Assisted Travel Based on Common Visibility and Navigation Meshes

Sebastian Freitag* Benjamin Weyers* Torsten W. Kuhlen*

Visual Computing Institute, RWTH Aachen University, Germany — JARA-HPC, Aachen, Germany

Abstract

The manual adjustment of travel speed to cover medium or large distances in virtual environments may increase cognitive load, and manual travel at high speeds can lead to cybersickness due to inaccurate steering. In this work, we present an approach to quickly pass regions where the environment does not change much, using automated suggestions based on the computation of common visibility. In a user study, we show that our method can reduce cybersickness when compared with manual speed control.

Index Terms: I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

1 Introduction

In many larger virtual environments, many regions are mostly empty, not modeled in detail, or simply of no importance to the user, such that users often have to travel through less interesting areas. If steering-based travel techniques are used, this either leads to longer travel times or the necessity to increase the travel speed. However, changing the travel speed manually requires the user to control an additional parameter (e.g., using a menu), potentially increasing cognitive load and error rates [3]. Therefore, several methods to automatically adapt travel speed have been proposed. While it is common to modify the speed based on the distance to the environment (e.g., [5]), this is impractical in ground-based interfaces, where the distance to the scene geometry is largely constant. Alternatively, it has been proposed to adapt travel speed inversely proportional to the viewpoint quality (a measure of the informativeness of a viewpoint) at the user's location [3]. However, this may lead to speed changes intransparent to the user. Furthermore, faster travel speeds are in general harder for the user to control and may lead to errors (e.g., overshooting the target) or cybersickness due to frequent direction changes caused by inaccurate steering [4].

Therefore, in this work, we propose an alternative ground-based travel approach, inspired by the observation that the parts of a path where the environment does not change significantly are often the ones that users want to pass quickly. These regions can be determined by finding areas of *common visibility*, i.e., areas in which the visible parts of the scene are similar. The proposed interface suggests to quickly move through these regions on a straight line to avoid errors and reduce cybersickness associated with changes in direction and acceleration. We evaluated our approach by conducting a formal user study.

2 Computation of Common Visibility

The visibility of the scene from a position can be represented by a *visibility histogram* $a = (a_1, ..., a_n)$, where the bin a_i represents the visual size of the i-th object in the scene, i.e., its area when projected onto a unit sphere around that position [2]. These histograms can be approximated by rendering a cube map of the scene at the viewpoint into an item buffer and counting the pixel area of each object [2]. To automatically determine the objects in a scene, we use the definition from [2], identifying all basic geometries as individual objects.

*e-mail: {freitag | weyers | kuhlen}@vr.rwth-aachen.de

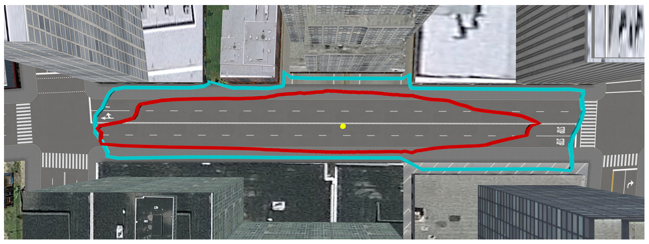
Figure 1: The region of common visibility of the yellow position in a city, with (cyan) and without (red) capping of the visual size of objects.

However, we additionally split object meshes into their connected components, considering each a separate object, to avoid that mesh parts with the same material across the scene (that may be stored as a single geometry) are recognized as a single large object.

Although the calculation of visibility is often too computationally expensive to be computed online, it is possible to determine visibility information by sampling the region of interest in a precomputation step and interpolate between these samples at runtime. To represent the region of interest for our method (navigable ground), we use *navigation meshes* (also called *navmeshes*). These encode the navigable surface of a virtual scene as a polygonal mesh, and are commonly used for automated pathfinding through 3D scenes. We use a navmesh to sample visibility information, using a default eye height above the vertices of the navmesh as sample points. Then, to compute the visibility histogram at any position in the navigable area at runtime, we determine the corresponding navmesh triangle and interpolate between the histograms determined at its vertices. As obstacles are implicitly encoded in the structure of the navmesh, this interpolation is never performed through an obstacle, avoiding large interpolation errors.

However, navmesh vertices may be far apart, reducing the accuracy of the interpolation. Therefore, we refine the navmesh when the visibility information at adjacent vertices differs too much. For each edge, we compute the *histogram intersection* $HI(a,b) = \sum_i \min(a_i, b_i) \in [0,1]$ of the (normalized) visibility histograms a and b of its vertices. If $HI(a,b) < \theta$ for some threshold $\theta \in [0,1)$, an edge split is performed, inserting a new vertex at the edge's midpoint and computing the visibility for the new position. In this work, we use $\theta = 0.8$, as this ensured low interpolation errors on our test scenes.

We define the *region of common visibility* for a position p with visibility histogram v_p to be comprised of all positions q where half of the visible parts of the scene is identical to p, i.e., all points with visibility histogram v_q where $HI(v_p, v_q) \geq 0.5$. To determine the boundary of the region of common visibility in some direction, the navmesh is sampled in that direction at every navmesh edge until the histogram intersection drops below 0.5. We then use binary search on the final navmesh triangle to find the exact position where it is 0.5. In addition, we also stop 1 m before the outer edge of the navmesh is reached.

Furthermore, large or close objects (with large visual size) have large entries in the visibility histogram and can therefore dominate the determination of common visibility. Therefore, we cap the visual size of each object at a relative size of 0.025 (about the size of a door at a distance of 2.5 m), distributing the excess on all other objects proportionally to their respective visual size. An example for a region of common visibility computed this way is illustrated in Figure 1.

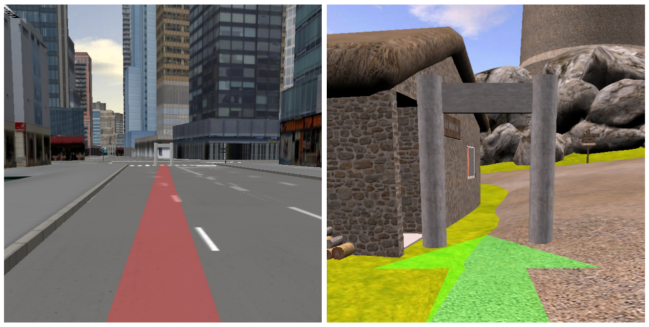
Figure 2: Visual feedback for a target suggestion in a city (left) and countryside (right) scene, indicating high (left) and low (right) speed and distance to the target.

3 Travel Method

Our travel method extends a simple ground-based steering-by-pointing technique, where the user points in the desired direction to go there. While traveling, the system tries to find a travel target candidate by computing the end of the region of common visibility in movement direction. To improve placement, the target position is corrected along the movement direction based on viewpoint quality (we select the position with the highest quality within ±40% of the distance). In addition to selecting more favorable targets, this improves the stability of the target position when it is updated while the user travels.

If the user can save time (at least 2 s) by traveling there quickly, a suggestion is made, visualized by an arrow on the ground and a gate at the target position (Fig. 2). The travel speed is chosen out of a set of discrete speed levels (to provide a better sense of the distance while traveling) so that the target is reached in at most 2 s, and indicated by the arrow color. Upon pressing a button, the user is transported to the target along a straight line. If the terrain on the way is uneven, the user's feet are kept at most 2 m above or 0.5 m below the ground while minimizing the number of (vertical) direction changes.

4 User Study

We conducted a user study to compare our approach (**A**) against a ground-based steering-by-pointing technique where the maximum speed could be changed using a menu (**M**). Both techniques used an ART Flystick 2, regulating the movement speed continuously between 0 and the maximum with the Flystick's joystick. We selected 2 m/s, 8 m/s, 24 m/s and 48 m/s as speed levels for both techniques, using 2 m/s in **A** for manual travel and the faster levels for suggestions.

The study was performed in a 5-sided CAVE system. We imitated a realistic scenario, having users travel medium distances and perform some local task at the target. It consisted of six phases, including one training phase for each technique, and one trial for each technique on each of two different, large scenes (a countryside and a city scene). After being informed about both techniques and the procedure, participants entered the CAVE and performed the training with the first technique, before the first trial with the same technique began. This was followed by the second training phase and the second trial with the second technique, and then trials 3 and 4 on the second scene, before participants left the CAVE to fill out questionnaires.

In each trial, participants had to follow signs to 5 houses, where they solved a simple search task (find a certain object and touch it with the Flystick), before traveling to the next house. When they reached a house, participants were asked to point in the direction of the last house, and the one where they started the trial. Then, they were asked how they were feeling, on a scale of 0 (how they felt before the experiment) to 7 (want to abort the experiment), similar to Fernandes et al.'s *discomfort score* [1]. A trial ended after finishing the fifth search task, after which participants could take a short break. The total procedure took 73 min. on average, 43 of which were spent in the CAVE.

Table 1: User study results.

Variable	**M**: mean	(SD)	**A**: mean	(SD)	p^*
traveled distance (per trial)	1819.4 m	(135.8)	1836.4 m	(206.8)	.584
time (per trial, w/o search)	176.5 s	(94.4)	199.5 s	(58.8)	.101
pointing error (last house)	19.5°	(11.3)	20.8°	(13.2)	.542
pointing error (first house)	43.4°	(21.3)	44.5°	(25.8)	.798
discomfort score	1.9	(1.8)	1.3	(1.4)	**.031**

Question (1: applies more to **M**, 7: more to **A**)	median	IQR	p^{**}
With which method did you reach the target faster?	3	[2,5]	**.014**
Which method was more precise?	2	[1,3]	**.000**
Which method was easier to use?	3	[2,4]	**.019**
Which method caused more dizziness/discomfort?	3	[2,5]	.057
Which method do you prefer overall?	2	[1,5]	**.000**

* independent-samples t-tests ** Wilcoxon signed-rank tests (against 4=neutral)

35 unpaid individuals participated in the study (8 female, 27 male, mean age 27; 9 VR professionals, 13 with previous VR experience, 13 first-time users), 3 of which aborted due to cybersickness. The order of techniques and scenes was counter-balanced, and participants were assigned each order combination balanced by gender and experience.

4.1 Results and Discussion

The effects of the travel technique on different measurements, as well as results from the questionnaires are summarized in Table 1.

Our method caused less cybersickness, as indicated by the discomfort score results (questionnaire results pointed in the same direction, but barely missed significance). The reason for this is probably that participants moved in straight lines, did not do any curves or speed changes while moving, and were less affected by uneven terrain, although some participants commented that the sudden stop when reaching a target made them dizzy. Furthermore, there were no significant differences regarding completion time, traveled distance, or pointing errors, suggesting that objectively, the methods otherwise performed similarly. However, participants preferred **M**, finding it easier to use and more precise. We suspect this to be due to participants feeling limited by the choice made by the target suggestions, as we observed anecdotally that users sometimes ignored well-placed suggestions, waiting for one that suited them better. This might be mitigated by giving users a better understanding of the method, e.g., by explicitly showing the edge of the region of common visibility.

5 Conclusion

We have presented a novel travel method that allows users to quickly move through less interesting environments, based on the computation of regions of common visibility and automatic suggestions. We showed in a user study that the approach reduced cybersickness compared to a technique using manual speed change, although participants still preferred the manual method. In the future, we want to alleviate the method's disadvantages by improving its interface, and by allowing users to choose between different suggestions.

References

[1] A. S. Fernandes and S. K. Feiner. Combating VR Sickness through Subtle Dynamic Field-of-View Modification. In *IEEE Symp. on 3D User Interfaces (3DUI)*, pages 201–210, 2016.

[2] S. Freitag, B. Weyers, A. Bönsch, and T. W. Kuhlen. Comparison and Evaluation of Viewpoint Quality Estimation Algorithms for Immersive Virtual Environments. In *ICAT-EGVE 2015*, pages 53–60, 2015.

[3] S. Freitag, B. Weyers, and T. W. Kuhlen. Automatic Speed Adjustment for Travel through Immersive Virtual Environments based on Viewpoint Quality. In *IEEE Symp. on 3D User Interfaces (3DUI)*, pages 67–70, 2016.

[4] J. J. LaViola Jr. A Discussion of Cybersickness in Virtual Environments. *ACM SIGCHI Bulletin*, 32(1):47–56, 2000.

[5] J. McCrae, I. Mordatch, M. Glueck, and A. Khan. Multiscale 3D Navigation. In *Proc. of the ACM Symposium on Interactive 3D Graphics and Games*, pages 7–14, 2009.