RayCursor: a 3D Pointing Facilitation Technique based on Raycasting

Marc Baloup, Thomas Pietrzak, Géry Casiez

RayCursor: a 3D Pointing Facilitation Technique based on Raycasting

Marc Baloup
Univ. Lille,
Inria, Lille, France
marc.baloup@inria.fr

Thomas Pietrzak
Univ. Lille,
UMR 9189 - CRIStAL, Lille, France
thomas.pietrzak@univ-lille.fr

Géry Casiez
Univ. Lille,
UMR 9189 - CRIStAL, Lille, France
gery.casiez@univ-lille.fr

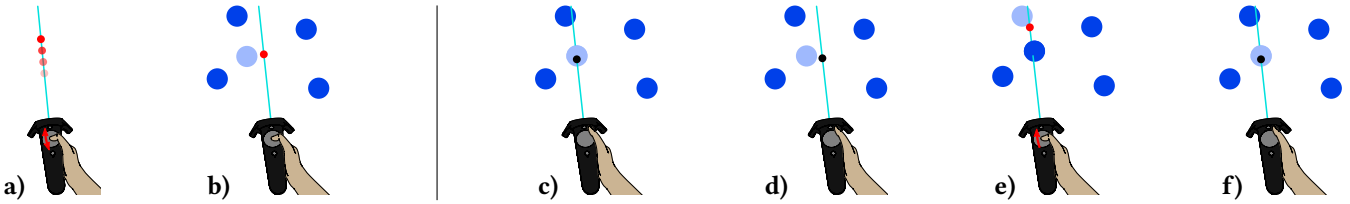

Figure 1: Illustration of manual *RayCursor*: a) the user controls a cursor along the ray using relative displacements of their thumb on the controller's touchpad; b) the target closest to the cursor is highlighted. Illustration of semi-auto *RayCursor*: c) by default, it works like *Raycasting*. The cursor (in black) is positioned at the intersection with a target; d) the target remains selected if the cursor moves out of the target, until it is closer to another target; e) the user can manually move the cursor using the controller's touchpad, to select another target (the cursor turns red to indicate manual mode); f) if the user does not touch the touchpad for 1s, the cursor returns to its behaviour described in c).

ABSTRACT

Raycasting is the most common target pointing technique in virtual reality environments. However, performance on small and distant targets is impacted by the accuracy of the pointing device and the user's motor skills. Current pointing facilitation techniques are currently only applied in the context of the virtual hand, *i.e.* for targets within reach. We propose enhancements to *Raycasting*: filtering the ray, and adding a controllable cursor on the ray to select the nearest target. We describe a series of studies for the design of the visual feedforward, filtering technique, as well as a comparative study between different 3D pointing techniques. Our results show that highlighting the nearest target is one of the most efficient visual feedforward technique. We also show that filtering the ray reduces error rate in a drastic way. Finally we show the benefits of *RayCursor* compared to *Raycasting* and another technique from the literature.

CHI 2019, May 4–9, 2019, Glasgow, Scotland Uk

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
https://doi.org/10.1145/3290605.3300331

CCS CONCEPTS

• **Human-centered computing** → **Virtual reality**; **Pointing**; *Gestural input*;

KEYWORDS

Virtual reality, pointing technique, visual feedforward

1 INTRODUCTION

Pointing is a fundamental task in any interactive system featuring a 2D or 3D interaction space. Numerous techniques have been proposed and refined over decades for 2D environments to exploit the degrees of freedom offered by new input devices (*e.g.* computer mouse, touch interfaces or eye trackers) or the characteristics of the objects displayed, trying to artificially reduce the distance to objects [13, 15] or increasing their width [3], for example. 3D environments also provide multiple techniques to accomplish this task. These techniques can be divided into two main categories: virtual hands and techniques based on *Raycasting* [4, 23]. Virtual hand techniques provide an isomorphic mapping between the real hand and the virtual one [1]. Despite many technique intended to improve *Raycasting* [6, 10, 14, 20, 22, 25–28], standard *Raycasting* and virtual hand remain the two default techniques available with devices such as the HTC Vive and their programming environment.

With a *Raycasting* technique, the user manipulates a ray whose origin and orientation are defined by those of a 6 degrees of freedom input device, in a way similar to the

manipulation of a laser pointer. When the ray intersects several objects, the one closest to the user can be selected. This technique allows to select targets with a difficulty increasing with longer distances and smaller object widths, due to limits in motion tracking and human motor capabilities. This technique is also affected by occlusion and distracting targets as only the closest intersecting target can be selected, requiring to change the ray position and orientation to select occluded targets.

Many techniques have been designed and refined to overcome these limitations. New input devices and interaction contexts offer new opportunities to improve pointing techniques. New trackers, as available on the HTC Vive, offer additional degrees of freedom such as trackpads that were not available before.

We propose an improved *Raycasting* called *RayCursor*[1], which uses a cursor on the ray. This technique has been designed primarily for immersive environments using the HTC Vive but could also be used in other 3D contexts featuring a 6 degrees of freedom (DOF) controller comprising a touchpad and buttons. The user can control the cursor along the ray using relative displacements of their thumb on the controller's touchpad. Similarly to the *Bubble Cursor* [13, 29], the target closest to the cursor can be selected when the user presses a button.

After presenting the related work on 3D selection techniques based on *Raycasting* and pointing facilitation techniques, we then describe the design of *RayCursor* and its characteristics: visual feedforward, cursor transfer function and ray filtering. We explain the general setup of our user studies, and a series of experiments investigating each of the characteristics of *RayCursor*. Finally, we detail a user evaluation, comparing *RayCursor* to standard *Raycasting* and the closest technique in the literature [26].

2 RELATED WORK

Our related work section focuses on selection techniques building on *Raycasting*. We start by presenting the different disambiguation mechanisms that have been proposed for these techniques. We then present techniques adding extra degrees of freedom to the ray before covering pointing facilitation techniques using target proximity.

Disambiguation techniques

Argelaguet and Andujar provide a taxonomy of the different techniques designed to improve *Raycasting* [1]. Most of these techniques are based on the use of a volume instead of a ray, requiring the use of disambiguation techniques for selection. They distinguish three groups of disambiguation mechanisms: manual, heuristic and behavioral.

The manual approach requires additional steps to manually select a target among those highlighted. For example, in the Flower Ray, Grossman *et al.* display in a pie menu the objects intersected by the ray [14]. The menu cone technique also displays the targets to disambiguate in a menu and the user performs a gesture to select the target of interest [25]. In a similar way the SQUAD technique proposed by Kopper *et al.* adapts the *Raycasting* to cast a sphere onto the nearest intersecting surface to determine which objects are available for selection [20]. The selectable objects are then distributed among four quadrants and the user refines the selection until the desired object can be selected. SQUAD showed significantly better performance compared to *Raycasting* for small target sizes and low densities but there was a significant performance degradation with large target sizes and high densities, due to the increased number of steps to select a target. Cashion *et al.* propose a variation of SQUAD, called Expand, adding the ability to zoom [6]. They show Expand performs faster than SQUAD for high object densities. Without relying on menus, the Depth Ray [14] uses a depth marker attached at a fixed position on the ray. When the rays intersects multiple objects, the one which is closest to the depth marker can be selected. The user can then adjust the position of the depth ray by changing the position and orientation of his hand. Grossman and Balakrishnan also proposed the Lock Ray technique that consists in locking all intersected objects before selecting one using the depth marker [14], building on the concept of cycling through the set selected objects [19]. However this did not improve movement time. Using a smartphone to select objects in the physical world, Delamare *et al.* proposed two techniques to disambiguate targets selected in a cone [11]. With P2Roll the user performs a rolling gesture to select the target of interest and a sliding gesture on the touch surface using P2Slide. Their techniques were only evaluated with up to 16 targets.

The heuristic approach applies some heuristics to determine the target the user is willing to select. The Flashlight technique, for example, highlights the object that is the closest of the central axis of the selection cone [22]. The Skicky-Ray is based on the *Raycasting* technique [28], and the last object intersected remains selectable until another one is hit. The virtual ray is bended towards the objects that can be selected, loosing some visual feedback to select another object. This technique has not been evaluated. Schimdt *et al.* proposed different pointing-based probabilistic selection algorithms to infer the target the user wants to select but it requires complex tuning of weighting schemes depending on the application [27].

[1] Additional material available at ns.inria.fr/loki/raycursor

Last, the behavioral approaches consider user's actions before the selection confirmation to determine the object to be selected. For example, IntenSelect uses a time-based scoring function to calculate the score of objects inside a conic selection volume and the object with the highest score can be selected [10]. In a similar way the Smart Ray continuously weights targets based on their proximity to the ray cursor [14]. However the latter shows lower performance compared to techniques such as the Flower Ray or the Depth Ray [14].

In summary, current interaction techniques that improve *Raycasting* require a disambiguation mechanism that needs extra steps to make a selection. When evaluated, these techniques show better performance for the selection of small objects in dense environments but they also show lower performance compared to *Raycasting* for the selection of large targets. Grossman and Balakrishnan showed that the Depth Ray gets better performance compared to the Lock Ray, Flower Ray or Smart Ray because of the lower time required for the disambiguation phase [14]. Instead of having to disambiguate between different targets using several steps, another approach is to add extra degrees of freedom to the *Raycasting* to help adjusting the target selection while manipulating the ray.

Adding extra degrees of freedom

Grossman and Balakrishnan simply added an extra degree of freedom by adding a fixed cursor at the middle of a ray [14], which proved to be the most efficient technique to select targets in a small volumetric display. However in the context of immersive environments such as VR headsets, the use of this technique would require significant displacements of the user to disambiguate between targets. Instead of having a cursor fixed on a ray, Ro *et al.* introduced the ability to adjust the depth of the ray using relative displacements of a finger on a smartphone touchscreen [26]. However their technique was not compared to other techniques and the transfer function used to control the length of the ray is not detailed. Recent studies combine hand movements with head and eye-tracking for pointing in augmented reality, such as Pinpointing [21]. This technique is much more precise than gaze-only-based techniques. However it shares disadvantages with *Raycasting*: sensibility to occlusion, hand tremor and input precision.

Pointing facilitation techniques

Various strategies were studied to facilitate pointing. For instance, semantic pointing expands targets in the motor space [3]. This technique was designed for 2D pointing, but another study extended it to 3D using a computer mouse on standard monitor [12]. This technique improves pointing performance in sparse environments but is affected by intervening targets on the way of the cursor (distractors) [8]. An alternative strategy is to replace pointing by symbolic gestures [18]. It alleviates issues due to pointing gestures, but it is not adapted to arbitrary targets. At the opposite, studies propose 3D gestures for pointing targets on 2D displays but these results are hardly applicable in the context of 3D targets selection [24, 32].

One of the most efficient pointing facilitation technique existing in 2D is to select the target closest to the cursor. For example, the Bubble Cursor displays a disk (a bubble) centered on a mouse cursor whose radius is adjusted with the distance to the closest target [13]. This technique is especially efficient when the density of targets is low, whatever the size of the targets. The main drawback remains the visual feedback introduced by the bubble constantly changing its radius. Guillon *et al.* have evaluated the impact of several visual feedbacks on performance for the Bubble Cursor and found that a simple highlight of the closest target is efficient [17].

Vanacken *et al.* developed a 3D version of the Bubble Cursor, called 3D Bubble, using a virtual hand technique to control a 3D pointer [29]. Their technique displays a 3D semi-transparent sphere enclosing the closest target. They show that the Depth Ray is more efficient than the 3D Bubble, that is more efficient than the *Raycasting*. Similarly, Vickers defined a sensitive cube around a 3D cursor manipulated by a wand [30]. When an object was found within the sensitive cube, the cursor jumped to the object.

In summary, disambiguation techniques appear efficient to select small targets in dense environments, but they overall increase movement time due to the extra steps they introduce. Selection techniques using target proximity appear efficient but they require the use of an appropriate feedback, especially in 3D. Inspired by the 3D Bubble, the Depth Ray and the adjustable length ray introduced by Ro *et al.*, we propose the *RayCursor*, a technique combining several of the benefits these techniques offer, without introducing extra disambiguation steps many techniques require.

3 RAYCURSOR

We describe the design of *RayCursor*, an improvement of *Raycasting*. First we add a cursor on the ray, that the user can manipulate. Then we add a strategy consisting in selecting the closest target to the cursor, similarly to the Bubble Cursor [13, 29]. We discuss variations of visual feedbacks, inspired by Guillon's work for 2D [17]. Then, we describe alternative transfer functions for the control of the cursor. Finally we explain filtering techniques we used on the ray to reduce tremor effects, with the *1€ Filter* [9].

Adding a cursor on the ray

The idea of a controllable cursor was introduced by Ro *et al.* to select targets in augmented reality environments using a mobile phone [26]. However this technique was not designed to facilitate the selection of small targets. Instead of using a mobile phone, the user performs forward-backward displacements on the touchpad of a Vive Controller located under their thumb. It would be possible to implement this technique on any other 6 DOF controller having at least an extra degree of freedom such as a wheel.

Gain function for cursor control

The transfer function, which computes the cursor movements, is an essential aspect of the interaction technique. Indeed, previous research showed that the transfer function influences pointing performance [7]. We consider two variables for the design of the transfer function:

- v_{pad} the speed of the contact point on the pad in m/s: this is a usual input for non-linear transfer functions.
- d_{cur} the distance between the hand and the cursor in m: because of depth, closer objects seem to be moving faster than farther objects. We make the hypothesis that it influences movement time.

The speed of the cursor is thus $v_{cur} = g(v_{pad}, d_{cur}) \times v_{pad}$. We propose several transfer functions, which we evaluate in a dedicated experiment.

Gain function depending on finger speed. Common non-linear transfer functions depend on the cursor speed [7]. This is due to the ballistic and correction phases of a pointing task. During the ballistic stage, the user wants to move as fast as possible to get close to the target. In the correction phase the user wants to select as precisely as possible the target. To do so, non-linear transfer functions use a lower gain at slow speeds and higher gains at high speed. We designed this transfer function as a bounded linear interpolation:

$$VitLerp(v_{pad}) = \begin{cases} k_1 & \text{if } x \leq v_1 \\ k_2 & \text{if } x \geq v_2 \\ k_1 + \frac{k_2 - k_1}{v_2 - v_1}(v_{pad} - v_1) & \text{otherwise} \end{cases}$$

Gain function depending on cursor position. With a constant gain function, the cursor speed seems to depend on the depth position: faster when it is closer, and slower when it is farther. We designed a transfer function based on a gain proportional to the cursor distance to the hand to alleviate this problem. It is tuned to ensure a usable minimum gain for closer distances. The formula is the following, with k the proportional factor, d_{cur} the distance between the hand and the cursor, and d a constant corresponding to an estimated distance between the user's eye and hand:

$$DistDep(d_{cur}) = k \times \sqrt{d_{cur}^2 + d^2}$$

Visual feedforwards

Guillon *et al.*'s work showed an influence of visual feedforward on pointing efficiency with a *Bubble Cursor* for 2D pointing [17]. In the case of selecting the nearest target, feedforward is necessary. Vanacken *et al.*'s study about the 3D bubble only used the bubble feedback [29]. We adapt to 3D interaction the feedforwards Guillon *et al.* designed for 2D interaction. We describe below these feedforwards along two dimensions: highlighting the target, and representing the distance between the cursor and the nearest target.

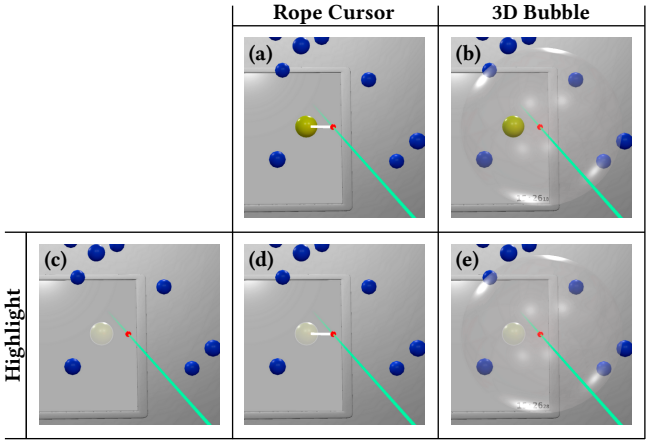

Figure 2: *RayCursor* with different visual feedbacks. The cursor is red and the ray is cyan. (c,d,e) *Highlight* : the nearest target is highlighted. (a,d) *Rope Cursor* [17] : a white ray binds the cursor and the nearest target. (b,e) *3D Bubble* [29] : a 3D bubble centered on the cursor encompassing the nearest target.

Target highlight. In our implementation, highlighting the closest target consists in applying a color lighter than the other targets (Figure 2 (c,d,e)). The advantage of this feedforward compared to the following ones is that visual clutter is minimal. We can also combine it with other visual feedforwards.

Representing target-cursor *distance.* Representing the distance between the cursor and the nearest target is likely to help determining the target that can be selected using the cursor. There are several ways to represent this distance. Drawing a semi-transparent sphere, centered on the cursor and whose radius is the distance *target-cursor*, is a 3D adaptation of the *Bubble Cursor* (Figure 2 (b,e)). Guillon *et al.* also proposed the *Rope Cursor*, which we adapt to 3D by displaying a white segment between the cursor and the nearest target (Figure 2 (a,d)). This visual feedback causes less visual clutter than the bubble. Another proposition of Guillon *et al.* is the Voronoï region of each target. This diagram represents

the region of influence for each target, taking into account the distance to each target. Although we implemented this visual feedforward as semi-transparent volumes, we discarded it because the visual clutter created makes it hard to use.

In section 5 we present a comparative study for combinations of the following visual feedbacks depicted in Figure 2: *3D Bubble*, *Rope Cursor* and *Highlight*.

Filtering the ray

An issue with *Raycasting* is the precision to select small targets. It is due both to hand tremor and noisy input. We propose to reduce jitter by filtering the ray. We apply filtering both on *Raycasting*, and on *RayCursor*, using the *1€ Filter* as it is fast, simple to tune, and offers a good trade-off between precision and latency [9]. We designed two filtering modes. In the first mode we filter the orientation of the ray but only to compute the intersection with virtual objects. The ray displayed is not filtered. We call this mode $1€_M$, as the ray is only filtered in motor space. In the second mode, we filter the orientation of the ray both in the motor and visual spaces ($1€_{VM}$). The advantage of $1€_M$ over $1€_{VM}$ is that the user experiences absolutely no extra delay but still benefits from filtering for the intersecting objects. In contrast, with $1€_{VM}$ the user has a more consistent feedback regarding the results of his actions.

We describe a comparative study of these two filter modes and an unfiltered *Raycasting* in section 5.

4 GENERAL EXPERIMENTAL SETUP

All the experiments in this paper use the same general setup as described in this section.

Apparatus

The experiment used a PC with a *HTC Vive* VR headset [31]. Participants manipulated a Vive controller in their dominant hand, and were not allowed to use the second hand. The ray was controlled using the six degrees of freedom of the controller, and the cursor was manipulated using the thumb on the touchpad (Figure 1). Target selection was done by pressing the controller's trigger using the index finger. The experiment application was coded in *C#* using *Unity 3D* and the *SteamVR* library.

Tasks

Unless stated otherwise, participants had to conform to the following instructions in all experiments. They had to stand in the middle of a 7m square room. The exact position was marked as a 70cm square on the ground. The experiment instructions were displayed on a virtual screen in front of the user (Figure 3). Participants were asked to select targets of varying positions, sizes and densities depending on the experimental conditions. All targets were at sight when the user was looking at the virtual wall with the virtual display. To allow a fair comparison of completion times and error rates between conditions, we generated, before the experiments, a single sequence of targets for each condition, used for all participants, all techniques and all blocks.

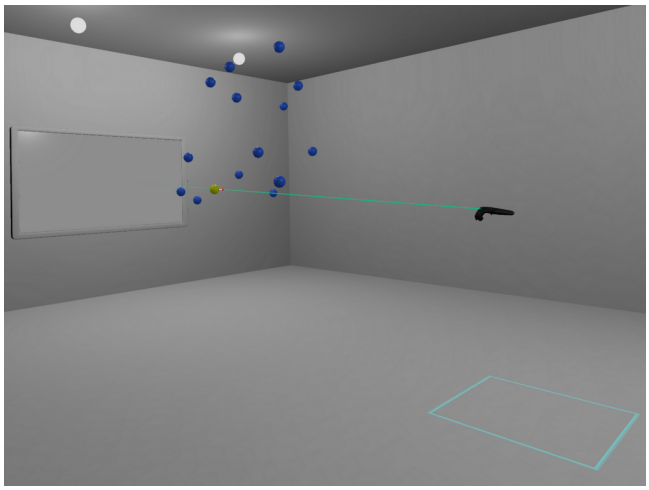

Figure 3: 3D scene used in the experiments. Participants stood in a cyan square drawn on the ground. Instructions were displayed on a virtual screen. Targets were displayed between the user and the screen.

All targets were blue except the one to be selected, which was yellow. The participant could select a target by pressing the trigger. A 10ms vibration of the controller informed the participant when the correct target was selected. When a participant pressed the trigger with no target selected (with *Raycasting*), the controller vibrated for 200ms, the correct target flashed green and the trial was marked as an error. When the participant selected a wrong target, the controller vibrated for 200ms, the selected target flashed red, the correct target flashed green and the trial was marked as an error. Participants could not move to the next target before correctly selecting the current one. The error rate was computed as the ratio between trials for which the correct target was not selected first and the total number of trials. Participants were instructed to remain around 4% error rate as a way to balance their speed/accuracy trade-off. The error rate was displayed on the virtual screen during breaks.

5 *RAYCURSOR* CHARACTERISTICS STUDIES

We present a series of experiments investigating each of the characteristics of *RayCursor*. We start with the study of different visual feedforwards to indicate the target to select, as it seemed to us as the characteristic most likely to affect performance. We then evaluate the different transfer functions we designed, and evaluate the effect of filtering the ray.

Visual feedforward

We proposed several visual feedforwards, based on previous work on similar techniques for 2D and 3D interaction [13, 17, 29]. Guillon *et al.*'s study in the 2D case concluded that highlighting the nearest target is the most efficient feedforward, while limiting visual clutter. We describe a similar study, comparing 3D versions of these visual feedforwards with *Raycasting* as a baseline. Considering previous work in 2D, our hypothesis is that highlighting the nearest target is the most efficient feedforward in 3D (H1). We used the *DistDep* transfer function to control the cursor and no filtering was used in any condition. We refer to [2] for more details about this study.

Methodology. Twelve participants (1 female, all right-handed, age mean=26, $\sigma = 4.3$) took part in this experiment. Two of them experienced Virtual Reality for their first time.

We used a within-subjects design, with factors: Technique, target Density, target Size and Block. The 6 techniques are *Raycasting* (*RC*) as the reference interaction technique, and the *RayCursor* with the 5 visual feedforwards, as depicted on Figure 2: *RopeCursor* (*Rope*), *Highlight* (*HL*), *3DBubble* (*Bub*), *3DBubble+Highlight* (*Bub+HL*), *RopeCursor+Highlight* (*Rope+HL*). The order of techniques was balanced between participants using a Latin square. The 2 target sizes were S_{Big} = 8cm and S_{Small} = 4cm. The 2 densities used were 15 targets (D_{Low}) and 40 targets (D_{High}). All the targets were spread out at pseudo-random positions in a 2m diameter sphere, whose center was 2m in front of the participants.

Participants were allowed to rest between blocks. The experiment design was: 12 participants × 6 Techniques × 3 Blocks × 2 Densities × 2 Sizes × 10 targets = 8,640 trials in total. The experiment lasted around 30min per participant.

Results. Our two main dependent variables are selection time and error rate.

Selection time. In this analysis, selection time refers to the time elapsed between two selections. Therefore the first trial of each sequence of 10 targets is discarded, as well as trials resulting in an error. A Box-Cox transformation with $\lambda = -1.2$ was applied to correct non-normal selection time residuals [5].

A repeated measures ANOVA[2] found a significant effect of Block ($F_{1.2,12.7} = 21.2$, $p < 0.001$, $\eta^2_G = 0.04$). Pairwise comparisons show significant differences between blocks 1 and the two following ones ($p < 0.003$, Block 1: 1.45s, 2: 1.29s, 3: 1.25s). We assume this difference is due to a learning effect, therefore we remove the first block from remaining analysis.

Subsequent analysis reveal a significant effect of Technique ($F_{5,55} = 5.4$, $p < 0.001$, $\eta^2_G = 0.1$). Pairwise comparisons show a significant effect between *Raycasting* and all the other techniques except *3DBubble* (*RC*: 1.39s, *HL*: 1.19s, *Bub*: 1.42s, *Bub+HL*: 1.22s, *Rope*: 1.23s, *Rope+HL*: 1.17s, $p < 0.027$).

The analysis shows a significant effect of Size ($F_{1,11} = 108.3$, $p < 0.001$, $\eta^2_G = 0.07$) and a Technique×Size interaction ($F_{5,55} = 28.1$, $p < 0.001$, $\eta^2_G = 0.05$). Pairwise comparisons only show significant differences between the techniques for *small* targets. *Raycasting* is significantly slower ($p < 0.0004$) than all the other techniques except *3DBubble* (*RC*: 1.63s, *Bub*: 1.48s, *HL*: 1.21s, *Rope*: 1.28s, *Rope+HL*: 1.20s).

We also observe a significant effect of Density ($F_{1,11} = 176.3$, $p < 0.001$, $\eta^2_G = 0.11$) and a Technique×Density interaction ($F_{5,55} = 5.2$, $p < 0.001$, $\eta^2_G = 0.01$). For *low* density, we observe a significant differences ($p < 0.01$) between *Raycasting* and all other techniques, except *3DBubble* (*RC*: 1.36s, *Bub*: 1.30s, *Bub+HL*: 1.13s, *HL*: 1.10s, *Rope*: 1.11s, *Rope+HL*: 1.07s).

Error Rate. The overall error rate is 4.7%, knowing that participants were instructed to keep around 4% of errors. Data were pre-processed using an Aligned Rank Transform (ART) to take into account the non-normal distribution [33]. Repeated-measures ANOVA reveals a significant effect of Technique ($F_{5,829} = 30.3$, $p < 0.001$). Pairwise comparisons show a significant differences between *Raycasting* (12.9%) and all the other techniques (*HL*: 2.2%, *Bub*: 3.8%, *Bub+HL*: 1.9%, *Rope*: 4.3%, *Rope+HL*: 3.3%, $p < 0.006$). We also observe a significant effect of Size ($F_{1,829} = 140.7$, $p < 0.0001$) and a Technique×Size

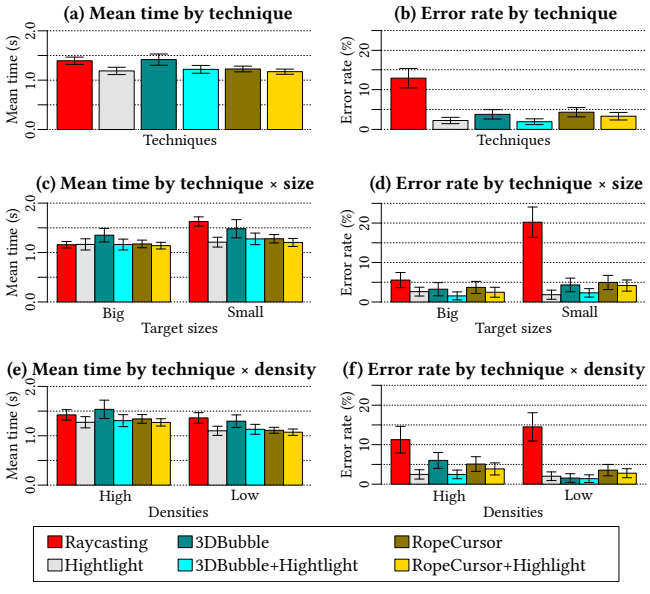

Figure 4: Mean times and error rates results for the visual feedforvard experiment, with 95% confidence intervals.

[2] All statistical analysis in the 4 experiments were performed using R, using $\alpha = 0.05$ and Greenhouse-Geisser corrections were applied to DoFs when sphericity was violated. We used Bonferroni corrections, where the p-values are multiplied by the number of comparisons. Detailed statistical analysis are available at ns.inria.fr/loki/raycursor.

interaction ($F_{5,829} = 26.4$, $p < 0.0001$). Pairwise comparisons only show that error rate with *Raycasting* is significantly higher for *small* (20.2%) targets than with *big* targets (5.6%). *Raycasting* also has a higher error rate for *small* targets than all the other techniques ($< 5\%$, $p < 0.0001$). The analysis also shows a significant Technique×Density interaction ($F_{5,829} = 5.8$, $p < 0.0001$). Pairwise comparisons show that *Raycasting* has a higher error rate for *low* density than all the other techniques ($< 3.6\%$, $p < 0.0001$).

Discussion. The results show that all visual feedforward used with *RayCursor* are overall more efficient than *Raycasting*. In particular *RayCursor* is overall faster for small targets while keeping a low error rate. Highlighting the nearest target is among the most efficient visual feedforwards (H1). Displaying a bubble does not allow to improve performance compared to *Raycasting*, especially for small targets or dense environments, certainly due to the higher visual clutter it introduces. These results are in line with Guillon *et al.*'s study for the 2D case.

Cursor transfer function

To check H2, we compared the performance of *VitLerp*, *DistProp*, and *VitLerp×DistDep* (a combination of *VitLerp* and *DistDep*) with 9 participants (2 female, all right handed, age mean=27.7, $\sigma = 6.5$). The parameters of each transfer function were tuned empirically to maximize performance. For *VitLerp*, the parameters were $k_1 = 30$, $k_2 = 150$, $v_1 = 0.05$ and $v_2 = 0.15$. For *DistProp*, the parameter was $k = 50$. The combination of *VitLerp* and *DistDep* (*VLDD*) consists in multiplying both gains. For *VLDD*, the parameters were $k_1 = 20$, $k_2 = 100$, $v_1 = 0.05$, $v_2 = 0.15$, $k = 1$ and $d = 0.55$.

The independent variables were transfer Function (*VitLerp*, *DistProp*, *VLDD*), target Position (*Near*, *Far*), Distance from the previous target (*Short*, *Long*), and Block (3). The targets to select were organized to combine long and short displacements, target far away and close to the user, to try to use all the possible range of speed and displacement for each function.

Selection time. A Box-Cox transformation ($\lambda = -0.4$) was applied to correct non-normal selection time residuals. A repeated measures ANOVA[2] showed a significant main effect of Function ($F_{2,16} = 3.8$, $p = 0.046$, $\eta^2_G = 0.07$). Pairwise comparisons show a marginally significant difference between *DistProp* and *VLDD* (*DP* : 2.79s ; *VLDD* : 2.39s ; $p = 0.05$). *DistProp* is slightly slower on average than *VLDD*. We also observe a significant effect of Position ($F_{1,8} = 223$, $p < 0.001$, $\eta^2_G = 0.55$) and a Function×Position interaction ($F_{2,16} = 25.1$, $p < 0.001$, $\eta^2_G = 0.16$). No significant difference was observed for the *Near* position. However for the *Far* position, pairwise comparison show a significant difference between *DistProp* (3.72s) and the two other functions (*VL* : 2.76s ; *VLDD* : 2.88s ; $p < 0.002$).

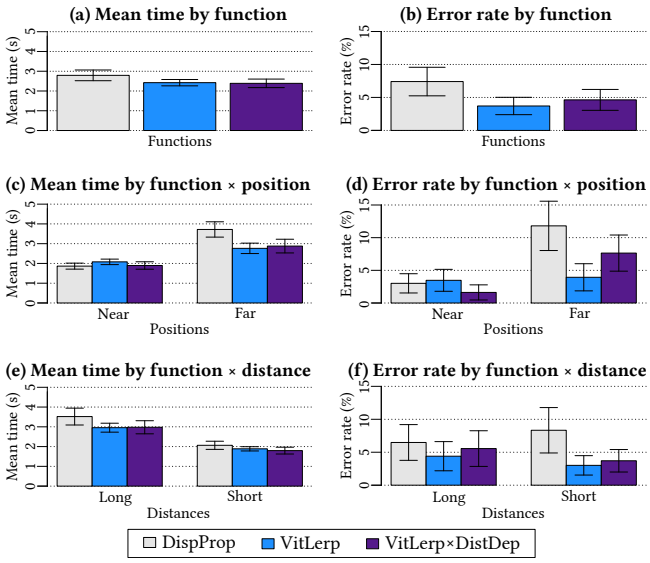

Figure 5: Mean times and error rates results from the transfert function experiment, with 95% confidence intervals.

Error Rate. Data were pre-processed using an ART to take into account the non-normal distribution. Repeated measures ANOVA shows a significant effect of Function ($F_{2,304} = 4.55$, $p < 0.0004$). Pairwise comparisons show a significant effect between *DistProp* and *VitLerp* (*DP* : 7.4% ; *VL* : 3.7% ; $p < 0.002$). Users globally made less errors with *VitLerp* than with *DistProp*. We also observe a significant effect of Position ($F_{1,304} = 29.6$, $p < 0.001$), and a Function×Position interaction ($F_{2,304} = 12.9$, $p < 0.0001$). Pairwise comparisons do not show statistical differences between transfer functions for *Near* targets. However we observed significant differences for *Far* targets between *DistProp* and *VitLerp* (*DP* : 11.8% ; *VL* : 3.9% ; $p < 0.008$).

We conclude that the transfer function influences the performance of *RayCursor* (H2 confirmed): *VitLerp* is faster and more reliable than *DistProp* for targets far away from the user and it is overall the most efficient function.

Ray filtering

In the previous experiments, smaller targets were harder to select because of hand tremor and input jitter. In this experiment we want to assess our hypothesis that filtering the ray will reduce selection errors for *Raycasting* (H3). While filtering has already been used in the literature to filter rays [20, 32], it was used in the context where the ray intersects a physical screen. We are not aware of previous work formally evaluating the effect of filtering for *Raycasting* on performance and error rate. Vogel *et al.* and Kopper *et al.* both used what corresponds to a primitive version of the *1€ Filter* [9]. We also used it as it appears to provide the best trade-off

between low jitter and low latency. In this experiment we only consider far away targets, which is the most difficult situation.

Methodology. Nine participants (2 female, all right handed, age mean=26.9, $\sigma = 6.25$) took part of this experiment. All of them already experienced Virtual Reality before.

We used a within-subjects design. Independent variables are: Technique and Block. The three techniques are described in section 3: *Raycasting* not filtered (*RC*), *Raycasting* filtered in the motor space ($1€_M$) and *Raycasting* filtered in the visual and motor spaces ($1€_{VM}$). The order of the three techniques was balanced among participants using a Latin square. We set the *1€ Filter* parameters to $\min_{cutoff} = 0.1$ and $\beta = 50$ according to empirical evaluations, in order to minimize latency and jitter. The selectable target was highlighted, according to the previous study on visual feedforward. For each technique, the participants had to perform 5 Blocks of 20 targets.

In summary, the experiment design is: 9 participants × 3 Techniques × 5 Blocks × 20 trials = 2,700 trials in total. The experiment lasted around 15min per participant.

Results. We analyze selection time and error to study the effects of filtering. There is an interruption between blocks, so we discarded the first trial of each block in the analysis. For selection time, we also discarded trials resulting in an error.

Selection Time. A Box-Cox transformation with $\lambda = -0.2$ was applied to correct non-normal selection time residuals. Repeated measures ANOVA^2 does not show a significant effect of Block ($F_{4,32} = 2.2, p = 0.09$) or Technique ($F_{1.2,9.8} = 2.0, p = 0.18$).

Error Rate. The overall error rate is 16.02%, despite the instruction to follow a 4% error rate. We applied a log transformation to correct non-normal error rate residuals. The analysis does not show a significant effect of Block ($F_{4,32} = 0.94, p > 0.05$). However we observe a significant effect of Technique ($F_{2,16} = 12.2, p < 0.001, \eta^2_G = 0.21$). Pairwise comparisons show a significant effect between *Raycasting*(25.2%) and the other techniques ($1€_{VM}$: 11.7%, $1€_M$: 11.1%, $p < 0.0001$).

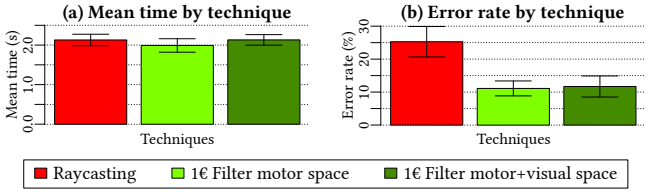

Figure 6: Mean times and error rates results from the *1€ Filter* experiment, with 95% confidence intervals.

Discussion. The results show that filtering the ray with the *1€ Filter* cuts selection errors by more than 50% (H3 confirmed). Whether only the motor space or both the visual and motor space are filtered does not influence selection errors. However our debriefing with participants reveals that 7/9 prefer when both the visual and motor space are filtered.

Discussion

Among the visual feedbacks we proposed for *RayCursor*, the efficient ones show the target that will be selected unambiguously. This is the case for all techniques highlighting the target, and the rope. Although several of the feedforwards have similar performance, we suggest just highlighting the nearest target, since this is the feedforward with the less visual clutter. It also suggests that the 3DBubble technique [29] would certainly also benefit from this type of feedforward.

Our studies about the transfer function for controlling the cursor showed that a non linear function of the cursor speed was the most efficient. This is in line with the literature and common transfer functions for computer mice and trackpads on a desktop [7]. We proposed a simplified function, parametrized with two input speed thresholds, and two extrema gains. The application designer can adapt these parameters to the size of the trackpad input area, and the distances the user has to travel. We informally observed that frequent forward and backward movements of the cursor over long distances slow down the technique. This brought us the idea of combining *Raycasting* and *RayCursor*, with a mechanism to teleport the cursor close to a point of interest, hence reducing travelling time.

Finally we showed that filtering the ray decreases selection errors. This is both beneficial for *Raycasting* and *RayCursor*. While our studies showed that either filtering just the motor space or both the visual and motor space increased performance, user preference tends to suggest it is better to filter both spaces.

In the following, we provide a last experiment to compare 1) *RayCursor* using the highlight feedback, tuned transfer function and filtered ray; 2) the same version of *RayCursor* but using a semi-automatic cursor positioning along the ray; 3) *Raycasting* using filtering; and 4) a recent technique from the literature, which features an adjustable ray [26].

6 COMPARATIVE STUDY

We evaluated every characteristic of *RayCursor*, and proposed optimal settings. In this section we compare two variations of the technique with *Raycasting* and the closest technique to ours in the literature [26]. The two variations are a manual control of the cursor, and a semi-automatic control of the cursor. The semi-automatic version (Figure 1, c-f) is an hybrid between *Raycasting* and *RayCursor*. Automatic cursor

control is enabled when the user does not touch the touchpad. In this mode when the ray intersects a target, the cursor moves automatically on the ray at the intersection point. If the ray moves out of the target, the target remains selectable using the proximity selection mechanism. If another target is closer to the cursor or intersects the ray, this new target is selected. If the user puts his finger on the touchpad, he can control the cursor position like in the manual version of *RayCursor* (the automatic control is disabled). If the user releases the touchpad more than 1s, the technique switches back to the automatic cursor behavior.

Raycasting and the two variations of *RayCursor* are filtered using the *1€ Filter* with the settings previously defined. Indeed, when *Raycasting* is not filtered its error rate is much higher than for other techniques according to our studies in section 5. This experiment will give us more insights on the performance improvements of *Raycasting* when it is filtered. We also implemented Ro *et al.*'s technique as it can be the closest technique to our work [26]. It also allows to measure the direct benefit of the use of the target proximity facilitation technique. As Ro *et al.* did not detail the transfer function used to control the length of their ray, we used for their technique the same transfer function used for *RayCursor*. Our hypothesis is that *RayCursor* with semi-automatic control of the cursor is faster and less error prone than others techniques tested (H4).

Methodology

Twelve participants (2 female, 1 left handed, age mean=27.6, $\sigma = 5.8$) took part of this experiment. Two of them experienced Virtual Reality for the first time. Six of them participated in previous experiments. The time between two of our experiments was at least 4 weeks, suggesting the acquired learning was minimal. The within subject design further reduces individual differences.

We used a within-subjects design, with factors: Technique, target Density, target Size and Block. The 4 techniques used are: *Raycasting* filtred with *1€ Filter* in visual and motor space (RC_f), the *RayCursor* with the cursor controlled manually by the user (*ManRCur*), the *RayCursor* with semi-automatic control of the cursor (*AutoRCur*) and the Ro *et al.*'s technique [26] (*Ro*). The order of techniques was balanced between participants with a Latin square. The 2 target sizes were $S_{Big} = 8$cm and $S_{Small} = 4$cm of diameter. The 2 densities used 30 targets (D_{Low}) and 60 targets (D_{High}). All the targets were spread out at random positions into 2 spheres of 60cm diameter, in front of the user, and centered at 80cm from the ground. The closer sphere was 1m in front of the user and the farther one at 4m. All targets were at sight when the user was looking at the virtual wall with the virtual display. The participant had to select a target alternately in the near and distant spheres. This condition

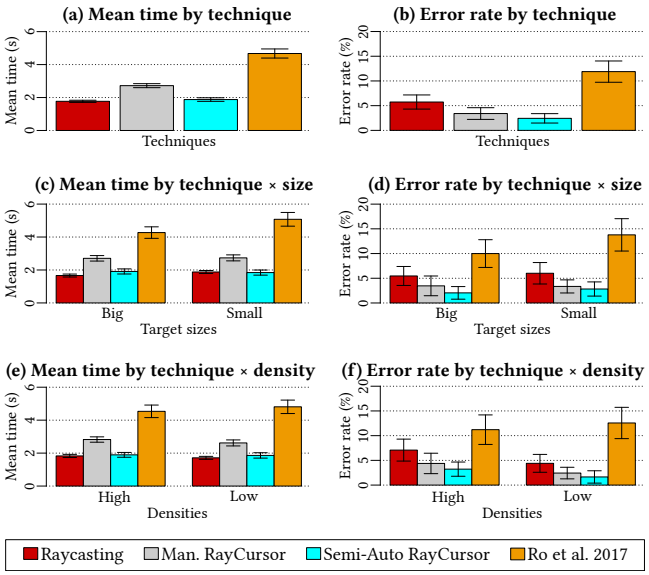

Figure 7: Mean times and error rates results for the comparative study, with 95% confidence intervals.

corresponds to the worse case scenario for *ManRCur* as the user has to constantly travel over long distances with the cursor along the ray. This is intended to point out the limits on *RayCursor*.

Participants were allowed to rest between blocks. The experiment design was: 12 participants × 4 Techniques × 3 Blocks × 2 Densities × 2 Sizes × 9 targets = 5, 184 trials in total. The experiment lasted around 30*min* per participant.

Results

Figure 7 shows selection times and error rates. We also discuss user preferences.

Selection time. In this analysis, selection time refers to the time elapsed between two selections. Therefore the first trial of each sequence of 10 targets is discarded, as well as trials resulting in an error. We also removed outliers trials, for which the selection time was above $mean + 3 \times sd$ for each technique. A Box-Cox transformation with $\lambda = -0.3$ was applied to correct non-normal selection time residuals. Repeated measures ANOVA[2] found a significant effect of Block ($F_{2,22} = 21.3$, $p = 0.001$, $\eta^2_G = 0.05$), with pairwise comparisons revealing a significant difference between blocks 1 and 3 ($p < 0.016$, Block 1: 2.96s, 2: 2.72s, 3: 2.59s). As these results do not clearly suggest a learning or fatigue effect, we kept all blocks for subsequent analysis.

We observe also a significant effect of Technique ($F_{1.4,15.9} = 120.0$, $p < 0.0001$, $\eta^2_G = 0.71$). Pairwise comparisons show a significant effect between *Ro* and all the other techniques (RC_f: 1.77s, *ManRCur*: 2.72s, *AutoRCur*: 1.88s, *Ro*: 4.67s, $p < 0.0001$), as

well as between *ManRCur* and all the others ($p < 0.0001$). *Ro* technique is significantly slower than *ManRCur*, that is significantly slower than the two last techniques. With the semi-auto *RayCursor* and *Raycasting*, the user just has to aim the right target with the ray in most cases. Therefore these techniques are faster than the others, which requires the user to move the cursor along the ray to reach the target.

The analysis shows a significant effect of Size ($F_{1,11} = 11.0$, $p < 0.007$, $\eta^2_G = 0.02$) and a Technique×Size interaction ($F_{3,33} = 11.8$, $p < 0.0001$, $\eta^2_G = 0.03$). Pairwise comparisons only show a significant difference for *Ro* technique between *Big* and *Small* targets (Ro_{Small}: 5.08s, Ro_{Big}: 4.27s, $p = 0.05$). We observe that our implementation of Ro *et al.*'s technique is slower for the *Small* targets. We also found a significant effect of Density ($F_{1,11} = 5.8$, $p < 0.035$, $\eta^2_G = 0.01$) and Technique×Density interaction ($F_{3,33} = 3.4$, $p < 0.03$, $\eta^2_G = 0.01$). Post-hoc analysis reveal that selection time increases for *Raycasting* for the *High* density ($p < 0.001$, *High*: 1.82s, *Low*: 1.71s).

Error Rate. The overall error rate is 5.9%, knowing that participants were instructed to keep around 4% of errors. Data were pre-processed using an ART to take into account the non-normal distribution. Repeated measures ANOVA does not show a Block effect ($F_{2,562} = 2.9$, $p > 0.05$). However it reveals a significant effect of Technique ($F_{3,549} = 25.5$, $p < 0.0001$). Pairwise comparisons show significant differences between Ro *et al.*'s technique (11.9%) and all the other techniques (RC_f: 5.7%, *ManRCur*: 3.8%, *AutoRCur*: 2.4%, $p < 0.0001$), as well as between RC_f and *AutoRCur* ($p = 0.013$). The high error rate of Ro *et al.*'s technique is probably due to the lack of proximity selection, compared to the manual *RayCursor*. Also, semi-auto *RayCursor* has lower error rate than *Raycasting*. We explain this effect by the proximity selection when the user deviates from the targeted object while trying to select it. There was also a significant effect of Size ($F_{1,549} = 11.9$, $p < 0.001$) showing higher error rate for the small target size (*Small*: 6.5%, *Large*: 5.2%). No significant effect of Density ($F_{1,549} = 0.01$, $p > 0.05$) was found.

User preferences. At the end of the experiment, each participant was instructed to rank the techniques according to their preference. A Friedman test revealed a significant effect of technique on user preference ($\chi^2(3) = 25.3$, $p < 0.0001$). Wilcoxon post-hoc analysis showed significant differences between Ro *et al.*'s technique (median rank = 4) and all others techniques (median ranks: RC_f: 2, *ManRCur*: 2, *AutoRCur*: 1, $p < 0.0001$) and between manual and semi-auto *RayCursor* ($p = 0.041$). These subjective results are in line with our analysis of error rates and selection time. A majority of participants ranked the semi-auto *RayCursor* first (9 out of 12).

Discussion

This experiment shows that our semi-auto *RayCursor* yields selection times similar to the filtered *Raycasting*, across different target sizes and densities. However, semi-auto *RayCursor* significantly improves error rates over the filtered *Raycasting* across the different conditions. It shows the efficiency of 1) the *1€ Filter* to reduce jitter; 2) the selection of targets by proximity; and 3) the semi-automatic placement of the cursor along the ray. When selecting targets far away the semi-auto *RayCursor* could have been negatively impacted by the ray hitting close targets and making the cursor suddenly jumps, but it was not the case. If jumps would occur due to targets approximately equally far away, the use of an hysteresis function would help solving the problem. Our manual *RayCursor* obtained lower time performance compared to the previously mentioned techniques, certainly due to the incessant cursor displacement required to move the cursor forward and backward from one trial to the other. However when comparing the manual *RayCursor* to Ro *et al.*'s technique, it clearly shows the benefits of proximity selection.

7 CONCLUSION

We presented *RayCursor*, a new interaction technique for 3D target selection in immersive environments. This technique is an improvement of *Raycasting* that uses a cursor on the ray to select the nearest target. Displaying a bubble was the typical visual feedforward for such a technique. However Guillon *et al.*'s showed that, in a 2D context, highlighting the nearest target is more efficient, and produces less visual clutter [16, 17]. We extended their results to 3D interaction, with similar conclusions. Despite the existence of previous work filtering the ray, we described the first evaluation of a filtered *Raycasting* formally evaluating its benefits. We showed that filtering the ray strongly reduces selection errors. This is both beneficial for *Raycasting* and *RayCursor*. Our results also demonstrate that transfer functions like the ones used on desktop interfaces are efficient for the control of a cursor on a ray. We recommend using a sigmoid function that depends on the cursor speed. Last, our comparative study shows that a filtered *Raycasting* has decent performance on several target sizes and densities. We also show that an hybrid technique between *Raycasting* and *RayCursor* has the lowest error rate, while being as fast as *Raycasting*. In 3D, selection often precede the manipulation of a 3D object. A side and important benefit of our technique is its ability to manipulate the object along the ray once it is selected, for example to bring it close to the user, something the standard *Raycasting* does not allow to do. Future work will focus on the use of *RayCursor* to enable both the selection and manipulation of 3D objects in immersive environments.

8 ACKNOWLEDGMENTS

This work was partially supported by the Inria IPL Avatar project.

REFERENCES

[1] Ferran Argelaguet and Carlos Andujar. 2013. A survey of 3D object selection techniques for virtual environments. *Computers and Graphics* 37, 3 (2013), 121 – 136. https://doi.org/10.1016/j.cag.2012.12.003

[2] Marc Baloup, Veis Oudjail, Thomas Pietrzak, and Géry Casiez. 2018. Pointing Techniques for Distant Targets in Virtual Reality. In *Proceedings of the AFIHM Conférence Francophone sur l'interaction Homme-Machine (IHM 2018) (Articles Scientifiques)*. Brest, France, 8. https://hal.archives-ouvertes.fr/hal-01899061

[3] Renaud Blanch, Yves Guiard, and Michel Beaudouin-Lafon. 2004. Semantic Pointing: Improving Target Acquisition with Control-display Ratio Adaptation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*. ACM, New York, NY, USA, 519–526. https://doi.org/10.1145/985692.985758

[4] Doug A. Bowman, Ernst Kruijff, Joseph J. LaViola, and Ivan Poupyrev. 2004. *3D User Interfaces: Theory and Practice*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

[5] George EP Box and David R Cox. 1964. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)* (1964), 211–252.

[6] Jeffrey Cashion, Chadwick Wingrave, and Joseph J LaViola Jr. 2012. Dense and dynamic 3d selection for game-based virtual environments. *IEEE transactions on visualization and computer graphics* 18, 4 (2012), 634–642. https://doi.org/10.1109/TVCG.2012.40

[7] Géry Casiez and Nicolas Roussel. 2011. No More Bricolage!: Methods and Tools to Characterize, Replicate and Compare Pointing Transfer Functions. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 603–614. https://doi.org/10.1145/2047196.2047276

[8] Géry Casiez, Nicolas Roussel, Romuald Vanbelleghem, and Frédéric Giraud. 2011. Surfpad: Riding Towards Targets on a Squeeze Film Effect. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2491–2500. https://doi.org/10.1145/1978942.1979307

[9] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 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 (CHI '12)*. ACM, New York, NY, USA, 2527–2530. https://doi.org/10.1145/2207676.2208639

[10] Gerwin de Haan, Michal Koutek, and Frits H. Post. 2005. IntenSelect: Using Dynamic Object Rating for Assisting 3D Object Selection. In *Proceedings of the 11th Eurographics Conference on Virtual Environments (EGVE'05)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 201–209. https://doi.org/10.2312/EGVE/IPT_EGVE2005/201-209

[11] William Delamare, Céline Coutrix, and Laurence Nigay. 2013. Mobile Pointing Task in the Physical World: Balancing Focus and Performance While Disambiguating. In *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '13)*. ACM, New York, NY, USA, 89–98. https://doi.org/10.1145/2493190.2493232

[12] Niklas Elmqvist and Jean-Daniel Fekete. 2008. Semantic Pointing for Object Picking in Complex 3D Environments. In *Proceedings of Graphics Interface 2008 (GI '08)*. Canadian Information Processing Society, Toronto, Ont., Canada, 243–250. http://dl.acm.org/citation.cfm?id=1375714.1375755

[13] Tovi Grossman and Ravin Balakrishnan. 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 (CHI '05)*. ACM, New York, NY, USA, 281–290. https://doi.org/10.1145/1054972.1055012

[14] Tovi Grossman and Ravin Balakrishnan. 2006. The Design and Evaluation of Selection Techniques for 3D Volumetric Displays. In *Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST '06)*. ACM, New York, NY, USA, 3–12. https://doi.org/10.1145/1166253.1166257

[15] Yves Guiard, Renaud Blanch, and Michel Beaudouin-Lafon. 2004. Object Pointing: A Complement to Bitmap Pointing in GUIs. In *Proceedings of Graphics Interface 2004 (GI '04)*. Canadian Human-Computer Communications Society, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 9–16. http://dl.acm.org/citation.cfm?id=1006058.1006060

[16] Maxime Guillon, François Leitner, and Laurence Nigay. 2014. Static Voronoi-Based Target Expansion Technique for Distant Pointing. In *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI 2014)*, Franca Garzotto, Paolo Paolini, Antonella De Angeli, Giulio Jacucci, Alessio Malizia, Maristella Matera, and Rosa Lanzilotti (Eds.). ACM, Como, Italy, 41–48. https://doi.org/10.1145/2598153.2598178

[17] Maxime Guillon, François Leitner, and Laurence Nigay. 2015. Investigating Visual Feedforward for Target Expansion Techniques. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 2777–2786. https://doi.org/10.1145/2702123.2702375

[18] Aakar Gupta, Thomas Pietrzak, Cleon Yau, Nicolas Roussel, and Ravin Balakrishnan. 2017. Summon and Select: Rapid Interaction with Interface Controls in Mid-air. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces (ISS '17)*. ACM, New York, NY, USA, 52–61. https://doi.org/10.1145/3132272.3134120

[19] Ken Hinckley, Randy Pausch, John C. Goble, and Neal F. Kassell. 1994. A Survey of Design Issues in Spatial Input. In *Proceedings of the 7th Annual ACM Symposium on User Interface Software and Technology (UIST '94)*. ACM, New York, NY, USA, 213–222. https://doi.org/10.1145/192426.192501

[20] Regis Kopper, Felipe Bacim, and Doug A Bowman. 2011. Rapid and accurate 3D selection by progressive refinement. In *2011 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, IEEE Computer Society, Washington, DC, USA, 67–74. https://doi.org/10.1109/3DUI.2011.5759219

[21] Mikko Kytö, Barrett Ens, Thammathip Piumsomboon, Gun A. Lee, and Mark Billinghurst. 2018. Pinpointing: Precise Head- and Eye-Based Target Selection for Augmented Reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 81, 14 pages. https://doi.org/10.1145/3173574.3173655

[22] Jiandong Liang and Mark Green. 1994. JDCAD: A highly interactive 3D modeling system. *Computers & Graphics* 18, 4 (1994), 499 – 506. https://doi.org/10.1016/0097-8493(94)90062-0

[23] Mark R Mine. 1995. *Virtual environment interaction techniques*. Technical Report. University of North Carolina, Chapel Hill, NC, USA.

[24] Mathieu Nancel, Olivier Chapuis, Emmanuel Pietriga, Xing-Dong Yang, Pourang P Irani, and Michel Beaudouin-Lafon. 2013. High-precision pointing on large wall displays using small handheld devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 831–840. https://doi.org/10.1145/2470654.2470773

[25] Gang Ren and Eamonn O'Neill. 2013. 3D selection with freehand gesture. *Computers & Graphics* 37, 3 (May 2013), 101–120. https://doi.org/10.1016/j.cag.2012.12.006

[26] Hyocheol Ro, Seungho Chae, Inhwan Kim, Junghyun Byun, Yoonsik Yang, Yoonjung Park, and Tackdon Han. 2017. A dynamic depth-variable ray-casting interface for object manipulation in ar environments. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE Systems, Man, and Cybernetics Society, 2873–2878. https://doi.org/10.1109/SMC.2017.8123063

[27] Greg Schmidt, Yohan Baillot, Dennis G. Brown, Erik B. Tomlin, and J. Edward II Swan. 2006. Toward Disambiguating Multiple Selections for Frustum-Based Pointing. In *Proceedings of the 3D User Interfaces (3DUI '06)*. IEEE Computer Society, Washington, DC, USA, 87–94. https://doi.org/10.1109/VR.2006.133

[28] Frank Steinicke, Timo Ropinski, and Klaus Hinrichs. 2004. Object selection in virtual environments with an improved virtual pointer metaphor. In *Computer Vision and Graphics: International Conference, ICCVG 2004*. Springer Netherlands, Warsaw, Poland, 320–326. https://doi.org/10.1007/1-4020-4179-9_46

[29] Lode Vanacken, Tovi Grossman, and Karin Coninx. 2007. Exploring the Effects of Environment Density and Target Visibility on Object Selection in 3D Virtual Environments. In *2007 IEEE Symposium on 3D User Interfaces*. IEEE Computer Society, Washington, DC, USA. https://doi.org/10.1109/3DUI.2007.340783

[30] Donald Lee Vickers. 1972. *Sorcerer's Apprentice: Head-mounted Display and Wand*. Ph.D. Dissertation.

[31] Vive. 2019. HTC Vive VR headset. https://www.vive.com/us/product/vive-virtual-reality-system/, retrieved January 7th, 2019.

[32] Daniel Vogel and Ravin Balakrishnan. 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 (UIST '05)*. ACM, New York, NY, USA, 33–42. https://doi.org/10.1145/1095034.1095041

[33] Jacob O. Wobbrock, Leah Findlater, Darren Gergle, and James J. Higgins. 2011. The Aligned Rank Transform for Nonparametric Factorial Analyses Using Only Anova Procedures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 143–146. https://doi.org/10.1145/1978942.1978963