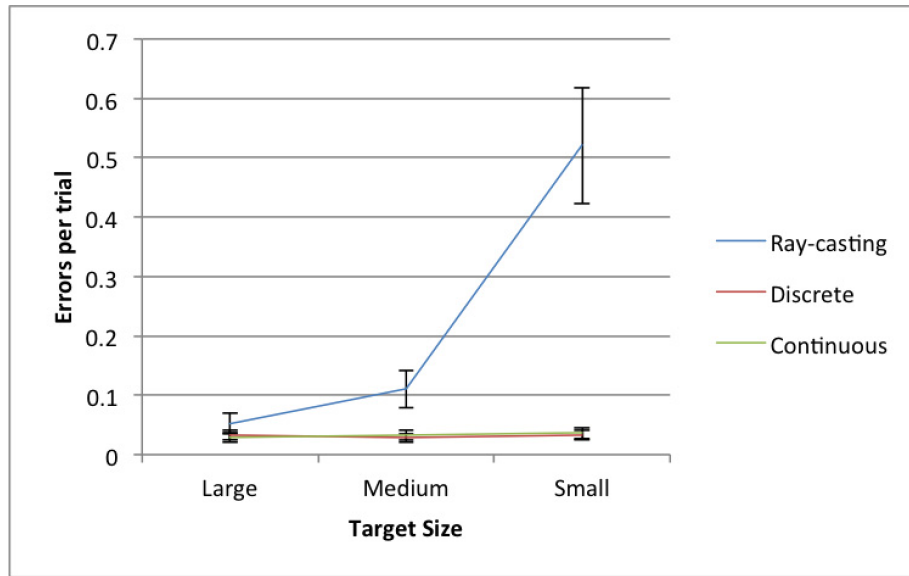


Figure 19 – Interaction between technique and target size. The error bars represent standard error.

This second analysis indicated a main effect of strategy ($F_{2,46} = 14.676, p < .001$). As can be seen in Figure 20, the large amount of zoom strategy provided more accuracy than the others for all sizes of targets. Even though the small amount of zoom and free zoom strategies had more errors, they still provided the same accuracy as the best case for ray-casting (i.e., when ray-casting is used with a large target).

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 questions by 17 participants. The other 7 participants preferred the discrete zoom technique for both questions. 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 progressive refinement techniques, even when taking into consideration larger targets. For strategy, 17 participants preferred to use a small amount of zoom, and 7 preferred to use a large amount of zoom.

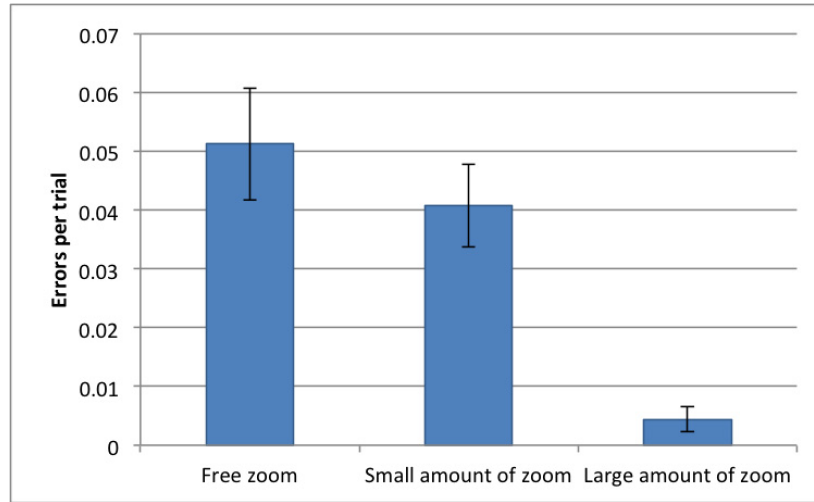


Figure 20 – Average number of errors per trial with each strategy. The error bars represent standard error.

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 ($n=17, W=33, p=0.44$), 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 21, 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.

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 5.08x, 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 21).

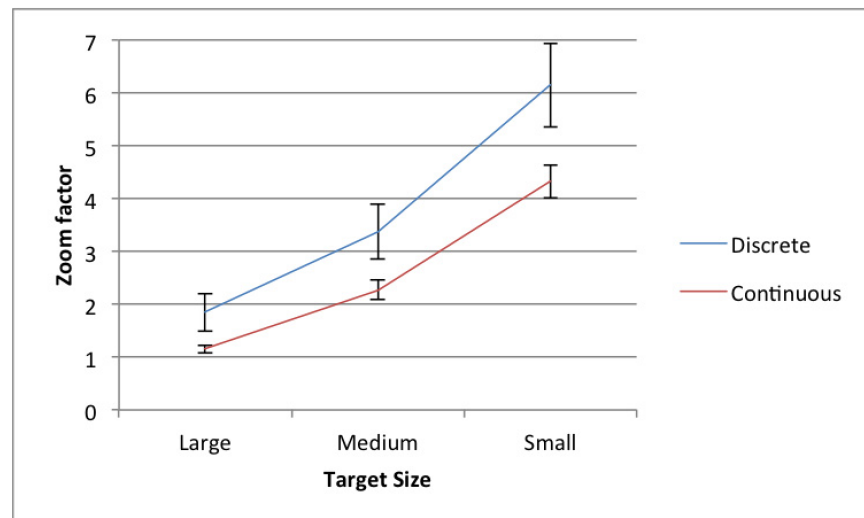


Figure 21 – Interaction between technique and target size. The error bars represent standard error.

4.2.5 Model Validation

As we did for experiment 1, we used information from both analytic and empirical evaluations to validate the predictive models we developed for the techniques. We first performed an analysis of the ray-casting data. Figure 22 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.

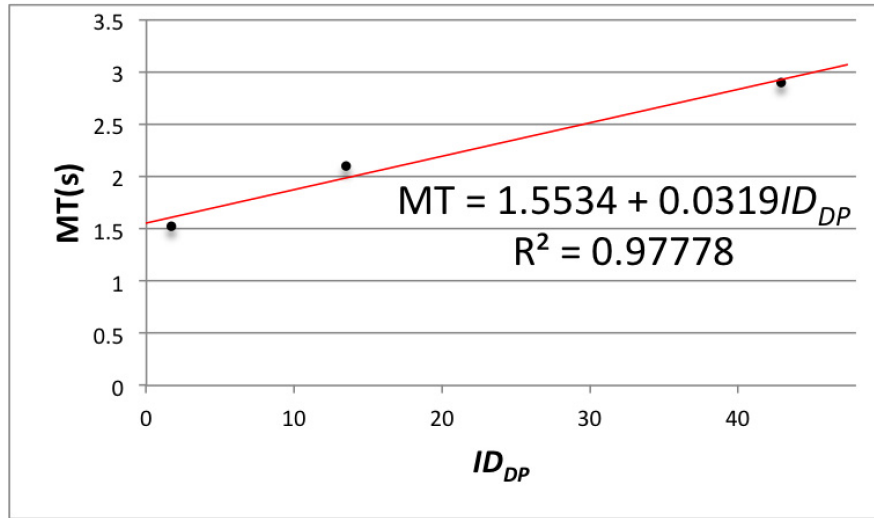


Figure 22 – Scatter plot and regression line for the ray-casting conditions.

The intercept of the regression line is 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.

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.

As can be seen in Figure 23 (small amount of zoom) and Figure 24 (large amount of zoom), our predictive models fit the data for both techniques with a very high level of fit.

We note that 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 is reduced 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 3.05s and 3.37s. The reason for that is the reduced ID_{DP} 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.

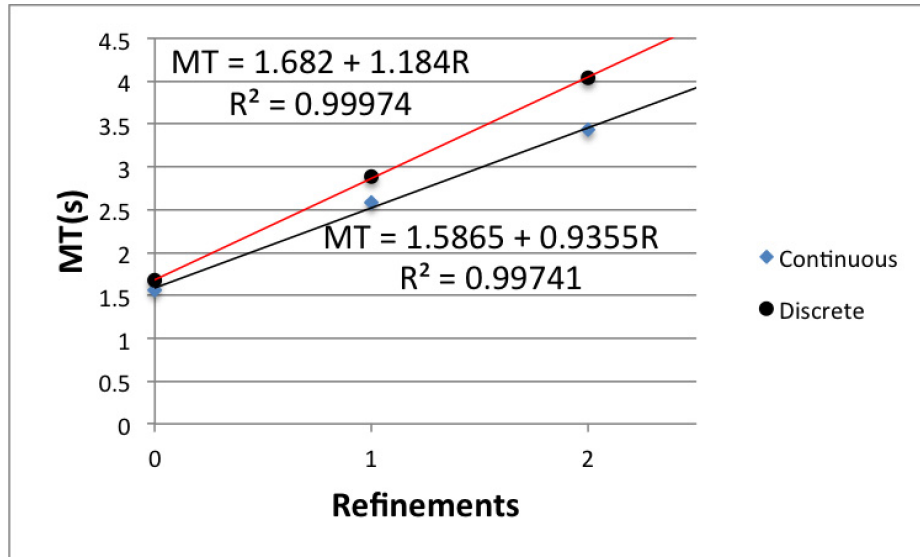


Figure 23 – Scatter plot and regression line for the continuous zoom and discrete zoom conditions with the small amount of zoom strategy. For the continuous zoom technique, refinement is defined by $\log_2 z$, where z is the final zoom level.

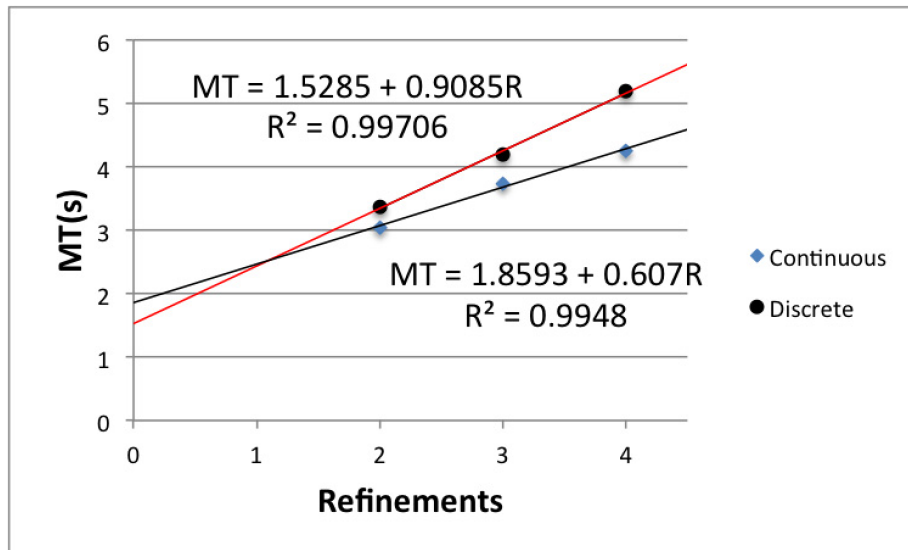


Figure 24 – Scatter plot and regression line for the continuous zoom and discrete zoom conditions with the large amount of zoom strategy. For the continuous zoom technique, refinement is defined by $\log_2 z$, where z is the final zoom level.

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} .

4.3 Discussion

The zooming techniques address the main limitation of SQUAD (the need for targets to be visually distinct). Using zoom as refinement steps, we decrease the number of selectable objects and increase the angular width of the target, making selection easier.

The analytic 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 by zooming more, for example, does increase accuracy but also increases the time to perform selection considerably. A strategy with just enough zooming to make the final target selection trivial seems 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. Furthermore, the investigation and evaluation of this trade-off helped us to partially answer our research question E1.1.

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 should 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. Finally, the results of our study helped us further answer our design question **D1** by showing once more that it is possible to design efficient progressive refinement techniques for single-object selection.

We also investigated the different strategy choices for refinement control to highlight differences between different design choices and begin to answer our evaluation question **E2**. 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 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 focused 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 efficiency is improved and the issues described by the users are addressed. 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 time for the final adjustment of the cursor would be comparable to continuous zoom. This hybrid technique would inherit the positive features of the continuous zoom techniques without the limitations of a long zooming time or difficulty in controlling the zoom window.

Considering both studies presented in this and the previous chapter, the main advantage of using progressive refinement techniques is accuracy. With SQUAD and the Zooming techniques, we achieved near-zero error rates in conditions where ray-casting could not. We found only a few cases (all with SQUAD), however, for which progressive refinement was also faster than ray-casting. For that reason, our next study focuses on modifying these techniques to provide the accuracy benefits without negatively affecting speed. Another important issue that we address in our next study is the real-world validity of the results. In both experiments discussed so far, we used controlled environments where users only performed selection tasks, in a non-realistic environment. In real-world tasks, however, users would be performing other tasks in addition to selecting targets, and environments exhibit varying degrees of complexity and clutter. Therefore, we evaluate which situations and scenarios are appropriate for the use of the different dimensions of progressive refinement.

5 Improvement and Evaluation of Progressive Refinement Techniques for Single-Object Selection

The work presented in chapters 3 and 4 showed the design of three different progressive refinement techniques: SQUAD, Continuous Zoom, and Discrete Zoom. Their performances were assessed in a comparison with the most commonly used immediate selection technique (ray-casting) in two different studies. In the next phase of our work, presented in this chapter, we focused on improving the design of our techniques for 3D selection to maximize performance and to continue answering our research question **D1**. Our goal was to design techniques that can be as fast and accurate as existing state-of-the-art immediate selection techniques. To evaluate our progress toward this goal, we compared the improved progressive refinement techniques to other state-of-the-art selection techniques, in addition to the baseline ray-casting technique, in both controlled environments and real-world scenarios. This helped us answer our evaluation questions **E1.1** and **E1.2**. This study also helped us investigate the effect of different design space options to further answer question **E2** through the comparison of our different techniques.

5.1 Improved Design of Single-Object Selection Techniques

For our final study of single-object selection techniques, we created two new selection techniques based on progressive refinement: Double Bubble and Flexible Rapid Incremental Zoom (FRIZ).

5.1.1 Double Bubble

Double Bubble is a new selection technique based on SQUAD, Expand (J. Cashion et al., 2012) and Bubble Cursor (Grossman & Balakrishnan, 2005). Similar to SQUAD and Expand, it implements selection in two phases: initial refinement using Bubble Cursor on screen-space, with the difference that it can only reduce its size down a pre-defined threshold; and, if there is more than one object inside the bubble, a menu where objects pop-out on the screen with original relative locations and distances preserved (Figure 25). Bubble Cursor is used in this second phase as well to allow fast direct selection of the target object. The next sections detail the design changes and rationale behind this technique for each phase separately.

Improving the Sphere-Casting Phase

SQUAD reduces the precision required to select small objects by casting a sphere into the environment and making all objects inside the sphere selectable. However, this approach is not very practical in cluttered environments, as the user may end up with a very large number of objects inside

the sphere, which will result in more time spent in the QUAD menu phase to disambiguate selection. The sphere in SQUAD has a large constant visual size that focuses on reducing the difficulty of every selection task to virtually zero, but it causes issues when performing selection in cluttered environments (as shown in chapter 3.2). Therefore, our first goal was to reduce the size of sphere without necessarily increasing the precision needed for pointing.



Figure 25 – Two selection phases of Double Bubble: Bubble Cursor using screen-space for initial in-context selection (left), and Bubble Cursor used to enhance selection of object in the menu.

In order to achieve this goal, we first implemented a 3D version of Bubble Cursor to automatically resize the sphere based on the distance between the ray and the objects in the environment, as well as the distance between the objects and the user. In this implementation, if there is no intersection between the ray and selectable objects, we cast a cone that is used to calculate candidates to be used as a reference to create a sphere. The candidate with the smallest distance in screen-space to the center of the cone is used to create a sphere with the minimum size necessary to encompass that object. If this size is smaller than a preset threshold size or if there is an intersection with the ray, we use the threshold as the size of the sphere to prevent it from becoming too small and requiring too much precision. This preset threshold size is smaller than in the original SQUAD. This way, the number of selectable objects for the first selection phase was reduced. However, this implementation highlighted issues related to the idea of sphere-casting itself.

In SQUAD, the sphere is always cast on the first surface it intersects. When objects are close together visually, but at different depths, it becomes hard to understand which objects will be sent to the second selection phase. This did not affect the results of our first study because all selectable objects were at the same depth, but it is a problem in 3D environments, as shown in Figure 26. Objects may

seem selectable because they are occluded by the sphere, but because they are not within the cast sphere's radius, they are not selectable. Therefore, a design goal was to reduce uncertainty in the first phase of selection.



Figure 26 – Illustration of the selection depth issue of sphere-casting with first person (left) and top-down view perspectives (right). Multiple objects are occluded by the sphere, as seen on the left image and marked by the white arrows on the right image, but are not selectable because they are not inside the sphere.

Our improved design for the first refinement phase solved the selection depth issue by using a slightly modified version of the original 2D Bubble Cursor. All visible objects are projected onto the screen as 2D objects, and only those objects are initially selectable. Selection is always done in screen space, and the bubble changes its radius based on the closest selectable pixel, which is associated with an object. The main difference to the original Bubble Cursor (Grossman & Balakrishnan, 2005) is that we introduced a minimum bubble size, and if the bubble contains more than one selectable object inside it when the user performs selection, they are sent to the second selection phase like in SQUAD. This implementation is shown in Figure 27, which has the user at the same position, pointing at the same place on the screen, and using the same minimum size as in Figure 26 to illustrate how they differed.

Improving the QUAD-Menu Phase

The second selection phase in SQUAD (refinement by QUAD-menu) requires very little precision because it reduces selection disambiguation to an iterative process of selecting the quadrant that contains the target object until it is the only one left. While this process can be very fast if the number of iterations is low, each refinement step involves a visual search time that can make it significantly slower when there is a large number of objects in the menu.



Figure 27 – Illustration of the 2D Bubble Cursor implementation in its minimum bubble size, with first person (left) and top-down view perspectives (right). It allows the selection of all visible objects inside the bubble, independent of depth (highlighted on the right).

In order to solve this issue, we created a few alternative designs that focused on different approaches. The first approach was to reduce the time required for selecting a quadrant by using simple gestures. In this case, we allowed users to flick the pointing device in the direction of the quadrant they want to select instead of pointing at it (e.g., flicking the device to the left selects the left quadrant). While this does seem to improve the overall time for selection of individual quadrants, it does not reduce the number of iterations or the visual search time to find which quadrant contains the target. Our second alternative solution was to remove the quadrants and display the objects in a grid, similar to the Expand technique (J. Cashion et al., 2012), and use direct selection of objects. Initially, this selection step was implemented with ray-casting, but precision requirements scaled very quickly as the number of objects inside the initial selection grew. To minimize this issue, we decided to use the original version of Bubble Cursor (Grossman & Balakrishnan, 2005) for the menu selection. However, Bubble Cursor itself does not solve this issue completely since its performance approaches that of ray-casting as screen clutter grows. While we limit the number of objects in the menu by using a minimum size for the sphere in the first phase, this can still be an issue in dense and cluttered environments.

All of these changes reduce the time it takes to select an object, but they do not solve SQUAD's main limitation related to its QUAD-menu: it requires the target object to be visually distinct, since selection is done out-of-context. We made a few different changes to our design to solve this issue. First, we made the background of the menu transparent and introduced a quick animation that shows the objects moving from their original position to their position in the menu. This allows users to

keep track of their target object as it is placed in the menu. Additionally, we show a line that connects the proxy objects in the menu to the original objects in the scene. Figure 28 shows this menu implementation.

The main drawback of this new menu implementation was that there was no specific order in which the selectable objects were distributed in the menu. In case users lose track of their target object during the initial animation, this creates an unnecessary visual search time in the case that the object is visually distinct (best case scenario), or it requires the user to point to each one of the candidate objects, follow the line that connects to the original object, and find the object they want (worst case scenario). In order to solve this issue, we changed our menu so that the objects are distributed based on their original spatial layout. For example, if object A is to the left of object B when they are selected, object A will be to the left of object B in the menu as well. This means relative locations of objects are preserved in the menu. Similarly, we also preserve relative distances between objects. For instance, if object C is really close to object A, but far from object B, this will be reflected in the menu. This is done by calculating the relative position of all the objects inside the sphere in screen space, and remapping that to the entire screen. In practice, this final version of our menu works like a popped-out and zoomed-in version of all objects that are selectable from the first phase, as seen in Figure 25.



Figure 28 – Initial implementation of the Double Bubble technique.

5.1.2 Flexible Rapid Incremental Zoom (FRIZ)

The best feature of our Discrete Zoom technique was the low precision it required in selecting the quadrants for zoom. Its main issue, however, was that every refinement phase required users to perform a new visual search. On the other hand, while Continuous Zoom required more precision than Discrete Zoom, it only required users to perform visual search once. Instead of improving each

technique separately, we decided to combine the strengths of each technique into one improved design.

To maintain low precision requirements, we use discrete zoom steps: every time the user presses the zoom in/out button, a pre-determined amount of zoom is used and a short animation is shown, like in the Discrete Zoom technique. A zoom preview window indicates what part of the screen the next level of zoom will show. Several alternatives for zoom window positioning were considered for the final design. The zoom window could always have the cursor at the center, which would allow users to zoom into regions of the screen not visible in the original view. The zoom window could be snapped to the edges of the screen to avoid this effect, but in either alternative the user would have to re-adjust the cursor position every time zoom is performed.

The solution was to then zoom in the direction of the cursor position and guarantee that the cursor will point at the same object in the 3D environment after zooming in or out. We do this by matching the position of the cursor in the current zoom level and the next level (illustrated by the zoom preview window), creating an off-axis projection frustum that has the cursor at the same position. This feature also eliminates visual search once a refinement is made.

Figure 29 shows the final design and implementation of this technique, with two refinement steps in which the cursor stays in place and the image simply grows around it, as indicated by the blue preview window.

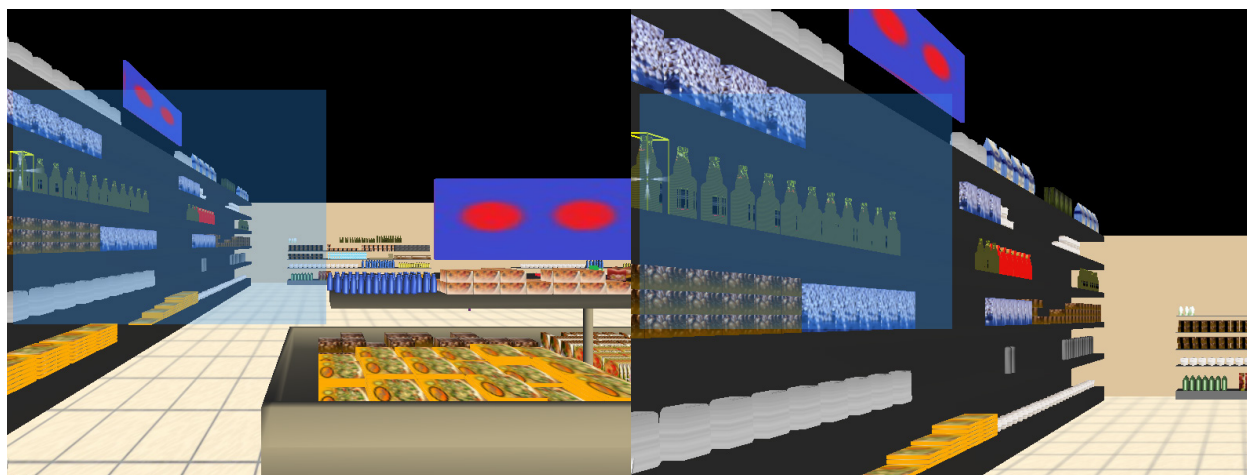


Figure 29 – Two refinement steps of the improved zoom technique.

5.2 Evaluation of Improved Technique Designs

In order to evaluate the techniques described in the previous section, we conducted a third experiment comparing Double Bubble and FRIZ to standard ray-casting as well as existing state-of-

the-art immediate selection techniques. We evaluated the same task as in the previous studies in four different conditions with varying levels of ecological validity. We considered this experiment to be a selection “shootout” since it compared the best progressive refinement and immediate selection techniques of which we were aware.

5.2.1 Goals and Hypotheses

The primary goal of this experiment was to assess the performance of progressive refinement techniques when compared to other state-of-the-art immediate selection techniques (and ray-casting as a baseline). We compared Double Bubble and FRIZ to two immediate selection techniques: DyCoDiR, which improves immediate selection by manipulating the **c/d ratio**; and Bubble Cursor, which uses a form of **snapping** to always make the target closest to the cursor selectable. As in our previous experiments, we focused on evaluating the main trade-off of progressive refinement techniques as compared to immediate selection techniques: using multiple selections versus performing a single selection step that requires more precision.

With this tradeoff in mind, we expected to find an interaction between technique and target size for the time to perform selection. We hypothesized that Double Bubble and Bubble Cursor would not be affected by target size, while ray-casting, FRIZ and DyCoDiR would be slow with small targets and fast with large targets. Additionally, we expected progressive refinement techniques to outperform other techniques in terms of accuracy, independently of target size.

We also expected there to be an interaction between technique and distractor density. While Double Bubble and Bubble Cursor techniques would be affected by the density of objects around the object, ray-casting, FRIZ and DyCoDiR should show no effects of density on time. We expected progressive refinement techniques to provide better accuracy, independent of distractor density.

Our secondary goal was related to the validation of our results by using varying levels of ecological validity. We used the abstract environment from the prior studies as one condition, and the “supermarket” environment from the first 3DUI Grand Prize to represent a more realistic virtual world. As suggested by Argelaguet and Andujar (2013), most studies, including our own, evaluate selection in isolation, while selection is usually mixed with other tasks in real-world scenarios. Thus, in the supermarket, we compared versions both with and without user navigation. We hypothesized that we would find similar results for both controlled and real-world environments, but that participants would take more time and be more accurate when they were allowed to navigate in the environment to move closer to target objects.

5.2.2 Experimental Design

We used a mixed design with repeated measures. Our between-subjects variable was *technique* (Ray-casting, Double Bubble, FRIZ, DyCoDiR, Bubble Cursor), and our within-subjects variables were *ecological validity* (abstract, supermarket, supermarket with navigation), *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 radius (referred to as *distractor density*) (16, 64, 256). We also included a fourth level of ecological validity, in which users selected all existing targets in the environment without any control for order or starting position (called supermarket-free). This condition was evaluated separately from others, since there was no guarantee of target size or number of distractors for each selection task. Thus, the design was 5 techniques $\times$ ((3 levels of ecological validity $\times$ 3 target sizes $\times$ 3 distractor densities) + 1 highest level of ecological validity). The order of presentation of the level of ecological validity was counterbalanced, with the exception of the highest level, which was always the last condition. Within each condition of ecological validity (except the last), each of the nine combinations of target size and distractor density was repeated four times and presented in random order. For the purpose of statistical analysis of the effect of navigation, we grouped our two controlled supermarket conditions into one and added navigation as a separate variable.

The dependent variables were time to complete the task, average number of errors made during the trials, distance traveled (in the case of the highest ecological validity condition), reported ease of use, reported difficulty to select small targets, and reported difficulty to select targets in high distractor density condition. The last three were measured using a Likert scale (1-7) after the completion of different conditions.

5.2.3 Apparatus

We used the same setup as in previous experiments: a back-projected VisBox-SX system, with only one projector (monoscopic) to display the experimental environment on a 2.29m $\times$ 3.05m screen. The resolution of the graphics was 1400px $\times$ 1050px. A wireless Intersense IS-900 Wand was used for controlling the cursor on the screen and confirming a selection, and a wireless PlayStation Navigation Controller was used for navigation. Navigation was done by simply pushing the analog joystick forward or backward to move in that direction, and pushing it left or right to turn in the respective direction.

The environment used for the lowest ecological validity condition consisted of circular objects, and the task was the same for all techniques and similar to our first experiment. The 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.

The environment used for the higher ecological validity conditions was a virtual supermarket, which had areas that combine both a high level of visual clutter and objects with very little visible area. The task followed the same template as in the controlled experiment. In the two intermediate ecological validity conditions (supermarket without and supermarket with navigation), we avoided visual search by showing all objects in grayscale before the task started, with the exception of two: the yellow circle used to start the task and the target object highlighted in red with an arrow on top. The rest of the environment was rendered in full color once the user selected the yellow circle and the arrow disappeared.

In the highest ecological validity condition (supermarket-free), all targets were highlighted red with the arrow on top. The arrows were visible through occluding objects, and changed color to indicate the object they were pointing at was occluded.

5.2.4 Participants

We recruited 45 voluntary unpaid participants from the campus community to perform the study and achieve a total of nine participants per technique. Participants' ages ranged from 18 to 33 years, with a median age of 21. Twenty of the participants were female.

5.2.5 Procedure

Participants were first given the informed consent form to read and sign. Upon agreement, they proceeded to complete the background questionnaire. After that, they were shown the experimental setting and the apparatus. The initial tutorial phase was done with the easiest condition in the lowest ecological validity condition (Abstract environment) so they could understand the technique and the experiment. After the tutorial was completed, participants were introduced to the first ecological validity condition in a practice session, in which they had to successfully complete selection of each target size and density combination once.

After practicing, they were reminded that they had to perform the trials as quickly as possible while trying not to make mistakes, and then performed selection of four sets of each of the nine target size and density 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 level of ecological validity. 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 only data recorded for these trials was the number errors. While this could result in missing data values, it was not the case since no participant required more than five attempts to complete a trial. The target position was the same for all attempts of a given trial.

After the end of each set of trials, participants completed a set of rating-scale questions and rested for up to two minutes if necessary, then moved on to the next condition. This was done for each level of ecological validity, following the same protocol, until all four conditions were completed. The highest level of ecological validity was always presented last, while the presentation of the other three was counter-balanced.

Finally, participants filled out a post-hoc questionnaire, giving final overall ratings for the technique and comparing the different ecological validity levels.

5.2.6 Results

We could not conduct a single analysis with ecological validity as a variable, given the differences between target sizes, distractor density, object shapes and so on, as these would create confounds and increase the variance in our data. While we tried our best to replicate conditions between abstract and supermarket conditions, they are essentially different tasks done because they are done in different environments. The best we can do, in this case, is to compare the results we get from different ecological validity levels to see if they are similar. Therefore, we conducted three separate analyses with the different levels of ecological validity: abstract-only (since all the target sizes and environmental densities were consistent); supermarket and supermarket with navigation (since both conditions had the same starting positions and targets, and we wanted to evaluate navigation as a variable); and supermarket-free (since it was a single continuous task without control for the selection tasks and free navigation). For the first two analyses, we evaluated independent variables technique (between-subjects), target size, and density (within-subjects), with five different techniques (Ray-casting, Double Bubble, FRIZ, DyCoDiR, Bubble Cursor), three levels of target size (small, medium, and large) and three levels of density (low, medium, high). For the final supermarket condition, we only considered our independent variable technique (between-subjects) and also included the analysis of another dependent variable (distance traveled).

Abstract-only

For this analysis, we performed mixed model factorial ANOVAs with repeated measures on the study's dependent variables: time to complete a task, and number of errors per trial.

Time

The first analysis of time compares our progressive refinement techniques to DyCoDiR, Bubble Cursor and Ray-casting in the abstract environment (lowest ecological validity condition). Overall, there was a main effect of technique ($F_{4,40} = 15.370, p < .001$). The mean time for completion was 1.399s for Bubble Cursor, 1.717s for Double Bubble, 2.071s for FRIZ, 2.148s for DyCoDiR, and 2.684s for Ray-casting. As predicted, Ray-casting was significantly slower than Bubble Cursor ($F_{4,40} = 15.370, p < .001$), DyCoDiR ($F_{4,40} = 15.370, p < .038$), Double Bubble ($F_{4,40} = 15.370, p < .001$), and FRIZ ($F_{4,40} = 15.370, p = .011$). Additionally, Bubble Cursor was significantly faster than FRIZ ($F_{4,40} = 15.370, p = .004$) and DyCoDiR ($F_{4,40} = 15.370, p = .001$). This does not support our hypothesis that progressive refinement techniques would not be slower than other state-of-the-art immediate selection technique. However, this result does not take errors or the other ecological validity conditions into account.

There was also a main effect of target size ($F_{2,80} = 191.071, p < .001$). As expected, pairwise comparisons showed that participants took significantly longer to select the small targets when compared to medium ($F_{2,39} = 136.739, p < .001$) and large targets ($F_{2,39} = 136.739, p < .001$), and medium targets when compared to large targets ($F_{2,39} = 136.739, p < .001$). 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 ($F_{8,80} = 36.586, p < .001$). As indicated previously, for all techniques, participants performed significantly slower as the targets got smaller. As hypothesized, there was a significant difference between different target sizes for Ray-casting ($F_{2,39} = 155.887, p < .001$ for all pairwise comparisons), FRIZ ($F_{2,39} = 27.789, p < .001$ for all pairwise comparisons), and DyCoDiR ($F_{2,39} = 37.019, p < .001$ for all pairwise comparisons). We found no significant difference for target size for Double Bubble ($\alpha = 0.05$). However, we did find a significant difference between large and medium targets for Bubble Cursor ($F_{2,39} = 5.187, p = .007$). This interaction can be seen in Figure 30.

There was a main effect of distractor density ($F_{2,80} = 8.409, p < .001$). Pairwise comparisons showed that participants took significantly longer to select the targets in high density condition as opposed to low density ($F_{2,39} = 9.865, p < .001$). However, no significant differences were found between low and medium densities or medium and high densities for $\alpha = 0.05$. We further examine these effects by analyzing the interaction between variables.

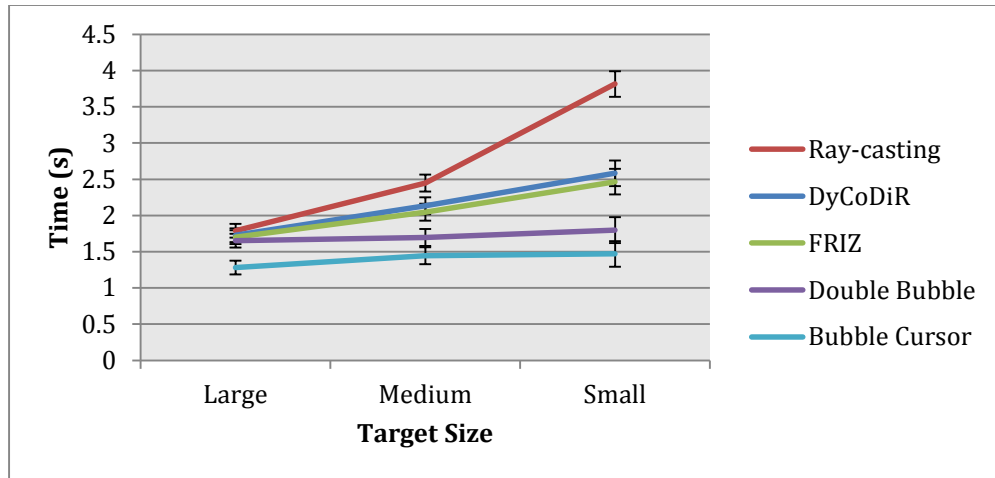


Figure 30 – Interaction between target size and technique for time in abstract-only.

The interaction between technique and distractor density was nearly significant ($F_{8,80} = 1.982$, $p=.059$). As expected, the pairwise comparison indicated a significant difference between low and high density conditions for Double Bubble ($F_{2,39} = 4.963$, $p=.012$) and Bubble Cursor ($F_{2,39} = 9.715$, $p<.001$). We found no significant difference for distractor density for the other techniques ($\alpha = 0.05$). This interaction can be seen in Figure 31.

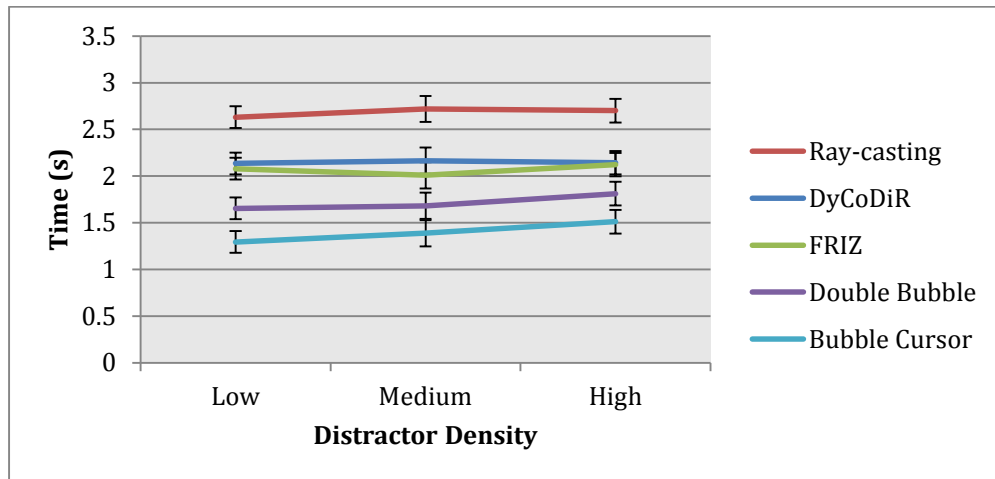


Figure 31 – Interaction between distractor density and technique for time in abstract-only.

There was no significant interaction between target size and distractor density, or between target size, distractor density and technique for $\alpha = 0.05$.

Errors

In this section, we present the results related to the average number of errors per trial in the Abstract

environment. As in the previous section, we compared our progressive refinement techniques to DyCoDiR, Bubble Cursor and Ray-casting in the abstract environment (lowest ecological validity condition). Overall, there was a main effect of technique ($F_{4,40} = 42.255, p < .001$). As predicted, Ray-casting had significantly more errors than Bubble Cursor ($F_{4,40} = 42.255, p < .001$), DyCoDiR ($F_{4,40} = 42.255, p < .038$), Double Bubble ($F_{4,40} = 42.255, p < 0.001$), and FRIZ ($F_{4,40} = 42.255, p = .011$). There was no significant difference between all the other techniques. This does not support our hypothesis that progressive refinement techniques would be more accurate than other state-of-the-art immediate selection techniques.

Additionally, we found a main effect of target size on accuracy ($F_{2,80} = 36.405, p < .001$). As expected, pairwise comparisons showed that, overall, participants made significantly more mistakes to select the small targets when compared to medium ($F_{2,39} = 34.229, p < .001$) and large targets ($F_{2,39} = 34.229, p < .001$), and medium targets when compared to large targets ($F_{2,39} = 34.229, p = .002$). By examining the interactions between all these variables, we are able to better understand these effects.

There was a significant interaction between technique and target size ($F_{8,80} = 26.667, p < .001$). By examining the pairwise comparisons, however, we only found a significant difference between different target sizes for Ray-casting ($F_{2,39} = 125.059, p < .001$ for all pairwise comparisons). We found no significant difference for target size for other techniques ($\alpha = 0.05$). This indicates that the effect of target size for Ray-casting was so strong that it caused the overall effect to be significant. This interaction can be seen in Figure 32.

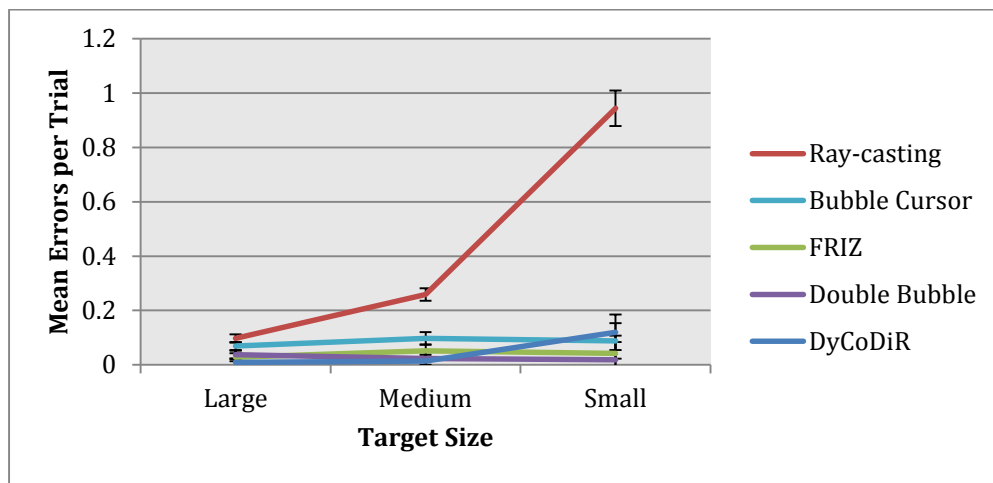


Figure 32 – Interaction between target size and technique for errors in abstract-only.

We did not find a main effect of distractor density on errors ($\alpha = 0.05$). However, the interaction between technique and distractor density was significant ($F_{8,80} = 4.089, p < .001$). The pairwise comparison indicated a significant difference between low and high density conditions for Bubble Cursor ($F_{2,39} = 3.707, p = .034$), as expected, but we did not find a significant effect for Double Bubble. Surprisingly, we also found a significant difference for Ray-casting ($F_{2,39} = 15.194, p < .001$ for comparisons between low and medium, and medium and high densities). As seen in Figure 33, participants made more errors in the medium distractor density condition when using Ray-casting. We found no significant difference for distractor density for the other techniques.

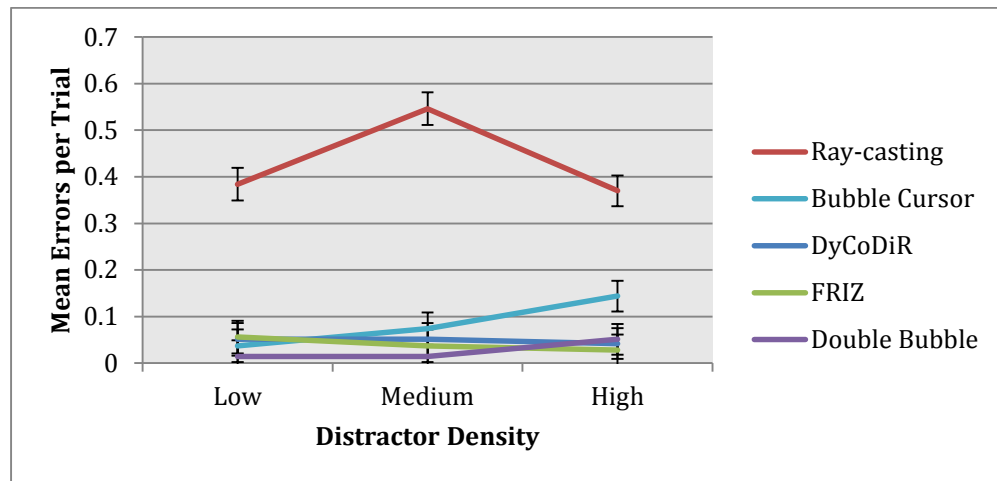


Figure 33 – Interaction between distractor density and technique for errors in abstract-only.

Supermarket without and with navigation

In the analysis of these next two levels of ecological validity, we also used mixed model factorial ANOVAs with repeated measures to evaluate the time to complete a task, and the number of errors per trial. We initially ran two separate ANOVAs, one without supermarket with navigation and another considering navigation as an independent variable and including supermarket with navigation. However, since the effects, interactions and pairwise comparisons were too similar between the two and validated the same hypotheses, we only report the complete analysis that includes supermarket with navigation.

Time

Surprisingly, there was no overall effect of technique for time ($\alpha = 0.05$). Since we did find such an effect in the Abstract environment, this result does not support our hypotheses that we would find

similar results in the different ecological validity conditions. Our hypothesis that all other techniques would be faster than ray-casting is also contradicted by this result. On the other hand, this supports the hypothesis that progressive refinement will not be slower than other state-of-the-art technique. Figure 34 illustrates the overall difference between the techniques in comparison to the abstract-only condition.

There was a main effect of target size ($F_{2,80} = 264.837, p < .001$). As expected and similar to the abstract-only analysis, pairwise comparisons showed that participants took significantly longer to select the small targets when compared to medium ($F_{2,39} = 165.306, p < .001$) and large targets ($F_{2,39} = 165.306, p < .001$), and medium targets when compared to large targets ($F_{2,39} = 165.306, p < .001$).

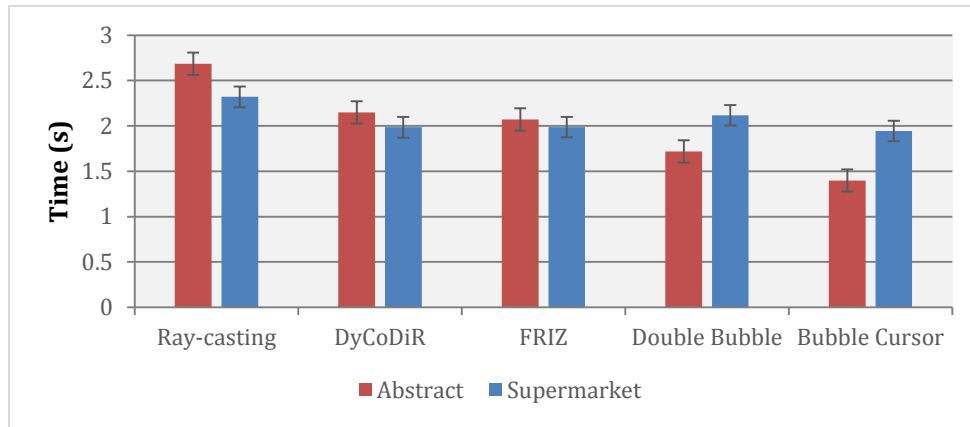


Figure 34 – Average time for completion of tasks in the Abstract-only and Supermarket conditions.

There was a significant interaction between technique and target size ($F_{8,80} = 14.667, p < .001$). As hypothesized, there was a significant difference between different target sizes for Ray-casting ($F_{2,39} = 101.242, p < .001$ for all pairwise comparisons), FRIZ ($F_{2,39} = 17.516, p < .001$ for comparisons between small and medium, and small and large; $p = .031$ between medium and large), and DyCoDiR ($F_{2,39} = 37.019, p < .001$ for all pairwise comparisons). Surprisingly, we also found a significant difference for target size for Double Bubble ($F_{2,39} = 3.944, p = .028$, with a significant difference between small and large targets, $p = .021$) and Bubble Cursor ($F_{2,39} = 42.427, p < .001$ for all pairwise comparisons). This interaction can be seen in Figure 35.

There was a main effect of distractor density ($F_{2,80} = 96.126, p < .001$). Pairwise comparisons showed that participants took significantly longer to select the targets in high density condition as opposed to low ($F_{2,39} = 103.144, p < .001$) and medium densities ($F_{2,39} = 103.144, p < .001$), as well as a significant difference between medium and low densities ($F_{2,39} = 103.144, p < .001$).

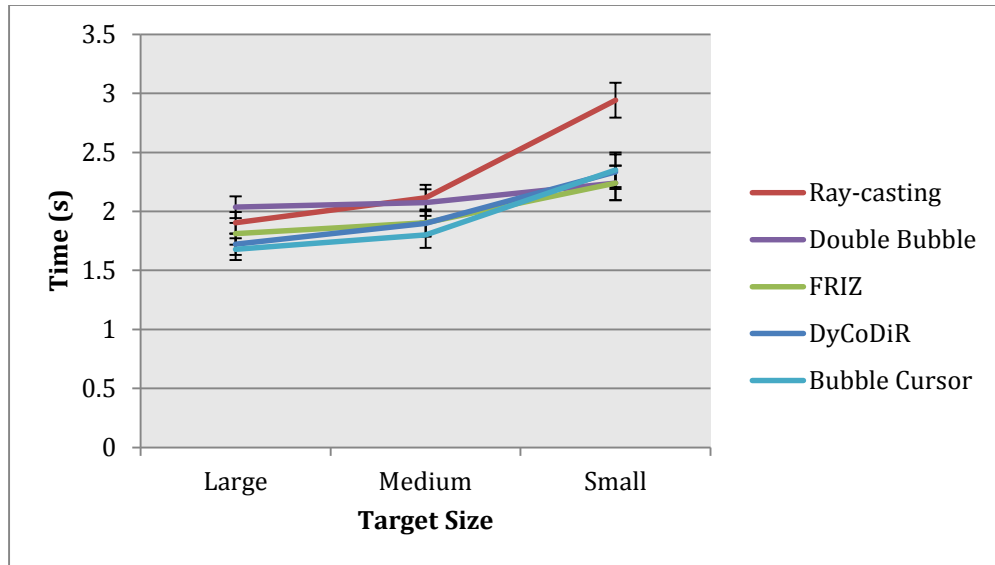


Figure 35 – Interactions between technique and target size in the supermarket conditions.

The interaction between technique and distractor density was significant ($F_{8,80} = 5.590, p < .001$). As expected, the pairwise comparison indicated a significant difference between all conditions for Double Bubble ($F_{2,39} = 47.4, p < .001$) and between low and high as well as medium and high conditions for Bubble Cursor ($F_{2,39} = 37.052, p < .001$). Surprisingly, we also found significant differences for distractor density for the other techniques (Ray-casting: $F_{2,39} = 17.782, p < .001$; FRIZ: $F_{2,39} = 4.124, p = .024$; DyCoDiR: $F_{2,39} = 18.028, p < .001$). This interaction can be seen in Figure 36.

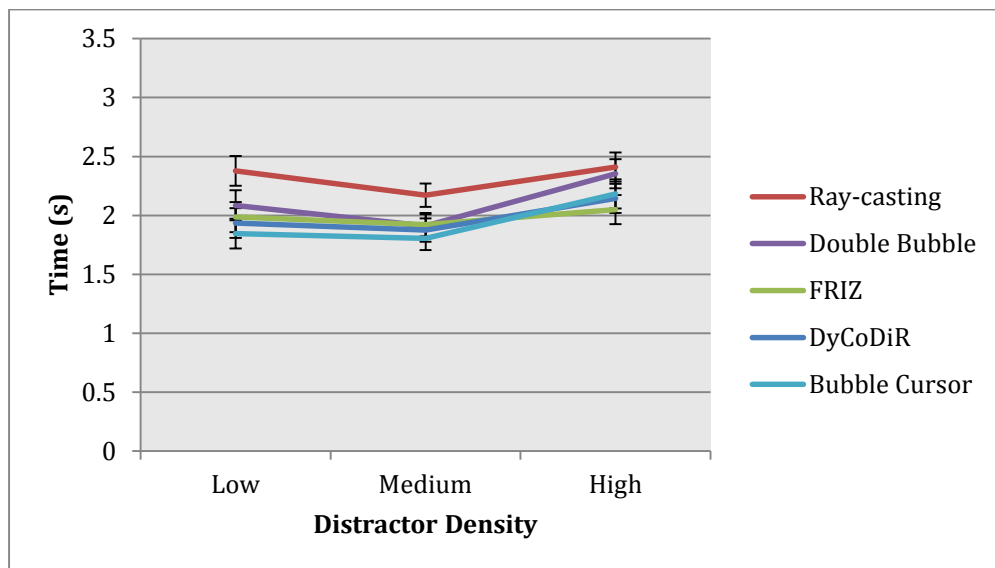


Figure 36 – Interactions between technique and distractor density in the supermarket conditions.

We also found a significant interaction between target size and distractor density ($F_{4,160} = 15.224$, $p < .001$, with significant differences between all pairwise comparisons at $\alpha = 0.05$), as well as between target size, distractor density, and technique ($F_{16,160} = 4.719$, $p < .001$), showing that these variables affect each other in more realistic environments.

We also considered navigation as an independent variable, and found a significant effect on time ($F_{1,40} = 7.027$, $p = .011$). As expected, we found that participants spent significantly more time to perform selection in the supermarket with navigation condition (higher ecological validity), even though they were not required to navigate (the mean selection time without navigation was 2.007s, while the mean selection time with navigation was 2.134s). The interaction between navigation and technique was not significant for $\alpha = 0.05$.

We also found significant interactions between navigation and target size ($F_{2,80} = 11.056$, $p < .001$, with a significant effect of navigation on medium and small targets for $\alpha = 0.05$), and navigation and distractor density ($F_{2,80} = 3.654$, $p = .03$, with a significant effect of navigation on low and high densities for $\alpha = 0.05$). These results support our hypothesis that navigation affects selection speed.

Errors

This section presents the analysis of the average number of errors per trial in both supermarket without navigation and supermarket with navigation conditions. We found a main overall effect of technique for errors ($F_{4,40} = 11.636$, $p < .001$). As expected and similar to the results of the abstract-only analysis, Ray-casting had significantly more errors than DyCoDiR ($F_{4,40} = 11.636$, $p = .005$), Double Bubble ($F_{4,40} = 11.636$, $p = 0.016$), and FRIZ ($F_{4,40} = 11.636$, $p = .007$). Surprisingly, we found no difference between Ray-casting and Bubble Cursor ($\alpha = 0.05$), and Bubble Cursor had significantly more errors than DyCoDiR ($F_{4,40} = 11.636$, $p < .001$), Double Bubble ($F_{4,40} = 11.636$, $p < 0.001$), and FRIZ ($F_{4,40} = 11.636$, $p < .001$). Again, this result does not support our hypothesis that we would find similar results in the different ecological validity conditions, but it does partially support the hypothesis that progressive refinement can be more accurate than other state-of-the-art immediate selection techniques. Figure 37 illustrates the overall difference between the techniques in comparison to the abstract-only condition.

There was a main effect of target size on the number of errors ($F_{2,80} = 36.531$, $p < .001$). As expected and similar to the abstract-only analysis, pairwise comparisons showed that participants took significantly longer to select the small targets when compared to medium ($F_{2,39} = 11.636$, $p < .001$)

and large targets ($F_{2,39} = 11.636, p < .001$). There was no significant difference between medium targets when compared to large targets for $\alpha = 0.05$.

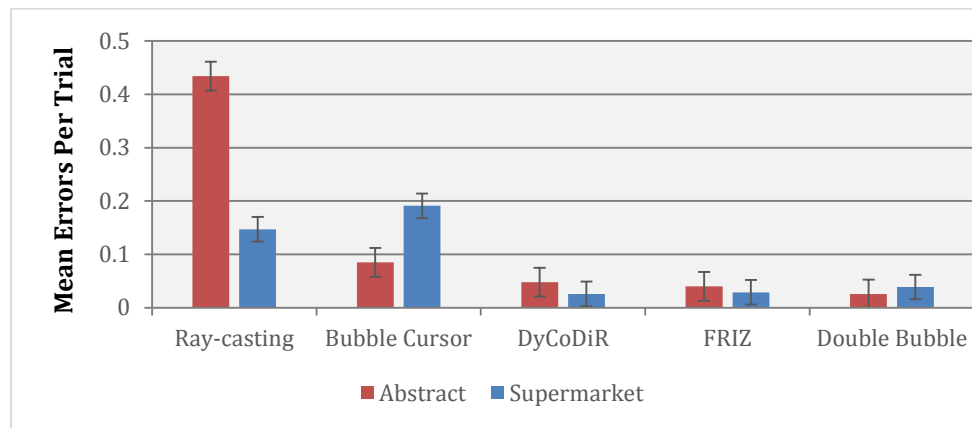


Figure 37 – Overall mean errors per trial for each technique in the abstract-only and supermarket conditions.

In order to understand these results, we looked at the interaction between technique and target size ($F_{8,80} = 11.197, p < .001$), shown in Figure 38. Similar to the abstract condition, participants made significantly more mistakes as targets get smaller with Ray-casting ($F_{2,39} = 101.242, p < .001$, for the pairwise comparisons between small and medium, and small and large target sizes). Additionally, as shown in the overall mean errors, we also found a significant effect for Bubble Cursor ($F_{2,39} = 17.516, p < .001$ for comparisons between small and medium, and small and large target sizes). Finally, these results further increase the evidence against the hypothesis about ecological validity, since Bubble Cursor was not significantly worse than other techniques in the abstract condition.

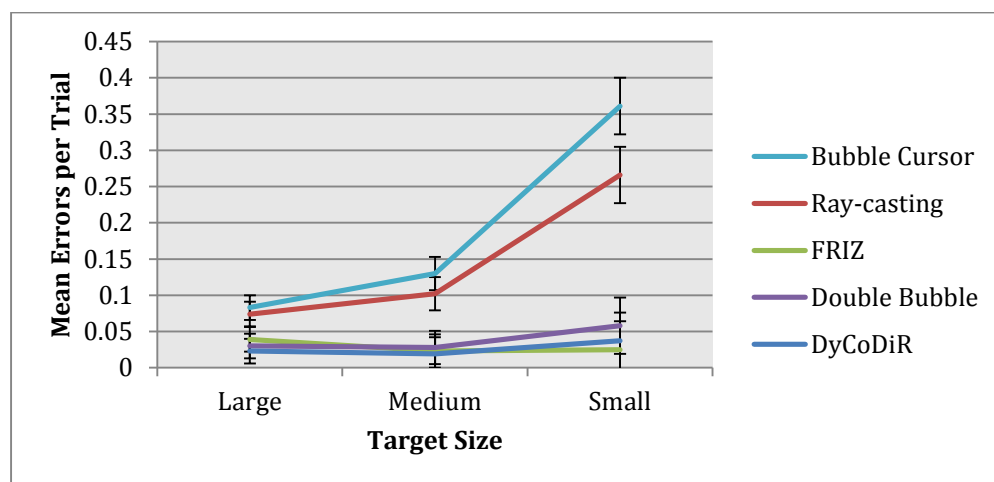


Figure 38 – Interaction between target size and technique for errors in the supermarket conditions.

Similar to target size, we found a main effect of distractor density ($F_{2,80} = 21.158, p < .001$). Pairwise comparisons showed that participants made significantly more mistakes when selecting targets in the high density condition as opposed to low ($F_{2,39} = 103.144, p < .001$) and medium densities ($F_{2,39} = 103.144, p < .001$). There was no significant difference between the medium and low density conditions ($\alpha = 0.05$).

The interaction between technique and distractor density was also significant ($F_{8,80} = 13.613, p < .001$). As expected, the pairwise comparison indicated a significant difference between medium and high density for Double Bubble ($F_{2,39} = 3.226, p = .05$) and between low and high as well as medium and high conditions for Bubble Cursor ($F_{2,39} = 41.562, p < .001$). Surprisingly, we also found significant differences for distractor density for Ray-casting ($F_{2,39} = 19.025, p < .001$), more specifically between low and medium with a significantly lower error rate for medium density targets. This interaction can be seen in Figure 39.

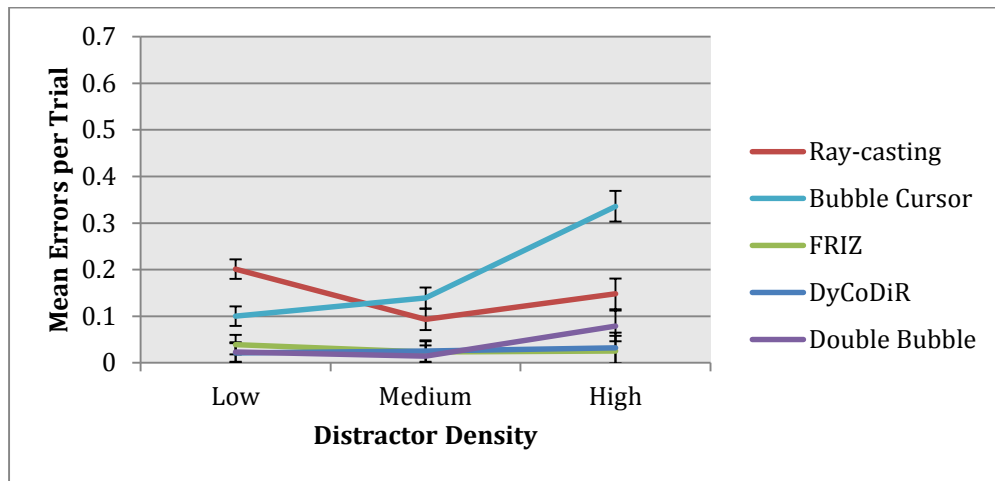


Figure 39 – Interaction between distractor density and technique for errors in the supermarket conditions.

Similar to the results related to time, we also found a significant interaction between target size and distractor density ($F_{4,160} = 5.08, p = .001$, with significant differences between pairwise comparisons for small and medium target sizes, low and high as well as medium and high densities at $\alpha = 0.05$). There was a significant interaction effect between target size, distractor density, and technique ($F_{16,160} = 3.883, p < .001$), further showing that these variables affect each other in more realistic environments.

We also analyzed the effects of navigation in errors, and found a nearly significant main effect ($F_{1,40} = 4.032, p = .051$). We found that participants make significantly fewer errors when they can navigate

towards the target. Investigating the significant interaction between navigation and technique ($F_{4,40} = 3.679, p=.012$), we only found significant differences between navigation conditions for Ray-casting ($F_{1,40} = 6.308, p=.016$) and Bubble Cursor ($F_{1,40} = 11.213, p=.002$). All other techniques did not present a significant effect of navigation on errors. These results partially support our hypothesis that navigation does affect accuracy in selection.

We also found significant interactions between navigation and target size ($F_{2,80} = 6.86, p=.002$, with a significant effect of navigation on medium and small targets for $\alpha = 0.05$); navigation, target size and technique ($F_{8,80} = 2.092, p=.046$); and navigation, distractor density and technique ($F_{8,80} = 2.513, p=.017$).

Supermarket-free

While we used a mixed model factorial ANOVA to analyze the other ecological validity conditions, there were no within-subjects variables or repeated measures in this condition. Thus, we used a one-way ANOVA and contrast analysis with Bonferroni correction to perform pairwise comparisons between the different techniques.

Time

There was no overall effect of technique on time ($\alpha = 0.05$). As in the other supermarket conditions, this result does not support our hypothesis that we would find similar results in the different ecological validity conditions or that all other techniques would be faster than ray-casting. However, this supports the hypothesis that progressive refinement will not be slower than other state-of-the-art technique. Figure 40 shows the average time to select a target for each technique.

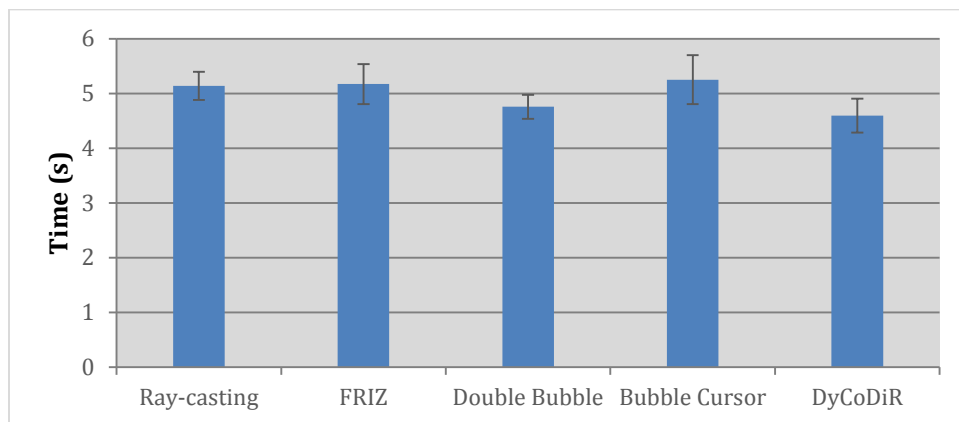


Figure 40 – Average time to select a target in the supermarket-free condition.

Errors

We found a main effect of technique on the number of errors made ($F_{4,40} = 3.287$, $p=.02$). The pairwise comparison showed that Double Bubble had significantly fewer errors than Bubble Cursor ($p=.029$), with 0.021 average errors per trial compared to 0.24, and that DyCoDiR had significantly fewer errors than Bubble Cursor as well ($p=.047$), with 0.034 average errors per trial compared to 0.24. This is illustrated in Figure 41. Surprisingly, there was no difference between ray-casting and any of the other techniques. This does not support our hypothesis that ray-casting would always cause more errors than other techniques.

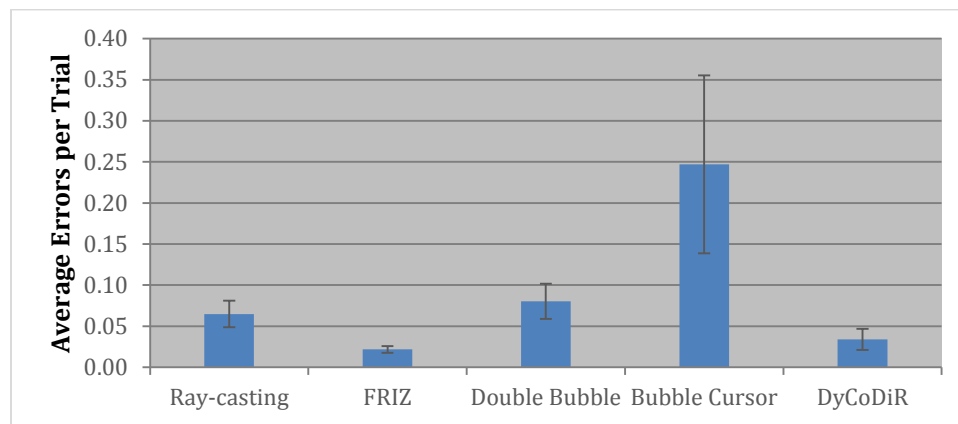


Figure 41 – Average number of errors per trial in the supermarket-free condition.

Distance Traveled

There was a significant effect of technique on the distance traveled to select all targets ($F_{4,40} = 7.474$, $p<.001$). The pairwise comparison between techniques showed that users who used Ray-casting and Bubble Cursor navigated more than all other techniques ($p<.01$ for all pairwise comparisons). The mean distance traveled was around 520m for both Ray-casting and Bubble Cursor compared to approximately 370m for the other techniques. There were no significant differences among the other techniques.

User Preference

For the subjective ratings regarding selection technique, participants were instructed to fill out a survey immediately after completing each level of ecological validity, and to rate the techniques on a seven-point scale for ease of use, the difficulty of using the technique when the targets were small, and the difficulty when selecting targets with high distractor density. We evaluated each question

with a Kruskal-Wallis test and subsequent Mann-Whitney tests for pairwise comparisons. At the end of all trials, we also asked participants to rate overall ease of use and ease of learning (both reported at the end of the ease of use subsection), and to choose the ecological validity conditions where it was easiest/hardest to select small targets and in high distractor density (reported at the end of their corresponding subsections).

Ease of Use

Each level of ecological validity was evaluated separately first. We found an overall effect of technique for the question about ease of use in the **abstract environment** ($H=13.325$, 4 *df.*, $p=.01$). Pairwise comparisons showed that Ray-casting ($mdn=4$) was ranked significantly more difficult to use than FRIZ ($mdn=2$) ($U=13$, $p=.013$), Double Bubble ($mdn=1$) ($U=11$, $p=.007$), and Bubble Cursor ($mdn=1$) ($U=13$, $p=.013$). There were no significant differences for Ray-casting ($mdn=4$) compared to DyCoDiR ($mdn=3$) ($U=29.5$, *insignificant*). Additionally, we found DyCoDiR ($mdn=3$) to be ranked significantly harder to use than Double Bubble ($mdn=1$) ($U=18$, $p=.037$), and nearly significant when compared to FRIZ ($mdn=2$) ($U=21$, $p=.075$), and Bubble Cursor ($mdn=1$) ($U=21$, $p=.074$).

We also found an overall effect of technique for reported ease of use in the **supermarket without navigation** condition ($H=12.018$, 4 *df.*, $p=.017$). Pairwise comparisons showed that Ray-casting ($mdn=3$) was ranked significantly more difficult to use than FRIZ ($mdn=2$) ($U=12$, $p=.009$), and Double Bubble ($mdn=2$) ($U=17$, $p=.029$). There were no significant differences for ray-casting ($mdn=3$) compared to DyCoDiR ($mdn=2$) ($U=26$, *insignificant*), and Bubble Cursor ($mdn=3$) ($U=35.5$, *insignificant*). We also found Bubble Cursor ($mdn=3$) to be ranked significantly harder to use than FRIZ ($mdn=2$) ($U=12$, $p=.009$) and Double Bubble ($mdn=2$) ($U=17$, $p=.029$).

We did not find a significant effect of technique for ease of use in the **supermarket with navigation** ($H=5.899$, 4 *df.*, *insignificant*) and **supermarket-free** conditions ($H=4.191$, 4 *df.*, *insignificant*).

In the final overall question about ease of use across all ecological validity conditions, we found no significant effect of technique ($H=7.113$, 4 *df.*, *insignificant*). We also found no significant effect of technique on the reported ease of learning ($H=6.714$, 4 *df.*, *insignificant*).

Difficulty when Selecting Small Targets

We found an overall effect of technique for the question about *ease of use* in the **abstract environment** ($H=23.620$, 4 *df.*, $p<.001$). Pairwise comparisons showed that Ray-casting ($mdn=7$) was rated significantly more difficult to use when selecting small targets than FRIZ ($mdn=4$) ($U=7.5$,

$p=.003$), Double Bubble ($mdn=2$) ($U=.5$, $p<.001$), Bubble Cursor ($mdn=3$) ($U=2.5$, $p=.001$), and DyCoDiR ($mdn=6$) ($U=15$, $p=.018$). Additionally, we found DyCoDiR ($mdn=6$) to be ranked significantly harder to use than Double Bubble ($mdn=2$) ($U=8.5$, $p=.003$) and Bubble Cursor ($mdn=3$) ($U=14.5$, $p=.019$), and nearly significant when compared to FRIZ ($mdn=2$) ($U=21$, $p=0.079$).

We also found an overall effect of technique for reported ease of use in the **supermarket without navigation** condition ($H=19.437$, 4 df , $p=.001$). Pairwise comparisons showed that Ray-casting ($mdn=6$) was ranked significantly more difficult to use to select small targets than FRIZ ($mdn=3$) ($U=8$, $p=.003$), Double Bubble ($mdn=3$) ($U=1$, $p<.001$), and DyCoDiR ($mdn=4$) ($U=26$, $p=.015$). There were no significant differences for ray-casting ($mdn=6$) compared to Bubble Cursor ($mdn=4$) ($U=25.5$, *insignificant*). We also found Bubble Cursor ($mdn=4$) to be ranked significantly harder to use for small targets than FRIZ ($mdn=3$) ($U=18.5$, $p=0.046$) and Double Bubble ($mdn=3$) ($U=6$, $p=0.001$). Finally, we also found a nearly significant difference between Double Bubble ($mdn=3$) and DyCoDiR ($mdn=4$) ($U=21$, $p=0.072$).

We did not find a significant effect of technique for ease of use in the **supermarket with navigation** ($H=6.36$, 4 df , *insignificant*) and **supermarket-free** conditions ($H=1.155$, 4 df , *insignificant*).

Difficulty when Selecting with High Distractor Density

For the question about difficulty selecting targets with high distractor density, we found no significant effects of technique on ease of use: **abstract** ($H=4.226$, 4 df , *insignificant*), **supermarket without navigation** ($H=8.926$, 4 df , *insignificant*), **supermarket with navigation** ($H=6.36$, 4 df , *insignificant*) **supermarket-free** ($H=1.155$, 4 df , *insignificant*).

5.3 Discussion

The design of Double Bubble and FRIZ addresses some of the major limitations and performance issues of SQUAD and the progressive refinement zooming techniques. Double Bubble uses a two-step selection process based on SQUAD, Bubble Cursor and the Expand technique to reduce the number of objects that are selected in the first step and reduce precision requirements in both steps. It also provides a number of cues to improve out-of-context selection in the second step, including conserving the original spatial layout of the selected objects. FRIZ is a technique that combines the strengths of both Discrete and Continuous Zoom to reduce the time it takes to zoom by using discrete steps and reduces the required precision to select the target by zooming around the cursor. These designs further support our hypotheses for our research question **D1**, which is about creating

selection techniques based on progressive refinement for single-object selection that can be as fast as immediate selection techniques, with the benefit of being easier to use and more accurate with less effort.

Our empirical evaluation also supported the research question **D1**, and focused on answering research questions **E1.1** and **E1.2**, which are related to the ecological validity of our results. We compared Double Bubble and FRIZ to Ray-casting and two state-of-the-art immediate selection techniques (Bubble Cursor and DyCoDiR) in a study with four different ecological validity conditions: an abstract environment, which had high control over our independent variables (target size and distractor density) like in our previous studies; a supermarket environment in which users performed very similar tasks as in abstract, but with less control over target size and distractor density; the exact same tasks in the supermarket environment with the added choice of navigation; and a final condition in the supermarket in which participants had to select all targets in a single continuous run, with as much navigation as needed and in no specific order.

While the goal of evaluating the different ecological validity conditions was initially considered a secondary goal, it turned out to be one of the most important results for this study. As we had hypothesized, all techniques were faster and more accurate than Ray-casting in the abstract condition. We also saw that Bubble Cursor was faster than FRIZ and DyCoDiR, while our progressive refinement techniques were not more accurate than Bubble Cursor or DyCoDiR. While these results do not entirely support our hypothesis that progressive refinement can be as fast as immediate selection techniques, the results for the intermediate ecological validity conditions were surprisingly different.

First of all, we did not find an overall effect of technique on time to select targets in the supermarket environment. This may be due to the increase in variance in the data since we did not have as much control over target sizes and distractor density, but it nonetheless means that, for example, Ray-casting was not slower than any other technique. Additionally, different than expected, we also found a significant effect of target size on time for Bubble Cursor and, to a lesser extent, Double Bubble. This effect can be partially explained if we look at our accuracy results. While our progressive refinement techniques and DyCoDiR were all significantly more accurate than Ray-casting when considering all target sizes and distractor densities, Bubble Cursor performed significantly worse than all, with the exception of Ray-casting.

This issue raised the question that, if the abstract environment was made to mimic real-world situations, how is it possible that the performance of Bubble Cursor is so different when other

techniques have similar performance? We believe this to be an issue inherent to these types of virtual environments. While we are able to replicate different target sizes and distractor densities in the abstract environment, we are not able to accurately portray the layout of objects in the environment. In the supermarket environment, either because of their shape or because they are at different depths, objects are always right next to each other and there are no empty spaces for Bubble Cursor to actually be useful, so it ends up requiring users to point directly to their targets most of the time, just like Ray-casting.

Also related to the results of the intermediate supermarket conditions, we found that navigation made selection slower, as expected, but also made it more precise for techniques that had significantly more errors without navigation (i.e., Bubble Cursor and Ray-casting). This means that participants compensated for the amount of precision required by the techniques by moving towards the object.

The results of the final supermarket condition show that there was no significant effect of technique on time, even though Ray-casting and Bubble Cursor required a significantly greater amount of navigation than others. Additionally, the only significant effects on errors were that Double Bubble and DyCoDiR were significantly more accurate than Bubble Cursor. These results might be interpreted to mean that Ray-casting is good enough in the context of real applications. However, in our other studies where ray-casting was directly compared to other techniques, our progressive refinement techniques were always preferred for ease of learning and ease of use. Moreover, while these results are certainly important, the fact that there was very little control over the way users performed the tasks (participants chose the order, how much they would navigate, etc) increased variance in the data and may have influenced the results. Differences might only be seen with a much larger sample.

Interestingly, user preference also shows that as we increase the level of ecological validity, the difficulty rating differences between techniques lessens. For example, we could not find a significant effect of technique on reported ease of use for the last two levels of ecological validity. Of course, comparing subjective ratings for different techniques when participants only tried one of them can be problematic, but we did see ratings consistent with our expectations for the first two levels of ecological validity. Another interesting result related to user preference is that, while DyCoDir performed as well as both progressive refinement techniques in the abstract and supermarket environments, it was rated harder to use. This was expected, as the mismatch between the absolute

position of the cursor and the relative position can be hard to understand, and the use of pre-defined velocity thresholds requires focus.

These results make a strong argument against using non-realistic task scenarios to evaluate selection. Evaluation in controlled, abstracted settings is an important tool, but care must be taken to replicate all the features of interest from the real world in the abstract environment. Furthermore, it must be understood that isolating selection from other tasks can hide important effects that the techniques may have on time or accuracy, and results may be misleading.

Finally, the combined results of all our studies so far show that progressive refinement can be as fast and is sometimes significantly more accurate than other state-of-the-art techniques. In addition, in all cases, they are considered easier to use and preferred overall in comparison to other immediate selection techniques. Considering all factors, we consider Double Bubble and FRIZ to be the “winners” of the “shootout.” They performed very well across all conditions, always in the top group for accuracy. Bubble Cursor was faster in the abstract environment, but performed much worse in the supermarket. DyCoDiR usually performed as well as (and sometimes better than) the progressive refinement techniques, but was not considered as easy to use.

Despite the positive results obtained by Double Bubble and FRIZ, we must acknowledge their limitations. For instance, we know there is a fine balance between the minimum bubble size in the first phase used to reduce the number of objects in the menu and the effort required to select objects using a bubble that size. In our study, we increased the size of the sphere in both the abstract and supermarket environments to be able to compare Double Bubble to the other techniques. This issue is directly linked to the scalability of the technique, which performs worse when distractor density is higher. Perhaps this issue could be solved by using the index of difficulty as a guide. Additionally, the same minimum size could be used in the menu phase to reduce precision requirements while increasing the number of refinements. FRIZ, on the other hand, could be further improved by using bubble cursor or any other snapping technique instead of simple Ray-casting during the selection phase.

Another important limitation that could be addressed in future studies is related to using these techniques in real-world applications. How much effect does zooming in/out for selection have in an application in which selection is not the primary task? How about having the Double Bubble resizing and snapping to different targets all the time, or having the explosion effect?

6 Design and Evaluation of Gesture-Based Progressive Refinement Techniques for Volume Selection

Our previous chapters presented the design of five different single-object selection techniques based on progressive refinement: SQUAD, Continuous Zoom, Discrete Zoom, Double Bubble, and FRIZ. These techniques were evaluated in three different studies that aimed to identify the trade-offs between progressive refinement and immediate selection for pointing-based selection. In this chapter, we present the design and evaluation of three new progressive refinement techniques that focused on improving the generality of the work.

Firstly, we wanted to design progressive refinement techniques with different metaphors and technologies. Newer devices such as the Kinect (Microsoft, 2012) and the Leap Motion Controller (Leap, 2012) enable gesture-based interaction by tracking the position and/or orientation of the whole human body or specific body parts like hands and fingers. Since these devices are based on markerless optical tracking, users are not required to hold or wear anything—their hands and bodies themselves are the input device. This sort of deviceless UI, also known as Natural User Interfaces (NUIs), holds the promise of free and natural interaction often envisioned in science fiction movies, reducing barriers to computing and empowering users through the use of gestures (Wigdor & Wixon, 2011). However, they have an important additional constraint: they make it much harder to initiate and confirm the selection since there are no physical constraints or buttons.

Additionally, we wanted to examine how progressive refinement can be used for different selection tasks, such as the selection of point clouds. Point clouds are sets of unstructured and unlabeled points generated by devices such as depth cameras and laser scanners. They can be used for a variety of different applications, from simple reconstruction of real-world objects to advanced mapping and spatial tracking for robots. However, while the point clouds generated by these devices may be accurate, their use is limited by the lack of semantic and structural information. In other words, while these devices may be able to create an accurate 3D representation of the world around them, it will contain no information about what is contained in that environment. Many scenarios may require the manual identification of parts of the point cloud (e.g., tools for a robot to use), and it is important to have a user interface that allows knowledgeable users to select and annotate these point clouds. This scenario was the theme of the IEEE 3DUI Contest 2014, which required participants to design a 3D user interface to solve this problem. In this task scenario, users have to select a large number of points from an even larger set (e.g., 1,000 out of 50,000 points). Using single-object selection techniques would be highly impractical for such a task, so using techniques that allow users to select

multiple objects at the same time is fundamental. In order to solve this issue, we designed techniques that use volume selection, in which users specify the regions of space that contain the points they want. Additionally, using a NUI device and 3D interaction to complete such tasks was interesting since point clouds are inherently 3D.

Finally, the design of progressive refinement techniques that used different interaction metaphors and for different tasks allows us to expand the design space and to compare different choices for the new dimensions in our design space. Additionally, these help answer our research question D2, which refers to the use of progressive refinement for different metaphors and selection tasks.

6.1 Leap Motion Controller

Before going into details about the techniques, we describe the NUI device that we used since our designs were heavily influenced by it. The Leap Motion Controller (hereafter called Leap) is a relatively new device that tracks the positions of hands and fingers. The Leap was released with the optimistic goal of allowing the user to interact with her computer in a whole new way through gestures. The device is said to offer sub-millimeter accuracy (down to 0.01 mm) and a processing rate reaching up to 290 fps, which results in a real-time tracking system, with a minimal latency claimed to be “imperceptible to the human eye.” However, in practice, this level of accuracy has not been proven. Weichert, Bachmann, Rudak, and Fisseler (2013) have been able to obtain an accuracy of 0.7 mm.

The hardware has a very simple configuration— two infrared CCD cameras and three infrared LED emitters, all packed in a small (8x3x1 cm) and light (45 grams) case—resulting in a very inexpensive device. The power of the Leap lies in the algorithms that process the data received by the cameras. Although we lack the specific details of the algorithms used, the hardware suggests that the Leap is an optical tracking system based on stereo vision. Because of the nature of the optical tracking system used, lighting conditions can have a major influence on tracking quality.

The Leap SDK is able to track predefined objects, like hands, fingers or some tools (e.g., a pencil), and it provides the position of these objects in a Cartesian space using the center of the device as the origin. The default interaction space is enclosed in a 60 cm³ cube centered over the device. By processing the positions of the objects over time, the software is able to determine if they have moved (translation or rotation) in order to translate these movements into recognizable gestures. Additionally, it provides a basic gesture detection system for few pre-defined gestures, like making circles, swiping or tapping. Finally, the user also has the option to configure some settings, like the height of the tracking space (from 7 to 25 cm) or the tracking priority (precision, speed, or balanced tracking).

6.2 Designing Point Cloud Selection

Our design process helped us quickly realize the limitations of the devices and interactions we set out to use. In particular, the small size of the Leap workspace, the lack of any physical indication of its limits, and finger occlusion were the biggest challenges we had to overcome. Additionally, a single precise selection was simply not possible due to the device and the nature of the task, which could, for example involve selecting multiple points in different regions of the point cloud. For these reasons, we decided to use a progressive refinement approach involving a two-step selection process that could be repeated as many times as needed: first *dividing* the point cloud into two, and then the *selection* of the set of points that will be kept and used from that point on.

6.2.1 Techniques for Dividing the Point Cloud

We designed three different techniques to perform the first step in the selection process: *Slice*, *Lasso*, and *Volume Brush*. In designing our selection techniques based on bare-hand gesture, we wanted metaphors that were easy to understand and that lent themselves to gestures that could be easily detected by the Leap device. Additionally, all techniques had two distinct versions: open-loop, in which activation and completion are based on movement speed, and closed-loop, which used button-based activation.

Slice

The metaphor we settled on for our first technique was that of a chef's knife, which is used to quickly cut off and, with the second selection step, swipe away unwanted parts of the food (or point cloud) until the desired portion is all that remains. Because this was the first metaphor we designed, the whole system is named "Slice-n-Swipe". To use it, the user inserts a single finger into the Leap workspace, and the 3D position of the fingertip and the hand is tracked by the Leap and displayed in the virtual workspace. Slicing can be done in two different modes: an **open-loop** feedback version (i.e., without live feedback and adjustment while performing the refinement step) of the technique that uses a rapid finger movement to specify the slicing plane; and a **closed-loop** version (i.e., with live feedback, allowing fine adjustments) that uses a button press to initiate slicing and releasing the button to complete it, allowing users to adjust the slice before committing to it. The open-loop version requires less precision, but also gives less control to the user. In both versions, the slice is a plane containing three points: the positions of the fingertip at the beginning and end of the movement, and the position of the hand at the end of the movement. The plane divides the 3D points into two subsets, and since it is infinite, precise depth or positioning of the finger is not required. This technique is shown in Figure 42.

Initially, we provided an indication of the current position of the hand and fingertip, as well as a line connecting the two. The line extended beyond the fingertip position to follow our chef's knife metaphor. During testing, however, estimating where the slice was going to be in 3D was a challenge, since there was no way to preview the slice without actually doing it. To solve this problem, we added a trail for the knife that allowed users to "practice" with slow finger movements before making the slice.

We also experimented with different ways to trigger the slice, trying to figure out the right speed thresholds and smoothing filters we could use, as well as how to prevent accidental triggers when entering or exiting the workspace. As we found in testing our earlier prototypes, all of these issues played a very important role in the overall usability of this technique, and we spent a significant amount of time on making sure these were just right.

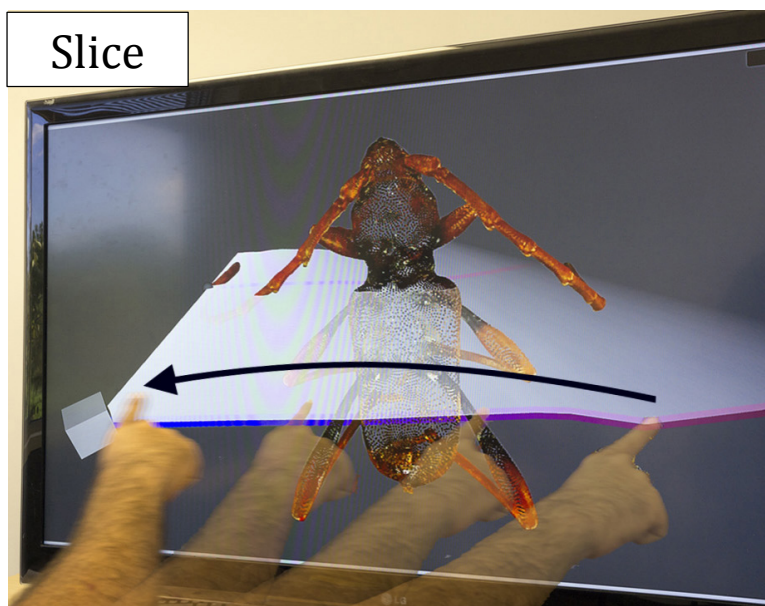


Figure 42 – Slice technique for dividing point clouds into two half spaces.

Lasso

Our second technique used a cookie cutter metaphor that works similarly to the lasso tool commonly found in image editing applications, in which the user draws a shape on the screen around the objects to be selected. Just like in the slice technique, only one finger is used. In this case, we created two different versions of the lasso technique: an **open-loop** feedback version in which users activate the lasso by moving the index finger quickly in any direction and close it by stopping finger movement; and a **closed-loop** feedback version, in which the lasso is activated by pressing a button and closed by releasing it. Since the open-loop version required users to keep moving to keep drawing the lasso,

it did not afford a large amount of precision. The closed-loop version, on the other hand, provided more control over the lasso and allowed users to adjust their selection as they were doing it. For both versions, once the lasso is activated, the system starts tracking the finger and the user can draw the shape of a lasso by moving the finger in the air. All the points inside the lasso are colored blue, and the points outside are colored red. This visual feedback is updated in real-time by the system, according to the current shape of the lasso. By releasing the button, the user creates the two point cloud subsets. This technique is illustrated in Figure 43.

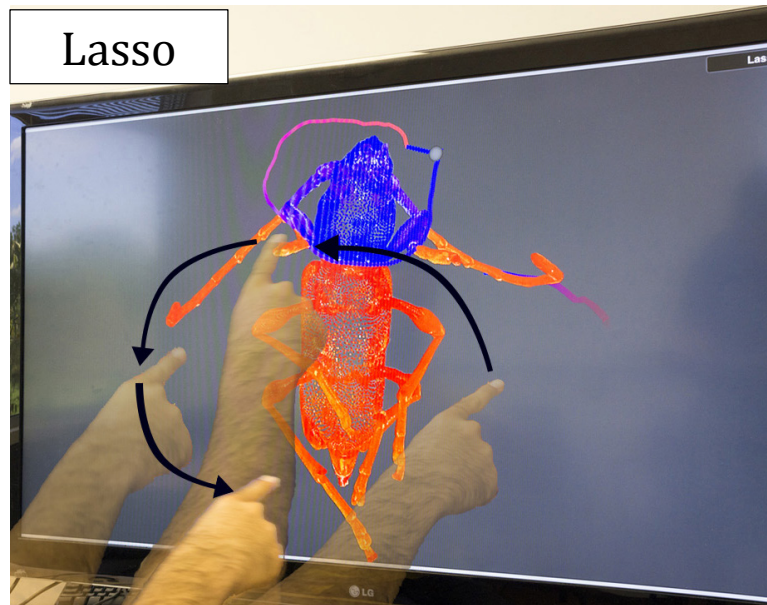


Figure 43 – Lasso technique for dividing point clouds into two half spaces.

Originally, the finger and hand orientation were also taken into account, and the lasso was formed by the 3D trail of the user's movement. This required a very high level of precision from the user, and the output ended up being not only hard to understand, but very imprecise. In order to reduce gesture complexity and decrease the amount of precision required from the user, our final design of the lasso uses position in screen space (2D) based on the projection of the tracked positions and points of the point cloud to the current viewpoint.

Volume Brush

The last technique we designed used a carving metaphor, in which the user controls a 3D cursor to pick out parts of the point cloud to work with. Our implementation works by placing the index finger and thumb in the Leap workspace in a pinching gesture. The position of the two fingertips is tracked and displayed to the user, and a spherical cursor is created at the centroid of these two points. The size of the cursor is determined by the distance between the fingers, which can be as big as the size

of the current set of selected points. Points inside and outside of the cursor are colored differently at all times. The Volume Brush can be used in two different ways, similar to the other techniques: an open-loop feedback version in which brushing starts by moving the cursor quickly and ends it when the cursor stops moving; and a closed-loop version in which brushing starts on button press and ends whenever the user releases the button. All points that currently are or were inside the cursor while brushing are colored blue, while all other points are colored red. By stopping the brush action, two point cloud subsets are created. Since the technique works by using the position of the fingers to create a three-dimensional cursor, it provides more precision than the previous techniques, but also requires more precision.

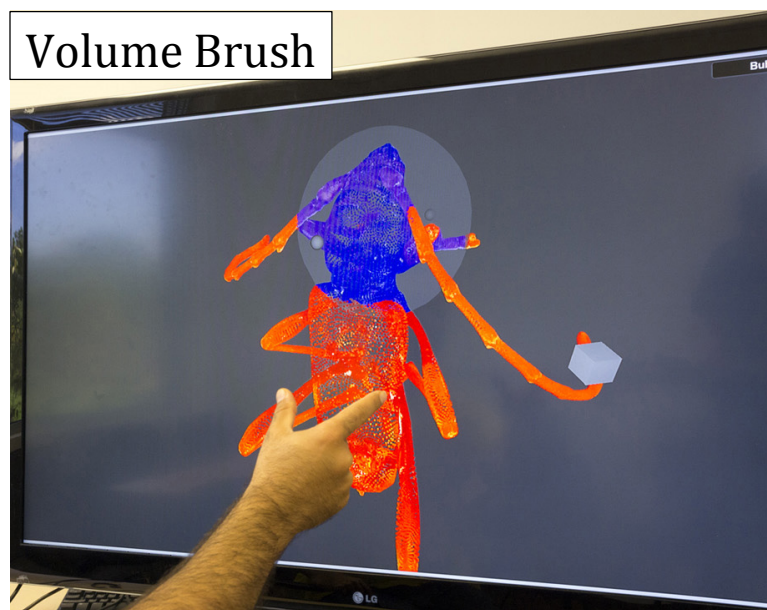


Figure 44 – Volume Brush technique.

In our initial design, this technique also included control over the shape and scale of individual axes of the cursor used to carve out the point cloud. However, these required either gestures that were too complicated to be executed or even detected by the Leap, or mode changes to use the same gestures in different ways. Additionally, the other two techniques cover cases where these different shapes would be useful. For these reasons, and following the progressive refinement idea of performing multiple rough interactions instead of single precise one, we decided to always use a sphere and only change scale uniformly.

6.2.2 Technique for Selection of a Point Cloud Subset

Once the two subsets of the point cloud are defined, the screen is divided vertically in two and the

point cloud is shown on both sides, with a different subset highlighted on each side. A rapid movement of the finger, roughly in the direction of one side, is detected as a *swipe*, and the system removes that subset of points from the selection set. This is illustrated in Figure 45. The user can undo the initial division by pressing a button. After selecting a subset, the viewpoint automatically zooms into the set of selected points. We found that this context-sensitive viewpoint adjustment was really important to avoid unnecessary navigation. In general, animations between states helped to clearly indicate what is happening in the system.

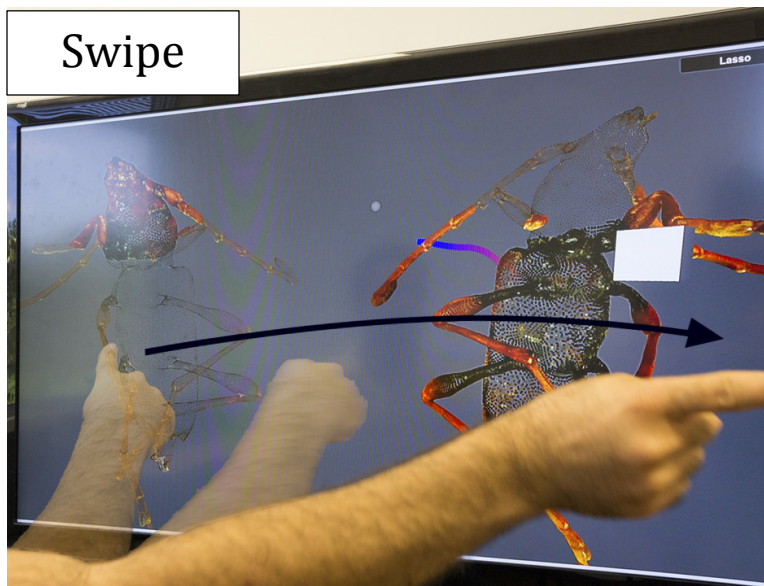


Figure 45 – Swipe to select one of the two point cloud subsets.

We tried other ways to show the two parts (colors, different positions of the parts) so the user could tell what had happened after the slice. We also tried different ways for the user to select the part they wanted to keep or throw away. Most of these, however, had their design coupled with the technique for creating the two subsets. For example, for the slice technique, we initially tried using color and the cutting plane orientation to determine direction of swipe. This, however, created an inconsistency with the other techniques, since they do not have a cutting plane.

6.3 Evaluation of Point Cloud Selection Technique Designs

The final experiment compared our three progressive refinement techniques for point cloud selection. We created a number of different tasks with varying level of difficulty, and compared the use of open-loop and closed-loop feedback.

6.3.1 Goals and Hypotheses

The primary goal of this experiment was to validate the use of progressive refinement for a different task than in previous experiments, as well as a different interaction metaphor. We compared Slice+Swipe, Lasso+Swipe, and Volume Brush+Swipe. Different than our previous experiments, we focused on comparing the trade-offs between requiring more precision and requiring more refinement steps when using a gesture-based interface to perform a volume selection task.

With this tradeoff in mind, we expected to find a significant effect of technique on time. We hypothesized that Slice would be simpler to use and therefore faster than Lasso, which in turn would be simpler to use and faster than Volume Brush. Additionally, we expected Slice to require more refinements than Lasso or Volume Brush, since it provides less precision. Similarly, we expected Volume Brush to require fewer refinements than others, since it provided full 3D control over volume selection.

Our secondary goal was related to the comparison between the open-loop and closed-loop feedback versions of the techniques. Participants had to perform the same tasks in three different conditions: open-loop only, closed-loop only, and both. We hypothesized that we would find participants trying to balance their use when given the chance: open-loop when less precision is needed, and closed-loop when more precision is needed. Given the always-on nature of the device we chose to use, we also expected participants to perform more accidental selections and have less accuracy with open-loop feedback as opposed to closed-loop.

6.3.2 Experimental Design

We used a mixed design with repeated measures. Our between-subjects variable was *technique* (Slice+Swipe, Lasso+Swipe, Volume Brush+Swipe), and our within-subjects variables were *task* (six total per technique, three common between all), and *feedback type* (open-loop, closed-loop, both). Thus, the final experimental design was 3 techniques x 6 tasks x 3 feedback types. The order of presentation of the feedback was counterbalanced for open-loop and closed-loop, and the “both” condition was always last. Within each condition of feedback type, each of the six tasks was presented in random order.

The dependent variables were time to complete the task, accuracy (percentage of target points removed and percentage of extra points remaining), number of refinements used, number of cancellations and number of undo actions (accidental selections which caused in cancellation of selection in the swipe phase or an undo after completing a refinement), type of feedback chosen in the “both” conditions, and our user preference metrics (preferred technique overall, preferred

technique when selection target was small, and preferred technique when there were too many points around the target).

6.3.3 Apparatus

Our setup was built around the Leap Motion device, used with the dominant hand to perform the selection tasks, seen in Figure 46. We also used a 3DConnexion SpacePilot Pro 3D mouse for viewpoint manipulation with the non-dominant hand. The 3D mouse provides six-degree-of-freedom isometric input, which enables camera translation and rotation in the 3D environment. Translational force causes the virtual camera to translate in the scene, while rotational force causes the camera to rotate about the point in the point cloud closest to the center of the screen. This makes it easy for users to view the dataset from any angle and zoom toward or away from the dataset. We also used a total of four different buttons to assist in the selection task and provide different functionality: highlight the target points, reset the viewpoint, activate the closed-loop interaction, and undo/cancel selection. Finally, we used a monoscopic 27-inch monitor (1920px x 1080px) as our display.



Figure 46 – Complete setup used in the evaluation of gesture-based progressive refinement techniques for point cloud selection.

We used a total of ten different tasks: *head*, *whiteboard*, *farm*, *post*, *tail*, *facade*, *train*, *chair*, *monitors*, and *stairs*. Two were used as training tasks for all conditions (*whiteboard*, *head*), and three of these were common for all techniques (*tail*, *farm*, *stairs*). Tasks had different difficulty, which was

determined by dimensionality, shape of the target selection, and percentage of target points relative to all points in the point cloud. We considered these parameters and created three difficulty levels (easy, medium, hard), and had one of each for the common tasks. The other tasks were used to measure specific characteristics of each technique. All tasks are shown in Figure 47.

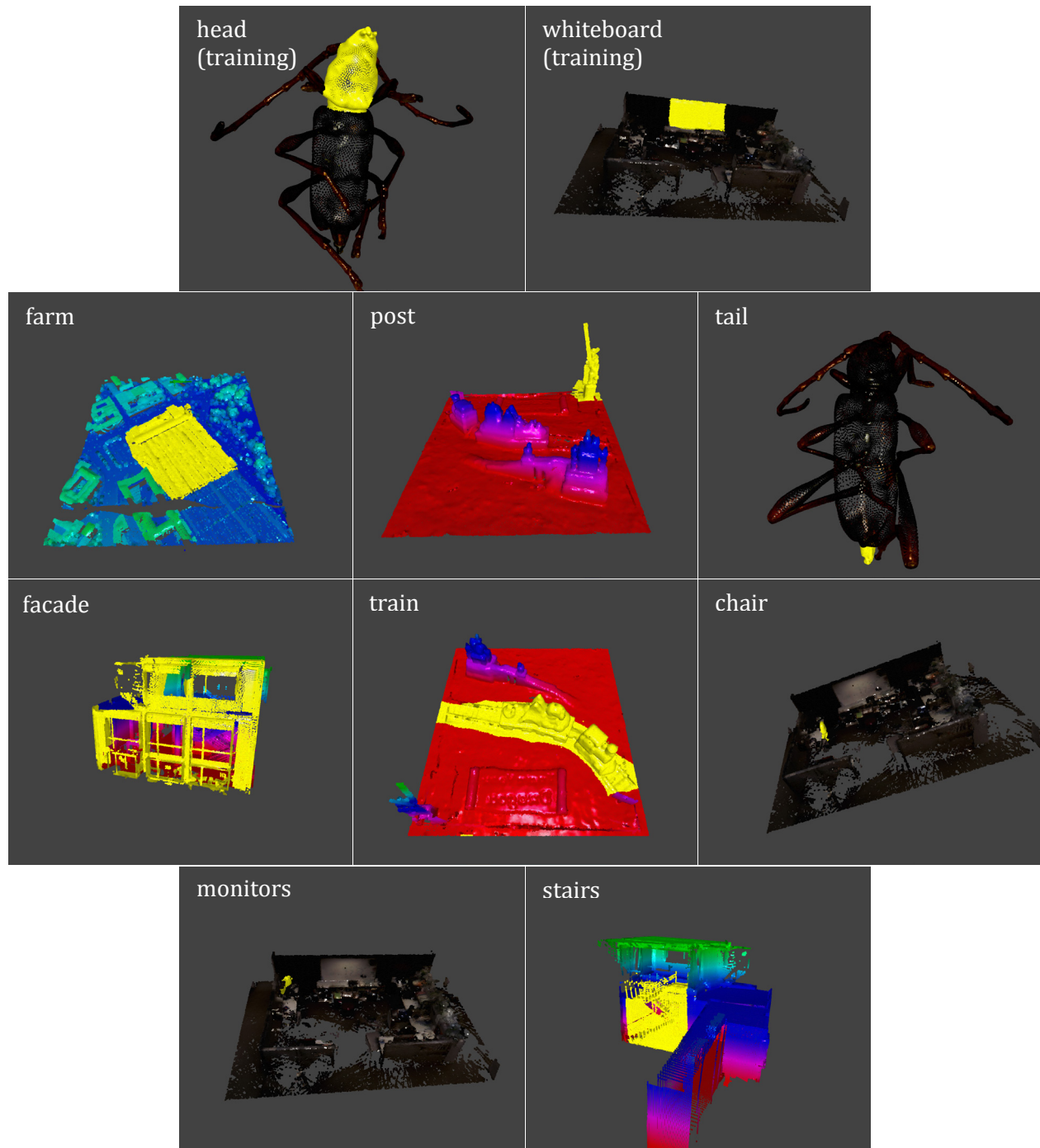


Figure 47 – All environments and tasks used in the evaluation of point cloud selection techniques, with target point clouds highlighted in yellow.

We used two bars to represent task progression, shown in Figure 48. The top bar represented the total target points the user has selected, which means it always starts full and green since all points are selected from start. The goal was to make sure it does not go below the threshold indicated in red, by avoiding the removal of target points. The second bar represented progress towards completing the removal of unwanted points. The goal was to make this bar reach the second threshold and turn green. The task was considered finished when both bars were green. Finally, we also showed a countdown timer for the current task.

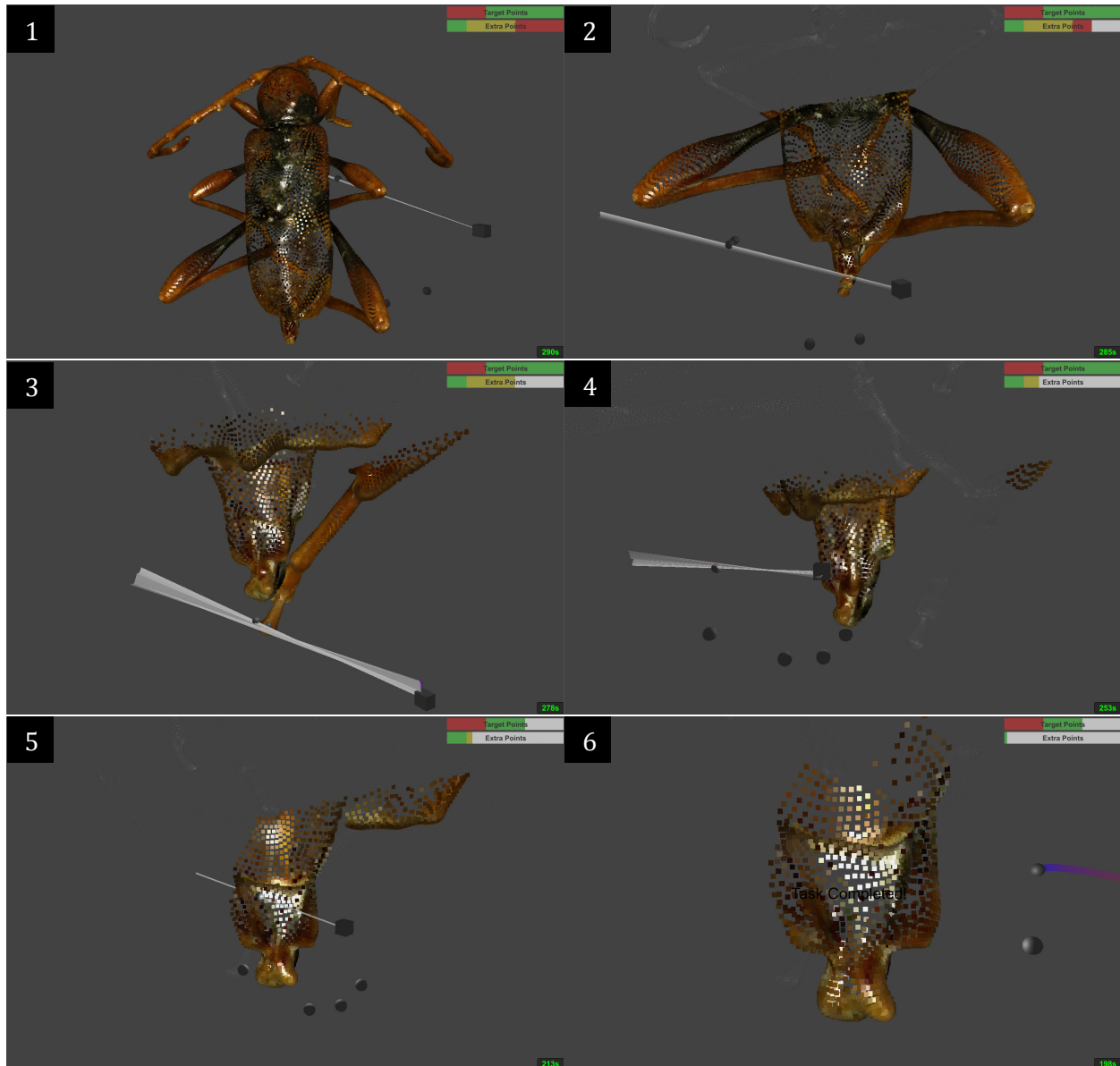


Figure 48 – Task progression bars and countdown timer during the completion of the tail task.

6.3.4 Participants

We recruited 24 voluntary unpaid participants from the campus community to perform the study, with a total of eight participants per technique. Participants' ages ranged from 19 to 34 years, with a median age of 21. Eight of the participants were female.

6.3.5 Procedure

Participants were first given the informed consent form to read and sign. Upon agreement, they proceeded to complete the background questionnaire. After that, they were shown the experimental setting and the apparatus. The initial tutorial phase was done with two tasks, shown in the previous section, and participants had to finish both before moving on to the trials. There was no time limit for these tasks.

After practicing, they were reminded that they had to perform the trials as quickly as possible while trying not to make mistakes, and then performed selection of all six tasks with the initial condition. During the trials, there was a maximum time for task completion set to five minutes. If participants exceeded this time limit, a message was shown ("Time elapsed!") and the next trial was loaded. If more than 5% of the target points were removed during the selection process, the message "Too many target points removed!" was shown, and participants were allowed to undo and continue. If less than 5% of the target points were removed and 30% or less extra points remained, the task was considered complete and the message "Task complete!" was shown.

This process was repeated for all feedback conditions, first with the counter-balanced presentation of open-loop and closed-loop, and the condition where participants could choose between the two (both) always presented last. After the completion of all trials, participants filled out a post-hoc questionnaire comparing the different feedback conditions.

6.3.6 Results

We conducted two separate analyses: feedback-focused analysis comparing different feedback types (open-loop and closed-loop), without considering the both condition to avoid including learning effects; and an evaluation of the choice of feedback in the both condition. For both these analyses, we ran separate mixed model factorial ANOVAs with repeated measures for four different sets of independent variables: all techniques and the three common tasks; Slice-only and all its six tasks; Lasso-only and all its six tasks; and Volume Brush-only and all its six tasks. Separating the techniques helped us better understand the effects of feedback and tasks.

Time

The first part of the analysis of time compares our three progressive refinement techniques (Slice, Lasso, Volume Brush), two types of feedback (open-loop, closed-loop) and three common tasks between them (farm, tail, stairs). Overall, there was a main effect of technique on time ($F_{2,21} = 5.653$, $p=.011$). The mean time to complete the tasks was 78.925s for Slice, 100.569s for Lasso, and 117.175s for Volume Brush, with standard error of 8.176s. The pairwise comparison confirmed that Slice was significantly faster than Volume Brush ($F_{12,34} = 2.237$, $p=.033$), as expected. There were no other significant differences between techniques. This supports our hypothesis that Slice would be the fastest technique, while Volume Brush would be the slowest.

While there was no main effect of feedback type ($F_{1,21} = 2.837$, *insignificant*), there was a nearly significant interaction between feedback type and technique ($F_{2,21} = 3.032$, $p=.07$). In the pairwise comparisons, we found there was a nearly significant difference between different feedback types for Slice ($F_{2,21} = 3.032$, $p=.064$) and Volume Brush ($F_{2,21} = 3.032$, $p=.058$). This interaction can be seen in Figure 49.

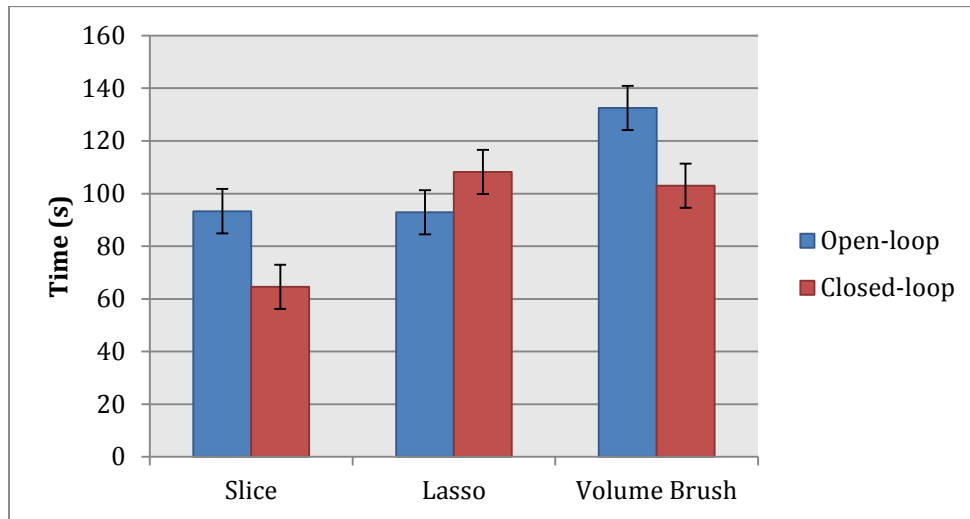


Figure 49 – Interaction between technique and feedback type for time.

As expected, we found a main effect of task on time ($F_{2,42} = 95.942$, $p<.001$). The mean time to complete the tail task was 49.115s, the farm task was 74.784s, and the stairs task was 173.311s, with standard error of 3.181s, 5.351s, and 10.849s, respectively. The pairwise comparisons showed significant differences between tail and farm ($F_{12,10} = 83.704$, $p=.001$), tail and stairs ($F_{12,10} = 83.704$, $p<.001$), and farm and stairs ($F_{12,10} = 83.704$, $p<.001$).

There was a nearly significant interaction between task and technique ($F_{4,42} = 2.505$, $p=.056$). However, the pairwise comparisons showed no significant effects between different combinations of conditions for $\alpha = 0.05$. Additionally, we found no significant interaction between task and feedback type, or between task, feedback type and technique.

The second part of our time analysis separates the techniques and compares two types of feedback (open-loop, closed-loop) and the respective six tasks for each technique (Slice: farm, post, tail, facade, monitors, stairs; Lasso: farm, post, tail, facade, chair, stairs; Volume Brush: farm, post, train, tail, chair, stairs). For this evaluation, we found a significant main effect of feedback type for Volume Brush ($F_{1,7} = 5.902$, $p=.045$). The pairwise comparison showed that closed-loop (mean = 92.941s) was faster than open-loop (mean = 124.064s) ($F_{1,7} = 5.902$, $p=.045$). There was no significant effect of feedback type for Slice ($F_{1,7} = .925$, *insignificant*) and Lasso ($F_{1,7} = .065$, *insignificant*).

We also found a significant effect of task for Slice ($F_{5,35} = 22.531$, $p<.001$), Lasso ($F_{5,35} = 21.795$, $p<.001$), and Volume Brush ($F_{5,35} = 23.777$, $p<0.001$). For all techniques, there were significant differences between their first four tasks (Slice: post, farm, tail, facade; Lasso: post, farm, tail, facade; Volume Brush: post, farm, tail, train), which were considered easier, and the last two (Slice: monitors, stairs; Lasso: chair, stairs; Volume Brush: chair, stairs) for $\alpha = 0.05$. These differences can be clearly seen in Figure 50, in which numbers represent the different tasks for each technique following the order listed previously.

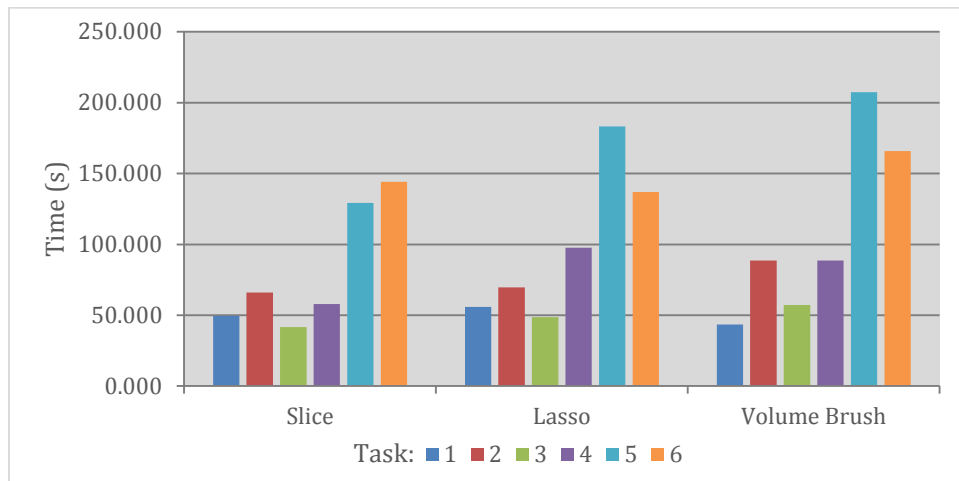


Figure 50 – Time for completion of different tasks for each technique.

Finally, we found no significant interaction between feedback type and task ($F_{5,35} = 1.434$, *insignificant*).

Accuracy

The first part of the analysis of accuracy looks into the two variables used to verify task completion: percentage of extra points left, and percentage of target points removed. Similar to time, we compare our three progressive refinement techniques (Slice, Lasso, Volume Brush), two types of feedback (open-loop, closed-loop) and three common tasks between them (farm, tail, stairs). Overall, we found no main effect of technique on extra points left ($F_{2,21} = 1.128$, *insignificant*) or goal points removed ($F_{2,21} = 1.855$, *insignificant*). The mean percentage of extra points left was 19.6% for Slice, 22.3% for Lasso, and 22.3% for Volume Brush, with standard error of 1.5%.

On the other hand, we found a main effect of feedback type on both extra points left ($F_{1,21} = 14.486$, $p=.001$) and goal points removed ($F_{1,21} = 7.075$, $p=.015$). The pairwise comparisons show that using closed-loop feedback resulted in both fewer extra points left ($F_{1,21} = 14.486$, $p=.001$) and fewer target points removed ($F_{1,21} = 14.486$, $p=.001$). These two results are shown in Figure 51. This supports our hypothesis that closed-loop results in more accurate selection.

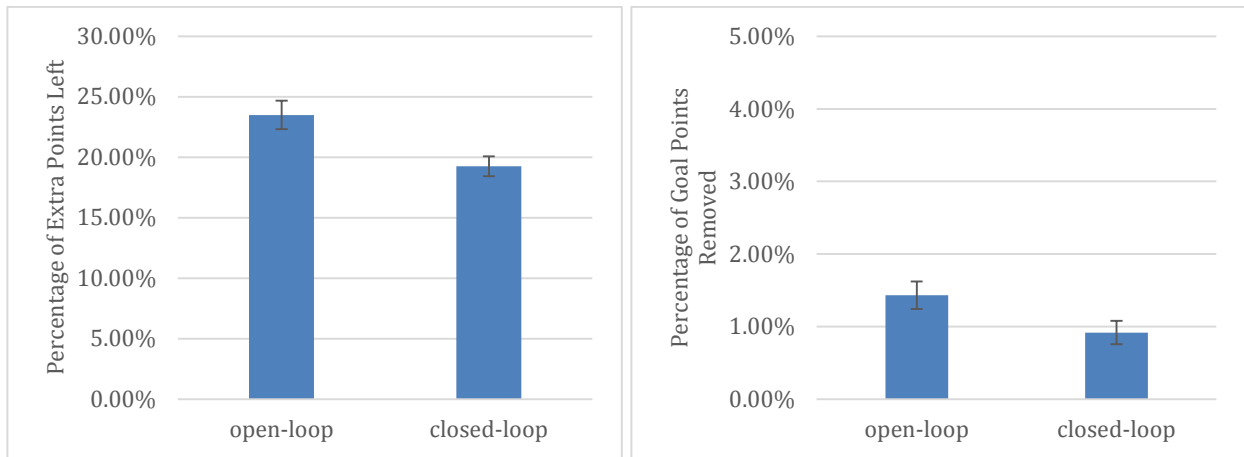


Figure 51 – Percentage of extra points left (left) and percentage of goal points left (right) for different feedback types.

There was no significant interaction between feedback type and technique for extra points left ($F_{2,21} = .124$, *insignificant*) or goal points removed ($F_{2,21} = 1.453$, *insignificant*).

As expected, we found a main effect of task on both extra points left ($F_{2,42} = 38.177$, $p<.001$) and goal points removed ($F_{2,42} = 17.505$, $p<.001$). The pairwise comparisons showed significant differences between tail and farm (extra points left: $F_{12,10} = 83.704$, $p<.001$; goal points removed: $F_{12,10} = 83.704$, $p=.023$), tail and stairs (nearly significant for extra points left: $F_{12,10} = 83.704$, $p<.071$; goal

points removed: $F_{12,10} = 83.704$, $p=.019$), and farm and stairs (extra points left: $F_{12,10} = 83.704$, $p<.001$; goal points removed: $F_{12,10} = 83.704$, $p<.001$).

There was no significant interaction between task and technique for extra points left ($F_{4,42} = 2.173$, *insignificant*) or goal points removed ($F_{4,42} = .407$, *insignificant*).

The second part of our analysis compares the different types of feedback (open-loop, closed-loop) and the respective six tasks for each technique (Slice: farm, post, tail, facade, monitors, stairs; Lasso: farm, post, tail, facade, chair, stairs; Volume Brush: farm, post, train, tail, chair, stairs) separately. For this evaluation, we found a significant main effect of feedback type for Slice (extra points left: $F_{1,7} = 7.821$, $p=.027$; goal points removed: $F_{1,7} = 25.376$, $p=.002$), with closed-loop providing more accuracy than open-loop. There was no significant effect of feedback type for Lasso (extra points left: $F_{1,7} = 4.52$, *insignificant*; goal points removed: $F_{1,7} = .321$, *insignificant*) or Volume Brush (extra points left: $F_{1,7} = .868$, *insignificant*; goal points removed: $F_{1,7} = 4.446$, *insignificant*).

We also found a significant effect of task for Slice (extra points left: $F_{5,35} = 14.834$, $p<.001$; goal points removed: $F_{5,35} = 5.293$, $p=.001$), Lasso (extra points left: $F_{5,35} = 5.203$, $p=.001$; goal points removed: $F_{5,35} = 3.509$, $p=.011$), and Volume Brush (extra points left: $F_{5,35} = 6.671$, $p<.001$; goal points removed: $F_{5,35} = 6.423$, $p<.001$). Similar to results for time, there were significant differences between their first four tasks (Slice: post, farm, tail, facade; Lasso: post, farm, tail, facade; Volume Brush: post, farm, tail, train), which were considered easier, and the last two (Slice: monitors, stairs; Lasso: chair, stairs; Volume Brush: chair, stairs) for $\alpha = 0.05$. These results further support the hypothesis about differences between tasks.

Refinements

Similar to the analyses of time and accuracy, we first looked into our three progressive refinement techniques (Slice, Lasso, Volume Brush), two types of feedback (open-loop, closed-loop) and three common tasks between them (farm, tail, stairs). We did not find a main effect of technique on the number of refinements ($F_{2,21} = .617$, *insignificant*). The mean number of refinements necessary to complete the tasks in common was 6.354 for Slice, 5.896 for Lasso, and 5.25 for Volume Brush, with standard error of 0.706. Although this does not support our hypothesis, we can see the expected overall trend of Slice requiring more refinements than Lasso, and Lasso requiring more refinements than Volume Brush.

On the other hand, we found a main effect of feedback type ($F_{1,21} = 32.016$, $p<.001$). As expected, the

pairwise comparisons showed that using open-loop feedback resulted in more refinements ($F_{1,21} = 32.016$, $p < .001$). Additionally, we found a significant interaction between feedback type and technique ($F_{1,21} = 22.202$, $p < .001$), shown in Figure 52. However, as is evident in this graph, we only found a significant difference between the different feedback types for Slice ($F_{1,21} = 22.202$, $p < .001$). Thus, this only partially supports our hypothesis that open-loop would result in more refinements.

As expected, we found a main effect of task ($F_{2,42} = 51.745$, $p < .001$). The pairwise comparisons showed significant differences between tail and farm ($F_{12,10} = 83.704$, $p < .001$), tail and stairs ($F_{12,10} = 83.704$, $p < .001$), and farm and stairs ($F_{12,10} = 83.704$, $p < .001$). We also found a significant interaction between task and technique ($F_{4,42} = 3.03$, $p = .028$). However, pairwise comparisons showed no significant differences between the different combinations of conditions.

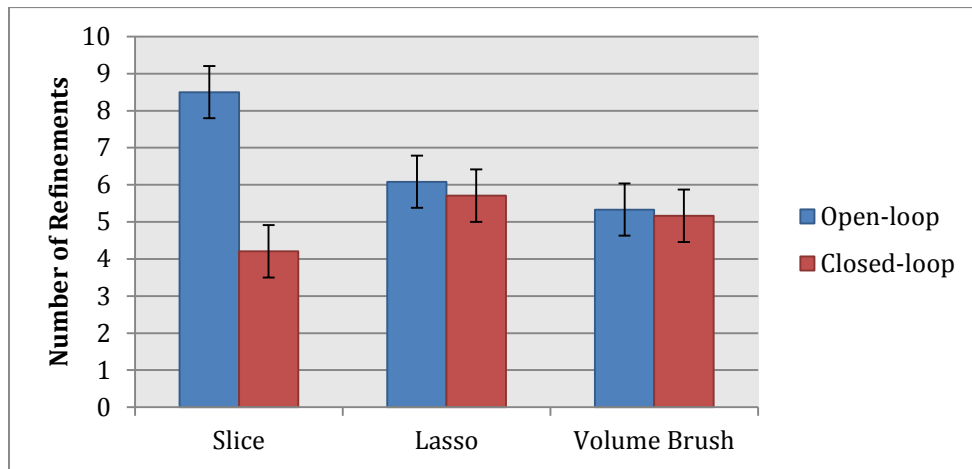


Figure 52 – Interaction between technique and feedback type for the number of refinements.

The second part of our analysis compares the different types of feedback (open-loop, closed-loop) and the respective six tasks for each technique (Slice: farm, post, tail, facade, monitors, stairs; Lasso: farm, post, tail, facade, chair, stairs; Volume Brush: farm, post, train, tail, chair, stairs) separately. We found a significant main effect of feedback type for Slice ($F_{1,7} = 248.904$, $p < .001$), with closed-loop reducing the number of refinements required to complete selection. There was no significant effect of feedback type for Lasso ($F_{1,7} = 1.063$, *insignificant*) or Volume Brush ($F_{1,7} = .405$, *insignificant*). Finally, we also found a significant effect of task for Slice ($F_{5,35} = 57.013$, $p < .001$), Lasso ($F_{5,35} = 20.274$, $p < .001$), and Volume Brush ($F_{5,35} = 17.012$, $p < .001$).

Mistakes

Our first analysis for the sum of the number of cancellations and undo actions, which we call the

number of mistakes, looked into our three progressive refinement techniques (Slice, Lasso, Volume Brush), two types of feedback (open-loop, closed-loop) and three common tasks between them (farm, tail, stairs). We did not find a main effect of technique on the number of mistakes ($F_{2,21} = .534$, *insignificant*). The mean number of cancellations and undo actions necessary to complete the tasks was 2.438 for Slice, 1.979 for Lasso, and 1.896 for Volume Brush, with standard error of 0.399.

More interestingly, we found a main effect of feedback type ($F_{1,21} = 17.765$, $p < .001$). As expected, the pairwise comparisons showed that using open-loop feedback resulted in more mistakes ($F_{1,21} = 17.765$, $p < .001$). Additionally, we found a significant interaction between feedback type and technique ($F_{1,21} = 5.688$, $p = .011$). The pairwise comparisons indicated a significant difference between the two feedback types for Slice ($F_{1,21} = 22.202$, $p < .001$) and Volume Brush ($F_{1,21} = 22.202$, $p = .015$). This partially supports our hypothesis that open-loop would result in more mistakes than closed-loop feedback.

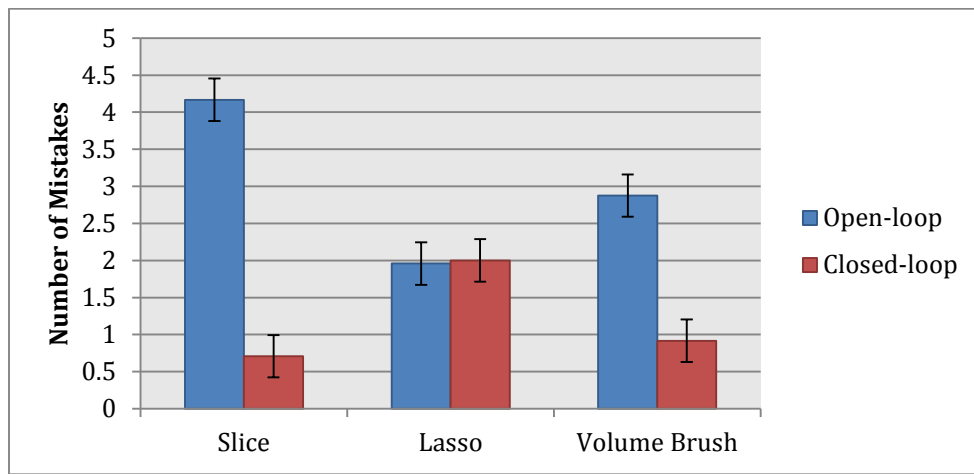


Figure 53 – Interaction between technique and feedback type for number of mistakes.

As expected, we found a main effect of task on the number of mistakes ($F_{2,42} = 21.936$, $p < .001$). The pairwise comparisons showed a nearly significant difference between tail and farm ($F_{12,10} = 83.704$, $p = .069$), and significant differences between tail and stairs ($F_{12,10} = 83.704$, $p = .002$), and farm and stairs ($F_{12,10} = 83.704$, $p < .001$), again supporting our hypothesis about task differences and difficulty. We did not find a significant interaction between task and technique ($F_{4,42} = 1.693$, *insignificant*).

The final part of our analysis compares the different types of feedback (open-loop, closed-loop) and the respective six tasks for each technique (Slice: farm, post, tail, facade, monitors, stairs; Lasso: farm, post, tail, facade, chair, stairs; Volume Brush: farm, post, train, tail, chair, stairs) separately. We found

a significant main effect of feedback type for Slice ($F_{1,7} = 12.188, p=.01$) and Volume Brush ($F_{1,7} = 24.938, p=.002$), with closed-loop reducing the number of mistakes. There was no significant effect of feedback type for Lasso ($F_{1,7} = .744, \text{insignificant}$). Finally, we found a significant effect of task for Slice ($F_{5,35} = 9.017, p<.001$) and Volume Brush ($F_{5,35} = 3.549, p=.011$). There was no significant effect of task for Lasso ($F_{5,35} = 2.282, \text{insignificant}$)

Chosen Feedback in 'Both' Condition

This statistical analysis is different than those previously presented, and focuses on the statistical comparison of the number of times open-loop and closed-loop were used when participants had the choice between them. We compared our three progressive refinement techniques (Slice, Lasso, Volume Brush), the number of times each feedback type was used (open-loop, closed-loop), and three common tasks between them (farm, tail, stairs) and evaluated their effects on the number of refinements. Firstly, we did not find a main effect of technique on the number of refinements ($F_{2,21} = .799, \text{insignificant}$), as expected given the results presented earlier in this section.

Interestingly, we found a main effect of feedback type used ($F_{1,21} = 6.72, p=.017$). As expected, the pairwise comparisons showed that open-loop (mean = 2.833) was used less than closed-loop overall (mean = 5.208) ($F_{1,21} = 6.72, p=.017$). This partially contradicted our hypothesis that participants would balance the use of the different types of feedback; this was only true for Slice. Additionally, we did not find a significant interaction between chosen feedback and technique ($F_{2,21} = 1.795, \text{insignificant}$). This interaction is shown in Figure 54.

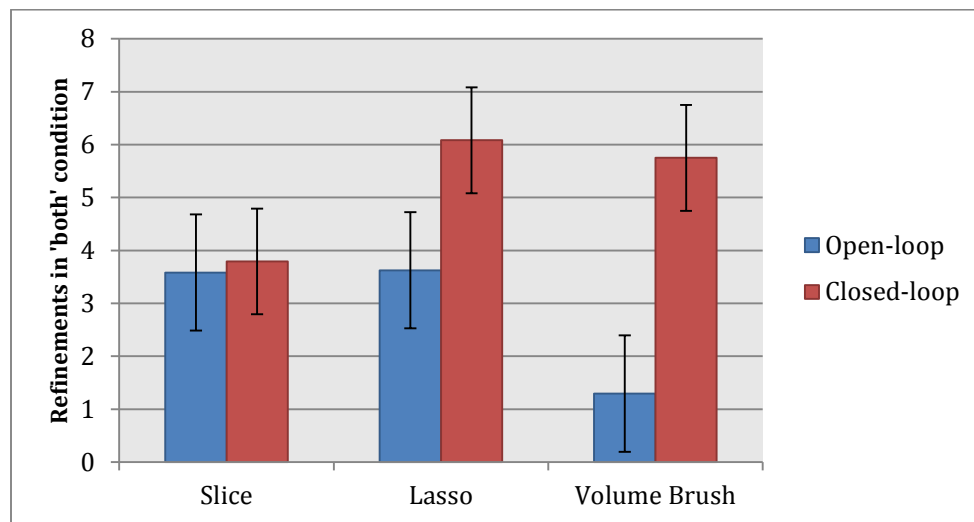


Figure 54 – Interaction between technique and chosen feedback.

Finally, we also found a main effect of task on the chosen feedback type ($F_{2,42} = 20.666, p < .001$). The pairwise comparisons showed a nearly significant difference between tail and stairs ($F_{12,10} = 83.704, p = .001$), and farm and stairs ($F_{12,10} = 83.704, p < .001$), again supporting our hypothesis about task differences and difficulty. We did not find a significant interaction between task and technique ($F_{4,42} = .817, \text{insignificant}$).

User Preference

In order to measure user preference, we asked participants to choose between the two feedback types (open-loop and closed-loop) overall, when target point clouds are small compared to the rest of the point cloud, and when the point cloud is too dense in the area surrounding the target. For the first two questions, one participant reported open-loop as the preferred feedback type (one of the participants who used Slice). For the last question, two participants who used Slice and one participant who used Volume Brush reported open-loop as their preferred method. All other participants preferred closed-loop, independently of the situation.

6.4 Discussion

This experiment focused on addressing our research question **D2**, which asked about the use of progressive refinement for multiple-object selection and new interaction metaphors. We answered this validation question through the design and evaluation of three progressive refinement techniques for point cloud selection. We based these techniques in different metaphors, and used in-air gesture-based input with the Leap Motion device. These techniques provide varying levels of control and precision to the user: Slice allows quick and easy selection through the specification of cutting planes; Lasso provides more precision by allowing users to draw lassos on the screen-space; Volume Brush gives complete 3D control over a volumetric cursor that is used to brush the point cloud. In other words, they differ in terms of complexity of the spatial region that can be selected, and this is traded off against the simplicity of the technique to understand and use (the more complicated techniques require more sophisticated strategies). Every technique is combined with Swipe, a technique that allows users to quickly select between the two sets of points specified with the first technique.

The differences in control between the techniques are highlighted by the mean number of refinements required for each technique. We saw an overall trend in which Slice required more refinements than Lasso, which required more refinements than Volume Brush. An interesting way to see this trend is to look at the refinement profile curves.

Refinement profiles are trend lines that show refinements across time, indicating how much of the point cloud was removed at each refinement and how much time was spent at each refinement as well. We created these curves by plotting each refinement/mistake as a data point in a scatter plot, separately for each participant and set of conditions. We then calculated the polynomial regression for those sets of points. This process can be seen in Figure 55, which highlights the creation of one trend line for the Slice technique with the stairs task and closed-loop feedback type.

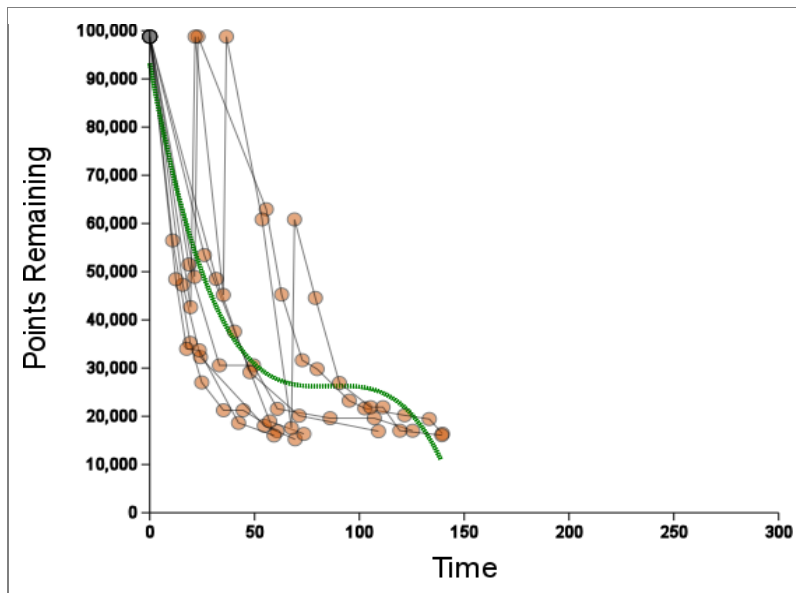


Figure 55 – Refinement profile for the Slice technique, stairs task, and closed-loop feedback.

Each point in the graph represents a refinement or a mistake, and each individual line represents a different participant. The green line is the calculated regression trend line. While these profiles do not present new statistical evidence for or against our hypotheses, the comparison between the trend lines for different techniques can help highlight the differences between techniques and illustrate the overall trends found with respect to the number of refinements and time, as shown in Figure 56.

We also continue to answer our research question **E2**, which asked about the evaluation of different design choices. More specifically, we looked into the sub-question **E2.5**, about the trade-offs between *closed-loop* and *open-loop* feedback. Across all our measures, closed-loop feedback presented better results. It was faster, more accurate, required fewer refinements, resulted in fewer mistakes, and was preferred over open-loop in every situation presented by the different tasks we created. This trend can be seen when we compare the refinement profile curves for all techniques and the different feedback types, illustrated in Figure 57. The dotted lines, representing open-loop feedback, always end after the solid lines, which represent closed-loop feedback. Additionally, while both types of

feedback seem to get close to the target selection quickly, participants spent more time performing precise selections to remove the last few extra points.

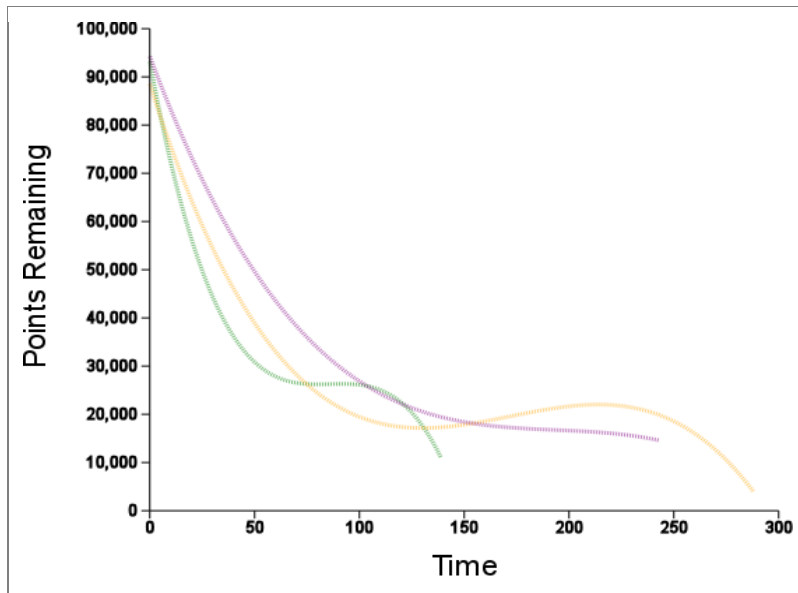


Figure 56 – Trend lines of all techniques for the stairs task and closed-loop feedback. Green line represents the trend line for Slice, purple line for Lasso, and yellow line for Volume Brush.

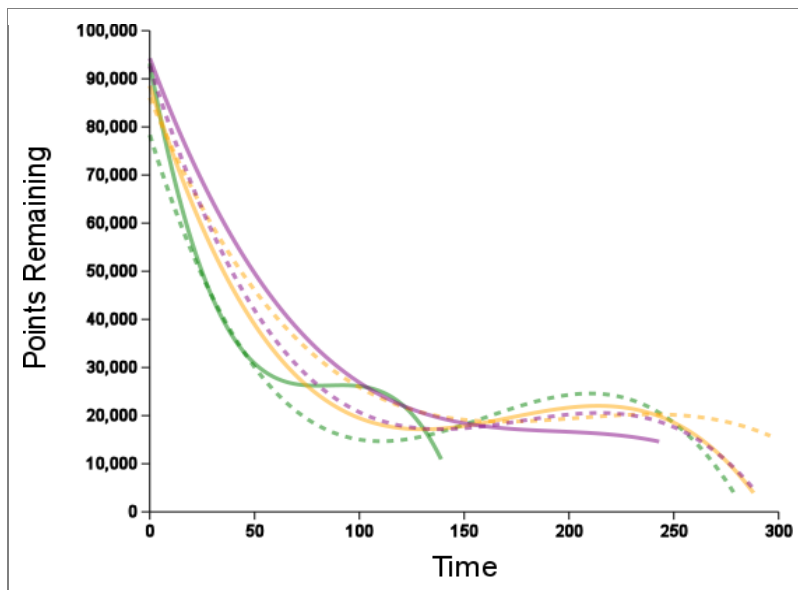


Figure 57 – Trend lines of all techniques and feedback types for the stairs task. Green lines represent the trend line for Slice, purple lines for Lasso, and yellow lines for Volume Brush. Solid lines show closed-loop feedback, and dotted lines show open-loop.

We believe that one of the main reasons for this difference between closed-loop and open-loop is related to the limitations of the input device we used. Some of these issues are inherent to the type

of device, which has always-on tracking and suffers from the live-mic issue for registration, continuation and termination of gestures (Wigdor & Wixon, 2011) in the case of open-loop feedback. Additional issues, such as the small interaction workspace, hand/finger occlusion, and large uncharacterized jitter make it even harder to create robust gesture detection (Teather, Pavlovych, Stuerzlinger, & MacKenzie, 2009). While we did implement a number of different safeguards to avoid these issues (e.g., a low-pass filter for finger position and velocity) and made sure participants were trained on how to avoid them, we did notice eventual accidental selections caused by tracking issues. These are evident in the analysis of the number of mistakes made, shown in the previous section, in which the number of mistakes made with the open-loop feedback versions of Slice and Volume Brush was significantly higher than with closed-loop. These issues were also a cause of frustration, reported by users in the post-experiment questionnaire. Additionally, while participants also reported that they understood that open-loop feedback was supposed to help them remove large unwanted parts of the point cloud faster, these issues resulted in the great majority of them always choosing closed-loop to avoid making mistakes. This is shown in our section about the chosen feedback in both condition.

Finally, considering all our results, we consider the closed-loop version of Slice the “winner” in this experiment. It was faster, had fewer refinements, and resulted in fewer mistakes than the other techniques. This conclusion is also evident in Figure 57. Additionally, while we see other techniques actually get closer to the target faster than Slice, closed-loop, they actually take much longer for the final refinements. We believe this is because of the nature of Slice, in which users always divide space in half using a plane as opposed to an arbitrary shape. This results in consistent selection difficulty across multiple refinements. For example, it may be easy to select large unwanted portions of the point cloud with Volume Brush and therefore get rid of them quickly in the initial selections, but the arbitrarily shaped point cloud it generates through brushing causes a large number of small detailed selections around the target points.

In a real-world application, selecting between the different tools between refinement steps would allow users to choose the best tools for the current job. For instance, using Slice to perform the initial refinements and generate a convex-shaped point cloud, and then choosing Lasso and Volume Brush to finalize selection as precision requirements increase from 2D to 3D and concave-shaped targets.

7 Design Space and Guidelines

Based on the progressive refinement techniques we designed, we identified five dimensions of the design space of progressive refinement selection techniques, which are shown in Figure 58. In this chapter, we explain the different dimensions of the design space and what we learned about them to answer our evaluation question **E2**.

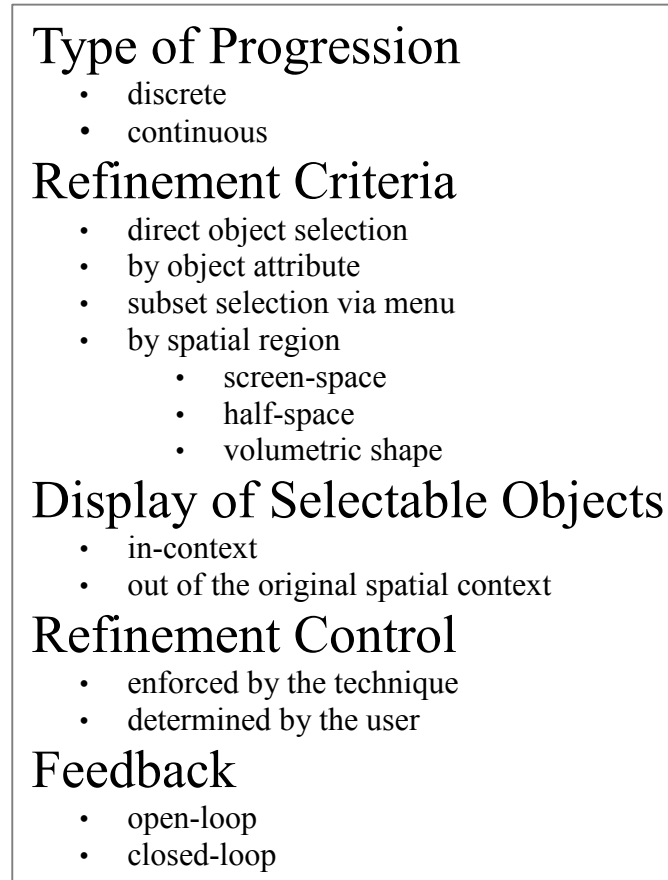


Figure 58 – Design space of selection by progressive refinement.

Type of Progression. 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 & Parker, 2004) and our continuous zoom technique.

In our second study, we compared these two types of progression, and while we did find continuous to perform better, it requires more precision from the user. However, these results were influenced by the fact that our Discrete Zoom technique design included a significant visual search time at each refinement step. In order to solve this issue, we combined the strengths of the Continuous Zoom with discrete progression in our third study to create a technique that performed as well as other state-

of-the-art immediate selection techniques: FRIZ. Overall, we found that discrete progression allows for lower input precision, and it requires less effort from the user. These results answered our research question **E2.1**, which was about the trade-offs between the different types of progression.

Refinement Criteria. 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 by selecting the object directly, like in the second phase of flower ray (Grossman & Balakrishnan, 2006) where the target object is selected to disambiguate selection. Refinement can also be done by the specification of attributes of the desired object, such as color, size or shape. Another example of refinement criteria is the quad menu refinement used in SQUAD, which uses subset specification via menu. It involves picking a subset of objects from a list or menu instead of from the environment. Finally, refinement can be done by selecting a spatial region, for example through the selection of an area in the screen space (e.g., Lasso), using planes to define half-spaces (e.g., Slice), or creating a volumetric shape (e.g., Volume Brush).

While we did not specifically test object selection or object attributes, we believe them to be impractical types of refinement progression, as they may require too many refinements or too much precision. We experimented with subset selection via menu in two of our studies (SQUAD, and Selection Shootout), and found that, while they can be fast, they require objects to be taken out of their original context. While this may be a disadvantage, we showed that by carefully designing our Double Bubble technique, out-of-context menu-based refinement can work as well as other approaches. Finally, we also tried the different options of selection of spatial region in all our technique designs: SQUAD uses selection by spatial region with a volumetric shape (sphere) in its first phase; Double Bubble uses the screen space to specify a 2D Bubble cursor for both of its phases; our zoom techniques uses the continuous or discrete specification of a view frustum; and each of our point cloud selection techniques uses a different type of spatial region specification – Slice uses a plane specification to define half-spaces, Lasso uses screen-space, and Volume Brush uses a sphere and brushing to define regions of space. In our last study, we were able to compare all types of spatial region selection, and found that the specification of half-spaces is faster, even when it requires more refinements with open-loop feedback. These results answered our research question **E2.2**, which was about the trade-offs between the different refinement criteria.

Display of Selectable Objects. 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 displayed out-of-context (as in SQUAD), through the use of menus, which may be sorted in some way or arranged randomly.

We have tested these two options in two different studies (SQUAD, and Selection Shootout). The implications of performing selection out of the object's original context are discussed in length in 3.3, 4, and 5.1.1. For example, this requires the target object to be visually distinct, since contextual cues are lost. The design of our Double Bubble technique contains a number of counter-measures to minimize this and other issues. The comparison between in- and out-of-context display of selectable object answers our research question **E2.3**, which was about the trade-offs between the different display of selectable objects options.

Refinement Control. The refinement control 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.

In our study about our zooming techniques, we compared user-controlled refinement against technique-enforced, and found no significant differences, even though technique-enforced was preferred by our participants. We have successfully designed techniques that fall under both categories, and found no major trade-offs between the two. This answers our research question **E2.4**, which was about the trade-offs between different refinement control options.

Feedback. The last dimension we identified is related to the interaction feedback loop for refinement. It can be either an open-loop, in which actions are constantly tracked and gesture registration, tracking, and completion are done in one discrete step, or closed-loop, in which users can adjust gestures and correct mistakes in real-time before they commit to a selection or refinement. All single-object selection techniques we created use closed-loop feedback, and the volume selection techniques have open-loop and closed-loop versions.

Our last study compared the two types of feedback directly, and we found closed-loop to be faster and more accurate than open-loop feedback, while requiring fewer refinements and resulting in fewer mistakes. However, we believe these results are closely linked to the type of device and interaction metaphors we chose: gesture-based input using the Leap Motion device. More

investigation is needed to fully answer our research question E2.5, which talks about the trade-offs between the different types of feedback.

8 Conclusions

In this work, we have presented the concept of progressive refinement for selection. We have designed a total of eight different techniques based on this concept and evaluated them through four user studies. In our first study (presented in chapter 3), we compared SQUAD to ray-casting, the most commonly used immediate selection technique based on the laser-pointer metaphor, and found that the former was always more accurate than and preferred over ray-casting. In addition, we found it to be faster in certain situations (small and medium target sizes, low distractor density around the target). SQUAD had a few limitations, however, mostly related to the fact that it takes the objects out of their original spatial context for its second selection phase (QUAD-menu).

In order to solve this issue, we designed a couple of techniques based on zooming and compared directly to ray-casting (presented in chapter 4). While we successfully solved the issue presented by SQUAD by always performing selection in-context, we found that ray-casting was faster than both zooming techniques in all conditions. Our Discrete Zoom design included a visual search time for every refinement, since it was based on selection of screen quadrants. On the other hand, our Continuous Zoom required too much precision from users since they had to continuously point to the target while performing zoom. Because of this, we also had to slow down the zooming animation to avoid mistakes. However, we found both techniques to be more accurate than regular ray-casting for all conditions.

Our third study had two goals: improve the design of the previous techniques to compare them against state-of-the-art immediate selection techniques, and to validate the results across a number of levels of ecological validity (shown in chapter 5). We created Double Bubble, a technique inspired by Bubble Cursor, SQUAD, and Expand, and FRIZ, a technique that combined the strengths of both zoom techniques into a new faster technique. We compared the improved progressive refinement techniques to Bubble Cursor and DyCoDiR. Our results showed that all techniques are faster and more accurate than ray-casting, and we found no significant differences in time against other immediate selection techniques. In some cases, progressive refinement techniques presented fewer errors, and they were considered easier to use. However, we found significant differences between the different levels of ecological validity, which showed how important it is to carefully design controlled environments to consider the limitations of all techniques.

In order to validate the concept of progressive refinement and show that it can be used with different interaction metaphors and methods, as well as to improve other selection tasks, we designed progressive refinement techniques based on gesture input for point cloud selection tasks (presented

in chapter 6). We created three techniques: Slice, which uses a plane specification gesture to create half-spaces in 3D; Lasso, which allows users to draw a lasso in screen space; and Volume Brush, which uses a resizable 3D cursor to provide full 3D selection through brushing. In our study, we found that Slice was faster, even though it requires more refinements to complete the tasks. We believe this is due to the fact that it is the simplest to use and understand (users simply perform a quick gesture to slice through the point cloud and cut it in half), and it requires less input precision (minimizing the limitations introduced by the device used: the Leap Motion controller). We also compared open-loop and closed-loop feedback versions of the techniques, and found that closed-loop was faster, more accurate and overall preferred, perhaps due to poor tracking and false-positives caused by the Leap. We also created the progressive refinement design space, which included all design dimensions found in previous work and those explored in our own work. Based on our results, we provided a set of recommendations related to each design dimension (presented in chapter 7).

8.1 Contributions and Limitations

Our main contributions can be summarized as follows:

- Definition of the concept of progressive refinement for selection
- Set of new single and multiple object selection techniques
- Design space with empirically validated design guidelines

We believe progressive refinement techniques should be used to improve usability of 3DUIs when selection accuracy is important and/or input precision is low. For example, in cluttered virtual environments, when there is a large number of small objects of interest, when using consumer-level input tracking, and when using gesture-based interaction. Through our four studies, we have shown that progressive refinement techniques can be as fast as state-of-the-art immediate selection techniques, are very accurate, are easier to use than other state-of-the-art immediate over other techniques, and are preferred over other techniques.

However, we must acknowledge the fact that most applications and tasks are too simple and do not need special purpose selection techniques, and in those cases a technique like ray-casting may be enough. Similar to immediate selection techniques that intend to solve the issues related to ray-casting, progressive refinement techniques require extra learning. While using immediate selection until progressive refinement is really needed can work (e.g., using the Bubble cursor size threshold like in Double Bubble), the cost of learning these techniques can offset the benefits of using them.

Additionally, there is an implicit assumption in all our studies that the devices affect the techniques

equally. For example, in our last study, the lack of precision in tracking for the Leap Motion device might affect Volume Brush more than Slice. This has to be taken into consideration when looking at our overall results (e.g., the comparison between open-loop and closed-loop) and when trying to replicate our studies, since different devices could result in better performance for techniques that require more precision.

8.2 Future Work

There are many directions that could be taken in future studies with progressive refinement. For example, gaming and scientific applications often involve selection of moving targets, and progressive refinement could be used as a way to improve these selections by using ghosting techniques (Hasan, Grossman, & Irani, 2011) or algorithms that predict selection (Jeffrey Cashion, Wingrave, & LaViola, 2013). Additionally, there are a number of other interaction modalities that could use progressive refinement to improve selection, such as touch-based interfaces and full-body interaction. In particular, for the rest of this section, we show some of our own early designs for progressive refinement selection for touch-based interaction.

One of the main design considerations for smartphone and tablet interfaces is that input is provided by the user's fingers. Unlike desktop PCs and our prior 3D user interfaces, there is no cursor involved in the interaction, and users simply touch directly what they want to select. In other words, the two phases of selection—*indication* of target and *commitment* of selection—are combined into a single action in touch-based interfaces, which means there is no way to predict where on the screen users will select their targets. For our zoom techniques, for example, this means that a zoom window preview would not show up until the user performs the zoom action. Thus, the design of progressive refinement techniques must accommodate for these issues. In the following section, we further validate the use of progressive refinement by proposing and discussing the design of techniques for touch interfaces.

The current design of zoom in Apple's iOS requires users to double-tap with three fingers to zoom into the selected region of the screen. Once zoom is complete, users can control the position of the zoom window by dragging with three fingers. The issue with this interface is that visual context is lost and users have to manually zoom out to reacquire it once the target is selected. For dynamic applications where context is important, such as viewing/editing videos with a timeline, this makes it impossible for users to zoom in, select a specific place in the timeline, and view the content of the video while manipulating it. In addition, the zoom in, drag, and zoom out inputs are cumbersome and likely to be difficult for users with motor disabilities.

Our design takes issues like this into account and focuses on improving selection performance without requiring as much motor precision from the user. Users touch the screen with two fingers to activate zoom. With the two fingers still pressed, they are presented with a zoom preview window, which they can move by moving their fingers. By releasing, the system automatically expands the selected window. If users simply tap with two fingers, zoom takes place immediately without the zoom preview window, allowing rapid zooming. Users can continue to zoom in until the target size is large enough. Once a target is selected with a single tap, the system automatically zooms out to the original view. Figure 59 illustrates this sequence of actions.

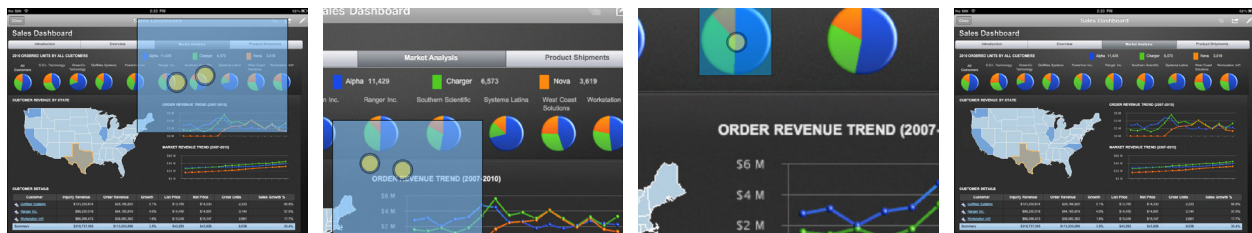


Figure 59 – Sequence of actions for the two-finger tapping technique. Yellow circles represent fingers touching the screen, and the blue square represents the zoom window preview.

An alternative design for this technique is to use quadrants, similar to the discrete zoom technique, to reduce the required precision. By touching the screen with two fingers, the quadrant menu becomes active. By moving the fingers around the screen, the user changes what quadrant is selected. Upon releasing, the system zooms into the selected quadrant. Again, users can simply tap with two fingers on a quadrant to zoom immediately, and can repeat this process until the target is large enough and then return to the original view. An example of how this technique works is presented in Figure 60. Unlike Apple's interface, these techniques can allow very rapid zooming based on imprecise and rough pointing. However, these first two designs may still require too much precision for users with severe motor issues.

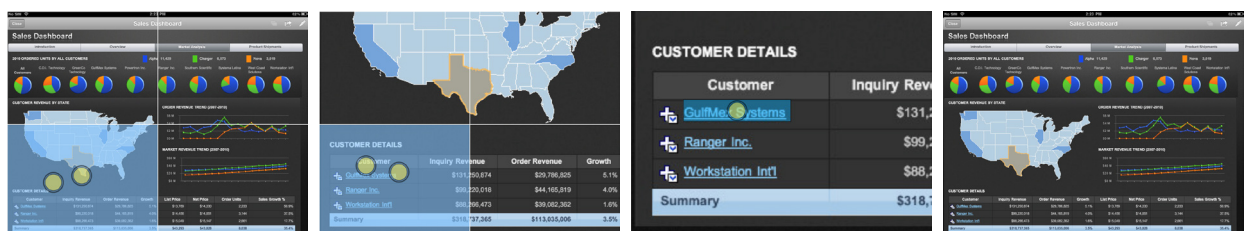


Figure 60 – Sequence of actions for the discrete zoom technique in a tablet application.

For people with more serious motor disabilities, we propose a switch interface to facilitate selection of the zoom area. Pressing the scan button causes the quadrants to become visible with one quadrant highlighted. Additional presses of the scan button change which quadrant is highlighted; pressing the selection button zooms into the highlighted quadrant. The sequence of actions performed by the users is illustrated in Figure 61. One open question for this design is how to finalize selection. Is it better to zoom in until there is only a single object visible, or to zoom in until there are fewer objects than some threshold number, after which the scan button will cause individual objects to be highlighted? Questions such as this will be investigated in our iterative design and evaluation process.



Figure 61 – Setup used for the switch interface and sequence of actions demonstrating the use of a progressive refinement technique combined with the switch interface in a tablet application.

A technique like SQUAD can be also used with touch-based devices. Similarly to the first zoom technique, a two-finger touch would highlight a circular region. When the fingers are lifted, users would be presented with the QUAD menu, which would allow them to select individual objects through a sequence of simple quadrant selections. Figure 62 illustrates the sequence of interaction for this modified version of SQUAD. However, using this with applications that use multiple identical buttons is not possible without grouping these with labels/identifiers.

Some users with motor control issues may not be able to extend or isolate one or two fingers. We will also investigate physical devices that can be held or strapped to the user that can be used as proxies for one and two-finger taps.

These designs, however, are not applicable to all scenarios found in real-world applications. For instance, while zooming provides selection in the object's spatial context and works fine when objects are stationary, moving targets would cause issues if the target moves out of the zoomed area. Entertainment applications, such as gaming, usually requires attention to be divided among several

subtasks, and eliminating the context by reducing the field of view (zooming) or creating a menu overlay may defeat the purpose of providing such selection interfaces to them.

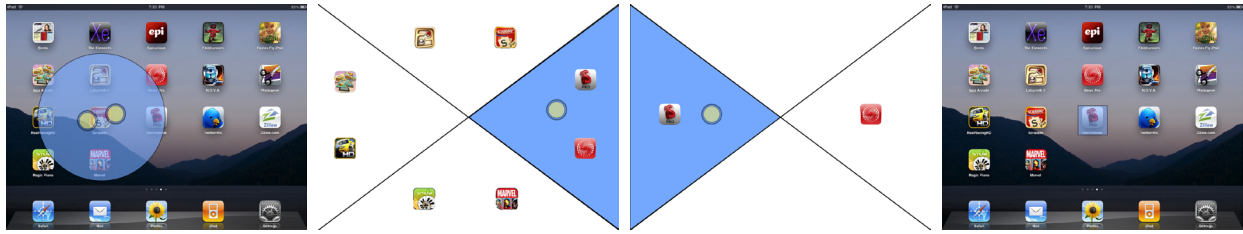


Figure 62 – SQUAD used on a tablet.

Alternative techniques that use concepts such as focus+context (Furnas, 1986) or explosion (Elmqvist, 2005) techniques can be designed. For instance, combining a dynamically changing control-display ratio with a fisheye view would allow users to increase the size of the target and increase precision at the same time, and would still provide a view of the overall context for the user (Figure 63), and it could be implemented with 3D input and touch-based devices.

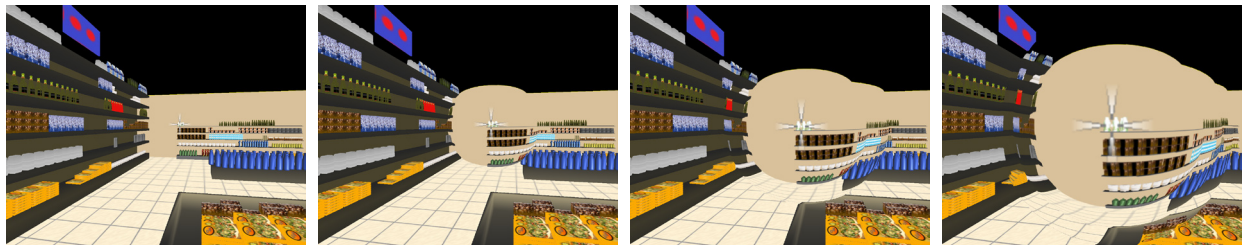


Figure 63 – Example of a focus+context technique gradually increasing the size of a target.

9 References

- Albinsson, P.-A., & Zhai, S. (2003). *High precision touch screen interaction*. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Ft. Lauderdale, Florida, USA.
- Appert, C., Chapuis, O., & Pietriga, E. (2010). *High-precision magnification lenses*. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Atlanta, Georgia, USA.
- Argelaguet, F., & Andujar, C. (2013). A survey of 3D object selection techniques for virtual environments. *Computers & Graphics*, 37(3), 121-136. doi: <http://dx.doi.org/10.1016/j.cag.2012.12.003>
- Bacim, F., Bowman, D., & Pinho, M. (2009). Wayfinding techniques for multiScale virtual environments *Proceedings of the IEEE Symposium on 3D User Interfaces* (pp. 67-74).
- Bacim, F., Kopper, R., Leal, A., Ni, T., & Bowman, D. A. (2010). 3DUI 2010 Contest Grand Prize Winners: Live Category Winners. *IEEE Computer Graphics and Applications*, 30(6), 93-96.
- Baudisch, P., Zotov, A., Cutrell, E., & Hinckley, K. (2008). *Starburst: a target expansion algorithm for non-uniform target distributions*. Paper presented at the Proceedings of the working conference on Advanced visual interfaces, Napoli, Italy.
- Bederson, B. B., & Hollan, J. D. (1994). Pad++: a zooming graphical interface for exploring alternate interface physics *Proceedings of the 7th annual ACM symposium on User interface software and technology* (pp. 17-26). Marina del Rey, California, United States: ACM.
- Benko, H., & Feiner, S. (2007, 10-11 March 2007). *Balloon Selection: A Multi-Finger Technique for Accurate Low-Fatigue 3D Selection*. Paper presented at the 3D User Interfaces, 2007. 3DUI '07. IEEE Symposium on.
- Benko, H., Wilson, A. D., & Baudisch, P. (2006). *Precise selection techniques for multi-touch screens*. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Montré#233;al, Qu#233;bec, Canada.
- Blanch, R., Guiard, Y., & Beaudouin-Lafon, M. (2004). *Semantic pointing: improving target acquisition with control-display ratio adaptation*. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.
- Bourgeois, F., & Guiard, Y. (2002). *Multiscale pointing: facilitating pan-zoom coordination*. Paper presented at the CHI '02 Extended Abstracts on Human Factors in Computing Systems, Minneapolis, Minnesota, USA.
- Bowman, D. A., Kruijff, E., LaViola, J. J., & Poupyrev, I. (2004). *3D User Interfaces: Theory and Practice*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc.
- Bowman, D. A., Kruijff, E., LaViola Jr, J. J., & Poupyrev, I. (2004). *3D user interfaces: theory and practice*. Addison-Wesley.
- Bragdon, A., DeLine, R., Hinckley, K., & Morris, M. R. (2011). *Code space: touch + air gesture hybrid interactions for supporting developer meetings*. Paper presented at the Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, Kobe, Japan.
- Burgess, R., Falcão, A. J., Fernandes, T., Ribeiro, R. A., Gomes, M., Krone-Martins, A., & de Almeida, A. M. (2015). Selection of Large-Scale 3D Point Cloud Data Using Gesture Recognition *Technological Innovation for Cloud-Based Engineering Systems* (pp. 188-195): Springer.
- Cashion, J., Wingrave, C., & LaViola, J. J. (2012). Dense and Dynamic 3D Selection for Game-Based Virtual Environments. *Visualization and Computer Graphics, IEEE Transactions on*, 18(4), 634-642. doi: 10.1109/TVCG.2012.40
- Cashion, J., Wingrave, C., & LaViola, J. J. (2013). *Optimal 3D selection technique assignment using real-time contextual analysis*. Paper presented at the 3D User Interfaces (3DUI), 2013 IEEE Symposium on.
- Debarba, H., Nedel, L., & Maciel, A. (2012, 4-5 March 2012). *LOP-cursor: Fast and precise interaction with tiled displays using one hand and levels of precision*. Paper presented at the 3D User Interfaces (3DUI), 2012 IEEE Symposium on.

- Elmqvist, N. (2005). *BalloonProbe: reducing occlusion in 3D using interactive space distortion*. Paper presented at the Proceedings of the ACM symposium on Virtual reality software and technology, Monterey, CA, USA.
- Fekete, J.-D., Elmqvist, N., & Guiard, Y. (2009). *Motion-pointing: target selection using elliptical motions*. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Boston, MA, USA.
- Findlater, L., Jansen, A., Shinohara, K., Dixon, M., Kamb, P., Rakita, J., & Wobbrock, J. O. (2010). *Enhanced area cursors: reducing fine pointing demands for people with motor impairments*. Paper presented at the Proceedings of the 23rd annual ACM symposium on User interface software and technology, New York, New York, USA.
- Forlines, C., Balakrishnan, R., Beardsley, P., van Baar, J., & Raskar, R. (2005). Zoom-and-pick: facilitating visual zooming and precision pointing with interactive handheld projectors *Proceedings of the 18th annual ACM symposium on User interface software and technology* (pp. 73-82). Seattle, WA, USA: ACM.
- Forsberg, A., Herndon, K., & Zeleznik, R. (1996). *Aperture based selection for immersive virtual environments*. Paper presented at the Proceedings of the 9th annual ACM symposium on User interface software and technology, Seattle, Washington, USA.
- Frees, S., Kessler, G. D., & Kay, E. (2007). PRISM interaction for enhancing control in immersive virtual environments. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 14.
- Fu, C.-W., Goh, W.-B., & Ng, J. A. (2010). *Multi-touch techniques for exploring large-scale 3D astrophysical simulations*. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Atlanta, Georgia, USA.
- Furnas, G. W. (1986). Generalized fisheye views *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 16-23). Boston, Massachusetts, United States: ACM.
- Grossman, T., & Balakrishnan, R. (2005). The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 281-290). Portland, Oregon, USA: ACM.
- Grossman, T., & Balakrishnan, R. (2006). The design and evaluation of selection techniques for 3D volumetric displays *Proceedings of the 19th annual ACM symposium on User interface software and technology* (pp. 3-12). Montreux, Switzerland: ACM.
- Grossman, T., Baudisch, P., & Hinckley, K. (2009). *Handle Flags: efficient and flexible selections for inking applications*. Paper presented at the Proceedings of Graphics Interface 2009, Kelowna, British Columbia, Canada.
- Guiard, Y., & Beaudouin-Lafon, M. (2004). Target acquisition in multiscale electronic worlds. *Int. J. Hum.-Comput. Stud.*, 61(6), 875-905. doi: 10.1016/j.ijhcs.2004.09.005
- Haan, G. D., Koutek, M., & Post, F. H. (2005). IntenSelect: Using Dynamic Object Rating for Assisting 3D Object Selection *Virtual Environments 2005* (pp. 201-209).
- Hansen, D. W., Skovsgaard, H. H. T., Hansen, J. P., & Møllenbach, E. (2008). Noise tolerant selection by gaze-controlled pan and zoom in 3D *Proceedings of the 2008 symposium on Eye tracking research & applications* (pp. 205-212). Savannah, Georgia: ACM.
- Hasan, K., Grossman, T., & Irani, P. (2011). *Comet and target ghost: techniques for selecting moving targets*. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.
- Hertzum, M., & Hornbæk, K. (2007). Input techniques that dynamically change their cursor activation area: A comparison of bubble and cell cursors. *Int. J. Hum.-Comput. Stud.*, 65(10), 833-851. doi: 10.1016/j.ijhcs.2007.05.001
- Jackson, B., Coffey, D., & Keefe, D. F. (2012). Force Brushes: Progressive Data-Driven Haptic Selection and Filtering for Multi-Variate Flow Visualizations. *Short Paper Proc. EuroVis*, 7-11.
- Jog, N. K., & Shneiderman, B. (1997). Starfield visualization with interactive smooth zooming *Proceedings of the third IFIP WG2.6 working conference on Visual database systems 3 (VDB-3)* (pp. 3-14). London, UK: Chapman & Hall, Ltd.

- Kabbash, P., & Buxton, W. A. S. (1995). *The "prince" technique: Fitts' law and selection using area cursors*. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Denver, Colorado, United States.
- Keefe, D. F., Zeleznik, R. C., & Laidlaw, D. H. (2008). *Tech-note: Dynamic dragging for input of 3D trajectories*. Paper presented at the 3D User Interfaces, 2008. 3DUI 2008. IEEE Symposium on.
- König, W. A., Gerken, J., Dierdorf, S., & Reiterer, H. (2009). *Adaptive Pointing --- Design and Evaluation of a Precision Enhancing Technique for Absolute Pointing Devices*. Paper presented at the Proceedings of the 12th IFIP TC 13 International Conference on Human-Computer Interaction: Part I, Uppsala, Sweden.
- Kopper, R. (2011). Understanding and Improving Distal Pointing Interaction *Computer Science* (Vol. PhD). <http://scholar.lib.vt.edu/theses/available/etd-07012011-195812/>: Virginia Tech.
- Kopper, R., Bacim, F., & Bowman, D. (2011, march). *Rapid and accurate 3D selection by progressive refinement*. Paper presented at the Proceedings of the IEEE Symposium on 3D User Interfaces.
- Kopper, R., Bowman, D. A., Silva, M. G., & McMahan, R. P. (2010). A human motor behavior model for distal pointing tasks. *International Journal of Human-Computer Studies*, 68(10), 603-615. doi: 10.1016/j.ijhcs.2010.05.001
- Leap. (2012). Leap Motion controller. from <https://www.leapmotion.com/>
- Liang, J., & Green, M. (1993). JDCAD: A Highly Interactive 3D Modeling System *3rd International Conference on CAD and Computer Graphics* (pp. 217-222). Beijing, China.
- Lucas, J. (2005). *Design and Evaluation of 3D Multiple Object Selection Techniques*, vol. M.S., Virginia Polytechnic Institute and State University, Blacksburg, VA, USA.
- MacKenzie, I. S. (1992). Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7, 91-139.
- Microsoft. (2012). Microsoft Kinect SDK. from <http://www.microsoft.com/en-us/kinectforwindows/>
- Mine, M. R. (1995). *Virtual Environment Interaction Techniques*. Chapel Hill, NC, USA: University of North Carolina at Chapel Hill.
- Mine, M. R., Brooks Jr., F. P., & Sequin, C. H. (1997). Moving objects in space: exploiting proprioception in virtual-environment interaction *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (pp. 19-26). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.
- Olwal, A., Feiner, S., & Heyman, S. (2008). *Rubbing and tapping for precise and rapid selection on touch-screen displays*. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Florence, Italy.
- Owada, S., Nielsen, F., & Igarashi, T. (2005). *Volume catcher*. Paper presented at the Proceedings of the 2005 symposium on Interactive 3D graphics and games.
- Peck, S. M., North, C., & Bowman, D. (2009). *A multiscale interaction technique for large, high-resolution displays*. Paper presented at the Proceedings of the 2009 IEEE Symposium on 3D User Interfaces.
- Pierce, J. S., Forsberg, A. S., Conway, M. J., Hong, S., Zeleznik, R. C., & Mine, M. R. (1997). Image plane interaction techniques in 3D immersive environments *Proceedings of the 1997 symposium on Interactive 3D graphics* (pp. 39-ff.). Providence, Rhode Island, United States: ACM.
- Prachyabrued, M., Ducrest, D. L., & Borst, C. W. (2011). *Handymap: a selection interface for cluttered VR environments using a tracked hand-held touch device*. Paper presented at the Proceedings of the 7th international conference on Advances in visual computing - Volume Part II, Las Vegas, NV, USA.
- Ramos, G., Robertson, G., Czerwinski, M., Tan, D., Baudisch, P., Hinckley, K., & Agrawala, M. (2006). *Tumble! Splat! helping users access and manipulate occluded content in 2D drawings*. Paper presented at the Proceedings of the working conference on Advanced visual interfaces, Venezia, Italy.
- Ren, G., & O'Neill, E. (2013). 3D selection with freehand gesture. *Computers & Graphics*, 37(3), 101-120.

- Roudaut, A., Huot, S., & Lecolinet, E. (2008). *TapTap and MagStick: improving one-handed target acquisition on small touch-screens*. Paper presented at the Proceedings of the working conference on Advanced visual interfaces, Napoli, Italy.
- Schaffer, D., Zuo, Z., Bartram, L., Dill, J., Dubs, S., Greenberg, S., & Roseman, M. (1993). Comparing fisheye and full-zoom techniques for navigation of hierarchically clustered networks *Proceedings of Graphics Interface* (pp. 87-96): Morgan-Kaufman.
- Shan, G., Xie, M., Gao, Y., & Chi, X. (2014). Interactive visual exploration of halos in large-scale cosmology simulation. *Journal of Visualization*, 17(3), 145-156.
- Shreiner, D., & Group, T. K. O. A. W. (2009). *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1*: Addison-Wesley Professional.
- Steed, A., & Parker, C. (2004). 3D Selection Strategies for Head Tracked and Non-Head Tracked Operation of Spatially Immersive Displays *8th International Immersive Projection Technology Workshop* (pp. 13-14).
- Steed, A., & Parker, C. (2005). Evaluating effectiveness of interaction techniques across immersive virtual environmental systems. *Presence: Teleoperators and Virtual Environments*, 14, 511-527.
- Stoppel, S., Hege, H.-C., & Wiebel, A. (2014). *Visibility-driven depth determination of surface patches in direct volume rendering*. Master's thesis, Fachbereich Mathematik und Informatik, Freie Universität Berlin, 2013. <http://forschung.awmw.org/PDF/stoppelMA.pdf>.
- Teather, R. J., Pavlovych, A., Stuerzlinger, W., & MacKenzie, I. S. (2009, 14-15 March 2009). *Effects of tracking technology, latency, and spatial jitter on object movement*. Paper presented at the IEEE 3DUI.
- Ulicny, B., Ciechomski, P. d. H., & Thalmann, D. (2004). Crowdbush: interactive authoring of real-time crowd scenes *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 243-252). Grenoble, France: Eurographics Association.
- Ulinski, A., Zambaka, C., Wartell, Z., Goolkasian, P., & Hodges, L. F. (2007). *Two handed selection techniques for volumetric data*. Paper presented at the 3D User Interfaces, 2007. 3DUI'07. IEEE Symposium on.
- Vanacken, L., Grossman, T., & Coninx, K. (2007). Exploring the Effects of Environment Density and Target Visibility on Object Selection in 3D Virtual Environments. *Proceedings of the IEEE Symposium on 3D User Interfaces*, 115-122.
- Weichert, F., Bachmann, D., Rudak, B., & Fisseler, D. (2013). Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, 13(5), 6380-6393.
- Wiebel, A., Vos, F. M., Foerster, D., & Hege, H.-C. (2012). WYSIWYP: what you see is what you pick. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12), 2236-2244.
- Wigdor, D., & Wixon, D. (2011). *Brave NUI world: designing natural user interfaces for touch and gesture*: Elsevier.
- Wingrave, C. (2009). *3DUI flavors beyond vanilla*. Paper presented at the SEARIS Workshop.
- Worden, A., Walker, N., Bharat, K., & Hudson, S. (1997). *Making computers easier for older adults to use: area cursors and sticky icons*. Paper presented at the Proceedings of the ACM SIGCHI Conference on Human factors in computing systems, Atlanta, Georgia, United States.
- Yu, L., Efstathiou, K., Isenberg, P., & Isenberg, T. (2012). Efficient structure-aware selection techniques for 3D point cloud visualizations with 2DOF input. *Visualization and Computer Graphics, IEEE Transactions on*, 18(12), 2245-2254.
- Zhai, S., Kong, J., & Ren, X. (2004). Speed-accuracy tradeoff in Fitts' law tasks: on the equivalency of actual and nominal pointing precision. *International Journal of Human-Computer Studies*, 61, 823-856.
- Zhao, S., Agrawala, M., & Hinckley, K. (2006). Zone and polygon menus: using relative position to increase the breadth of multi-stroke marking menus *Proceedings of the SIGCHI conference on Human Factors in computing systems* (pp. 1077-1086). Montré#233;l, Qu#233;bec, Canada: ACM.

Zhou, W., Correia, S., & Laidlaw, D. H. (2008). *Haptics-Assisted 3D Lasso Drawing for Tracts-of-interest Selection in DTI Visualization*. Paper presented at the IEEE Visualization Poster Compendium.