# Evaluating Automatic Parameter Control Methods for Locomotion in Multiscale Virtual Environments

Jong-In Lee
Simon Fraser University,
Adobe Research
jonginl@sfu.ca

Paul Asente
Byungmoon Kim
asente@adobe.com
bmkim@adobe.com
Adobe Research

Yeojin Kim
Ewha Womans University,
Adobe Systems
kim@adobe.com

Wolfgang Stuerzlinger
Simon Fraser University
w.s@sfu.ca

## ABSTRACT

Virtual environments with a wide range of scales are becoming commonplace in Virtual Reality applications. Methods to control locomotion parameters can help users explore such environments more easily. For multi-scale virtual environments, point-and-teleport locomotion with a well-designed distance control method can enable mid-air teleportation, which makes it competitive to flying interfaces. Yet, automatic distance control for point-and-teleport has not been studied in such environments. We present a new method to automatically control the distance for point-and-teleport. In our first user study, we used a solar system environment to compare three methods: automatic distance control for point-and-teleport, manual distance control for point-and-teleport, and automatic speed control for flying. Results showed that automatic control significantly reduces overshoot compared with manual control for point-and-teleport, but the discontinuous nature of teleportation made users prefer flying with automatic speed control. We conducted a second study to compare automatic-speed-controlled flying and two versions of our teleportation method with automatic distance control, one incorporating optical flow cues. We found that point-and-teleport with optical flow cues and automatic distance control was more accurate than flying with automatic speed control, and both were equally preferred to point-and-teleport without the cues.

## CCS CONCEPTS

• **Human-centered computing** → **Human computer interaction (HCI)**; User studies.

## KEYWORDS

VR navigation, Automatic control, Point-and-teleport, multiscale virtual environments

**ACM Reference Format:**
Jong-In Lee, Paul Asente, Byungmoon Kim, Yeojin Kim, and Wolfgang Stuerzlinger. 2020. Evaluating Automatic Parameter Control Methods for Locomotion in Multiscale Virtual Environments. In *26th ACM Symposium on Virtual Reality Software and Technology (VRST '20), November 1–4, 2020, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3385956.3418961


*VRST '20, November 1–4, 2020, Virtual Event, Canada*

ACM ISBN 978-1-4503-7619-8/20/11…$15.00
https://doi.org/10.1145/3385956.3418961

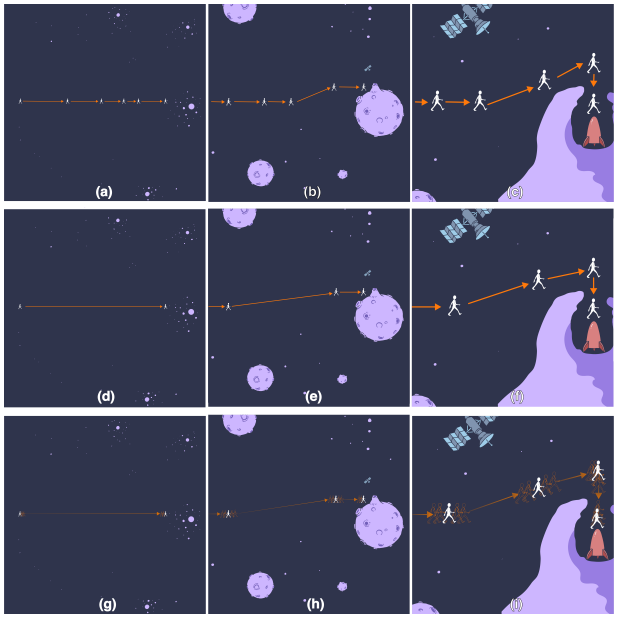


**Figure 1: Navigation that requires both long-distance and fine-grained movement in an unconstrained multiscale VE. Images (a)-(c) illustrate a user teleporting with manual distance control. (d)-(f) show our method, which automatically controls the distance based on the proximity of the surroundings and enables efficient travel. Our second method provides continuous visual updates (g)-(i), which lets users track the direction to the target more effectively. Reversing the paths would require mid-air navigation, which could not be accomplished with traditional point-and-teleport.**

## 1 INTRODUCTION

Locomotion is a core navigation task in Virtual Environments (VEs). Changing poses through a Virtual Reality (VR) locomotion interface lets users find appropriate viewpoints for activities like inspection, selection, and manipulation. *Multiscale VEs* present special challenges for such activities. These environments occur in many fields, including cosmology, medicine, and architecture [9, 17, 23, 40]. Feature sizes and the distances between them vary widely, often by many orders of magnitude. Such VEs contain features ranging from mountains to molehills, or even from planets to individual plants. While exploring a new environment, users must be able to quickly traverse the large gap between major objects without getting lost in

empty space [11] while still having sufficient fine-grained control to inspect and manipulate small details. Using constant travel parameters, especially speed, would be impractical. A speed that is low enough for a detailed exploration would be too slow to cross the vast empty spaces between major objects, and one that is suitable for large spaces would make it impossible to accurately move among small objects (Figure 1). Furthermore, high speeds are known to cause overshooting—passing an intended target [38].

To resolve this problem, the user must be able to change the travel parameters. However, manually controlling speed, acceleration, and gain for travel at different scales is challenging. Extra cognitive load, longer travel times, and greater distances lead to higher navigation error rates [1]. It is useful to distinguish between surface-viewpoint and unconstrained-viewpoint VE navigation, a distinction that has been made in previous work [25, 34]. In the first, a user virtually stands on a ground plane or another object, with a viewpoint some distance above the standing position. In the second, the viewpoint can be positioned arbitrarily in space. Surface-viewpoint VE navigation is simpler since there is always some geometry present that the user can use to specify a destination. With unconstrained-viewpoint VE navigation, there might be nothing near the intended destination. Flying is one technique for navigating a multiscale VE with an unconstrained viewpoint. Researchers have proposed several automatic speed control techniques for it [1, 27, 33, 38]. However, automatic distance control for the other major navigation interfaces, such as teleportation, remain unstudied.

Since Bowman et al. [6] showed that pointing-based techniques are comfortable to use through the decoupling of viewing and moving actions, and perform better than gaze-based technique, teleportation had been studied in VR research [3, 5, 13]. After decades, Bozgeyilkli et al. presented a similar technique called "Point-and-Teleport" [7]. It showed that point-and-teleport is intuitive, easy-to-use, and does not necessarily lead to spatial disorientation, as long as the orientation is preserved. This makes it the dominant egocentric technique for constrained-viewpoint navigation, with the user's viewpoint bound to surfaces, and it is now commonly used in commercial VR applications. However, such point-and-teleport techniques require users to specify a target destination on a 2D surface, making them impossible to use for unconstrained-viewpoint VE navigation. One of a few exceptions is the work of Drogemuller et al. [8], which uses pointing direction and manual distance control to specify a destination in space.

In this paper we present a novel approach to automatically control the user's teleport distance, letting the user easily teleport to destinations in multiscale VEs by simply aiming the controller. It uses the distance between the user's viewpoint and the closest surface combined with the gradient computed from the distance field of the scene. To evaluate our new method, we conducted a user study in a large open VE and compared our method with flying with automatic speed control, currently the most prominent locomotion technique in Multiscale VEs, and with teleportation with manual distance control. The results showed that our method decreases task completion time, task load, and allows more precise navigation in a large open VE. However, due to its discontinuous nature, participants found it challenging to use in terms of spatial updating. To address this issue, we implemented a variant of our method incorporating continuous movement, and assessed its effectiveness in another study. The results showed that this variant enabled users to navigate more accurately in a dense VE.

## 2 RELATED WORK

Teleportation has been widely used for various VR applications [12, 32, 35, 36]. In VR research, portal-based teleportation for a CAVE was proposed by Freitag et al. [13], which let users create a "target portal" by pointing at any visible position and entering a "start portal" to jump to the target. Authors reported that this technique frequently caused a loss of orientation after exiting the portal. Liu et al. [22] later presented redirected teleportation for HMDs, which is similar to Freitag's approach, but addressed the disorientation issue by creating portals that gradually redirect the user towards the center of the tracking space.

Since Bowman et al. identified the advantages of pointing-based locomotion [6], several subsequent studies investigated pointing-based teleportation. Bolte et al. [5] proposed a technique in which users aim their heads to the target and then physically jump forward to execute teleportation. Even though this technique does not require an additional input device, it is impractical for a long usage due to its physical inefficiency. Bozgeyikli et al. [7] later proposed a point-and-teleport technique that was easier and less physically fatiguing. Since then, many researchers studied its properties. Xu et al. [45] found that a user's performance in building spatial knowledge was not significantly different between point-and-teleport, joystick, and walk-in-place techniques. Also, Frommel et al. [14] found that point-and-teleport induced the least discomfort compared with teleportation to predefined positions, automatic locomotion, and joystick-based techniques. Bhandari et al.[4] presented the "Dash" teleportation technique that moved the user's viewpoint continuously to the destination. While there was no difference in VR sickness between Dash and regular teleportation, Dash significantly improved the path integration process, which mentally updates one's position and orientation from the initial position [24]. Researchers also introduced different techniques that enable predefining the orientation before teleporting[10, 15]. These decreased the need to correct the orientation afterwards, but required extra time and effort to specify that orientation before travel execution. The techniques and user evaluations in these studies focused on 2D constrained-viewpoint teleportation where the user is always moving parallel to a ground plane. However, for unconstrained-viewpoint VE navigation, one must be able to specify destinations in mid-air, whether there is geometry nearby or not, making these techniques unusable. Drogemuller et al. [8] let users teleport to arbitrary destinations by pointing the controller and specifying a distance through a trackpad, but their VEs were neither particularly large nor multiscale. They performed extensive evaluations, but not in the kinds of environments we are investigating.

When navigating in a multiscale VE, travel parameters like speed should be controlled in a way that lets users explore the scene easily. Constant speed without control can lead to various problems. A user may feel frustrated if travel speed is either too slow, substantially increasing travel completion time, or too fast, leading the user to overshoot and forcing them to turn around and readjust the viewpoint frequently. However, manual speed control increases the complexity of the interface. Trindade and Raposo [38] found that users tend to make errors manually controlling the speed, which

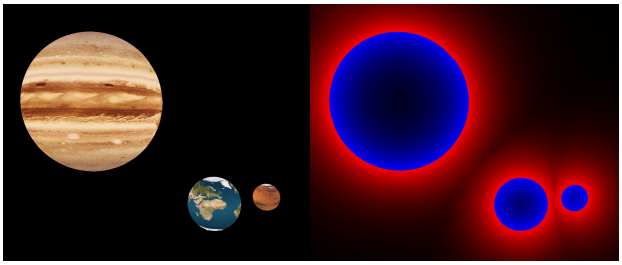

**Figure 2: The figure on the left shows 3D objects rendered with raycasting and the right shows a visualization of a plane through the signed distance field. Values are negative in the blue area and positive in the red area. The level of saturation shows the magnitude of the value.**

leads to increased travel times and lower usability. Researchers have studied various approaches to automatic speed control for flying. Since Mackinlay et al. [26] first suggested a speed control method based on the target distance, Ware and Fleet [41] presented a method that considers distances to visible points and found that both the minimum and average distance work well. McCrae et al. [27] proposed a cubemap-based approach that generated six depth maps by looking from the viewpoint in the six axial directions with 90° frustums, then computed a weighted average from the depth maps to control the speed. Trindade and Raposo [38] improved this method for multiscale VE navigation by using an exponential weighted average between the global minimum and the distance to the closest object in the view direction. This avoided having nearby out-of-sight objects influence the global minimum, which slowed the travel speed drastically when traveling close to any objects. Papoi and Stuerzlinger [33] further improved on this approach by weighing the contribution of all scene content in the forward direction depending on its closeness to the view vector. Argelaguet et al. [1] developed an approach that adjusted the speed to maintain constant optical flow. They found no significant difference between their method and other distance-based approaches. Argelaguet and Maignant then developed the GiAnt technique, which automatically adjusts flying speed and the VE scale factor to keep the perceived speed constant and to avoid diplopia [2]. Taunay et al. proposed a spatial partitioning heuristic for controlling the speed in static scenes and later extended it for dynamic scenes through distributed computation [37]. Even though previous work presented different approaches for automatically controlling flying speed, automatic distance control for teleportation remains unstudied. Here, we present a new technique that automatically controls distance to enable users to teleport in any pointed direction even when nothing exists in that direction in the air.

## 3 AUTOMATIC DISTANCE CONTROL

Our new automatic distance control technique uses a signed distance field of the entire scene for distance calculations. We generate this distance field once for the whole scene using an efficient hierarchical octree-based method [21], which we modified to support very large spaces with a high level of detail. The distance of each cell was calculated as the Euclidean distance between its center and the center of the closest non-void voxel. The field enables users to benefit from automatic distance control for unconstrained exploration in any unknown environment. This differentiates our approach from the NaviFields method, which requires predefining target positions [30]—a time-consuming approach this is not feasible for large environments. Figure 2 visualizes a distance field around three spherical objects. Because distances can be precomputed with a distance field, the computation at runtime is much faster than for the cubemap approaches, which require substantial computation at each frame [27, 33, 38]. While our user studies use static scenes and precomputed distance fields, it is also possible to support dynamic scenes by updating distance fields in real-time with GPU computation or distributed methods [37].

We calculate the teleportation displacement based on the distance field and its gradient. The length $l$ of the teleportation vector from the position of the controller $\mathbf{c}$ is:

$$l = f(\mathbf{c}) + k\mathbf{d} \cdot \nabla f(\mathbf{c}) \tag{1}$$

In the above equation, $f(\mathbf{c})$ is the value of the distance field at $\mathbf{c}$, and $k$ is a weight that controls how much the field gradient affects the distance. Using a pilot study, we determined that setting $k$ to 1 prevented users from colliding with an object that they are approaching while still minimizing the number of teleports. $\mathbf{d}$ is a unit vector in the direction pointed to by the controller (Figure 3). $l$ is affected by the distance from the controller to the closest surface as given by the distance field function. Then, the pose of the controller $\mathbf{d}$ influences the final value of $l$. The value increases when the controller is aimed in the same direction as the gradient vector and decreases when the controller is aimed against it. Since the gradient vectors always point outward from the closest surface, this method prevents users from teleporting directly onto a surface. It also lets them quickly teleport away from nearby surfaces. If a user stands on a large surface, points up into mid-air, and triggers multiple teleports, the distance will gradually increase as the user moves further from the surface. Conversely, the distance will gradually decrease as the user approaches an object, reducing the potential for overshooting. When the teleport direction is parallel to a surface, the distance depends on the height above the surface, with larger heights triggering longer teleports. When traversing a narrow tunnel, the gradient vectors in the centerline are perpendicular to the movement direction, which makes the second term of the equation zero, and thus the distance is only determined

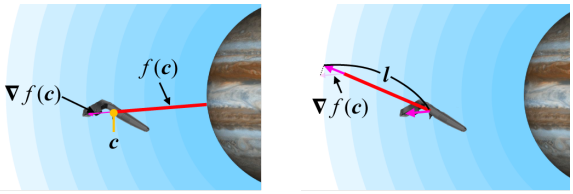


**Figure 3: The teleport distance is based on the distance to the closest object and the gradient at the controller position. (a) illustrates the position of the controller c, the closest distance $f(\mathbf{c})$ retrieved from the distance field, and the gradient at the controller's position $\nabla f(\mathbf{c})$. In (b), the gradient is projected onto the pointing vector and increases the length of the ray by the magnitude of the projected vector.**

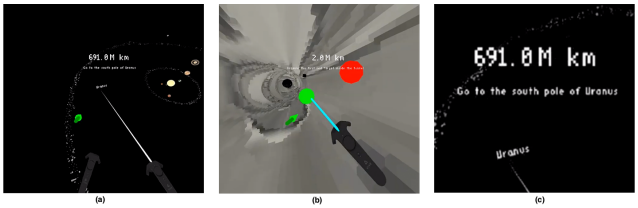


**Figure 4: The experiment included two sessions. (a) In the first session, PlanetTask, participants were asked to travel large distances and collect targets in a solar-system like environment. (b) In the second session, TunnelTask, they were asked to travel through a tunnel and to collect targets along its walls. (c) The instruction for the current task, displayed below the current teleport distance.**

by the first term. We did not incorporate viewing direction into the equation to decouple pointing action from head movement. This enables users to even point behind them to back away from a surface while facing it to get a better view.

# 4 USER STUDY 1: LOCOMOTION WITH AUTOMATIC PARAMETER CONTROL IN A LARGE OPEN ENVIRONMENT

We conducted a user study to evaluate our automatic distance control for point-and-teleport (AutoTeleport) in comparison to automatic speed control for flying (AutoFlying) and to manual distance control for point-and-teleport (ManualTeleport) .

## 4.1 Design

A single-variable within-subject design with three navigation *Techniques* were used: ManualTeleport, AutoTeleport, AutoFlying. For AutoTeleport and AutoFlying, users only had to point the controller held in their dominant hand and push the touchpad to teleport or to fly. We designed an automatic speed control similar to previous work [27] that uses the minimal distance from the user's viewpoint to the nearest geometry to determine flying speed.[1] To develop ManualTeleport for mid-air point-and-teleport, we built on a technique from previous work, where users control the distance using the touchpad [8].[2] The $y$-touch position determined the speed with which the teleport distance changes—touching the upper or lower part of the touchpad made the distance longer or shorter, with the rate of change controlled by how far the touch was from the center. The distance increased or decreased by up to $0.2m$ every 10 milliseconds. This mapping was determined through a pilot study, where we aimed to balance the goals of letting users quickly set a desired distance while not being too sensitive, to avoid large over- and undershoots. To improve the technique, we developed additional features. First, we moved the distance control input to the controller held in the non-dominant hand so that the the dominant hand controlled solely pointing and executing teleportation. This let users change distance and point-and-click at the same time. In the pilot study, we observed that users could travel to intended positions quicker and more comfortably compared with the interface that combined all functionality onto a single controller. Second, we let users reset the distance to its initial value by pressing the distance-control touchpad. Results showed that users sometimes increased the distance so much that it took a long time to return to a reasonable value.

Montello's taxonomy identifies two components of navigation: locomotion and wayfinding [31]. The scope of this study was locomotion, the motor component of navigation. As this study was not about an aided wayfinding task [43], we did not incorporate guidance and visualization techniques for selecting distant targets, such as that done by Mendes et. al [29].

The study was split into two sessions. First, we asked participants to navigate to targets in a solar-system-like multiscale environment, with all targets being placed far apart (PlanetTask). Second, participants were asked to acquire targets while travelling through a tunnel (TunnelTask).

## 4.2 Participants

We recruited 15 participants for the study from the local university. There were ten males and five females, and the average age was 24.3 (SD=4.09). All participants had experience with 3D games. They were compensated with 15 Canadian dollars for their participation.

## 4.3 Apparatus and Environment

Our experiment was conducted with an HTC Vive HMD, with 1080 × 1200 pixels for each eye, and Vive controllers for input. The physical space was 3m × 3m, spacious enough to let participants turn freely and perform point-and-teleport in any direction.

To evaluate our distance control method, we sketched the solar system in the Canvox system[3]. The VE contained celestial bodies of varying scales, with a sun, nine planets and hundreds of asteroids in two asteroid belts. The diameter of the sun was more than 100 times larger than that of the smallest object in the asteroid belts.

The HUD interface always showed the current teleport distance or the flying speed, instructions for the current task, and a target direction indicator (Figure 4), rendered as a 3D green arrow pointing towards the target. In ManualTeleport and AutoTeleport, the interface included a cyan vector with a green sphere at the far end to visualize where the user would end up with a teleportation using the current settings. In AutoFlying, it included a cyan cone at the tip of the controller to visualize the direction of flying.

## 4.4 Task

In the first part of the experimental task session, they were asked to "collect" 14 targets as quickly as possible by navigating to them. A target was considered to be collected when the user's HMD position overlapped with it. The targets were positioned far from each other (Figure 4(a)). The user could not see the next target from the current one, because they were either much too far away to be visible or were hidden behind other celestial bodies. The label of the planet with the current target was always visible regardless of the distance. Participants were shown a description of how to

[1] We did not use Trindade et al.'s method [38] because it uses a single ray, which introduces instability in large environments. We did not use GiAnt [2] because it also scales the environment. Other work has been done since our user study [33, 37].
[2] Funk et al.'s technique [15] is newer, but it only supports targets on surfaces and does not allow mid-air teleportation.

[3] Canvox is a volumetric VR painting system that uses an efficient dynamic octree data structure together with GPU acceleration [21]. We used this system because of its hierarchical-distance field computation and octree traversal faster than other 3D game engines. High-performance of these features was crucial for our method to provide smooth navigation experience without any notable latency and error in teleportation distance calculation.

get to the next target's location, for example, "Go to the south pole of Uranus," and asked to follow a direction indicator to find the way to the target. We refer to this as the PlanetTask. The second task required subjects to collect 20 targets along the walls of the tunnel mentioned above as quickly as possible (Figure 4(b)). We call this the TunnelTask. All the targets in both task were at the same locations across different conditions.

## 4.5 Procedure

First, participants were asked to complete a brief questionnaire about their background. After the survey, they were asked to wear the HMD and hold both of the controllers while standing in the middle of the experimental space. They were encouraged to rotate their body freely, but asked to stay roughly in the middle. Participants encountered the three conditions, (AutoTeleport, ManualTeleport, and AutoFlying) in counterbalanced order to cancel learning effects. In each condition participants were first instructed in the use of the current navigation interface and conducted seven practice travel tasks. Before conducting the first part of the main experimental task, they were asked to read the task instructions out loud to make sure that they fully understood the current task. Before starting TunnelTask, they also performed six practice tasks collecting three targets in the tunnel. To prevent collisions, the interfaces did not let users jump or fly into any objects. AutoTeleport prevented this automatically. If the user tried to jump into an object with ManualTeleport, the interface ignored the action and played an error sound. Similarly, the interface played an error sound and ignored the action when the users tried to fly into an object using AutoFlying. After finishing both parts of the experimental task, participants were asked to fill out a simulator sickness (SSQ) [20] and NASA task load index (NASA-TLX) [18] questionnaire. Then they experienced the other two experimental conditions in sequence. Users then completed a post-task questionnaire. All techniques had identical settings for the view parameters. The average study duration was about 90 minutes.

## 4.6 Result

We analyzed the collected data for task completion time and task load of all three methods. We conducted inferential analysis using repeated-measures ANOVA with $\alpha = 0.05$ in R.

*4.6.1 Task Completion Time.* For PlanetTask, Mauchly's test showed that the sphericity assumption was violated for *Technique*. Therefore, we adjusted the degree of freedom using Greenhouse-Geisser correction ($\epsilon = 0.81$). The one-way ANOVA identified a significant difference on task completion time between control methods ($F(1.62, 22.71) = 30.69, p < .0001$). Post-hoc analysis with Tukey-HSD revealed that the ManualTeleport method was significantly slower than AutoTeleport ($p < .0001$) and AutoFlying ($p < .0001$), while AutoTeleport and AutoFlying were not different from each other ($p = 0.87$) (Figure 5). The ANOVA for TunnelTask revealed that the task completion times between control methods was again significantly different ($F(2, 28) = 30.69, p < .0001$). Unlike the first task, pairwise comparison showed that the AutoTeleport method was not only significantly faster than ManualTeleport (p < .0001), but also than AutoFlying (p < .05). AutoFlying and ManualTeleport were significantly different from each other (p < 0.001).

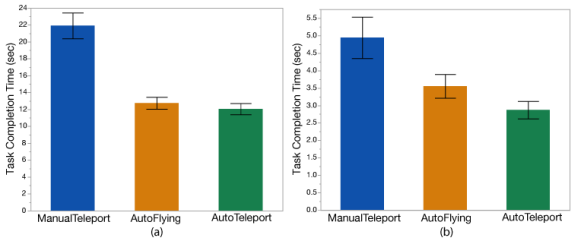


**Figure 5: (a) In PlanetTask, AutoTeleport reduced task completion time significantly compared with ManualTeleport, but there was no difference from AutoFlying. (b) However, in TunnelTask, AutoTeleport significantly reduced completion time compared with the others. Error bars show 95% confidence intervals.**

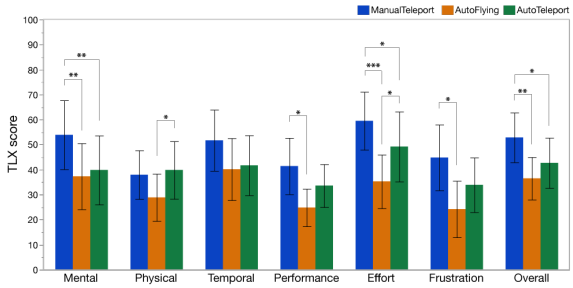


**Figure 6: There were significant differences between *Techniques* for the most categories except for Temporal demand. The overall task load of ManualTeleport was significantly higher than AutoFlying and AutoTeleport. There was no difference between AutoFlying and AutoTeleport. Error bars show 95% confidence intervals.**

*4.6.2 Number of Trigger Presses.* The average number of trigger presses for each control method were: ManualTeleport ($M = 26.09, SD = 13.92$), AutoFlying ($M = 10.13, SD = 7.41$), AutoTeleport ($M = 23.54, SD = 2.01$). An ANOVA identified a significant difference between control methods in terms of trigger presses ($F(2, 28) = 22.329, p < .0001$). Post-hoc analysis with Tukey's HSD revealed that the number of trigger presses for AutoFlying is significantly smaller than AutoTeleport ($p < .0001$) and ManualTeleport ($p < .0001$), but there is no difference between AutoTeleport and ManualTeleport.

*4.6.3 Overshoot.* We detected an overshoot when the dot product of the last and the current travel direction became negative, i.e., when participants reversed their direction. The average number of overshoot occurrences for each control method were: ManualTeleport ($M = 37.8, SD = 6.55$), AutoFlying ($M = 23, SD = 8.94$), AutoTeleport ($M = 21, SD = 6.50$). An ANOVA identified a significant difference between control methods in terms of occurrences of overshoot ($F(2, 28) = 37.97, p < .0001$). Post-hoc analysis with Tukey's HSD revealed that the number of overshoot occurrences of ManualTeleport is significantly higher than AutoFlying ($p < .0001$) and AutoTeleport ($p < .0001$), but there is no difference between AutoFlying and AutoTeleportFor the most of the TLX sub-categories except for Temporal demand, there were significant difference between *Technique* (Figure 7).

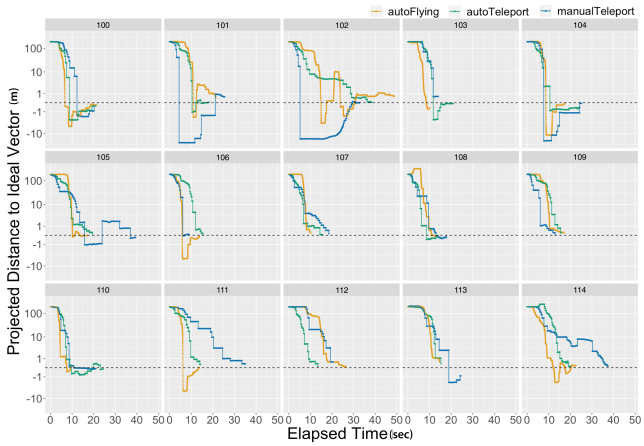


**Figure 7: The graphs show each participant's trajectory with projected distance for conducting task #8 in PlanetTask: going to the south pole on Uranus from Pluto, which was outside the outer asteroid belt. The number above each graph is the participant ID. To get the projected distance, a position vector from the current target to the user's HMD position was projected onto the ideal vector from the current target position to the previous one. Participants tended to overshoot more with ManualTeleport and AutoFlying than with AutoTeleport.**

*4.6.4 Collision.* The average of collisions for each *Technique* were: ManualTeleport ($M = 0.21, SD = 0.57$), AutoFlying ($M = 0.15, SD = 0.71$), AutoTeleport ($M = 0.14, SD = 0.50$). An ANOVA did not identify a significant difference between *Technique* in terms of the number of collisions ($F(2, 28) = 1.14, p = 0.33$).

*4.6.5 Task Load Index.* The averages of the overall task load of each control method measured with the NASA-TLX questionnaire were: ManualTeleport ($M = 54.00, SD = 18.57$), AutoFlying ($M = 37.53, SD = 15.71$), AutoTeleport ($M = 43.06, SD = 19.38$). An ANOVA identified a significant difference between control methods in terms of overall task load score ($F(2, 28) = 11.77, p < .0001$). Post-hoc analysis with Tukey's HSD revealed that the overall task load of ManualTeleport is significantly higher than AutoFlying ($p < .001$) and AutoTeleport ($p < .03$), but there is no difference between AutoFlying and AutoTeleport (Figure 6).

*4.6.6 Simulator Sickness Questionnaire.* The average SSQ scores of each *Technique* were: ManualTeleport (M = 7.31, SD = 9.81), AutoFlying (M = 13.52, SD = 18.11), and AutoTeleport (M = 7.19, SD = 9.08). For PlanetTask, Mauchly's test of sphericity showed that the assumption was violated for *Technique*. Therefore, we adjusted the degree of freedom using Huynh-Feldt correction ($\epsilon = 0.65$). One-Way ANOVA showed that there was a significant difference on the SSQ score between *Techniques* (F(1.31,18.31)=4.06, p<0.05). Post-hoc analysis with Tukey's HSD revealed that the SSQ score of AutoFlying was significantly higher than AutoTeleport ($p < .05$), and higher than ManualTeleport but not significantly so ($p = 0.05$). There was no significant difference between ManualTeleport and AutoTeleport.

*4.6.7 Post-Task Questionnaire Response.* Nine participants preferred AutoFlying to ManualTeleport and AutoTeleport. Six of them explained that the method was *"easy to use"*, and four answered that it was *"comfortable"*. Several explained why AutoFlying was the easiest. P100 explained that *"Flying was smoother and more realistic and familiar."* P108 described using AutoFlying as having *"Not much physical and mental effort and less frustration. It is a lot more comfortable than other methods because I get a feeling of control."* Those who did not prefer AutoFlying explained that the method *"causes nausea"* (P109), *"motion sickness"* (P112) when the participant went through cluttered spaces, and that it was difficult to get used to it because of *"abrupt speed changes"* (P107) so it *"speeds up too fast, and [slows] down too much"* (P111).

Three participants reported that they preferred AutoTeleport. Most of them explained the reason was that it was more *"easy to use"* than the others. P113 described AutoTeleport as *"didn't have to think too much by looking at the distance above."* P114 commented that *"it gives me more confidence to get me where I wanted."* Five participants who did not prefer AutoTeleport responded that it was physically *"tiring"* and required multiple presses to get to the intended position. Three participants commented about the discontinuity of travel with point-and-teleport. P103 described the method as *"jumpy"* and P101 described it *"didn't flow as smoothly."*

Three participants preferred ManualTeleport. They explained that *"I had control over my decisions and preferences"* (P107), and *"I prefer to have the level of control"* (P109). Of those that did not prefer ManualTeleport, six said it was demanding and difficult to use. The other six mentioned they had difficulty estimating the distance with ManualTeleport and overshot the targets frequently.

## 4.7 Discussion

Our study showed that AutoTeleport reduced overshooting and enabled traveling a large distance as quickly and efficiently as one of the prominent methods, AutoFlying. Users traversed small tunnels more quickly than either ManualTeleport or AutoFlying. Our approach reduced the under- and overshooting associated with manual control, which delayed completing tasks. AutoTeleport also reduced the overall task load compared to ManualTeleport.

Manual distance control methods are inherently prone to errors because humans' ability to estimate distance is limited, which is even worse in a VE [44]. Teleporting with ManualTeleport incurred frequent overshooting because participants could not estimate the distance or control it precisely. AutoFlying was also worse than AutoTeleport in this regard, because it was challenging to steer to the target when flying speed was too fast. While the number of overshoot occurrences was different between *Techniques*, there was no significant difference in the number of collisions.

Manual control was even more challenging when the destination was so far away that the teleportation destination marker (in this study, the green sphere) became invisible and the user had only the distance indicator to rely on. This limitation could be alleviated by providing additional visual guidance and widgets to enhance user's distance estimation [16, 39] and performance on selecting distant objects [29]. These issues remain to be studied in future work.

However, our method has a few limitations. First, the automatically computed distance might not always match the distance

the user wants to travel to achieve their navigation goal. For instance, when there is no object close to the desired viewpoint, AutoTeleport can compute a longer-than-desired distance. Also, when moving away from an object, the gradually-increasing distance means that several teleports are often needed to travel large distances. Manual control can require fewer teleports if the user can judge the distance well. Our results show that with ManualTeleport four participants managed to move away faster and also got closer to the next target more quickly and with fewer teleports. However, two of them (P101, P109) also overshot afterwards, which negated the benefits of the manual control. This issue could be addressed by using a hybrid method that affords both automatic and manual control, i.e., a method that lets the user freely adjust the distance if the computed one does not match the user's intent.

Furthermore, most participants preferred AutoFlying to the two point-and-teleport techniques due to its easy control and continuous movement. In a previous study, steering-based techniques like flying were shown to improve spatial updating because users continuously perceive the scene while travelling [42]. Thus, we incorporated optical flow cues [4] into AutoTeleport , and conducted a second study to see whether the cues had a positive effect. The environment was designed to be more complex and cluttered, to increase the potential effect of optical flow, which tends to be small in vast open spaces. We also added a feature that automatically repeats teleports to mitigate the physical fatigue from repeated trigger presses in the teleport techniques.

# 5 USER STUDY 2: LOCOMOTION WITH AUTOMATIC PARAMETER CONTROL IN A DENSE VR

In contrast to the vast open and sparse environment in Study 1, Study 2 was conducted in a dense and closed VE. We compared the original point-and-teleport technique with automatic distance control to a variant with optical flow cues [4] to see if such cues have the same effect for path integration that the flying technique affords in a complex multiscale VE. To address the issue that users in the first study found it tiring to do multiple presses in point-and-teleport, we added automatic teleport repeats as long as the user held the trigger button down. Here, we call the new point-and-teleport with optical flow AutoDash, and the original AutoTeleport. The VE had four rooms with different levels of scale and each room had two parts: a maze-like corridor and a room cluttered with different objects.

## 5.1 Design

This study used as within-subject design with two independent variables, the navigation *Technique* and the level of *Scale*. *Technique* had three values: AutoTeleport, AutoDash, and AutoFlying. As long as the touchpad was held down, Automatic triggering repeated the teleportation at short intervals. This interval was customized by each participant during the training ($M = 0.12$, $SD = 0.03$). The movement speed of AutoDash was set inversely proportional to this custom interval, as the user dashed faster with a smaller interval and slower with a larger one. *Scale* had four levels: 1, 1/8, 1/64, and 1/512. The order of levels of *Scale* was randomized while *Techniques* were counter-balanced with a Latin square.

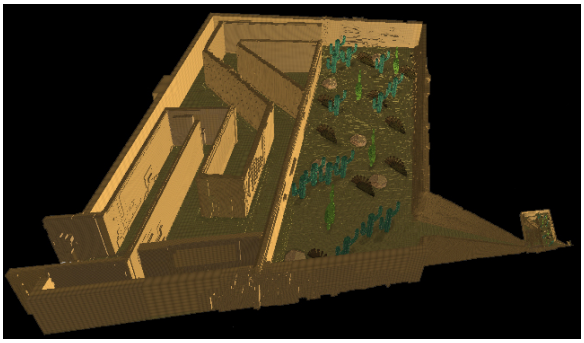

**Figure 8: In User Study 2, the VE had four rooms with the same structure and object layout but decreasing scale levels (1:1, 1:8, 1:64, 1:512).**

## 5.2 Participants

We recruited 12 participants for the study. Since recruitment was limited due to COVID-19, we used participants from our lab ($n = 3$) and acquaintances ($n = 9$). Seven participated remotely with their own VR setup: three had a full set of equipment with an HMD with a controller and two lighthouses, and four had the HMD with one controller and one lighthouse. People who had only one lighthouse were asked to set it up the lighthouse in a way that minimized tracking issues during the experiment. The average age was 29.3 years (SD = 2.31 years). Four participants were female and eight male. All were right-handed. Ten participants had experience with VR systems, and seven had logged 20+ hours. They were compensated with 15 Canadian dollars for their participation.

## 5.3 Apparatus and Environment

The hardware setup and input space for in-person experiments were identical as the first user study. The VE design and the navigation tasks were inspired by previous work on navigation [1, 33], with an additional room at an even smaller scale. The VE had thus four rooms at different scales, each with the same structure and object layout (Figure 8). The first part of each room was a maze-like corridor. In the second part, rather than using simple 3D primitives as obstacles, we made objects (two types of cacti, plants, and rocks) with more complex structures and textures to make it easier to identify any effects of the optical flow cues. All techniques required only one-handed control—pointing the controller in a direction and using the touchpad to execute teleportation or flying.

## 5.4 Task

The task was to collect a set of red spherical targets (Figure 9) as quickly as possible, using AutoTeleport, AutoDash or AutoFlying, as in previous work [1, 33]. A target was "collected" when the user put the top of the controller into it. This was different from Study 1, which used HMD position, because it was not feasible to put one's head into the target at the smallest level of scale. Participants were asked to take the shortest path possible. Targets were in mid-air and at different vertical heights and distances to the nearest surface, forcing participants to use unconstrained-viewpoint navigation to change their altitude continuously during the task.

## 5.5 Procedure

At the start, participants filled out a consent form and completed a pre-questionnaire. During the experiment they were encouraged

**Figure 9: Similar to Study 1, targets appeared as red spheres and the green direction indicator pointed to the current target to reduce disorientation.**

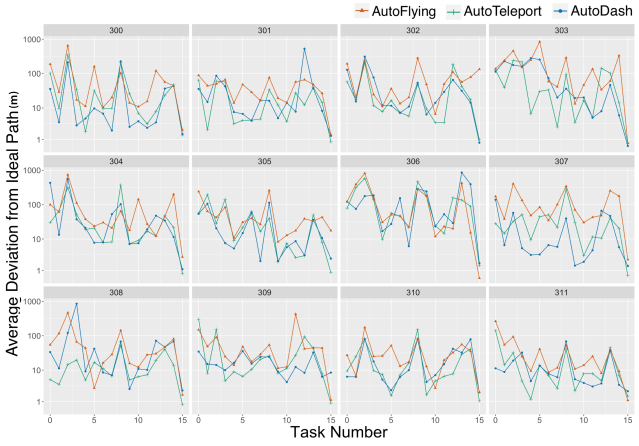


**Figure 10: The graphs show the average deviation of the viewpoint from the ideal path at scale 1 for each participant. The value was computed as the average distance between the participant's position and the line connecting the previous and current target for the task. The top number of each graph is the participant ID, the X-axis is the task number, and the Y-axis is in meters at a logarithmic scale.**

to stay in one place but to freely turn around. They were asked to use the controller with their dominant hand. At the beginning of each condition, there was a practice session with 20 targets to acquire. During this session, they were asked to set the interval for automatically triggering repeated teleportation. In the actual session, 15 targets were in each room, 60 targets in total. Participants filled a Simulator Sickness Questionnaire before and after each condition, and a NASA-TLX questionnaire after each condition. After the experiment, they filled a post-questionnaire regarding their preference on techniques and the reasons for their preference.

## 5.6 Results

*5.6.1 Task completion time.* The assumption of sphericity was violated for *Scale*. We used Greenhouse-Geisser correction ($\epsilon = 0.96$ for *Technique*, $\epsilon = 0.53$ for *Scale*, and $\epsilon = 0.66$ for *Technique* × *Scale*). ANOVA of *Technique* and *Scale* versus Time found no significant effects on task completion time for *Technique* ($F(1.92, 21.14) = 2.64, p = 0.09, \eta^2 = 0.04$) nor *Scale* ($F(1.60, 17.61) = 3.50, p = 0.06, \eta^2 = 0.11$). There was no interaction between *Technique* and *Scale* ($F(3.95, 43.43) = 0.41, p = 0.80$).

*5.6.2 Task Load Index.* We averaged the NASA-TLX score across all scales to analyze task load for each *Technique*. The averages of the overall task load of each *Technique* were: AutoTeleport (M = 26.64, SD = 25.67), AutoDash (M = 25.69, SD = 19.45), and AutoFlying (M = 28.21,SD = 24.05). ANOVA found no effect of *Technique* on the TLX score ($F(2, 22) = 0.22, p = 0.80, \eta^2 = 0.002$).

*5.6.3 Simulator Sickness Questionnaire.* Similar to the TLX, we averaged SSQ ratings of each *Technique* across all scales. The average SSQ score of each *Technique* was: AutoTeleport (M = 17.84, SD = 14.54), AutoDash (M = 10.83, SD = 13.12), and AutoFlying (M = 17.84, SD = 22.77). ANOVA identified no significant difference on the SSQ score between *Techniques* ($F(2, 22) = 2.48, p = 0.10$).

*5.6.4 Average Deviation from Ideal Path.* To measure the navigation accuracy for each *Technique*, we computed the average deviation from the ideal path (i.e. the shortest path from the current to the next target) for each task. Mauchly's test of sphericity was violated for *Scale* and *Technique* × *Scale*. We used Greenhouse-Geisser correction ($\epsilon = 0.88$ for *Technique*, $\epsilon = 0.34$ for *Scale*, and $\epsilon = 0.31$ for *Technique* × *Scale*). The two-way ANOVA of *Technique* and *Scale* versus average distance showed that both *Technique* ($F(1.75, 19.26) = 9.89, p < 0.01, \eta^2 = 0.05$) and *Scale* ($F(1.00, 11.05) = 33.40, p < 0.001, \eta^2 = 0.63$) had significant effects on average distance to the ideal travel path. There was an interaction between *Technique* and *Scale* ($F(1.87, 20.55) = 9.98, p < 0.001$). Post-hoc tests with Tukey-HSD revealed significant differences between techniques at scale 1 (AutoTeleport: M = 48.38 *m*, AutoDash: M = 56.40 *m*, AutoFlying: M = 89.68 *m*). There was no difference between AutoTeleport and AutoDash, but AutoFlying was different from AutoTeleport ($p < 0.0001$) and AutoDash($p < 0.0001$). The deviation from the shortest path of the participants' trajectories with AutoFlying was larger than for AutoTeleport and AutoDash (Figure 10). There was no significant difference at the other three levels of scale.

*5.6.5 Post-questionnaire Response.* Half of the participants ($n = 6$) preferred AutoDash, and the other half preferred ($n = 6$) AutoFlying. Participants who preferred AutoDash explained that the technique *"made it easier to reach the target"* and *"prevents overshoot"*, while incurring *"less motion sickness."* P310 described the experience with AutoDash as *"It provided a semblance of flying without the problem of overshooting."* P311 stated that *"I had more control with AutoDash than AutoTeleport and had less dizziness than AutoFlying."* Three participants who did not like AutoDash explained that the technique was *"hard to adapt [to]."* People who preferred AutoFlying mentioned that it was more *"natural"* and *"easy to use."* Six who did not like it reported that it incurred *"dizziness"* and *"motion sickness."* Three of them explained that it was hard to reach a nearby target using AutoFlying. P301 described that is was *"hard to control near the goal target."* P311 reported *"some dizziness in my head when changing direction"* close to the target.

Six participants did not prefer AutoTeleport due to the optical discontinuity that made it difficult to track the relative position of the target. They stated that they *"missed the target."* P301 explained

it was *"tiring and confusing"* and even felt *"nausea because of [discontinuous] rendering."* P310 described the technique as *"too jittery"* and P311 responded that it created *"more temporal/mental load to locate my position and figure out the next target."*

### 5.7 Discussion

While there was no significant difference in task completion time and work load, AutoDash was preferred to AutoTeleport, since it let users reach targets more easily, likely through the continuous visual updates. Moreover, the technique enabled users to travel with less trajectory deviation from the ideal path than AutoFlying.

The results of our study suggest that in a dense VE, point-and-teleport techniques with automatic distance control enable users to travel more accurately and closer to an ideal path than flying with automatic speed control. Due to the nature of steering-based techniques, users' trajectories with AutoFlying had the largest deviations from the ideal path. Flying techniques are known for being difficult to control [28], which may even get worse when the speed is too high. At the largest scale (1), the automatic speed was too high, which led to deviations from the ideal path, in turn requiring users to turn their heads more to track the target. This might be the cause of the increase in simulator sickness [19].

The task and environment did not induce substantial amounts of simulator sickness. Even though the questionnaire results did not reveal any significant difference between teleportation techniques and flying, four participants reported that AutoFlying was motion-sickness-inducing and made them feel exhausted. Further studies with a larger participant pool might reveal a statistical difference here. In regards to the preference, while half of the participants chose AutoDash due to its comfort and accuracy, the other half chose AutoFlying due to the familiarity of interaction.

## 6 GENERAL DISCUSSION

In this section, we discuss the advantages, disadvantages, and possible use cases for each automatic parameter controlled technique in multiscale VEs.

### 6.1 Point-and-Teleport with Automatic Distance Control

Point-and-teleport with automatic distance control combines the known benefits of teleportation in reducing simulator sickness with the usefulness of automatic speed control to facilitate navigation in large multiscale VEs. It removes the need to manually specify the teleport distance, reducing the overall task load required of users. Through optical flow cues, it improves the user's path integration, which flying techniques also afford.

However, the cost of constructing the distance field can require more computation than other approaches, especially for dynamic scenes. This can be mitigated through methods to update the distance field on the fly using a distributed approach [37]. Another potential disadvantage is that point-and-teleport with automatic distance control is unfamiliar to users. More experience might be needed before the technique achieves its full potential.

Beside the scenarios we explored, other use cases include travelling through a space filled with a great number of small particles as might be generated by a simulation, and exploring an internal structure with many levels of scale, such as a human body with elements modeled at the cell level or an airplane with small screws.

### 6.2 Flying with Automatic Speed Control

Flying with automatic speed control also facilitates multiscale navigation. It prevents most over- and undershooting in multiscale VEs, enabling users to quickly approach targets. Since steering-based locomotion is familiar through its use in VR applications, users can quickly adapt to the addition of automatic speed control.

However, when the level of scale of surrounding environment changes quickly, flying speed also changes abruptly. Also, in a large scale environment the computed speed can be too high, which might make users struggle to reach targets. Both of these increase the potential for navigation errors and frustration. To recover from such errors, they must look around more frequently, which might incur simulator sickness.

Flying with automatic speed control might be more appropriate for travelling in VEs with simple structures or where scale changes gradually. Examples include tunnel structures, such as a model of a lung with different vessel sizes or a passage with varying width in a model of a large power plant.

## 7 CONCLUSION

Our major contribution is a new automatic distance control method for 3D teleportation, enabling users to easily and efficiently travel both to distant places or through small tunnels in multiscale virtual environments. Unlike previous work, our method does not require heavy real-time computation and can be used on many systems, including portable ones. We also showed the benefits and drawbacks of automatic distance control for teleportation through user studies. In Study 1, our method outperformed manual distance control in terms of task completion time, travel accuracy, and task load. Moreover, our method enabled users to travel faster in a small tunnel than automatic speed control for flying, while significantly reducing simulator sickness. In Study 2, point-and-teleport with our method let users travel more closely to the ideal path, i.e., users were more accurate in their navigation. Modifying our method by adding optical flow and automatic repeat made it more popular with the users. Several open questions remain. First, we plan to investigate faster methods to compute the distance field. Our current, unoptimized implementation takes five minutes, which is only usable for precomputation in static environments. We plan to investigate other approaches. When navigating a narrow twisted path like a maze, our current implementation dramatically decreases the teleport distance because walls are always close to the user. We plan to address this with different weights for different directions relative to the view direction. We also plan to investigate the automatic adaptation of the user's scale based on the distance field of the scene. Finally we would like to explore the effects of combining different locomotion techniques in multiscale environments.

## ACKNOWLEDGMENTS

We thank the artist Jini Kwon for creating great environments for the experiments and designing the main figure. We thank Adobe Research for supporting the project.

## REFERENCES

[1] Ferran Argelaguet. 2014. Adaptive navigation for virtual environments. In *2014 IEEE Symposium on 3D User Interfaces (3DUI)*. 123–126. https://doi.org/10.1109/

3DUI.2014.7027325
[2] Ferran Argelaguet and Morgant Maignant. 2016. GiAnt: stereoscopic-compliant multi-scale navigation in VEs. In *Proceedings of the 22nd acm conference on virtual reality software and technology*. 269–277.
[3] Niels H Bakker, Peter O Passenier, and Peter J Werkhoven. 2003. Effects of head-slaved navigation and the use of teleports on spatial orientation in virtual environments. *Human factors* 45, 1 (2003), 160–169.
[4] Jiwan Bhandari, Paul Macneilage, and eelke folmer. 2018. Teleportation without Spatial Disorientation using Optical Flow Cues. In *Proceedings of Graphics Interface 2018*. 153–158.
[5] Benjamin Bolte. 2011. The Jumper Metaphor: An Effective Navigation Technique for Immersive Display Setups. In *Proceedings of Virtual Reality International Conference*, Vol. 1. 2–1.
[6] Doug. A. Bowman, David Koller, and Larry F. Hodges. 1997. Travel in immersive virtual environments: an evaluation of viewpoint motion control techniques. In *Proceedings of IEEE 1997 Annual International Symposium on Virtual Reality*. 45–52. https://doi.org/10.1109/VRAIS.1997.583043
[7] Evren Bozgeyikli, Andrew Raij, Srinivas Katkoori, and Rajiv Dubey. 2016. Point & Teleport Locomotion Technique for Virtual Reality. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '16)*. ACM, New York, NY, USA, 205–216. https://doi.org/10.1145/2967934.2968105
[8] Adam Drogemuller, Andrew Cunningham, James Walsh, Maxime Cordeil, William Ross, and Bruce Thomas. 2018. Evaluating navigation techniques for 3d graph visualizations in virtual reality. In *2018 International Symposium on Big Data Visual and Immersive Analytics (BDVA)*. IEEE, 1–10. https://doi.org/10.1109/BDVA.2018.8533895
[9] Dynamoid. 2016. COSM Worlds. [steamVR]. https://store.steampowered.com/app/467480/C_O_S_M/
[10] Carmine Elvezio, Mengu Sukan, Steven Feiner, and Barbara Tversky. 2017. Travel in large-scale head-worn VR: Pre-oriented teleportation with WIMs and previews. *Proceedings - IEEE Virtual Reality* (2017), 475–476. https://doi.org/10.1109/VR.2017.7892386
[11] George Fitzmaurice, Justin Matejka, Igor Mordatch, Azam Khan, and Gordon Kurtenbach. 2008. Safe 3D navigation. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*. 7–15.
[12] Adrien Fonnet, Florian Melki, Yannick Prié, Fabien Picarougne, and Gregoire Cliquet. 2018. Immersive Data Exploration and Analysis. In *Student Interaction Design Research conference*. Helsinki, Finland. https://hal.archives-ouvertes.fr/hal-01798681
[13] Sebastian Freitag, Dominik Rausch, and Thorsten Kuhlen. 2014. Reorientation in virtual environments using interactive portals. In *2014 IEEE Symposium on 3D User Interfaces (3DUI)*. 119–122. https://doi.org/10.1109/3DUI.2014.6798852
[14] Julian Frommel, Sven Sonntag, and Michael Weber. 2017. Effects of Controller-based Locomotion on Player Experience in a Virtual Reality Exploration Game. In *Proceedings of the 12th International Conference on the Foundations of Digital Games (FDG '17)*. ACM, New York, NY, USA, 30:1–30:6. https://doi.org/10.1145/3102071.3102082
[15] Markus Funk, Florian Müller, Marco Fendrich, Megan Shene, Moritz Kolvenbach, Niclas Dobbertin, Sebastian Günther, and Max Mühlhäuser. 2019. Assessing the Accuracy of Point & Teleport Locomotion with Orientation Indication for Virtual Reality Using Curved Trajectories. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. ACM, New York, NY, USA, 147:1–147:12. https://doi.org/10.1145/3290605.3300377
[16] Chris Furmanski, Ronald Azuma, and Mike Daily. 2002. Augmented-reality visualizations guided by cognition: Perceptual heuristics for combining visible and obscured information. In *Proceedings. International Symposium on Mixed and Augmented Reality*. IEEE, 215–320. https://doi.org/10.1109/ISMAR.2002.1115091
[17] Google. 2016. Google Earth VR. [steamVR]. https://arvr.google.com/earth
[18] Sandra G Hart. 1986. NASA Task load Index (TLX). Volume 1.0; Paper and pencil package. (1986).
[19] Ping Hu, Qi Sun, Piotr Didyk, Li-Yi Wei, and Arie E Kaufman. 2019. Reducing simulator sickness with perceptual camera control. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–12. https://doi.org/10.1145/3355089.3356490
[20] Robert S. Kennedy, Norman E. Lane, Kevin S. Berbaum, and Michael G. Lilienthal. 1993. Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness. *The International Journal of Aviation Psychology* 3, 3 (July 1993), 203–220. https://doi.org/10.1207/s15327108ijap0303_3
[21] Yeojin Kim, Byungmoon Kim, and Young J. Kim. 2018. Dynamic Deep Octree for High-resolution Volumetric Painting in Virtual Reality. *Computer Graphics Forum* 37, 7 (2018), 179–190. https://doi.org/10.1111/cgf.13558
[22] James Liu, Hirav Parekh, Majed Al-Zayer, and Eelke Folmer. 2018. Increasing Walking in VR Using Redirected Teleportation. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology (UIST '18)*. ACM, New York, NY, USA, 521–529. https://doi.org/10.1145/3242587.3242601
[23] The Body VR LLC. 2016. The Body VR. [steamVR]. https://thebodyvr.com
[24] Jack M. Loomis, Roberta L. Klatzky, and Reginald G. Golledge. 2001. Navigating without vision: basic and applied research. *Optometry and vision science : official publication of the American Academy of Optometry* 78, 5 (2001), 282–289. https://doi.org/10.1097/00006324-200105000-00011
[25] Mayra D. Barrera Machuca, Wolfgang Stuerzlinger, and Paul Asente. 2019. Smart3DGuides: Making Unconstrained Immersive 3D Drawing More Accurate. In *25th ACM Symposium on Virtual Reality Software and Technology* (Parramatta, NSW, Australia) *(VRST '19)*. Association for Computing Machinery, New York, NY, USA, Article 37, 13 pages. https://doi.org/10.1145/3359996.3364254
[26] Jock D. Mackinlay, Stuart K. Card, and George G. Robertson. 1990. Rapid Controlled Movement Through a Virtual 3D Workspace. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '90)*. ACM, New York, NY, USA, 171–176. https://doi.org/10.1145/97879.97898
[27] James McCrae, Igor Mordatch, Michael Glueck, and Azam Khan. 2009. Multiscale 3D navigation. *Proceedings of the 2009 symposium on Interactive 3D graphics and games - I3D '09* (2009), 7–14. https://doi.org/10.1145/1507149.1507151
[28] Daniel Medeiros, Antönio Sousa, Alberto Raposo, and Joaquim Jorge. 2019. Magic Carpet: Interaction Fidelity for Flying in VR. *IEEE transactions on visualization and computer graphics* (2019). http://dx.doi.org/10.1109/TVCG.2019.2905200
[29] Daniel Mendes, Daniel Medeiros, Eduardo Cordeiro, Maurício Sousa, Alfredo Ferreira, and Joaquim Jorge. 2017. PRECIOUS! Out-of-reach selection using iterative refinement in VR. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*. IEEE, 237–238. doi.org/10.1109/3DUI.2017.7893359
[30] Roberto A Montano Murillo, Elia Gatti, Miguel Oliver Segovia, Marianna Obrist, Jose P Molina Masso, and Diego Martinez Plasencia. 2017. NaviFields: Relevance fields for adaptive VR navigation. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. 747–758. doi.org/10.1145/3126594.3126645
[31] D. R. Montello. 2001. International encyclopedia of the social & behavioral sciences. In *International encyclopedia of the social & behavioral sciences*, Neil J. Smelser and Paul B. Baltes (Eds.). Elsevier, 14771–14775.
[32] Mtschoen-unity. 2019. EditorXR. https://github.com/Unity-Technologies/EditorXR
[33] Domi Papoi and Wolfgang Stuerzlinger. 2019. Improved Automatic Speed Control for 3D Navigation. In *2019 Computer Graphics International (CGI)*. 278––290. https://doi.org/10.1007/978-3-030-22514-8_23
[34] Christopher Scharver, James Patton, Robert Kenyon, and Eric Kersten. 2005. Comparing adaptation of constrained and unconstrained movements in three dimensions. In *9th International Conference on Rehabilitation Robotics, 2005. ICORR 2005.* 434–439. https://doi.org/10.1109/ICORR.2005.1501136
[35] Bethesda Softworks. 2017. Fallout 4 VR. [steamVR].
[36] Bethesda Softworks. 2018. The Elder Scrolls V: Skyrim VR. [steamVR].
[37] Henrique Taunay, Daniel Medeiros, and Alberto Raposo. 2019. A Distributed Approach for Automatic Speed Adjustment during Navigation in 3D Multiscale Virtual Environments. In *2019 21st Symposium on Virtual and Augmented Reality (SVR)*. IEEE, 140–146.
[38] Daniel Ribeiro Trindade and Alberto Barbosa Raposo. 2014. Improving 3D navigation techniques in multiscale environments: a cubemap-based approach. *Multimedia Tools and Applications* 73, 2 (Nov. 2014), 939–959. https://doi.org/10.1007/s11042-012-1127-8
[39] Kengo Uratani, Takashi Machida, Kiyoshi Kiyokawa, and Haruo Takemura. 2005. A study of depth visualization techniques for virtual annotations in augmented reality. In *IEEE Proceedings. VR 2005. Virtual Reality, 2005.* IEEE, 295–296. http://dx.doi.org/10.1109/VR.2005.1492802
[40] Orbital Views. 2018. Overview. [steamVR]. https://www.overviewexperience.com
[41] Colin Ware and Daniel Fleet. 1997. Context Sensitive Flying Interface. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics (I3D '97)*. ACM, New York, NY, USA, 127–130. https://doi.org/10.1145/253284.253319
[42] Tim Weißker, André Kunert, Bernd Fröhlich, and Alexander Kulik. 2018. Spatial updating and simulator sickness during steering and jumping in immersive virtual environments. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 97–104. http://dx.doi.org/10.1109/VR.2018.8446620
[43] Jan M Wiener, Simon J Büchner, and Christoph Hölscher. 2009. Taxonomy of human wayfinding tasks: A knowledge-based approach. *Spatial Cognition & Computation* 9, 2 (2009), 152–165.
[44] Bob G. Witmer and Paul B. Kline. 1998. Judging Perceived and Traversed Distance in Virtual Environments. *Presence: Teleoperators and Virtual Environments* 7, 2 (April 1998), 144–167. https://doi.org/10.1162/105474698565640
[45] Mengxin Xu, Maria Murcia-López, and Anthony Steed. 2017. Object location memory error in virtual and real environments. In *2017 IEEE Virtual Reality (VR)*. 315–316. https://doi.org/10.1109/VR.2017.7892303