International Journal of Human-Computer Studies 120 (2018) 36–48

Contents lists available at ScienceDirect

International Journal of Human-Computer Studies

journal homepage: www.elsevier.com/locate/ijhcs

VOTE: A ray-casting study of vote-oriented technique enhancements

Alec G. Moore*, John G. Hatch, Stephen Kuehl, Ryan P. McMahan

University of Texas at Dallas, Richardson TX, USA

ARTICLE INFO

Keywords:
VOTE
Precise selection
Ray-casting

ABSTRACT

When making selections within 3D user interfaces (3DUIs), a user can fail to select a desired target despite indicating that target during most of the interaction process. This is due to numerous factors that can negatively impact which object is being indicated during the final confirmation step. In this paper, we present a novel vote-oriented technique enhancement (VOTE) for 3D selection that votes for indicated object each interaction frame and then selects the object with the most votes during confirmation. VOTE can be applied to nearly any 3D selection technique, as it does not require additional user input and does not require any prior knowledge of the environment or task. To demonstrate the effectiveness of VOTE, we present a ray-casting selection study that compared traditional, Snap-To, and VOTE ray-casting techniques for a standard multidirectional selection task. The results of our study show that VOTE afforded faster selections than traditional ray-casting and resulted in fewer incorrect selections than the Snap-To enhancement. Additionally, VOTE yielded significantly better effective throughput than traditional ray-casting and the Snap-To enhancement for selections within clustered environments.

1. Introduction

Some 3D interaction tasks, particularly selection, can require high levels of precision and are difficult for users to accomplish (LaViola et al., 2017). For instance, many users struggle to use a ray-casting technique to point at distant targets. Similarly, most users will find it difficult to grab a small object with a simple virtual hand implementation.

The difficulties involved in 3D selections requiring high precision are due to many factors. For example, using large muscle groups is almost always required for 3D selections, which causes difficulty due to the disproportionate representation of muscle groups in the brain (Zhai et al., 1996). Because small muscle groups normally have increased innervation, input devices that make use of these small groups (e.g., a joystick being controlled by the fingers) should have performance advantages over devices that are operated primarily by large muscle groups (e.g., a handheld tracked remote) (Accot and Zhai, 2001). Another closely related difficulty is hand tremor, in which physiological oscillations and movements during button clicks can result in inaccurate selections and manipulations (Benko, Wilson, and Baudisch, Precise Selection Techniques for Multi-touch Screens, 2006). This is exacerbated by the fact that typical 3D interactions do not involve any form of supporting surface, which can result in the user experiencing more muscle fatigue, and in turn, more hand tremors (Hincapié-Ramos et al., 2014).

Another source of difficulties for precise 3D selections can be the control-to-display (CD) ratio that a technique employs. While techniques that use transfer functions with a low CD ratio afford users new capabilities, such as virtually reaching objects outside of their physical reach (Poupyrev et al., 1996). In particular, CD ratios below one usually result in reduced precision in terms of user performances. Additionally, some 3D selection tasks may influence the effective CD ratio. For example, in the case of ray-casting, small wrist movements result in large positional changes at far distances, which increases the required amount of precision on the part of the user (Kopper et al., 2010). Finally, the accuracy and precision of the tracking system used can influence the effective CD ratio of a technique (McMahan et al., 2014).

To improve the precision of 3D selections, researchers have developed several technique enhancements that address some of these factors and their negative influences on user performance. We classify these enhancements into four broad categories of approaches: input filters, dynamic CD ratios, progressive refinement, and heuristics-based scoring. In our related work section, we examine different approaches in these categories, as well as some of their typical shortcomings.

In this paper, we present a new type of heuristics-based enhancement that we call VOTE (vote-oriented technique enhancement). The

* Corresponding author.
E-mail addresses: alec@utdallas.edu (A.G. Moore), john.hatch@utdallas.edu (J.G. Hatch), unstephenk@gmail.com (S. Kuehl), rymcmaha@utdallas.edu (R.P. McMahan).

https://doi.org/10.1016/j.ijhcs.2018.07.003
Received 24 May 2017; Received in revised form 6 June 2018; Accepted 4 July 2018
Available online 06 July 2018
1071-5819/

key insight that VOTE was designed after is that users often indicate their desired target while using a selection technique, but another object may be inadvertently indicated at the time of confirmation. For example, the action of pressing a button to confirm a selection can change the position of the user's hand, and ultimately, the object being indicated. The basic concept of VOTE is to vote every interaction frame for the object currently being indicated, then when the user confirms a selection, the object with the most votes is selected. Unlike other heuristics-based enhancements, VOTE does not require any knowledge of the environment, aside from what object the underlying technique is indicating. It does not require additional user input, unlike some dynamic CD ratio enhancements and progressive refinement. Additionally, it limits the negative effects of hand tremor without negating user actions, which input filters often fail to do.

To demonstrate the efficacy of VOTE, we conducted a ray-casting study that employed a multidirectional selection task like prior studies (Teather and Stuerzlinger, 2013; Teather et al., 2009). In the study, we compared a traditional (unmodified) ray-casting technique, a previously developed Snap-To (object) ray-casting enhancement, and our VOTE version of ray-casting. The results of our ray-casting study show that VOTE was significantly faster than traditional ray-casting and resulted in significantly fewer incorrect selections than Snap-To ray-casting. We also found the effective throughput of VOTE to be significantly better than Snap-To ray-casting for clustered environments, which indicates VOTE is the superior ray-casting selection enhancement.

After discussing the effectiveness and efficiency of our VOTE ray-casting technique, we present a generalized framework for designing VOTE enhancements for nearly any 3D interaction task (i.e., selection, manipulation, travel, or system control). We identify the methods and parameters of VOTE and discuss they apply to various 3D interaction techniques. We also identify several open research questions regarding the VOTE methods and parameters.

2. Related work

In this section, we first briefly discuss common 3D selection techniques and their use cases. We then describe several enhancements for 3D selection that researchers have developed and investigated in the past. We organize these enhancements according to the four broad categories discussed in the introduction: input filters, dynamic CD ratios, progressive refinement, and heuristics-based scoring. While we cover many prior enhancements in this section, this is by no means an exhaustive list. For those interested in other approaches, we recommend reading the survey of 3D object selection techniques by Argelaguet and Andujar (2013).

2.1. Common 3D selection techniques

Based on the taxonomy presented by LaViola et al. (2017), there are five broad categories of 3D selection techniques: grasping, pointing, surface, indirect, and bimanual.

Grasping metaphors emulate the natural ability to interact with a 3D world by grasping and manipulating objects with one's hands. The most common grasping metaphor is the simple virtual hand technique, which directly maps the user's physical hand motion to a virtual hand's motion. However, this approach naturally limits selection to those objects within the user's own physical reach. Some grasping techniques attempt to overcome this limitation by using non-linear mappings to extend the virtual hand further out as the physical hand extends from the user's body. The Go-Go technique employs such a non-linear mapping once the physical hand is extended beyond a predefined distance threshold (Poupyrev et al., 1996). However, non-linear grasping techniques, such as Go-Go, restrict the user's ability to easily select objects at certain distances because small movements of the physical hand can result in large jumps of the virtual one.

Pointing metaphors emulate the natural tendency to indicate objects by pointing at them, which allows objects beyond the arm's reach to be selected. LaViola et al. (2017) separated pointing techniques into either vector-based pointing or volume-based pointing. Vector-based pointing techniques rely on a single vector to determine which object the user intends to select. Ray-casting, the focus of this research, represents this vector as a virtual ray, which is usually attached to the user's virtual hand to directly control the vector. Image plane selection is another vector-based pointing technique; however, it uses a vector that originates from the user's viewpoint and extends through the virtual hand (Pierce et al., 1997). In general, vector-based pointing techniques are efficient selection techniques except when a high degree of angular precision is required, such as when selecting small objects or distant objects. Volume-based pointing techniques address this precision issue by allowing objects within a volume to be selected, as opposed to objects being intersected by a vector. For example, the flashlight technique imitates pointing at objects with a flashlight by using a cone attached to the user's virtual hand, instead of a virtual ray (Liang and Green, 1994). However, volume-based pointing techniques require some method of disambiguating which object should be selected when multiple objects are present in the volume (LaViola et al., 2017). In the case of the flashlight technique, the object closest to the centerline of the cone is selected first.

Surface-based techniques use multi-touch surfaces for selection by controlling the depth of virtual objects relative to the surface (LaViola et al., 2017). For example, the balloon selection technique employs the metaphor of controlling the height of a balloon (the selection cursor) by using the non-dominant hand to pull the bottom of its string from the dominant hand holding it (Benko and Feiner, 2007). Similarly, the triangle cursor uses the distance between two touch points to define the vertical position of its cursor while the midpoint of the touch points defines the cursor's horizontal position (Strothoff et al., 2011). The precision of a surface-based selection technique is dependent upon the size of its 3D cursor. A small cursor requires precise positioning of the touch points that control it. On the other hand, a large cursor requires less precision, but can make it difficult to disambiguate which object to select when the cursor is touching multiple objects.

Indirect and bimanual selection metaphors involve many of the same concepts and issues as grasping and pointing. Indirect control-space techniques use a multi-touch surface to control a cursor on the primary display and image plane selection to select objects under the cursor (LaViola et al., 2017). Indirect proxy techniques provide local representations of objects that can be used to remotely select and manipulate those objects (McMahan et al., 2014). For example, the world-in-miniature technique uses a grasping metaphor for selection (Stoakley et al., 1995) while the voodoo dolls technique uses image plane selection to initially create proxies (Pierce et al., 1999). Similarly, bimanual selection techniques often employ a grasping metaphor or a pointing metaphor for selecting objects. The Spindle technique uses bimanual input to define a virtual spindle that serves for grasping (Mapes and Moshell, 1995) while the iSith technique uses two virtual rays to define an interaction point at the closest midpoint between the rays (Wyss et al., 2006).

2.2. Filtered input enhancements

Our first category of selection enhancements is based around the concept of using filters to improve the precision of input. One of the simplest filters used for human input is the low-pass filter. Low-pass filters work by attenuating high frequency noise while allowing low frequency input to pass through relatively unmodified (Plonus, 2001). Several variations of low-pass filters have been used in efforts to improve user inputs. Some low-pass filters also utilize a variable cutoff frequency based on the speed of input, such as the dynamic recursive filter used by Vogel and Balakrishnan (2005) and the “1€ filter” created

by Casiez et al. (2012). This approach allows for lower jitter with higher latency at low speeds and high jitter with low latency at higher speeds.

Applying a Kalman filter to input is another approach to enhancing selections. Kalman filters work by providing an efficient solution of the least squares method by essentially estimating a prediction of values for a certain time, obtaining feedback in the form of a measurement, and then using that information to further correct its future predictions each update (Welch and Bishop, 1995). Kalman filters are often used in systems with noisy input, such as optical hand tracking and gesture recognition (Suarez and Murphy, 2012) and using inertial measurement units for full-body motion tracking (Yun and Bachmann, 2006), and can provide good estimates despite the inherent noise. Kalman filters can also be used to improve natural hand jitter. Matveyev et al. (2003) used a high-latency, low-jitter Kalman filter to smooth the input used for selection calculations while using the low-latency, natural-jitter input for visual feedback.

One issue with using input filters is optimizing the thresholds applied to the input. If the thresholds are too strict, motions intended by the user can be negated. For example, if the user intentionaly makes high-frequency movements, a low-pass filter may negate those actions. Alternatively, if the thresholds are too lenient, they will permit more jitter and hand tremor, which are likely to negatively affect selection. Furthermore, it is questionable whether perfect jitter compensation would even allow for sufficient precision in cases like pointing, given the limitations of the human motor system (König et al., 2009). As demonstrated in our study, VOTE ameliorates the limitations of the human motor system with regard to tremor without negating any user actions.

2.3. Dynamic control-display ratio enhancements

One approach to dynamically modifying the CD ratio of a technique is to have the user explicitly specify an interaction mode, with each mode employing its own CD ratio. An example of this approach is the RayToRelative pointing technique developed by Vogel and Balakrishnan (2005). With this technique, the user can use an absolute pointing mode with a one-to-one CD ratio by pointing with a semi-clinched fist at the display screen. When more precision is necessary, the user can then use an open-hand posture to switch to a relative pointing mode that uses a large CD ratio. In their study, Vogel and Balakrishnan (2005) found that the RayToRelative technique afforded significantly fewer errors than traditional ray-casting. A similar technique is the Absolute and Relative Mapping (ARM) ray-casting technique (Kopper et al., 2010), in which the user activates a precise interaction mode by pressing a button with the non-dominant hand.

Another approach to dynamically modifying the CD ratio is to implicitly base it on the velocity of the user's movements. One of the first techniques to take this approach was the precise and rapid interaction through scaled manipulation (PRISM) virtual hand technique developed by Frees and Kessler (2005). PRISM was designed around the tendency that users will often slow their movements when attempting to be precise. Given this insight, PRISM employs a scaled motion mode when the user's physical hand velocity is less than a scaling-constant threshold. In this mode, the CD ratio dynamically increases as the velocity of the user's hand decreases to afford greater precision. Frees and Kessler (2005) also used a recovery mode, in which the CD ratio decreased at a constant rate, until any discrepancies between the user's physical and virtual hands were eliminated. It is worth noting that a PRISM virtual hand technique inherently improves the precision of manipulating objects, in addition to improving the selection of objects. Adaptive pointing (König et al., 2009), is very similar to PRISM, except it was designed specifically for ray-casting, instead of simple virtual hand.

The main limitation of dynamic CD ratio techniques is that they require additional user input to change the CD ratio. The RayToRelative and ARM ray-casting techniques require explicit user input in the form of an open-hand posture or a button pressed by the non-dominant hand, respectively. The PRISM and Adaptive pointing techniques implicitly rely on the velocity of the user's hand to slow when the user is attempting to be precise. Unlike these dynamic CD ratios, VOTE does not require additional user input, explicitly or implicitly.

2.4. Progressive refinement techniques

Our third category of enhancement techniques use the concept of selection by progressive refinement, which is the process of continually reducing the set of selectable objects until only the desired target remains (Kopper et al., 2011). One of the first techniques to use progressive refinement was the sphere-casting refined by QUAD-menu (SQUAD) technique developed by Kopper et al. (2011). With SQUAD, the user first selects a group of potential targets using a sphere-casting technique like ray-casting, with the exception that a spherical volume is cast instead of a virtual ray. A quad menu is then used to display the set of potential targets in four quadrants. Each time the user selects a quadrant, the quad menu is repopulated with that quadrant's set of potential targets, until the quadrant selected contains only one object. In an empirical study, Kopper et al. (2011) found that SQUAD was significantly faster than traditional ray-casting when selecting small targets; however, ray-casting was significantly faster when selecting large targets. They also found that using SQUAD resulted in nearly no errors, while ray-casting induced significantly more errors.

Another progressive refinement technique is the Expand method developed by Cashion et al. (2012). The Expand technique is very similar to SQUAD, except it displays the set of potential targets in a virtual grid within the current view, as opposed to a quad menu in a separate context. Additionally, with the Expand technique, the user can cause the objects to transition from their original positions to the virtual grid and back by moving the handheld controller away from or closer to the display screen, respectively. Cashion et al. (2012) compared Expand to traditional ray-casting, a zoom-based variant, and SQUAD for a series of selection tasks. They found that Expand was significantly faster than the zoom variant and SQUAD, and that Expand resulted in significantly better precision (i.e., fewer errors) than all the other techniques. Cashion et al. (2013) also compared Expand to Bendcast (which is described in the following section) and found that Expand resulted in significantly fewer errors but slower completion times.

The Starfish selection technique developed by Wonner et al. (2012) is another progressive refinement technique. In Starfish, users control a 3D pointer surrounded by a starfish-shaped closed surface, where each extremity ends on near targets. The Starfish technique has two modes: a move mode, wherein the “head” of the starfish is translated with the tracked hand, and a select mode, in which the shape and position of the starfish are locked and the user can move the pointer within the Starfish volume down an arm to select the object at the end of that arm. Wonner et al. (2012) haven't yet conducted a formal comparison study between Starfish and SQUAD, but preliminary results showed that Starfish's selection time and error rate were promising. Unlike SQUAD and Expand, Starfish doesn't need to overlay the potential objects over the scene for the refinement step, but it still requires the positioning of a selection volume, followed by refinement to complete selection.

The major limitation of progressive refinement techniques is that they generally require multiple selections to eventually select one target. In many cases, this results in slower selection times despite improved precision (Kopper et al., 2011; Cashion et al., 2012; Cashion et al., 2013). Additionally, many of these techniques require screen real estate to display either a quad menu or a virtual grid over the current view of the virtual environment. Hence, they may not be appropriate to use for some 3DUIs, such as virtual reality (VR) applications. Unlike progressive refinement enhancements, VOTE does not require additional user input in the form of multiple selections and does not require any screen real estate.

2.5. Heuristics-based scoring enhancements

The final category of enhancements that we will discuss is based on the concept of using heuristics to assign scores to potential targets and then selecting the target with the best score during confirmation. One of the simplest examples of a heuristic-style approach for ray-casting is to assign scores based on the angle offset from the original ray, within some angular threshold, and then point the ray being cast to the object with the smallest angle offset score. The Snap-To ray-casting technique developed by Wingrave et al. (2002) was one of the first enhancements of this style. Cashion et al. (2013) developed the Bendcast technique, which behaves just as the Snap-To technique, except it uses an onscreen cursor instead of a ray for feedback. They compared Bendcast to their Expand technique and found that Bendcast was significantly faster but resulted in more errors. To the best of our knowledge, neither Snap-To ray-casting nor Bendcast had been formally compared to traditional ray-casting, prior to our research.

Another simple heuristic-based scoring technique is the Bubble Cursor, originally developed by Grossman and Balakrishnan (2005), which dynamically resizes the radius of the selection cursor to always touch the closest target. Vanacken et al. (2007) developed a 3D version of the Bubble Cursor, in which the radius of a semitransparent sphere would automatically enlarge to enclose the nearest object. If the sphere would intersect multiple objects to completely enclose the nearest one, a second semitransparent sphere would be displayed to enclose the nearest object instead. Vanacken et al. (2007) compared the 3D Bubble Cursor to a traditional simple virtual hand implementation and found that the Bubble Cursor afforded faster selection times.

For a more-complex example of a heuristics-based scoring enhancement, the IntenSelect technique, developed by de Haan, Koutek, and Post (2005), leverages the insight that users will generally point in the direction of their target for a short period of time before confirming the selection. IntenSelect uses an object-scoring function and a conic volume to assign scores to objects based on their distances from the casted ray. Objects close to the ray are assigned higher scores while objects near the edges of the conic volume are assigned lower scores. These scores are accumulated over multiple interaction frames and when the user confirms a selection, the object with the greatest score is chosen. In an informal study, de Haan et al. (2005) found that IntenSelect was faster than traditional ray-casting.

Another heuristics-based scoring technique is the Hook method developed by Ortega (2013) for selecting 3D moving objects in a dense environment. Hook was designed based on the insight that if a target is moving in an unpredictable but non-Brownian motion, the user cannot anticipate future positions and will be forced to chase the target. To account for this, Hook scores every possible target based on its distance to the cursor and, like IntenSelect, accumulates these scores over multiple interaction frames before selecting the target with the highest score when the user makes a confirmation. To avoid system inertia, only a limited number of the closest targets will have their scores increased, while all other scores will be decreased. However, this number of closest targets must be determined based on the total number of targets, the surrounding spherical volume of the cursor, and the volume that the targets are moving in. Ortega (2013) compared Hook to traditional ray-casting and the 3D Bubble Cursor. He found that Hook was significantly faster and resulted in significantly fewer errors than the other two techniques for selecting moving targets in dense environments.

Another heuristics-based scoring technique is the Intent-Driven Selection (IDS) virtual hand method developed by Periverzov and Ilieş (2015). The IDS technique was specifically designed for finger-based grasping with the insight to use the posture of the virtual fingers as an indication of the user's level of confidence in selecting a particular object. The IDS technique uses a proximity sphere defined by the curvature of the fingers to define a subset of targets and to assign scores to possible targets. The technique then employs a complex scoring function that accounts for the strength of intent, the movement of the user's hand, and several other factors to determine which object had the greatest score and should be selected upon a confirmation. Periverzov and Ilieş (2015) compared the IDS technique to a traditional finger-based virtual hand technique and found that IDS significantly improved selection time in the cases of selecting large occluded objects and small objects during tracking jitter.

A major limitation of these prior heuristics-based scoring techniques is that their underlying scoring functions require knowledge of the environment. For example, all the techniques mentioned, from Snap-To ray-casting to IDS, require knowledge of all possible targets within the environment to effectively score them and function. VOTE, on the other hand, only requires knowledge of what the underlying technique is indicating each frame. Additionally, VOTE is a simple enhancement that relies on a simple queue of votes to determine which object had the most votes during the underlying technique's confirmation step.

3. VOTE for ray-casting

In this section, we describe our implementation of VOTE for ray-casting. First, we discuss how traditional ray-casting is commonly implemented for 3DUIs. Then, we explain how VOTE utilizes the calculations of the traditional technique to build a queue of votes, which is then used to select the object with the most votes during the confirmation step.

3.1. Traditional ray-casting

Traditional ray-casting usually relies on two methods, one for indicating objects and one for confirming selections. The indication method is often implemented by ray-casting a vector into the 3D environment and determining which virtual object is intersected first. With popular games engines like Unity and Unreal, this is easily accomplished with engine functions that only consider game objects with predefined physics-related attributes (e.g., colliders). With more-polished implementations of ray-casting, the length of the virtual ray is then adjusted to end at the intersected object to provide better visual feedback.

When called, the confirmation method returns the currently intersected object as the selected object. In most implementations, the confirmation method is called when a specific device button is pressed or clicked. However, other input events can be used as well. For some applications, selecting an object may result in the object being attached to the virtual ray for manipulation. In other applications, selecting an object may execute some functionality, such as calling a menu command or displaying a visual effect.

It is important to note that the indication and confirmation methods for traditional ray-casting are usually processed every interaction frame, in that order. For game engines like Unity and Unreal, these interaction frames are equivalent to graphics frames, as game developers are recommended to process user input each frame update, before rendering occurs. Hence, like all interaction techniques processing user input on frame updates, user performance will suffer from latencies in rendering with most implementations of traditional ray-casting.

3.2. VOTE ray-casting

As discussed in the introduction, VOTE was designed around the fact that users often indicate their desired target while using a selection technique, but other objects may be inadvertently selected during the confirmation step. To leverage this, VOTE maintains a queue of votes and uses a popular vote to select the object with the most votes.

Every interaction frame, VOTE adds a timestamped vote to the queue for the object being indicated by the ray-casting technique's indication method. If an object is not indicated by the indication method (i.e., no objects are currently being intersected by the virtual ray), a

timestamped “abstain” vote is added to the queue instead. When the confirmation method is called, VOTE first expires any votes older than a predefined threshold. For the presented study, we used a threshold of 0.2 s, which we chose after some preliminary pilot testing of our VOTE ray-casting technique (see Section 5.5 for a detailed discussion of the expiration parameter). Once old votes are expired, VOTE then selects and returns the object with the most votes, as opposed to the currently intersected object.

This approach allows VOTE to address many of the precision issues discussed before. For example, consider the situation in which the user is pointing at the desired object and then points at the wrong object during confirmation due to pressing the button. With traditional ray-casting, the wrong object would be selected. However, with VOTE, the desired object would have more votes than the wrong one and would be selected instead. Similarly, consider the scenario in which the user passes over the desired target and confirms while not pointing at any objects. With traditional ray-casting, nothing would be selected, but again, with VOTE, the desired object would have the most votes and would be selected.

4. Ray-casting selection study

To determine whether VOTE can improve the precision of 3D selection techniques, we decided to investigate a ray-casting selection task. We chose ray-casting because it is a common, straightforward selection technique that would allow us to easily demonstrate the effects of VOTE when dealing with distant objects and dense virtual environments, such as clusters of objects. In addition to comparing VOTE to traditional ray-casting, we decided to also include the Snap-To technique (Wingrave et al., 2002). Like VOTE, the Snap-To technique is a heuristics-based scoring enhancement. Additionally, to the best of our knowledge, neither it nor the very similar Bendcast technique (Cashion et al., 2012) had been formally and empirically compared to traditional ray-casting.

4.1. Selection task

For the selection task, we used a multidirectional selection task similar to the one described by Soukoreff and MacKenzie (2004) and used by Teather et al. in prior selection studies (Teather and Stuerzlinger, 2013; Teather et al., 2009). The task required the user to select the blue target from a ring of black targets. The blue target would alternate sides of the ring while continuing clockwise (seeFig. 2). To mimic making selections in both dense and sparse virtual environments, we used clustered and non-clustered versions of the multidirectional tapping task. For the clustered version, the ring consisted of 12 targets with close spacing between adjacent targets, and the user was required to only iterate through all the targets once, for a total of 12 selections. For the non-clustered version, the ring consisted of 6 targets with moderate spacing between adjacent targets, and the user was required to iterate through all the targets twice, for a total of 12 selections (see Fig. 3). In both versions of the task, the diameter of the ring was 1 m, and the diameter of the targets was 0.17 m, to accommodate the clustered version. Finally, to mimic making selections at various distances within the virtual environment, we placed the rings at distances of 5 m, 10 m, and 15 m from the user, while maintaining the 1 m diameter.

4.2. Experimental design

We conducted a within-subjects, repeated-measures experiment to evaluate the efficacy of VOTE for the ray-casting selection task. There were three independent variables. The first variable was the technique: traditional ray-casting, Snap-To ray-casting, and VOTE ray-casting. The presentation order of the technique variable was counterbalanced between subjects to negate any potential learning effects. The second variable was whether the task was clustered or non-clustered. Finally, the third variable was the distance from the user that the ring of targets was placed: 5 m, 10 m, or 15 m.

To ease participants into the task, we chose to gradually increase the task difficulty of the task conditions. For each technique, the user would start with an untimed training phase that consisted of a 5-m, non-clustered version of the selection task. After the experimenter verified that the participant understood and could complete the task, the timed portion of the study would begin. The participant would again start with the 5-m, non-clustered task condition. The selection task then progressed to the 5-m, clustered condition, followed by non-clustered and clustered versions of the 10-m task, and then finally the 15-m conditions (see Table 1). Due to the high difficulty index of the 15-m, clustered condition, we set a time limit of 30 s for completing each task condition. For every selection task trial (i.e., target), our study software kept track of the time to select the object, the number of missed selection attempts, and the number of incorrect selection attempts, as participants could make multiple misses and incorrect selections per trial.

4.3. Apparatus

To provide six-degree-of-freedom (6DOF) hand tracking, we used an infrared Vicon tracking system consisting of 16 MX cameras (see Fig. 4) to track retroreflective markers attached to a handheld controller. This allowed us to track the position and orientation of the user's hand within a 4-m by 4-m tracking area. For the handheld controller, we used a Nintendo Wii Remote (see Fig. 5B). We chose to use the Wii Remote to provide discrete button input for the ray-casting techniques via a wireless Bluetooth connection. For the button input, we used the Wii Remote's B button, which provides a trigger-like form factor.

For the visual display, we used an Oculus Rift development kit 2 (DK2) head-mounted display (HMD). The DK2 provides a 100° diagonal field of view with a display resolution of 960 × 1080 pixels per eye and a 75 Hz refresh rate. The DK2 has a goggle-like form factor with elastic straps and weighs approximately 440 g. The DK2 usually uses an internal inertial measurement unit (IMU) to provide head-orientation tracking, while an external camera unit is used to optically track infrared LEDs located on the front of the HMD to provide positional tracking updates. However, to avoid tracking interference between the infrared Vicon system tracking the Wii Remote and the Rift camera tracking the DK2′s infrared LEDs, we used the Vicon system to track the position of the DK2 by attaching retroreflective markers to it (see Fig. 5A). We still used the DK2′s internal IMU to update head orientation at the necessary 75 frames per second (fps).

To enable a tetherless experience, we used an Alienware 13 laptop placed within a mesh backpack to run our VR applications. The Alienware 13 included an Intel Core i7-5500U processor with dual core and 3.0 GHz turbo boost, 16 GB dual-channel DDR3L memory, and a 2GB NVIDIA GeForce GTX 960M graphics card. The VR applications were developed using the Unity game engine and maintained framerates of 75 fps to match the refresh rate of the DK2 display. Custom scripts were used to assess the input data provided by the Vicon tracking system, the DK2′s IMU, and the Wii Remote.

The virtual environment used as the background for the selection task was a virtual bowling alley (see Fig. 6). We chose this for the environment because it provided visual depth cues, most notably the converging lanes. Additionally, we did not consider it to be distracting as a background, due to its monochromatic walls and minimum mesh geometry. For all three techniques, visual feedback for the virtual ray included being rendered green when an object could be selected and black when an object would not be selected.

Though our apparatus permitted 6DOF head tracking, we wanted to keep users from walking around during the selection task or completing it from different viewpoints. To restrict these possible user movements, we implemented a one-meter by one-meter area in the center of the tracking space that served as the confines for the selection task. When

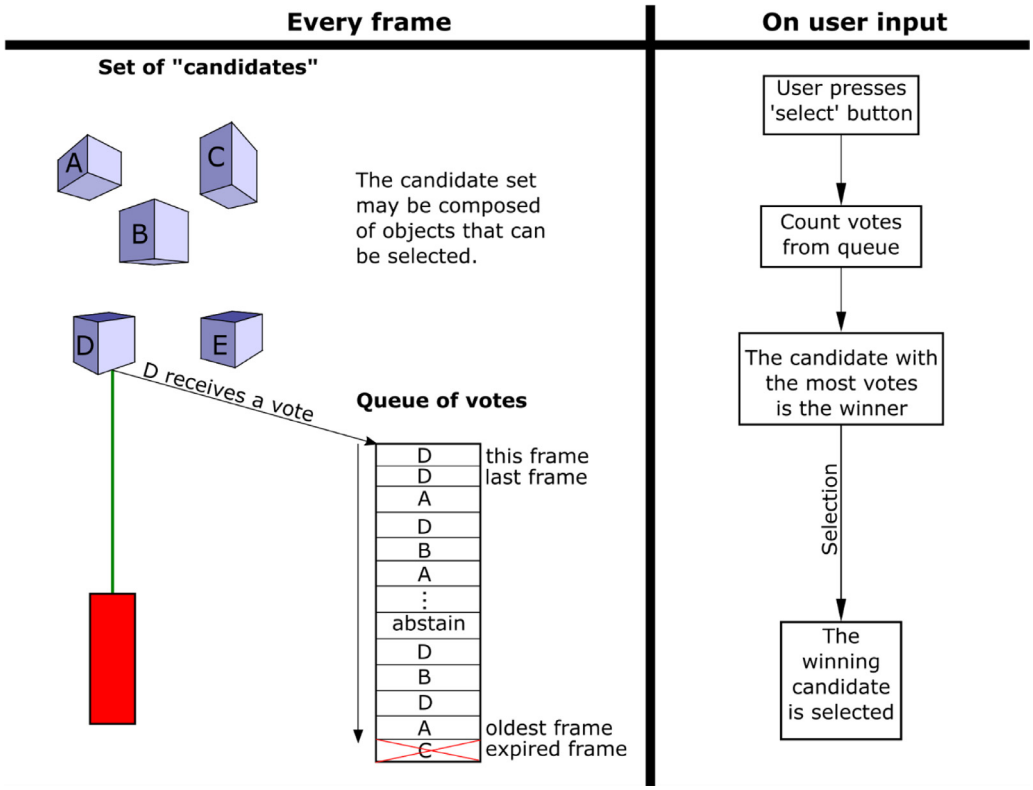

Fig. 1. A visual representation of VOTE for ray-casting.

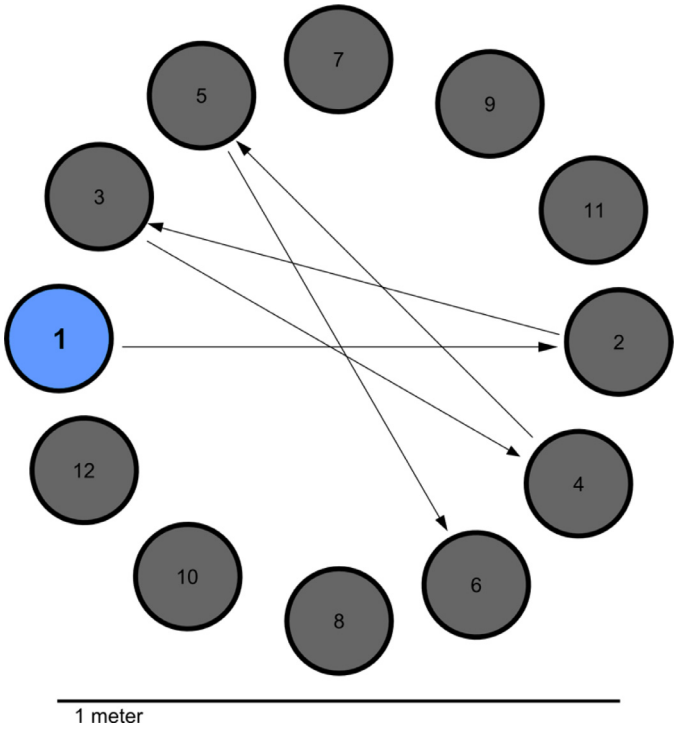

Fig. 2. The clustered version of the multidirectional selection task used for our ray-casting selection study.

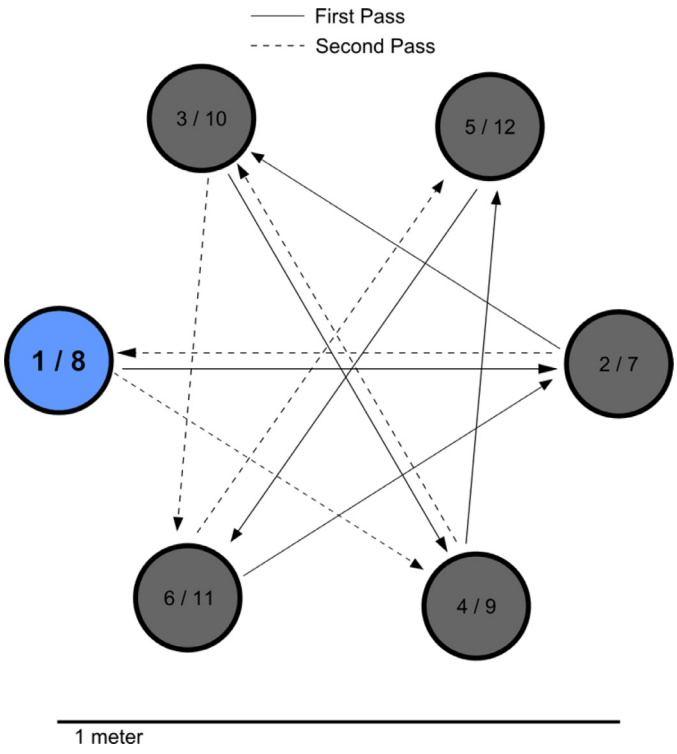

Fig. 3. The non-clustered version of the multidirectional selection task used for our ray-casting selection study.

participants stepped outside of these confines, we would pause the selection task and replace the bowling alley environment with an abstract space. The abstract space contained a green volume representing the area that the participant needed to return to and a caution tape that represented the physical boundaries of the tracking space (see Fig. 7).

During the training phase, this out-of-bounds feature was demonstrated to participants, and they were informed to return to the green area to continue the selection task. Upon returning, the task timer would start again. Note, only two participants ever ventured outside of the confines during the actual study.

Table 1
The ordering of selection task conditions per technique.

	Clustering	Distance
Task 1	Non-clustered	5 m
Task 2	Clustered	5 m
Task 3	Non-clustered	10 m
Task 4	Clustered	10 m
Task 5	Non-clustered	15 m
Task 6	Clustered	15 m

Fig. 4. The Vicon tracking system and space used for the study.

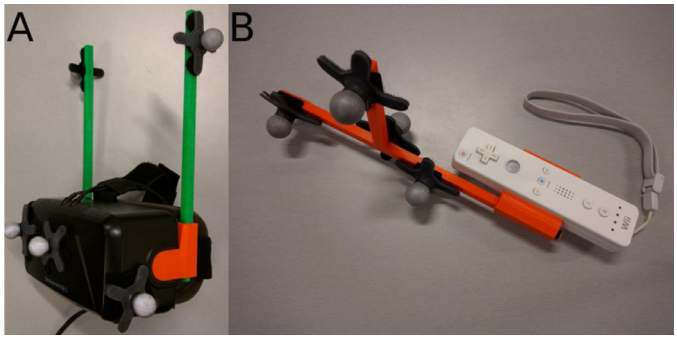

Fig. 5. The devices used for the Study: A) a Vicon-tracked HMD afforded 6DOF head tracking, and B) a Vicon-tracked handheld controller afforded 6DOF pointing.

4.4. Procedure

The following procedure was reviewed and approved by The University of Texas at Dallas institutional review board (IRB). Once recruited, the participant signed an informed consent form and filled out a background survey. After this, the experimenter would help the participant equip the mesh backpack, HMD, and Wii Remote. The experimenter would then guide the participant to the middle of the Vicon-tracked area and provide instructions on how to complete the training phase. The participant was informed that to select the highlighted object, he or she would need to point the virtual ray in the environment at the blue object and press the B button. Furthermore, the participant was informed that as a form of feedback, the ray would remain colored black, if a valid selection could not be made that frame (i.e., a miss would occur). Otherwise, the ray would turn green whenever an object could be selected. Finally, the participant would be informed to complete the selection tasks as quickly as possible.

Once trained and ready to proceed, the participant would be presented with the first technique condition, which would be determined by his or her assigned technique ordering. Again, for each technique condition, the participant was presented with six selection tasks. If the participant exceeded 30 s for any one task condition, the system would automatically move on to the next task. After completing all six task conditions, the participant was given the system usability scale (SUS) questionnaire (Brooke, 1996) to subjectively evaluate the selection technique that they just experienced. Note, participants were never informed of which technique they were experiencing and were given the same instructions for each condition. After completing the first SUS questionnaire, the participant was then presented with the second and third technique conditions, in the same manner as the first. Finally, after completing all three technique conditions and SUS questionnaires, participants were given a final exit survey and thanked for participating in our study.

4.5. Hypotheses

Given our careful design of VOTE and our ray-casting selection study, we had several hypotheses for the results of the study.

H1. VOTE will allow users to complete the selection task with fewer missed selections than traditional ray-casting. With traditional ray-casting, the user may not be pointing to an object at the time of confirmation, which will result in a missed selection. However, because VOTE selects the object with the most votes in its queue at the time of confirmation, it does not matter if the user is currently pointing at an object or not.

H2. VOTE will allow users to complete the selection task with fewer incorrect selections than traditional ray-casting. Using the same logic as the previous hypothesis, we hypothesized that traditional ray-casting may result in more errors due to hand jitter causing the ray to point to another object at confirmation. On the other hand, VOTE will still select the object with the most votes in its queue.

H3. VOTE will allow users to complete the selection task faster than traditional ray-casting. Building on the previous hypotheses, we expected that users would be able to complete the selection task more quickly with VOTE due to the increased precision that it offers over traditional ray-casting.

H4. VOTE will be perceived as more usable for the selection task than traditional ray-casting. Given our expectations that VOTE would reduce misses, reduce errors, and improve completion times, we hypothesized that users would perceive it as more usable than traditional ray-casting.

H5. Snap-To ray-casting will allow users to complete the selection task with the fewest missed selections, compared to both VOTE and traditional ray-casting. Because Snap-To ray-casting snaps the virtual ray to the nearest object within a threshold, we expected that users would rarely, if ever, miss when using the technique.

H6. Snap-To ray-casting will result in the greatest number of incorrect selections, compared to both VOTE and traditional ray-casting. Based on the prior research conducted by Cashion et al. (2013), which indicated that Bendcast produced significantly more errors than the Expand technique, we predicted that Snap-To ray-casting would also result in more errors because Snap-To ray-casting is similar to Bendcast.

H7. VOTE and Snap-To ray-casting will allow users to complete the selection task in similar completion times. Because both VOTE and Snap-To are designed as enhancements, we expected the two techniques to yield similar selection times.

H8. VOTE will provide greater effective throughput than both Snap-To and traditional ray-casting. Considering all the previous hypotheses regarding missed selections, incorrect selections, and completion times, we expected that VOTE would yield the greatest

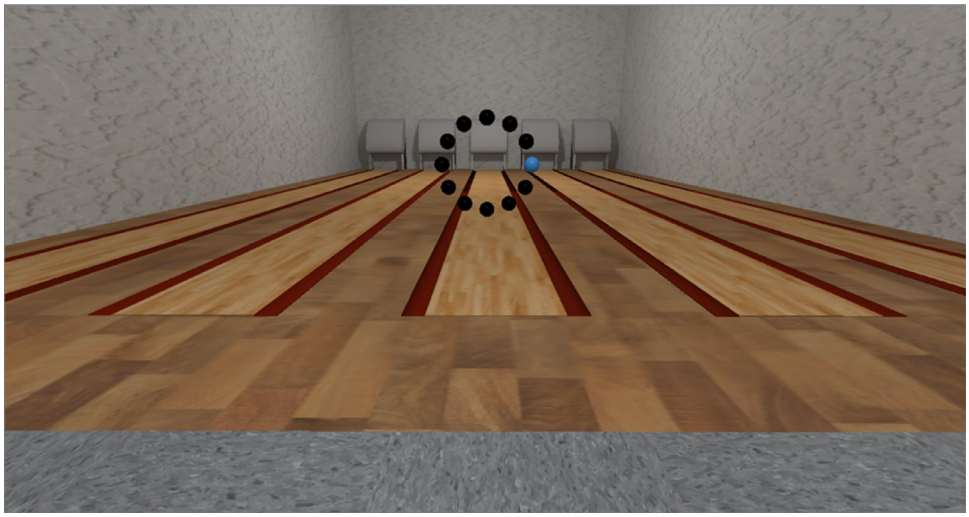

Fig. 6. The bowling alley environment used for the study.

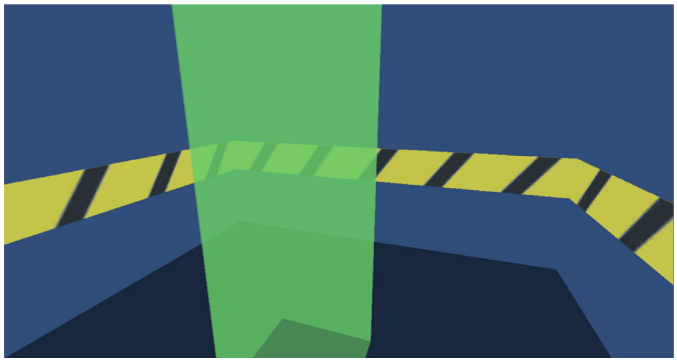

Fig. 7. The abstract space used to keep participants confined to the same approximate viewpoint (indicated by the green volume).

effective throughput for the selection task.

4.6. Participants

We recruited 24 unpaid participants (22 male, 2 female) through university mailing lists for this study. Their mean age was 20.9 years, within a range of 18 to 27 years. All the participants were right-handed except for one participant. Based on self-reported background data, 18 of the participants played video games on a regular basis (i.e., at least one hour per week). Only 9 of the participants had a previous VR experience. Each participant was assigned to one of six ordering cohorts to counterbalance the presentation order of the three techniques between subjects.

4.7. Results

For each metric, we conducted a three-way (technique, clustering, distance) repeated-measures analysis of variance (ANOVA) at a 95% confidence level. Degrees of freedom were corrected using Greenhouse-Geisser estimates of sphericity when Mauchly's test of sphericity indicated that the assumption of sphericity had been violated. Tukey's post hoc tests were used to determine which pairs of conditions were significantly different.

4.7.1. Selection time

We found a significant three-way interaction effect for technique, clustering, and distance on individual selection times, $F(4, 92) = 3.007$, $p = 0.022$ (see Fig. 8). For non-clustered tasks, traditional ray-casting was significantly slower than VOTE, and both traditional ray-casting and VOTE were significantly slower than Snap-To. For clustered tasks, traditional ray-casting was significantly slower than VOTE at all distances, and significantly slower than Snap-To at 10 m and 15 m. There were no significant differences in selection times between VOTE and Snap-To for clustered tasks.

4.7.2. Incorrect selections

We did not find a significant three-way interaction for technique, clustering, and distance on the percentage of incorrect selections per target, $F(2.540, 58.427) = 1.224$, $p = 0.307$. We did find a significant two-way interaction between technique and clustering, $F(1.135, 26.097) = 40.388$, $p < 0.001$ (see Fig. 9). For clustered tasks, Snap-To resulted in significantly more incorrect selections than VOTE, which induced significantly more incorrect selections than traditional ray-casting. For non-clustered tasks, Snap-To yielded significantly more incorrect selections than traditional ray-casting and VOTE, and there was not a significant difference between traditional ray-casting and VOTE. We did not find a significant interaction effect between technique and distance, $F(1.932, 44.443) = 1.720$, $p = 0.192$.

4.7.3. Missed selections

We did not find a significant three-way interaction for technique, clustering, and distance on the percentage of missed selections per target, $F(2.533, 58.261) = 0.221$, $p = 0.850$. We also did not find a significant interaction between technique and clustering on the percentage of misses, $F(1.458, 33.533) = 2.094$, $p = 0.150$. However, we did find a significant interaction between technique and distance on the percentage of missed selections, $F(2.363, 54.358) = 19.865$, $p < 0.001$ (see Fig. 10). For each distance, traditional ray-casting resulted in significantly more missed selections than VOTE, which resulted in significantly more missed selections than Snap-To.

4.7.4. Effective throughput

In addition to the direct performance statistics, we also computed the effective throughput of the ray-casting techniques, a value that incorporates both speed and accuracy. To calculate effective throughput, we used the following equation for each technique, as presented by Teather and Stuerzlinger (2013).

$$TP_e = \frac{\log_2\left(\frac{D_e}{W_e} + 1\right)}{MT} = \frac{ID_e}{MT} \quad (1)$$

In Eq. 1, D_e is the average of the measured movement distance with the given technique. W_e, which represents the effective width of a target

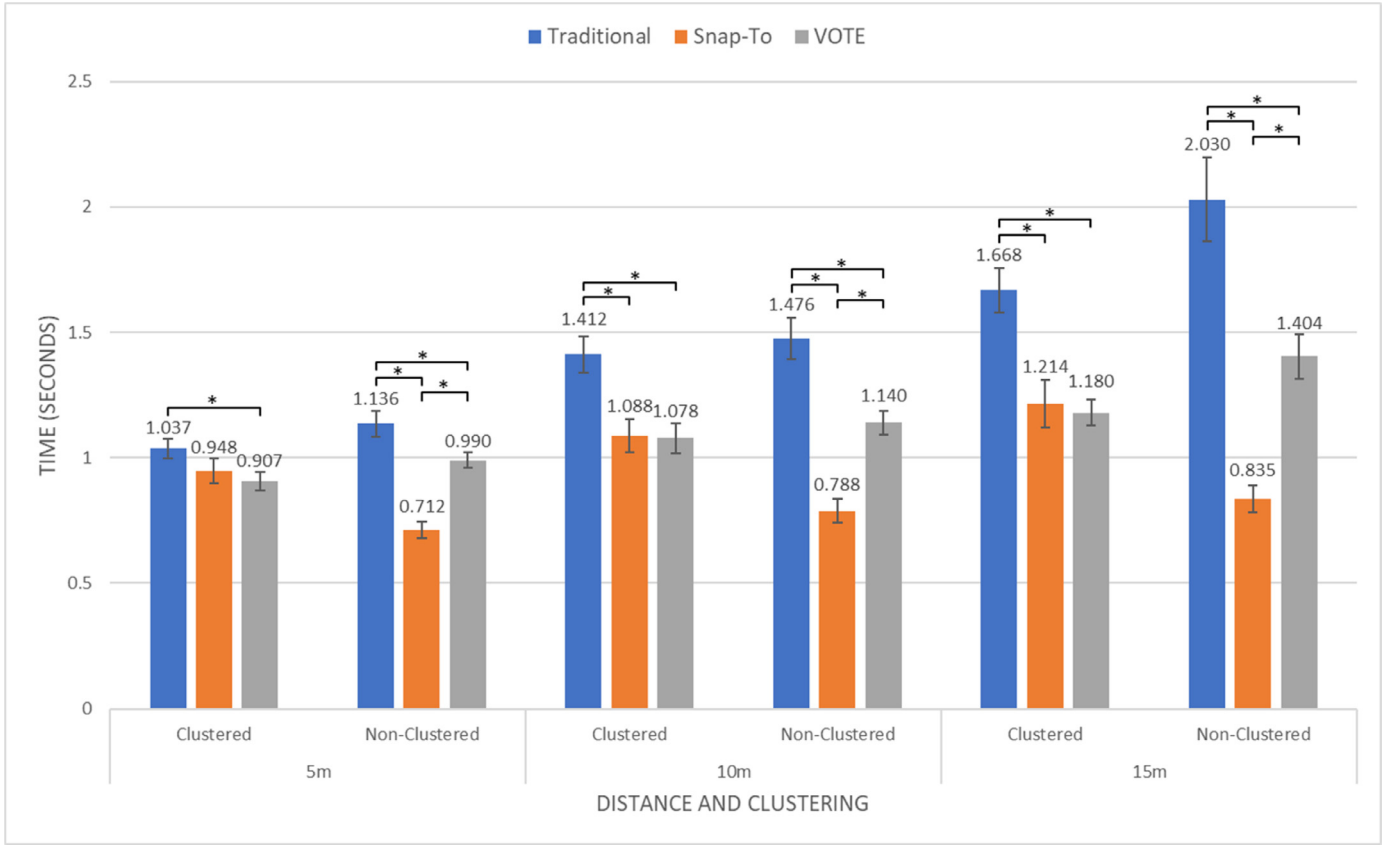

Fig. 8. Mean selection times (with standard error bars) by technique, distance and clustering state. Asterisks represent pairwise significant differences.

object, is determined by projecting the closest point of the virtual ray onto the task axis, finding the standard deviation of the projection's distance from the center of the target, and multiplying by 4.1333. *MT* is the movement time. The logarithmic term, also known as the index of difficulty (ID_e), represents the overall difficulty of the task. Because this equation was designed for 2D tasks, we had to determine the intercept of the virtual ray with the target plane at 5, 10, or 15 m, depending on the current task, and used this intercept as the cursor. Using this formula, we calculated the effective throughput of each participant for all three techniques in all six clustering-distance task conditions.

We did not find a significant three-way interaction for technique, clustering, and distance on the effective throughput of the techniques, $F(3.131, 62.614) = 1.190$, $p = 0.322$. We did find a significant two-way interaction between technique and clustering, $F(1.823, 36.468) = 33.513$, $p < 0.001$ (see Fig. 11). For clustered tasks, Snap-To resulted in significantly lower effective throughput than traditional ray-casting, which was significantly less effective than VOTE. For non-clustered tasks, there was not a significant difference between traditional ray-casting and the two enhancements. Finally, we did not find a significant interaction effect between technique and distance, $F(3.058, 61.154) = 2.129$, $p = 0.105$.

In Table 2, we provide the means of the ID_e and TP_e values for each technique. It is worth noting that the index of difficulty for Snap-To is lower than the other two techniques because it increases the activation area of targets, which results in a higher effective width. As reported by Wingrave et al. (2002), increasing the width of targets can cause users to decrease their accuracy. Additionally, Argelaguet and Andujar (2013) note that a decrease in the index of difficulty can in turn

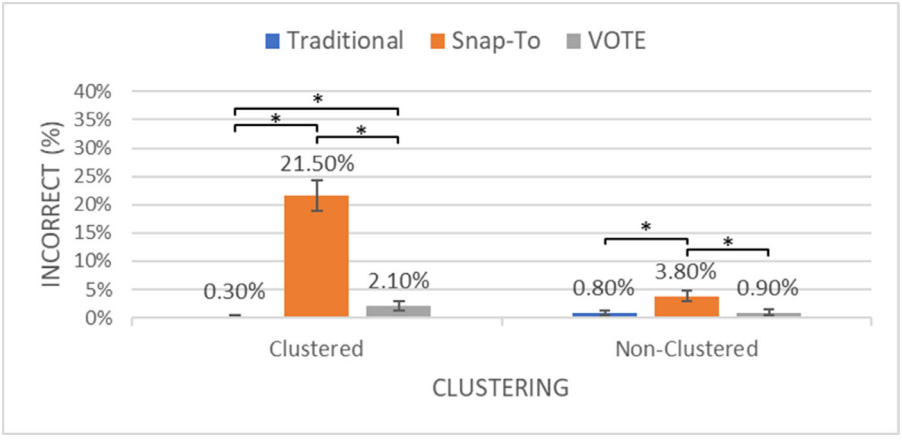

Fig. 9. Mean percentage of incorrect selections (with standard error bars) by technique and clustering state. Asterisks represent pairwise significant differences.

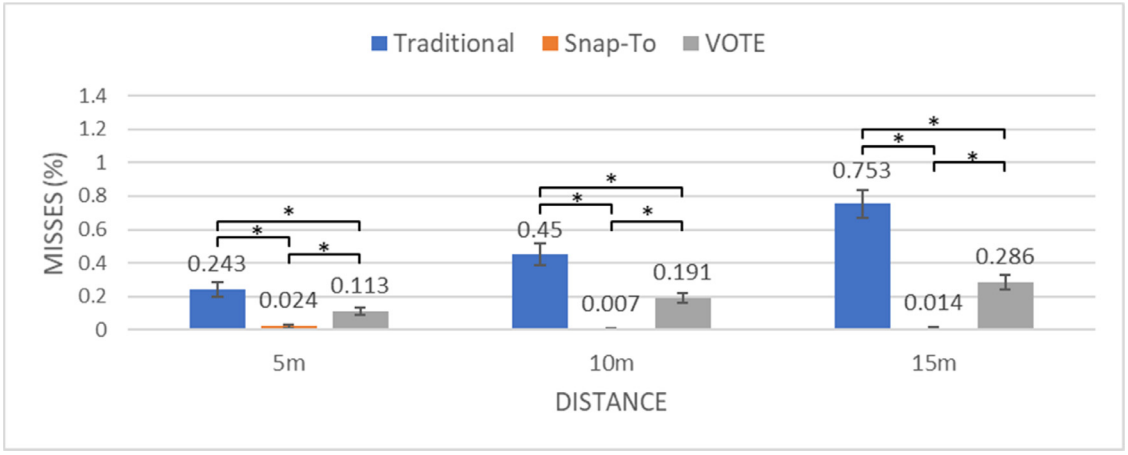

Fig. 10. Mean percentage of missed selections (with standard error bars) by technique and distance. Asterisks represent pairwise significant differences.

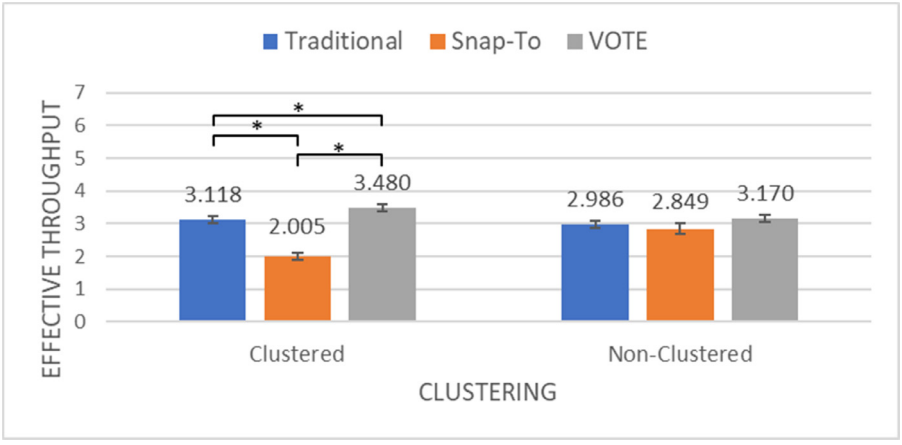

Fig. 11. Mean effective throughput (with standard error bars) by technique and clustering state. Asterisks represent pairwise significant differences.

Table 2
Means of effective index of difficulty and effective throughput values by technique.

Technique	ID_e mean	TP_e mean
Traditional	4.222	3.029
Snap-To	2.072	2.339
VOTE	3.585	3.298

lead to a decrease of throughput.

4.7.5. Perceived usability

We did not find a significant main effect of technique on the SUS usability scores, $F(1.374, 31.591) = 2.329$, $p = 0.129$. However, VOTE ($M = 83.646$, SD = 2.041) did receive higher usability scores than traditional ray-casting ($M = 75.938$, SD = 2.712) and Snap-To ray-casting ($M = 81.042$, SD = 3.583) on average.

4.8. Discussion of results

In this section, we discuss the results of our ray-casting selection study.

4.8.1. VOTE affords fast interactions

Based on the results of our ray-casting selection study, we have concluded that VOTE affords faster selections. In terms of speed, VOTE allowed users to complete the tasks significantly faster than traditional ray-casting, which confirmed our H3 hypothesis. Additionally, VOTE allowed users to make selections with significantly fewer misses than traditional ray-casting, which confirmed H1. However, traditional ray-casting resulted in significantly fewer incorrect selections than VOTE, which disproved H2. When we compared the effective throughput of the two techniques, which balances speed and accuracy, we found a significant difference between VOTE and traditional ray-casting for clustered tasks, which partially supports H8.

Considering these results together, it is clear that VOTE allowed users to make selections at faster speeds by reducing the chances of missed selections. However, due to an increased likelihood of missing, participants tended to use the traditional ray-casting much more slowly, which resulted in fewer incorrect selections. Therefore, we can confidently claim that VOTE should be used instead of traditional ray-casting when completion times and speed are important, or the task involves clustered selections.

4.8.2. VOTE likely provides performance versatility

We should point out that traditional ray-casting is not necessarily better than VOTE when accuracy is needed. Though the results of our study indicate that traditional ray-casting yielded significantly fewer incorrect selections, the result is likely due to participants attempting to use VOTE as quickly as possible, as we instructed for all three techniques (see Section 4.4), but using traditional ray-casting slower due to the increased probability of a miss. Had we changed the task and informed participants to be as accurate as possible, we believe that VOTE would not have yielded significantly more incorrect selections than traditional ray-casting, especially considering that the effective throughput of VOTE was better than traditional ray-casting for

clustered tasks and the two techniques were not statistically different for non-clustered tasks. Obviously, this needs to be confirmed in follow-up studies, but we are confident that VOTE is a versatile technique enhancement that can allow for accurate selections when needed and has already been demonstrated to afford fast selection times. Furthermore, the results of the SUS questionnaires indicate that users preferred VOTE to traditional ray-casting, though the results were not significant, and therefore, neither prove nor disprove H4.

4.8.3. VOTE is better than Snap-To

In our ray-casting selection study, we found that Snap-To ray-casting afforded significantly faster selection times than VOTE for non-clustered tasks, which disproved H7. It also yielded significantly fewer missed selections than VOTE, which affirmed H5. However, it also resulted in significantly more incorrect selections, proving H6. But, most importantly, Snap-To ray-casting afforded a significantly lower effective throughput than VOTE for clustered tasks. These results indicate that VOTE does a better job balancing the requirements of speed and accuracy, while Snap-To ray-casting prioritizes speed. For most applications, some degree of accuracy is needed, else users will have to take additional actions to undo incorrect interactions and mistakes. Considering this, it appears that VOTE is the better enhancement compared to the Snap-To technique.

5. A framework for designing VOTEs

In this section, we present a framework for designing vote-oriented technique enhancements for a broad range of 3D interactions beyond just 3D selection tasks. We discuss the methods and parameters required to develop a VOTE. This discussion raises several open research questions, including how VOTE can be applied to various types of 3D interaction (i.e., selection, manipulation, travel, and system control).

5.1. Indication method

VOTE requires an indication method from the underlying 3D interaction technique to cast votes. An indication method should return one or more candidates for the VOTE to add timestamped votes to its queue. If no candidate is returned, the VOTE should add a timestamped “abstain” instead.

For 3D selection techniques, indication methods should be readily available. For example, as described in Section 3, traditional ray-casting implements its indication method by ray-casting a vector into the 3D environment and determining which virtual object is intersected first. Other pointing-based selection techniques implement similar methods. Grasping-based selection techniques, such as simple virtual hand, also implement indication methods, usually based on the virtual hand colliding with other objects.

For 3D manipulation techniques, underlying indication methods that indicate candidate positions and orientations are usually not explicitly available. Manipulation techniques that employ constraints, such as snapping positions to a grid (LaViola et al., 2017), are some of the few techniques that provide such explicit indication methods. However, nearly every 3D manipulation technique dynamically updates the 6DOF position and orientation of its target each interaction frame. These continuous values could be used as candidates for six independent elections, especially if the values were truncated to some predefined level of precision to increase the likelihood of casting multiple votes for the same candidate in any election. However, whether this approach would be effective or not is an open research question.

For some 3D travel techniques, explicit indication methods are readily available. For example, selection-based travel techniques employ the same types of indication methods that 3D selection techniques do (LaViola et al., 2017). For these techniques, implementing VOTE is no more difficult than for their related 3D selection techniques. However, for many 3D travel techniques, appropriate indication methods are not clearly available. Consider steering, which allows the user to continuously control the direction of motion (LaViola et al., 2017). While the direction of motion is clearly indicated every interaction frame, it is likely not a useful type of input data to conduct an election on, as the user will likely update the direction every frame based on the previous frame's motion (i.e., the user is making new decisions every frame). However, identifying appropriate indication methods for 3D travel techniques is a potential research direction.

For some 3D system control techniques, such as graphical menus, indication methods are normally provided by the selection technique used to interact with them. Occasionally this is as straightforward as ray-casting or virtual hand on a 2D graphical menu. However, for other types of system control, such as voice commands and gestural commands, identifying appropriate indication methods is an open research question, as these system control techniques often indicate and confirm commands at the same time (LaViola et al., 2017).

5.2. Vote weighting method

Though not required, VOTE can employ a vote weighting method when adding votes to its queue. In our VOTE ray-casting implementation, we used an implicit one-to-one weighting method (i.e., each frame, one indicated object received exactly one vote). However, there are numerous possibilities and conditions that could be used for weighting votes.

For example, with ray-casting, we can compute the instantaneous angular velocity of the handheld device, subtract it from a predefined maximum angular velocity, and multiply any votes cast by the result. With this weighting method, the values of votes fractionally decrease as the angular velocity of the user's hand increases. With slower input movements, votes will have greater fractional weights and are more likely to influence which object is selected. This weighting method is similar to the previous PRISM (Frees and Kessler, 2005) and Adaptive pointing (König et al., 2009) techniques, in that it attempts to leverage the insight that users move more slowly when trying to be precise in their 3D interactions.

Other vote weighting methods can be used to weight votes according to the difficulty of the indication task. For example, when using a pointing metaphor to select objects, we can compute the visual angle of an object, subtract it from a predefined expected visual angle, and multiply any votes cast by the result. This weighting method results in objects with small visual angles, such as small or distant objects, receiving votes of greater value than objects with large visual angles. In turn, it should be easier to select objects that are normally difficult to select.

The methods presented above are only a small set of the possible vote weighting methods that VOTE could employ. There are many open research questions regarding weighting methods. What weighting method is the best for grasping-based selection techniques? What weighting method is the best for pointing-based selection techniques? What weighting method is the best for 3D manipulation techniques? What weighting method is the best for 3D travel techniques? These are just a few of the questions possible for investigation.

5.3. Confirmation method

Like the indication method, VOTE requires a confirmation method from the underlying 3D interaction technique. A confirmation method indicates that VOTE should expire old votes and conduct an election using its queue of votes to determine the selected candidate. Without a confirmation method, VOTE does not know when to make a final selection.

Many 3D interaction techniques employ some sort of a confirmation method. For 3D selection techniques, the confirmation is usually explicit, such as pressing a device button to make a selection. Similarly, most 3D manipulation techniques use an explicit confirmation method,

such as releasing a device button to release an object at its final position and orientation. On the other hand, for many 3D travel techniques, a confirmation method is not readily available. Selection-based travel techniques provide explicit confirmation methods, like they provide explicit indication methods. However, many 3D travel techniques employ continuous motion, such as walking metaphors and steering metaphors. Finally, most 3D system control techniques afford some type of confirmation method, from confirming the selection of a graphical menu element to simultaneously issuing and confirming a gestural command.

5.4. Election method

VOTE must employ its own election method on its queue of votes to make a final selection. For this research, we employed a plurality election method for our VOTE ray-casting implementation, in which the object with the highest number of votes, but not necessarily the majority, was selected. A majoritarian election method could be employed, in which a candidate must receive the majority of the votes (i.e., more than 50% of the votes cast, including abstentions) to be selected. Another possibility is to use a multi-winner weighting method, in which each candidate with a predefined number of votes (less than the majority) is selected. This method is relevant for multiple-object selection techniques (LaViola et al., 2017).

In addition to determining how a final selection is made from the queue of votes, the election method can also be used to modify the votes beforehand. For example, one such modification is to reweight votes based on their age, specifically to apply greater weights to newer votes. This gives priority to the most recent user input while deprioritizing older input. This election method could potentially serve as an alternative to using a larger or smaller expiration parameter for the queue of votes, which is discussed in the next section. Aside from reweighting votes according to their age, election methods could be used in numerous ways to modify votes before determining a final selection.

Like the vote weighting method, there are many open research questions regarding the election method: Do plurality methods or majoritarian methods result in the better user performance? How should multi-winner methods be implemented for multiple-object selection techniques? Does reweighting votes according to their age improve user performance? What are other effective methods for modifying votes before making a selection?

5.5. Expiration parameter

Finally, VOTE requires an expiration threshold to prevent input inertia (i.e., accumulating votes) from negatively impacting the current confirmation step. As mentioned in Section 3.2, during preliminary pilot testing of our VOTE ray-casting technique, we started with an expiration threshold of 0.5 s. However, we quickly found that this threshold would result in incorrect selections for some quick participants, due to the old target retaining more votes in the queue than the new target. We eventually settled on a threshold of 0.2 s for our formal study. Given our results, that threshold appears appropriate for our ray-casting selection task. However, determining an optimal expiration threshold is an open research question. Additionally, it is likely that the optimal threshold will vary among the different types of 3D interaction tasks, and possibly between different types of techniques for the same task (e.g., pointing metaphor vs. grasping metaphor).

6. Conclusions and future work

In this paper, we have presented VOTE, a new and simple approach to enhancing the usability of 3D selection techniques. VOTE is designed around the insight that users often indicate their desired target when using a selection technique, but due to many factors, an undesired target may be chosen when confirming the selection. VOTE addresses this issue by voting each interaction frame for the currently indicated object and then selecting the object with the most votes during confirmation, regardless of whether it is currently being indicated or not.

To demonstrate the effectiveness of VOTE, we conducted a ray-casting selection study, comparing traditional ray-casting, Snap-To ray-casting, and VOTE ray-casting. The results of that study indicate that VOTE affords faster selections than traditional ray-casting and likely has the versatility to also provide accurate selections when needed. The study also indicates that VOTE is a better enhancement for ray-casting than the Snap-To technique enhancement, considering incorrect selections and effective throughput.

Finally, we presented a framework for designing VOTEs for 3D interaction techniques other than ray-casting, include alternative metaphors for 3D selection, 3D manipulation, 3D travel, and 3D system control techniques. This framework is based on an indication method, a vote weighting method, a confirmation method, an election method, and an expiration threshold. In describing this framework, we raised several research questions for how VOTE can be used most effectively to improve 3D interactions.

For future work, we are currently focused on exploring the versatility of VOTE by applying it to the simple virtual hand technique for 3D selections. We have yet to complete the study, but informal pilot testing indicates that VOTE will also improve virtual hand selections. Hence, we are confident that VOTE will prove to be a simple, versatile, and effective technique enhancement. We are also planning to investigate the efficacy of VOTE for selections in environments with moving targets, similar to the environments used to evaluate the Hook technique (Ortega 2013).

Conflict of interest

There are no conflicts of interest.

References

Accot, J., Zhai, S., 2001. Scale effects in steering law tasks. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1–8.
Argelaguet, F., Andujar, C., 2013. A survey of 3D object selection techniques for virtual environments. Comput. Graph. 37, 121–136.
Benko, H., Feiner, S., 2007. Balloon selection: a multi-finger technique for accurate low-fatigue 3d selection. In: Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI), pp. 79–86.
Benko, H., Wilson, A.D., Baudisch, P., 2006. Precise selection techniques for multi-touch screens. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 1263–1272.
Brooke, J., 1996. SUS: a 'quick and dirty' usability scale. In: Jordan, P.W., Thomas, B., Weerdmeester, B.A., McClelland, I.L. (Eds.), Usability Evaluation In Industry 189. Taylor & Francis Ltd, London, United Kingdom, pp. 189–194.
Cashion, J., Wingrave, C., LaViola, J.J., 2012. Dense and dynamic 3D selection for game-based virtual environments. IEEE Trans. Vis. Comput. Graph. 18, 634–642.
Cashion, J., Wingrave, C., LaViola, J.J., 2013. Optimal 3D selection technique assignment using real-time contextual analysis. In: Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI), pp. 107–110.
Casiez, G., Roussel, N., Vogel, D., 2012. 1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2527–2530.
de Haan, G., Koutek, M., Post, F.H., 2005. IntenSelect: using dynamic object rating for assisting 3d object selection. In: Proceedings of the 11th Eurographics Conference on Virtual Environments, pp. 201–209.
Frees, S., Kessler, G.D., 2005. Precise and rapid interaction through scaled manipulation in immersive virtual environments. In: Proceedings of IEEE Virtual Reality, 2005, pp. 99–106.
Grossman, T., Balakrishnan, R., 2005. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 281–290.
Hincapié-Ramos, J.D., Guo, X., Moghadasian, P., Irani, P., 2014. Consumed endurance: a metric to quantify arm fatigue of mid-air interactions. In: Proceedings of the 32nd annual ACM Conference on Human Factors in Computing Systems, pp. 1063–1072.
König, W.A., Gerken, J., Dierdorf, S., Reiterer, H., 2009. Adaptive pointing - design and evaluation of a precision enhancing technique for absolute pointing devices. In: Proceedings of the IFIP Conference on Human-Computer Interaction, pp. 658–671.
Kopper, R., Bacim, F., Bowman, D.A., 2011. Rapid and accurate 3D selection by progressive refinement. In: Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI), pp. 67–74.
Kopper, R., Bowman, D.A., Silva, M.G., McMahan, R.P., 2010. A human motor behavior

model for distal pointing tasks. Int. J. Hum. Comput. Stud. 68, 603–615.
LaViola, J.J., Kruijff, E., McMahan, R.P., Bowman, D.A., Poupyrev, I., 2017. 3D User Interfaces: Theory and Practice, second ed. Addison-Wesley, Boston.
Liang, J., Green, M., 1994. JDCAD: a highly interactive 3D modeling system. Comput Graph 18 (4), 499–506.
Mapes, D.P., Moshell, J.M., 1995. A two-handed interface for object manipulation in virtual environments. Presence 4 (4), 403–416.
Matveyev, S., Göbel, M., Frolov, P., 2003. Laser pointer interaction with hand tremor elimination. In: Proceedings of HCI International, pp. 376–740.
McMahan, R.P., Kopper, R., Bowman, D.A., 2014. Principles for designing effective 3D interaction techniques. Handbook of Virtual Environments: Design, Implementation, and Applications, second ed. CRC Press, pp. 285–311.
Ortega, M., 2013. Hook: heuristics for selecting 3D moving objects in dense target environments. In: Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI), pp. 119–122.
Periverzov, F., Ilieş, H., 2015. IDS: the intent driven selection method for natural user interfaces. In: Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI), pp. 121–128.
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. In: Proceedings of the ACM Symposium on Interactive 3D Graphics (I3D '97), pp. 39–44.
Pierce, J.S., Stearns, B.C., Pausch, R., 1999. Voodoo dolls: seamless interaction at multiple scales in virtual environments. In: Proceedings of the ACM Symposium on Interactive 3D Graphics (I3D '99), pp. 141–145.
Plonus, M., 2001. Electronics and Communications for Scientists and Engineers. Academic Press.
Poupyrev, I., Billinghurst, M., Weghorst, S., Ichikawa, T., 1996. The go-go interaction technique: non-linear mapping for direct manipulation in VR. In: Proceedings of the 9th annual ACM symposium on User interface software and technology, pp. 79–80.
Soukoreff, R.W., MacKenzie, I.S., 2004. Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI. Int. J. Hum. Comput. Stud. 61, 751–789.
Stoakley, R., Conway, M.J., Pausch, R., 1995. Virtual reality on a WIM: interactive worlds in miniature. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 265–272.
Strothoff, S., Valkov, D., Hinrichs, K., 2011. Triangle cursor: interactions with objects above the tabletop. In: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, pp. 111–119.
Suarez, J., Murphy, R.R., 2012. Hand gesture recognition with depth images: a review. In: Proceedings of the 21st IEEE International Symposium on Robot and Human Interactive Communication (2012 IEEE RO-MAN), pp. 411–417.
Teather, R.J., Stuerzlinger, W., 2013. Pointing at 3d target projections with one-eyed and stereo cursors. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 159–168.
Teather, R.J., Pavlovych, A., Stuerzlinger, W., MacKenzie, I.S., 2009. Effects of tracking technology, latency, and spatial jitter on object movement. In: Proceedings of the 2009 IEEE Symposium on 3D User Interfaces (3DUI), pp. 43–50.
Vanacken, L., Grossman, T., Coninx, K., 2007. Exploring the effects of environment density and target visibility on object selection in 3D virtual environments. In: Proceedings of the 2007 IEEE Symposium on 3D User Interfaces (3DUI), pp. 117–124.
Vogel, D., Balakrishnan, R., 2005. Distant freehand pointing and clicking on very large, high resolution displays. In: Proceedings of the 18th annual ACM symposium on User interface software and technology, pp. 33–42.
Welch, G., & Bishop, G. (1995). An introduction to the Kalman filter, http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html.
Wingrave, C.A., Bowman, D.A., Ramakrishnan, N., 2002. Towards preferences in virtual environment interfaces. In: Proceedings of the Eurographics Symposium on Virtual Environments (EGVE). 2. pp. 63–72.
Wonner, J., Grosjean, J., Capobianco, A., Bechmann, D., 2012. Starfish: a selection technique for dense virtual environments. In: Proceedings of the 18th ACM symposium on Virtual reality software and technology, pp. 101–104.
Wyss, H.P., Blach, R., Bues, M., 2006. iSith - intersection-based spatial interaction for two hands. In: Proceedings of the IEEE Symposium on 3D User Interfaces (3DUI), pp. 59–61.
Yun, X., Bachmann, E.R., 2006. Design, implementation, and experimental results of a quaternion-based Kalman filter for human body motion tracking. IEEE Trans. Robot. 22, 1216–1227.
Zhai, S., Milgram, P., Buxton, W., 1996. The influence of muscle groups on performance of multiple degree-of-freedom input. In: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 308–315.