Virtual reality (VR) enables users to achieve what may not be possible in the physical world. Though many user interfaces have been developed for simulating or adapting real-world features (such as providing realistic tactile feedback [5]), it has long been argued that the real power of VR lies in creating a “better” reality by utilizing “magical” techniques that while being unrealistic, provide a better user experience [1, 46, 63, 65, 67]. One primary advantage of such interaction techniques is to overcome human limitations in terms of cognitive, perceptual, physical, and motor capabilities [46]. For example, existing techniques enable the user to interact with distant objects [78] and teleport around virtual environments [37], which are impossible in the physical world. This research focuses on one such interaction—selecting fully-occluded targets in VR.

The challenge of interacting with fully-occluded targets is prevalent within virtual environments. Structural elements, like walls, can easily hide and prevent users from accessing the objects behind them [23, 47, 79] (see Figure 1). In another example, high-dimensional data visualizations are also likely to obscure a datapoint of interest from being acquired by analysts [6, 20, 54, 76]. Further, when building 3D models in virtual environments, it might be cumbersome to select and thus manipulate hidden components, such as an engine hidden inside a virtual model of a motor vehicle [4].

However, existing selection techniques in VR are limited in their effectiveness for selecting fully-occluded targets. Based on the available literature on the topic, we argue that the main challenges are (1) the deficiency of the formulation of the problem in VR and general strategies to solve it; (2) the lack of effort in combining 3D occlusion management techniques to facilitate the discovery phase of the selection process [3]; and (3) the absence of a thorough evaluation and comparison of techniques that manipulate the key factors related to fully-occluded target selection. We aim to fill these gaps in this paper.

We first formulate the fully-occluded target selection problem and frame an approach to address it. We then derive a design space, which inspired seven potential techniques for selecting fully-occluded targets in VR. We present a user study that compares these techniques based on the realism of the scene. Among the selectable objects, there is commonly one primary target that the user intends to interact with, while all the selectable objects act as distractors. The target can switch when the user’s intention changes. A target is defined to be fully-occluded from a viewpoint if it can not be seen from any viewing direction of the user.

In this section, we first introduce previous work regarding occluded target selection in VR. Previous work normally assumed that the target location was known or only partially hid the target. In addition, important factors, such as occlusion layers, target depths, object densities, and the estimation of target locations, can affect technique performance. Our findings from both studies and distilled recommendations can inform the design of future VR systems that offer selections for fully-occluded targets.
is the whole visualization. After that, the user can use a supporting technique to “locate” the fully-occluded target and then pinpoint the target to perform the selection.

In line with the problem space discussed above, we propose a general strategy to tackle the fully-occluded target selection problem. Once a user has a selection intention, a visualization of the target needs to be displayed to make it “visible” to the user from their viewpoint. However, as the technique will not know which object the user is aiming at, a group of potential objects, possibly within the user’s estimated occurrence areas, will be presented. Next, the technique helps the user to disambiguate the list of selectable objects and provides activation feedback when an object is being pointed at. Finally, the technique should allow the user to select an object by pressing a trigger and receive confirmation feedback. The general strategy described above for selecting fully-occluded targets is summarized in Figure 2.

3 RELATED WORK

In this section, we first introduce previous work regarding occluded target selection in VR. We then present the techniques related to visualization and selection, which are the two main steps in our general problem-solving strategy.

3.1 Occluded Target Selection in VR

Previous research in VR has explored object selection under cluttered and occlusion conditions, mostly for partially-occluded targets. One successful technique is Depth Ray [38], which attaches a movable marker onto the selection ray. The object intersected by a ray and closest to the marker can be selected. In another study, Depth Ray was shown to be effective for selecting a target that was completely occluded by making the objects adjacent to the ray semi-transparent [72]. However, the target was first visible to the user for 2 seconds before it got fully-occluded by distractors, which is unlikely to be the case in real applications. Furthermore, only one layer of occlusion was introduced, and the target shape was different from the distractors, which made the task much simpler. In real scenarios, multiple layers of occlusion can be presented, and the target can also be similar to the distractors [54]. Future work has refined [111], and applied similar techniques for collaborative work [2] and object manipulation [57].

Some techniques try to separate objects in a cluster by translating them into new positions. For example, Flower Ray uses a two-step approach: the user first point at an object cluster through a virtual ray, and then press to trigger to separate them into a marking menu [38, 44]. An updated version of Flower Ray uses a fixed-size cone to replace the ray to avoid missing small targets [26]. Both techniques have not been tested under dense conditions, where objects could still be partially- or fully-occluded even if they are translated to new positions.

Progressive refinement techniques that take the advantages of rearranging the objects in a more organized way, commonly into a new view, can also be suitable for selecting fully-occluded targets. SQUAD [7, 42] allows users to cast a sphere onto multiple objects and interactively narrow down their selections using a quad-menu. Another technique called Expand [9, 18, 19] (not in VR HMD) enables users to zoom into the target area and reorganize the objects onto a grid for a second phase selection. Expand was shown to perform faster than SQUAD in dense environments. Later works extend such techniques by using a mobile touchscreen as input [25] and arranging objects in different layouts (circular layout rather than a grid) [49, 56]. However, none of them have been formulated under the context of fully-occluded target selection, nor have they been thoroughly compared to other techniques presented in this section. Nevertheless, we drew inspiration from these techniques when developing our techniques for fully-occluded target selection in VR.

There are some other techniques that are promising for selecting fully-occluded targets: flexible pointer [51] uses a curved ray which could bypass the distractors, iSith [77] determines the target by using the interaction point of two rays, VirtualGrasp [78] retrieves an object by simulating the gesture as if grasping the target object, X-Ray Vision [40] reveals hidden content by looking at a “scaffolding pattern”, and Outline Pursuits [64] selects an occluded target by matching its outline with smooth pursuit eye movement [73]. While these techniques provide interesting concepts, substantial tweaks would be needed for them to be suitable for general fully-occluded target selection scenarios. For example, VirtualGrasp [78] can not deal with objects with an identical shape.

We summarise the following three gaps in the literature:

- The fully-occluded target selection problem has not been established in VR. Previous work normally assumed that the target location was known or only partially hid the target. In addition, important factors, such as layers of occlusions, were not identified.

- Limited work has tried to combine occlusion visualizations to support the discovery phase of the targets, as they mainly focus on the selection phase. However, as fully-occluded targets can cause some uncertainties with their locations, visualizations that help with the search phase are essential.

- A thorough evaluation and comparison of different types of techniques that could be potentially used for fully-occluded target selection are missing.

3.2 3D Occlusion Visualization

Elmqvist and Tsigas reviewed fifty 3D occlusion management techniques for visualizations [31] and extracted five design patterns from these techniques. Next, we highlight important work in the three patterns that are more relevant to our research.

Multiple Viewports. The multiple viewports pattern is characterized by embedding alternate (often separate) viewports/windows to the main view. Examples include World In Miniature (WIM) [67, 70], which generates a small, handheld copy of the entire world, and Worldleets [33], which inserts multi-perspective viewpoints of an environment into the main view [13, 58, 76]. Recent work presents 3DMini-map [79], which helps to convey distance and direction information of off-screen and occluded targets. However, selecting objects directly on these visualizations is still underexplored.

Virtual X-Ray. The virtual X-ray pattern makes objects visible by turning occlusion surfaces invisible or semi-transparent. Making front objects transparent can benefit the discovery of objects that hide behind [27, 30, 41, 68, 82]. However, it is known to suffer from the “Superman’s X-ray vision” problem [48]—when there are too many occlusion layers, users are not able to make sense of the depth relationships of objects. Others have explored a cutaway view [16, 23, 28, 34], which eliminates or cuts holes over unwanted distractors.

Volumetric Probes. Volumetric probes normally use a probe to transform objects by removing or separating them. The above-mentioned disambiguation techniques, which reorganized potential targets on a new
view [74], could be counted as one stream. Other techniques have attempted to scale [22], translate [8, 15, 29, 32, 55], and distort [17, 24] objects in the scene in order to reveal the hidden objects. The transformation of the object needs to be carefully controlled so that the object is not occluded by new distractors, especially in a dense environment [32].

3.3 Selection Techniques in VR

There are two main categories of selection techniques in VR: virtual hand and virtual pointing [3]. Since a plethora of techniques have been proposed under those two categories, we direct interested readers to surveys on the topic [3, 46], and more recent works [11, 12, 71, 80]. RayCasting [11, 50] is one of the most common techniques for 3D object selection in virtual environments. In RayCasting, a visible ray emanates from the tracked hand position to the direction of where the hand is pointing at, and the first object that is intersected by the ray can be selected [46]. Despite its usefulness, the performance of RayCasting deteriorates when selecting distant or small objects. Researchers have been actively seeking solutions to enhance its performance, especially in dense environments [36, 72]. Recent work has compared different visual feedforwards for RayCasting and suggests that highlighting the nearest target was the most efficient way in terms of selection performance [11]. Another approach is to try to minimize input noise with the use of algorithms and computational models [11, 80]. We utilized some of the techniques mentioned above to strengthen our fully-occluded target selection techniques. We illustrate this aspect in more detail in the description of the developed techniques.

4 DESIGN SPACE

Our general strategy for selecting fully-occluded target suggests that the problem can be solved in five steps (visualization, disambiguation, activation, selection, and feedback). Here, we focus on the three main steps, which are visualization, disambiguation, and selection. We maintain the other two the same across all the techniques. The activation indication was provided by outlining the target, and the confirmation feedback was given by sound. Regarding the three focused steps, we have identified the following six primary considerations for designing fully-occluded target selection techniques.

Visualization Patterns: which type(s) of the visualization pattern, among the ones that are identified by Elmquist et al. [31] (typically multiple viewpoints, virtual X-ray, and volumetric probe) is/are utilized to visualize the target?

Visualization Size: what is the size of the visualization area? Are we applying the visualization to only limited areas, or more extensive areas (even the whole environment)? For instance, to visualize the objects, we can make a small area transparent, however, we can also tweak the whole scene to do so.

Visualization Versatility: will users be able to specify which area(s) they want to apply the visualization? How precise can it be (in an arbitrary shape or a constrained region)? In real use cases, the users will have different estimations of where the target might occur, thus it is important to define their belief/guess accurately.

Disambiguation Invariances: when applying the disambiguation technique, which property (or properties) of the original objects will be maintained? These properties may include object position, size, relation, and appearance. For example, if we are asked to select a datapoint among other datapoints that have the same appearance, re-arranging all of them into new positions might not be ideal.

Selection Techniques: what type of selection techniques will be applied? Are we embedding selection enhancement techniques or filter out the noisy input? These decisions are likely to be highly related to selection performance. In the initial exploration, we mainly focus on the selection techniques that are based on pointing (Raycasting) and virtual hands without adding selection enhancements.

Input Modality: which input modality (modalities) are used for selecting the fully-occluded target? While many types of input modality exist (voice, gaze, gesture, etc.), we focus on controller input. A survey of currently available controllers on the market showed that most controllers were equipped with at least a touchpad or a joystick (2 degrees-of-freedom input, 2DOF), a trigger (1DOF input), and buttons (only on/off). The controller itself can be 3DOF (only rotation can be detected) or 6DOF (both rotation and translation can be recognized). Different techniques might need to employ different inputs. In our research, we used a joystick, a trigger, and a button of an Oculus Touch controller throughout the studies. The design space can be expanded in the future when investigating other input modalities to achieve the functionalities of each technique (e.g. hand-tracking).

5 POTENTIAL TECHNIQUES

Based on the design space, we developed the following nine potential techniques with several iterations and pilot tests. These techniques are summarized and visualized in Figure 3. The following technique descriptions adhere to the design space. For an explicit mapping between the design space and the techniques, please refer to our supplementary materials.

Alpha Cursor: this technique is inspired by previous work that attaches a movable cursor onto the selection ray [11, 38]. With AlphaCursor, users control the cursor to come closer or go deeper into the environment at a constant speed by pushing the joystick forward or backward (see Figure 3b). In contrast to previous work, if the distance between the cursor and the user is larger than the distance between an object to the user, the object becomes fully transparent. The transparency manipulation is applied to the whole environment, and all objects maintain their original position and size during the disambiguation phase. RayCasting, which uses the trigger for selection confirmation, then selects the desired object.

Flower Cone: in FlowerCone (see Figure 3c), users select objects in two phases. First, the user controls a cone to match the estimated area of where the target might occur. The size of the cone can be adjusted by tilting forward/backward the joystick. When pressing the trigger, the user enters the second selection phase, in which all objects within that cone are presented on a grid. The user can select the target directly on the grid with RayCasting, or, if the target is not on the grid, the user can press the button to go back and resize the cone again. This technique combines visualization and disambiguation by using the grid layout. The visualization size can be controlled through the size of the flat circular base of the cone. However, the grid layout changes the original location and size of the object.

Gravity Zone: as shown in Figure 3d, GravityZone translates all objects in the scene to come closer or further away in a constant speed by tilting the joystick forward or backward. If the distance between an object and the user is smaller than a threshold, the object will be fully transparent. It is similar to AlphaCursor in that both of them make the objects transparent based on their relative depth. However, in contrast to AlphaCursor, GravityZone manipulates all the objects in the scene rather than the cursor. The location and size of the objects are changed during the translation, but their relative position is not altered. Raycasting is used to make the selection.

Grid Wall: inspired by Expand [19], in this technique, when the user presses the controller button, all objects are arranged on a grid (see Figure 3e) with a constant scale factor. We did not use the zoom-in feature from Expand as it can make participants dizzy in VR. GridWall completely reorganizes all objects in the scene to a new location with a different size. The user can select the target on the grid with RayCasting. The original location information of the object is temporarily lost with the grid layout.

Lasso Grid: with LassoGrid, users draw a trace in any shape by long-pressing the trigger (see Figure 3f). All objects within the trace, are presented on a grid layout when releasing the trigger for the second stage of selection. If the trace is not closed, the program completes it automatically. RayCasting is used to select the target on the grid. Pressing the button allows the user to go back and draw the trace again.

Magic Ball: inspired by previous work [79] (which only explored visualization rather then selection), MagicBall removes unselectable distractors and creates a 3D mini-map of all the selectable objects inside a transparent sphere (see Figure 3g). The objects’ size and the distance between each other are both scaled-down, but the relative size and location information are both maintained. The user can select directly on the semi-transparent object proxies by moving the tip of the virtual
will have different estimations of where the target might occur, thus it is important to define their belief/guess accurately.

We applying the visualization to only limited areas, or more extensive with the use of algorithms and computational models [11, 80]. We have attempted to scale [22], translate [8, 15, 29, 32, 55], and distort [17, 24] objects, we can make a small area transparent, however, we can also change the environment at a constant speed by pushing the joystick forward or backward (see Figure 3b). In contrast to previous work, if the distance between the cursor and the user is larger than the distance to the target, the object becomes fully transparent. We identified a set of factors that we hypothesized could impact the selection performance directly, it can cause distraction and are quite likely to appear in real application scenarios (unwanted objects are spread across the whole environment).

Pressing the button allows the user to go back and draw the trace again. The transparency manipulation is applied to the whole environment, and all objects maintain their original position and size during the dis- occlusion. Different sizes of the occurrence area is likely to influence the target searching time.

Our general strategy for selecting fully-occluded target suggests that the selection techniques that are based on pointing (Raycasting) and grid wall, users control the cursor to come closer or go deeper into different size. The user can select the target on the grid with RayCasting. Different techniques might need to employ different inputs. In our research, we placed six distractors around the target (front, behind, up, down, left, and right). Density space is the distance between the six distractors surrounding the target.

6.2 Experimental Setup

To frame the experimental task for this research, we first consulted the past literature regarding target selection in 3D space. Existing tasks with perceivable patterns (e.g., [11, 66, 69, 81]), which users were required to select a set of fixed targets in a constant sequence, are not applicable in our case. This is because we wanted to vary the occurrence areas, which requires some randomness in the allocation of the target. Meanwhile, tasks based on interaction scenarios with the presence of some degree of unexpectedness, such as a game [19], might pose challenges to the control of variables. We decide to use more controlled tasks, which would still allow the randomization of target locations (such as [7, 38, 45, 53, 72]). However, as there is little work regarding fully-occluded target selection, we had to develop a new and reusable experimental task. Based on previous research, we designed the task as follows.

In the task, the user aimed to select a fully-occluded target among a set of distractors in a virtual environment. The target and the distractors had different colors, and the colors were generated from a pre-prepared list (we used seven colors in our case which were chosen to be easily distinguishable, see Figure 4). The task was divided into two phases:
We recruited 21 participants (13F/8M), aged between 19-39 (M = 24.5 ± 4.3). The study employed a within-subjects design where we compared the techniques for selecting fully-occluded targets in virtual environments. In terms of selection time, most techniques (all except GridWall) felt significantly faster than GridWall (N=7) felt significantly slower. The search task required users to search for a specific target in a 2D grid. In contrast, the repeat task required them to finish as fast and as accurately as possible. The results show that performance improved for most techniques when the objects appeared and when the selection was made (p = 0.036 for AlphaCursor and p < .001 for others) except GridWall (p = .219).

There was a statistically significant difference between TECHNIQUES regarding error rates (F(2,30) = 3.710, p = .002). Post-hoc analysis indicated that FlowerCone had a significantly higher error rate than AlphaCursor (p = .003) and GravityZone (p = .002).

7.3.3 Search Task - High Complexity

TECHNIQUE had a significant main effect on selection time, with a large effect size (F(2,221) = 13.276, p < .001, η²_p = 0.399). GridWall had a significant main effect on selection time, with a large effect size (F(2,221) = 13.276, p < .001, η²_p = 0.399).

We collected both performance data and subjective feedback from participants. Both selection time (the elapsed time between when the objects appear and when the selection is made) and error rate (the percentage of error trials for each condition) were recorded. We also measured the easiness of the techniques with the Single Easement Questionnaire [60] and the intrusiveness caused by them [52] on a 7-point scale. In addition, we asked participants to provide their preference ranking after finishing each technique and optionally also provide free-form feedback. We monitored the experiment from a computer, which showed the user’s current view in VR, to observe the use of the techniques.

The whole procedure lasted around 40 minutes for each participant. At the beginning of the study, participants were briefed about the purpose of the research and signed a consent form. They also completed a pre-experiment demographic questionnaire. After that, they were introduced to the VR device and the experimental task, where we required them to finish as fast and as accurately as possible. They then wore the VR headset and familiarised themselves with the virtual environment. Next, they proceeded to the formal experiment within a fixed physical area. The experiment was divided into seven parts (corresponding to the evaluation of seven techniques). In each part, there were three phases: practice, perform formal trials, and answer questions. In the practice phase, participants were taught about how to use the technique, and they could practice it as long as they wanted until they got familiar with it. Then they completed a series of formal trials. Finally, they were asked to complete the questionnaires mentioned above. Participants were allowed to rest between each condition. They were compensated with a $10 voucher at the end of the study.

7.3 Performance Results

In total, we collected 4704 data points (21 participants × 7 techniques × 2 complexities × 2 tasks × 8 repetitions) from the experiment. To analyze selection time, we discarded trials in which participants made a wrong selection (374 error trials, 8.0%), and removed outliers, in which the selection time was above three standard deviations from the mean (mean + 3sd) in each condition (92 trials, 2.0%). Such outliers are typically removed as they are likely to not represent the typical selection performance (e.g., small distraction during the experiment), and can skew results in a particular condition [72, 80]. The data regarding selection time were shown to be normally distributed (evidence from Kolmogorov-Smirnov tests and visual inspections), while the error rate data were not normally distributed and underwent pre-processing through Aligned Rank Transform (ART) [11, 75]. Next, we performed a repeated-measures ANOVA (RM-ANOVA) and Bonferroni-adjusted pairwise comparisons in each experiment scenario to analyze the selection time and error rate in each experimental condition.

The degrees of freedom produced by RM-ANOVA regarding selection time was adjusted using Greenhouse-Geisser correction. Both results are summarized in Figure 5.
We conducted an initial exploration and evaluation of the seven potential techniques for selecting fully-occluded targets in virtual environments. The study employed a within-subjects design where we compared the performance of the participants across two levels of complexity—low and high. The detailed parameters are as follows: density, higher target depths, and larger density space than the lower complexity condition. We ensured that there were considerable differences by modifying the variables of interest mentioned above, this task can be considered as a complex task. In total, we collected 4704 data points (21 participants × 75 users × 2 conditions × 7 techniques). In each part, participants were taught about how to introduce to the VR device and the experimental task, where we introduced the exact p-value when p < 0.05, ** = p < 0.01, and *** = p < 0.001. There was a statistically significant difference between techniques regarding error rates (F(6,120) = 3.204, p = 0.06). GravityWall was shown to have significant lower error rate than MagicBall (p = 0.01).

### 7.3.3 Repeat Task - Low Complexity

**Technique** was found to have a statistically significant effect on selection time, with a large effect size (F(2, 379) = 26.061, p < 0.001, η² = 0.566). GravityZone took the least time for selection when compared to most other techniques (α < 0.001), except AlphaCursor (p = 0.078).

There was no statistically significant difference between **Techniques** regarding error rates (F(6,120) = 0.945, p = 0.466).

### 7.3.4 Repeat Task - High Complexity

**Technique** was found to have a statistically significant effect on selection time, with a large effect size (F(3, 282) = 18.412, p < 0.001, η² = 0.479). LassoGrid was the fastest, but was similar to FlowerCone and GridWall (p = 1.000). LassoGrid was significantly faster than AlphaCursor (p = 0.031) and the remaining techniques (p < 0.001).

There was a statistically significant difference between **Techniques** regarding error rates (F(6,120) = 6.491, p < 0.001). AlphaCursor (p = 0.018) and GravityZone (p < 0.001) had much lower error rates than MagicBall. GravityZone led to less error than SmashProbe (p = 0.029).

### 7.3.5 Search Task vs. Repeat Task

In terms of selection time, most techniques (all p < 0.003) had significant improvements in the repeat phase of the low complexity condition except GridWall (p = 0.045) and MagicBall (p = 0.057). For high complexity condition, there was no statistically significant effect of task on selection time for GridWall (p = 0.970) and SmashProbe (p = 0.143), but there was for all the others (MagicBall: p = 0.031 and others: p < 0.001).

Regarding error rates, only MagicBall (p = 0.035) and SmashProbe (p = 0.043) improved in the low complexity condition. No significant difference was revealed in the high complexity condition (all p > 0.05).

### 7.4 User Feedback Results

The overall easiness and intrusiveness of the techniques were calculated by averaging the 7-point Likert scale results. We also computed the mean ranking and counted the number of first/second place for each technique. The results from both questionnaires are summarized in Table 1.

In terms of the free-form feedback, the comments were mostly focused on GridWall, MagicBall, and SmashProbe. Several participants (N=7) felt GridWall was somewhat “boring” because it simply arranged all the objects in a 2D grid. In contrast, SmashProbe was seen as “fun” to use (N=4). Some participants thought MagicBall provided a good overview of the objects (N=4) but was quite difficult for selecting the target when the object number was high (N=3).

### Table 1. The mean value (standard error) of easiness rating, intrusiveness rating, and preference ranking for all the techniques in Study 1.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Easy</th>
<th>Intrusiveness</th>
<th>Rank</th>
<th>#1/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlphaCursor</td>
<td>5.38</td>
<td>1.90</td>
<td>4.05</td>
<td>1/2</td>
</tr>
<tr>
<td>FlowerCone</td>
<td>5.81</td>
<td>1.95</td>
<td>3.38</td>
<td>4/3</td>
</tr>
<tr>
<td>GravityZone</td>
<td>5.86</td>
<td>1.76</td>
<td>3.05</td>
<td>5/3</td>
</tr>
<tr>
<td>GridWall</td>
<td>6.33</td>
<td>1.57</td>
<td>3.48</td>
<td>6/1</td>
</tr>
<tr>
<td>LassoGrid</td>
<td>5.76</td>
<td>1.86</td>
<td>3.86</td>
<td>1/7</td>
</tr>
<tr>
<td>MagicBall</td>
<td>4.33</td>
<td>3.19</td>
<td>5.19</td>
<td>0/3</td>
</tr>
<tr>
<td>SmashProbe</td>
<td>4.76</td>
<td>3.10</td>
<td>5.00</td>
<td>3/1</td>
</tr>
</tbody>
</table>

### 7.5 Summary and Discussion

The results show that performance improved for most techniques when participants moved from the search task to the repeat task. This is particularly true for the complex tasks, where selection time was significantly shortened in the repeat task. However, GridWall did not gain an advantage from the repeated selection, as the object order was randomized on the grid. SmashProbe did not improve significantly in the high complexity condition during the repeat phase. The selection phase of these techniques took a significantly longer time to complete when compared to the searching phase. As the repeat task was a replay of the previous task, the learning effect can also reduce the selection time and help users correct errors. Interestingly, the ranking of the techniques based on selection time almost did not change from the...
search task to the repeat task. This is likely caused by the fact that the first selection only narrowed down the participants’ estimation of the "occurrence area" of the target, while some searching was still needed in the subsequent selection.

For low complexity tasks, GravityZone and AlphaCursor performed better (both with shorter selection time and lower error rates). GridWall also yielded good performance, whereas other techniques were shown to take more time or have higher error rates. One possible cause for this is that for simpler tasks, GravityZone, AlphaCursor, and GridWall can reveal the target quickly with straightforward manipulations, while techniques like FlowerCone and LassoGrid required an extra layer of area specification. The performance of SmashProbe was comparable to FlowerCone and LassoGrid. In contrast, MagicBall was the slowest, mostly because it required some precision to select the small proxies of objects.

For high complexity tasks, techniques that arranged the objects on a grid were the most successful in terms of selection performance, with GridWall, LassoGrid, and FlowerCone clearly outperforming other techniques. For instance, performance when using AlphaCursor and GravityZone suffered when the task got complex. Searching the target became difficult for participants, as once missing the target, which was surrounded by the sea of distractors, the participant had to move back and forth (the cursor of AlphaCursor or the object clusters of GravityZone) to search for them. Navigating to the correct depth where the target located was cumbersome. Similarly, SmashProbe performed poorly, as the target can sometimes "jump" to places where it was still fully-occluded by others.

Furthermore, participants found that if they kept spreading all objects in such dense environments, it would lead to significant distraction. The complex scenario also further exacerbated the problems with MagicBall. This is because participants needed to have very high precision for selecting the duplicates of the objects, while pressing the trigger on the controller could easily cause hand tremors [14], which can lead to the wrong selection. Regarding easefulness and intrusiveness, all techniques were rated better than the middle point of the 7-point Likert scale. Participants rated GridWall the easiest technique, which also caused the least distraction. However, participants felt bored when using this technique as it no longer felt like 3D interaction. GravityZone, LassoGrid, FlowerCone, and AlphaCursor all got positive feedback in terms of these two scales.

On the other hand, MagicBall and SmashProbe were rated lower, given the difficulty of selecting targets with these techniques. MagicBall can cause a wrong selection due to the handshaking, and SmashProbe might lead to an unexpected spread of the objects when performing the selection. However, they were both seen as interesting by the participants. MagicBall built a nice overview of the objects, while SmashProbe created a level of unexpectedness, which could be fun for gaming purposes [59]. With regard to preference ranking, GravityZone was ranked highest, followed by techniques that employed the grid feature and AlphaCursor. Based on the study results and our observations, we extracted a set of design lessons for different kinds of scenarios and application purposes regarding fully-occluded target selection.

1.1. Use techniques with the grid feature (GridWall, LassoGrid, and FlowerCone) for dense environments. Our results showed that these techniques had much better performance in complex tasks. However, according to the design space, be aware that these techniques would not preserve the original scene (the original locations of objects).

1.2. Depth-based techniques (AlphaCursor and GravityZone) provide simple solutions to lower complexity tasks. They can also preserve the location information of objects. However, when many distractors are clustered with the target, it might be difficult for these techniques to navigate to the exact depth where the target is located.

1.3. A smaller-scaled duplicate of the whole environment (like MagicBall) can help provide location awareness in virtual environments [79]. However, requiring users to perform direct selection on the small object proxies can pose challenges, such as hand tremors [14].

1.4. It can be beneficial to use techniques that have some sort of unpredictability for recreational purposes (like SmashProbe). However, in dense and complex environments, such unexpectedness can obstruct the primary selection task. In addition, applying 3D features in a virtual environment rather than only using 2D surfaces (e.g., GridWall) could lead to a more enjoyable experience.

After summarizing the findings from this first study, we were interested in refining the most promising techniques further. We also wanted to explore how specific environmental factors (like the size of the occurrence areas, occlusion layers, target depths, and object densities) would affect the performance of the techniques.

8 Technique Refinement

The seven techniques can be categorized into three sets: grid-based (GridWall, LassoGrid, and FlowerCone), depth-based (GravityZone and AlphaCursor), and others (MagicBall and SmashProbe). As the techniques in the different sets are better suited for different application purposes, and the ones within a set have similar strengths and weaknesses, we decided to improve them according to their general features. Based on the results and our observations from Study 1, we refined the techniques as follows.

We first improved the techniques that used grids (GridWall, LassoGrid, and FlowerCone). In the experiment, we found that the RayCasting technique for selecting objects on the grid sometimes led to errors during the fast-paced movements, as a correct selection was confirmed only when the ray was "crossing through" the target. Therefore, we decided to add selection enhancement techniques for RayCasting in the grid selection phase. We highlighted the nearest object to the ray and confirmed the selection on it when the user pressed the trigger, as this was shown to be the most efficient visual feedforward by recent work [11] (see Figure 6a). Additionally, to preserve depth information of the objects, we scaled the distances between the objects on the grid and the user according to the object’s real distances to the user. This also adds some 3D features on the 2D grid surface. Although the visual size of the objects changed, our selection enhancement ensured that the effective size for selection was the same. Furthermore, we identified that randomizing the positions of the objects on the grid every time (for search and repeat tasks) was not efficient when there were a large number of objects. It could be more effective if objects were arranged by their distance to the ray (or center of the cone), from the closest to the farthest.

We also aimed to enhance the depth-based techniques (GravityZone and AlphaCursor). For some users, we observed that when the distractors in the front of the target were fairly close to it, navigating to the exact depth where the target located was laborious. Meanwhile, using a constant cursor speed might not be ideal for every user, as some might need it to be faster, while others want it to be slower. As a result, we made the cursor speed adjustable through the joystick input. The harder it was pushed/tilted, the faster the cursor became.

In addition, we found that most users had difficulties when selecting object proxies inside the mini-map, especially in the case where there was a large number of objects. To improve the selection, we combined the grid feature, which was shown to be effective for selection into MagicBall. Instead of employing the forward and backward movements of the joystick to translate the mini-map (which was not that useful
according to our inspection), it was used to scale up and down the transparent point cursor, which was used for selecting objects. Once the selection trigger was pressed, and the scaled-up cursor enclosed more than one object proxy, the objects that were inside the cursor would be arranged onto a grid for the second phase of selection (see Figure 6b). Users could still select objects from the mini-map directly if only one object proxy collided the cursor.

We first picked two techniques, LassoGrid from the grid-based techniques and GravityZone from the depth-based techniques, according to the empirical performance and user feedback, and applied the refinements as mentioned above. We also improved MagicBall, as many users preferred the small overview of objects, and the main problem with the technique was the difficulty caused by selection. Consequently, we evaluated the three refined techniques (LassoGrid+, GravityZone+, and MagicBall+) in the second study.

9 Study 2 - In-Depth Evaluation

To have a more thorough understanding of how different environmental factors might affect the performance of the techniques, we conducted a second study based on three refined techniques (GravityZone+, LassoGrid+, and MagicBall+).

9.1 Environmental Factors

Initially, we were interested in five essential environmental factors (occurrence area, occlusion layer, environmental density, target depth, and density space) which can have a substantial impact on target selection with the different techniques. However, evaluating all of them might pose a high workload for participants.

As a result, we combined environmental density and density space to one single factor called area density, as both these factors are related to the number of distractors inside a space unit. Area density specified the density of the objects within the occurrence area, intending to maintain the same level of difficulty for techniques within the targeting area. We assumed that objects within the targeting area might raise more challenges than the ones that were spread around the whole space. In this case, the density of the objects within the whole environment (outside of the occurrence area) would be set as constant. We ended up with four environmental factors, which are OCCURRENCE AREA, AREA DENSITY, OCCLUSION LAYER, and TARGET DEPTH.

9.2 Method

We recruited another set of 16 participants (9F/7M) between the ages of 20-32 (M = 24.6 ± 3.3) with different educational backgrounds from a local university campus. All participants had normal or corrected-to-normal vision. They rated their familiarity with VR as moderate (3.6 ± 1.7 on a 7-point scale). We used the same apparatus and devices as in the first study.

The study employed a within-subjects design with five factors: TECHNIQUE (GravityZone+, LassoGrid+, and MagicBall+), OCCURRENCE AREA (small and large), AREA DENSITY (low and high), OCCLUSION LAYER (less and more), and TARGET DEPTH (low and high). The details of the variables are summarized in the supplementary material. Three techniques appeared in a random sequence, while for each technique, we varied the four counterbalanced environmental factors. The techniques were well-distributed in terms of their order according to our post-hoc evaluation. In this study, only the search task was used, rather than including both search and repeat task, because 1) it decreased the workload of the participants, 2) we found techniques had similar rankings based on the selection time for both tasks, and 3) the search phase is likely to be more relevant to real application scenarios. In total we collected 3072 trials of data (16 participants × 3 techniques × 2 occurrence areas × 2 area densities × 2 occlusion layers × 2 target depths × 4 repetitions).

As with Study 1, we gathered selection time and error rate data, and observed each experiment. Additionally, we used two standardized questionnaires to assess the task workload and user experience. The workload was measured by RAW NASA-TLX [39], and the user experience was quantified by the short version of the User Experience Questionnaire (UEQ-S) [61]. These questionnaires are more comprehensive than the ones used in the first study, which were used for the simplicity of the experiment given the higher number of tested techniques. Both questionnaires were presented inside the virtual environment as previous work has shown that it can reduce study duration and user disorientation [62].

The study lasted approximately 35 minutes for each participant. A similar procedure as the first study was used. Participants were compensated with a $10 voucher.

9.3 Results

As in the first study, we discarded the error trials (174 errors, 5.7%) and the outliers (78 trials, 2.5%) to analyze the selection time. We employed a RM-ANOVA with Greenhouse-Geisser correction for analyzing the effect of each factor. Pairwise comparisons with Bonferroni adjustment were used for technique comparison. Error rate data was transformed using ART [75] and was then analyzed through a RM-ANOVA. Regarding user feedback, we summarised the results from the questionnaires in Table 2 and Figure 7.

Since we were interested in how the techniques were affected by different environmental factors, we only present the effects and interactions related to the factor TECHNIQUE.

9.3.1 Selection Time

A RM-ANOVA indicated that TECHNIQUE (F1, 375.20.627 = 20.039, p < .001, ηp2 = 0.572) had a significant main effect on selection time, with a large effect size. A post-hoc test revealed that LassoGrid+ was significantly faster than GravityZone+ (p < .001) and MagicBall+ (p < .001). GravityZone+ was also indicated to be faster than MagicBall+ (p = .048).

There were interaction effects between TECHNIQUE × AREA DENSITY (F1,1,356.20.336 = 6.090, p = .015, ηp2 = 0.289), TECHNIQUE × OCCLUSION LAYER (F1,864.27.954 = 7.365, p = .003, ηp2 = 0.329), and TECHNIQUE × TARGET DEPTH (F1,747.26.206 = 14.584, p < .001, ηp2 = 0.493), all with medium to large effect size. We present these interaction effects in Figure 8. No other interaction effects were found. Although there was no interaction between TECHNIQUE and OCCURRENCE AREA, OCCURRENCE AREA itself did have a significant main effect on selection time (F1,15 = 61.186, p < .001, ηp2 = 0.803).

<table>
<thead>
<tr>
<th>Technique</th>
<th>Pragmatic</th>
<th>Hedonic</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>GravityZone+</td>
<td>1.31 (&gt;avg.)</td>
<td>1.38 (&gt;avg.)</td>
<td>1.34 (&gt;avg.)</td>
</tr>
<tr>
<td>LassoGrid+</td>
<td>1.83 (exc.)</td>
<td>1.77 (good)</td>
<td>1.80 (exc.)</td>
</tr>
<tr>
<td>MagicBall+</td>
<td>1.56 (good)</td>
<td>2.05 (exc.)</td>
<td>1.80 (exc.)</td>
</tr>
</tbody>
</table>

Table 2. The results from the short version of User Experience Questionnaires (UEQ-S) which outline the pragmatic quality, hedonic quality, and overall quality of each technique. In the table, “>avg.” means “above average”, “exc.” means “excellent”.

Fig. 7. The results from Raw NASA-TLX Questionnaires. Error bars indicate the 95% confidence interval. Statistical significant effects are marked (** = p < .001).
9.3.2 Error Rate

TECHNIQUE had a significant main effect on error rate ($F_{2,705} = 8.027$, $p < .001$). A post-hoc test showed that GravityZone+ (5.96%, $p = .032$) and LassoGrid+ (3.51%, $p < .001$) had statistically significantly lower error rates than MagicBall+ (7.52%), with no significant difference between the two ($p = .490$).

A RM-ANOVA also revealed further interactions among the other factors. However, as the error rate was relatively low for all techniques (according to [11,66]) and better techniques clearly outperformed the worse ones in terms of performance (lower error rate techniques as well as lower selection time), we do not examine these results here in further detail. For detailed statistics, please refer to our supplementary materials.

9.4 Summary and Discussion

In this section, we first examine the influence of the four environmental factors on the three techniques and then compare the input techniques from different perspectives.

Occurrence area had a similar impact on all techniques—as it increased, the selection time of the three techniques also increased significantly. This was expected as an inaccurate prior estimate of where the target might be located (larger occurrence area) leads to a higher search time, prolonging the selection process.

Area density affected the selection performance of all techniques, but the magnitude of the effect was different, as indicated by the significant interaction effect. As the density increased, search and selection for GravityZone+ and MagicBall+ became much more difficult than for LassoGrid+. One potential reason for this difference is the fact that organizing the objects on a grid-like 2D layout demanded less effort for searching rather than the original clustered and overlapped 3D arrangements [19].

Occclusion layer only influenced the performance of GravityZone+. With GravityZone+, it can feel somewhat cumbersome navigating through multi-layers of distractors. However, when using LassoGrid+, which arranged objects in the target area on a grid, and MagicBall+, which provided a quick overview of all the objects, users were not impeded by these layers at all.

Target depth affected the performance of GravityZone+ and MagicBall+, but not of LassoGrid+. LassoGrid+ was invariant to the change of target depth, as it transformed the 3D region to a grid, regardless of the real depth of the objects. Although we added depth information on LassoGrid+, it only changed the visual size of the objects, but not its effective size with the selection enhancement technique [35,66]. For GravityZone+, when the target was located further, participants were required to navigate more to reach it, thus induced longer selection time. However, participants spent more time in selecting the lower depth target using MagicBall+ than the higher depth ones. This was because participants were observing the whole environment from the outside of the mini-map, lower depth target actually looked farther away. Hence, there might be more distractors on the way of getting the goal target.

After seeing how each environmental factor affected the performance of the techniques, we compared the techniques in terms of different measurements below. The performance data were consistent in terms of selection time and error rates. LassoGrid+ had the lowest selection time and error rate, while for MagicBall+ they were the highest. The NASA-TLX results also show a similar trend. Participants were more satisfied with their performance and had lower frustration and mental workload levels when using LassoGrid+. Concerning the UEQ-S results, LassoGrid+ was shown to have excellent pragmatic value, while MagicBall+ was rated outstanding in the hedonic quality. They both had excellent overall quality. However, GravityZone+ was rated just above-average on all aspects of the UEQ-S. It seemed to suffer from the “middle children syndrome” [43], where it did not look as innovative as MagicBall+ and was not as effective as LassoGrid+.

Therefore, its ratings from the participants were relatively low.

10 Design Recommendations

Based on the results from both studies, we distill design recommendations regarding choosing input techniques for the selection of fully-occluded target in virtual environments.

R1. When the goal of the task is rapid selection, we suggest using grid-based techniques (GridWall, FlowerCone, and LassoGrid+) to ensure optimal user performance. Use LassoGrid+ when it is difficult to decide which one of them to use, as it allows users to define their estimate of where targets might occur freely, and only one trigger/button will be needed for the whole selection process. Consider adding selection enhancement techniques (like highlighting the closest object) to improve performance further.

R2. When maintaining the object location information is essential (e.g., 3D plots), we recommend using depth-based techniques (AlphaCursor and GravityZone+) or MagicBall+. Our results indicate that GravityZone+ should be favored if better performance is needed. AlphaCursor and MagicBall+ can be used when it is not desirable to move the objects in the scene.

R3. If the technique is used for recreational purposes (like game applications), consider use SmashProbe and MagicBall+ as they are more exciting or have higher hedonic quality. However, avoid using SmashProbe when there are too many objects in the scene, as it could be very distracting.

R4. Be sure to consider the environmental factors (occlusion layers, target depths, object densities, and the estimation of target location) of the application and how they might influence the performance of the technique. If the environment constantly changes, as a rule of thumb, use LassoGrid+ as it was shown to be relatively robust in terms of performance.

11 Demonstrations

Based on our findings, we have developed two proof-of-concept demonstrations in VR showing the techniques in real application scenarios (see Figure 9). The first demo shows an ocean exploration scenario in VR, which belongs to the case of exploring complex 3D data visualizations. Users are immersed under the ocean and surrounded by a large
number of underwater creatures. With our techniques, they can select an animal of interest that lives in certain areas or is hidden by corals to delve into its detailed information (like name, habitat, life cycles, etc.). A similar scenario would be to explore specific locations occluded by buildings in a 3D city visualization. The second demo mimics a 3D modeling scenario. Users can acquire fully-occluded objects in the scene and perform consequent manipulations like translation and duplication. Both applications are demonstrated in the supplementary video.

12 LIMITATIONS

We have identified several limitations in our work. First, for simplicity, we simulated a user’s estimated area of where a target might occur only in a circular form. However, in a real-world scenario, multiple occurrence areas can exist, and they can be in any shape, even with some depth.

Second, we did not fine-tune the parameters of all the techniques through user studies, as it was not the primary goal of this work. For example, instead of arranging objects on a grid, other layouts are also possible (e.g., rings [10]), which could further improve the performance of the techniques.

Third, we did not include unselectable objects in the scene, as we envision a superimposed scenario that pulls out the unselectable objects for the ease of selection. However, future work might want to investigate how unselectable objects can be embedded into the scene and how various properties related to these objects (like sizes and placements) can affect the selection.

Fourth, our experiments feature more abstract tasks that enabled us to control the variables of interest precisely, however, we did not evaluate technique performance under practical scenarios. To strike a balance between internal and external validity of our findings, though two proof-of-concept demonstrations are provided, more work is necessary to understand how the techniques can perform and how we can adjust them in realistic workflows. For example, future work can explore how the techniques could be applied to disambiguate vertex or edge selection in 3D modeling applications.

13 CONCLUSION

In this paper, we explored fully-occluded target selection in virtual reality environments. Based on the existing literature on the topic, we highlighted three open challenges within this research topic in terms of problem formulation, combining occlusion visualization with selection techniques, and in-depth evaluation. To address them, we first framed a general problem-solving strategy and, according to that, devised the design space. We then designed seven potential techniques and evaluated them through a user study.

Based on the study results, we derived design implications and refined the most promising techniques. We conducted a second study to analyze how four environmental factors (occlusion layers, target depths, object densities, and the estimation of target locations) influence technique performance. Based on our findings, we offer a set of distilled recommendations for future virtual reality systems that offer fully-occluded target selection. We believe our design approaches and proposed techniques can trigger the creation of exciting user interfaces and applications related to fully-occluded selection. Future work can optimize further the techniques, as well as develop new methods for selecting fully-occluded targets in VR.

REFERENCES


