An Enhanced Steering Algorithm for Redirected Walking in Virtual Environments

Mahdi Azmandian* Rhys Yahata* Mark Bolas*† Evan Suma*

*USC Institute for Creative Technologies †USC School of Cinematic Arts

Abstract

Redirected walking techniques enable natural locomotion through immersive virtual environments that are considerably larger than the available real world walking space. However, the most effective strategy for steering the user remains an open question, as most previously presented algorithms simply redirect toward the center of the physical space. In this work, we present a theoretical framework that plans a walking path through a virtual environment and calculates the parameters for combining translation, rotation, and curvature gains such that the user can traverse a series of defined waypoints efficiently based on a utility function. This function minimizes the number of overt reorientations to avoid introducing potential breaks in presence. A notable advantage of this approach is that it leverages knowledge of the layout of both the physical and virtual environments to enhance the steering strategy.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual reality

1 Introduction

For many applications of immersive virtual environments, it is desirable to support interactions that allow users to move their bodies naturally, similar to the way people move in the real world. However, supporting walking is often not feasible because the dimensions of the physical tracked space will ultimately limit the size of the virtual world that may be navigated through natural body movement. To address this problem, researchers have developed *redirected walking*, a technique that manipulates the mapping between physical and virtual motions to steer the user away from the boundaries of the physical space [2].

While redirection introduces a conflict between visual motion and vestibular sensation, these manipulations will ideally remain imperceptible to users so long as they do not exceed human sensitivity thresholds, which previous researchers have measured empirically [3]. Though previous researchers have attempted to make resetting techniques less intrusive by introducing distractors, these techniques typically require a temporary interruption of the user's exploration, which may break presence in the virtual world. Therefore, it is generally advisable to apply redirected walking as efficiently as possible in order to minimize the number of resets required to handle failure cases.

To optimize redirected walking algorithms, recent work has examined different strategies for steering the user during redirection[1][4]. However, these algorithms all rely upon a single general heuristic based on the location of the steering target in physical space. We suggest that a more sophisticated approach considering the architectural layout of the virtual environment would yield an improvement over previous strategies. In this work, we present a method for planning a path through a virtual environment and calculating redirection parameters such that the user can visit a set of defined waypoints with a minimal number of resets. This work may be immediately applicable for applications where these navigation waypoints are known in advance, such as a virtual tour, and may also be extended in the future to determine waypoints dynamically.

*e-mail: {mazmandian, ryahata, bolas, suma}@ict.usc.edu

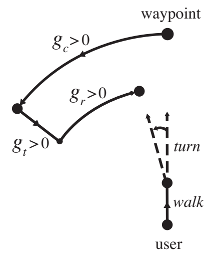


Figure 1: Influence of gains on waypoint translation in physical space during one time step. As the user steps forward, the waypoint rotates due to g_c, moves towards the user due to g_t, and rotates due to g_r. The dashed lines show the user's change in orientation to correct for the waypoint's rotation.

2 Method

Background. Three different types of self-motion gains have been identified in the literature: (1) translation gains, (2) rotation gains, and (3) curvature gains. *Translation gains* involve repositioning the virtual model towards (positive) or away (negative) from the user, increasing or decreasing the perceived displacement in the virtual world. *Rotation gains* involve rotating the virtual model about the user during head rotation, effectively increasing or decreasing the perceived rotation in the virtual world.[2] *Curvature gains* also involve rotating the virtual model about the user, but are instead applied during translation. This is normally applied as a user walks forward towards a point in the virtual world, resulting in a curved path towards the target while perceiving a straight walking path.

Problem. Given a virtual environment M and a physical tracking space S, construct a path consisting of a sequence of waypoints $\{w_i\}_{i=0}^{n}$ in M such that if the user traverses the path he shall remain in S. We expect that the user will navigate to each waypoint in the planned path consecutively. Each segment of the path will use appropriate translation, rotation and curvature gains to constrain the user's path to the bounds of S. If the user is about to leave S, the user's orientation may be reset. We first address how the position and the rotation of the user and the model change when a user moves toward a waypoint. As the user moves through S, the model may be translated and rotated around the user in response to translation, rotation and curvature gains.

Waypath Prediction. Given the user position P_u, user rotation R_u, the model's position and rotation, and the next waypoint's position, calculate the user's and model's position and rotation when the user reaches the waypoint. Assume translation gain g_t, rotation gain g_r

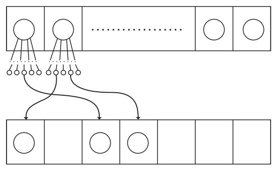


Figure 2: Configuration class representatives in consecutive iterations. The top row represents the input configurations. Each circle is a representative of a configuration class. After the tree is expanded, only the best representative leaf node for each output configuration class is retained for the next iteration.

and curvature gain g_c are applied (g_c is expressed in unit of angle per unit of length) as shown in Figure 1. To address this problem, we derived a closed-form solution that handles resets.

2.1 Tree Search

Knowing how the reorientation techniques affect the model and user, we must decide how to effectively combine them. We now determine what value within the acceptable range each parameter must take to create an "effective" path. The efficacy of a path is gauged by a utility function U. U is defined as the additive inverse of the number of resets triggered along that path, $u(\delta) = -\delta$. This utility function could be refined by including other factors such as number of times each redirection technique was applied.

An exhaustive search is performed to find the most effective solution. In order to constrain the branching factor, we limit each redirection parameter to a finite subset of values within empirically determined thresholds [3]. When a boundary is reached, a reset is triggered, and the reset angle is limited to a finite set of possible values. We construct a tree with the root as the initial configuration. Child nodes are generated based on different options for the gain parameters and reset angles. Each root to leaf path corresponds to a possible solution. By performing a graph search algorithm such as BFS or DFS we can visit the leaf nodes and determine the optimal solution.

If the number of options for translation, rotation and curvature gain and reset angles are o_t, o_r, o_c, and o_d respectively, the depth of the tree will be in the order of $\max\{o_t o_r o_c, o_d\}^{n+\delta max}$ where δmax is the maximum number of resets triggered. To limit the search space, we define a *Configuration* as the pair (P_u, R_u). Although there are infinite P_u and R_u within the model space, not all are "substantially" different. Our approach discretizes the space of possible configurations.

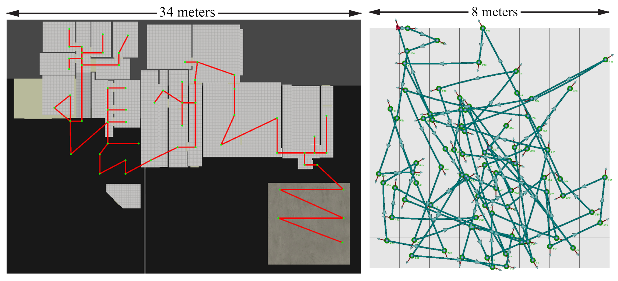
2.2 Configuration Classification

An exhaustive search of the tree expands similar subtrees thus increases memory overhead. To control this, our approach uses configurations as metric with which to prune the tree. A configuration is a discrete position and bearing of the user in S. S is divided into c_p equal-sized cells. Rotations are divided into c_r equal intervals in the range $(-\pi, \pi]$. Two nodes are in the same configuration class if they share the same position cell and rotation interval.

The pruning algorithm groups nodes of the same configuration class and only retains the one with the highest efficacy value based on a utility function (Figure 2). We apply this prune when the tree expansion depth has reached a multiple of the pruning index i_p. By pruning, we guarantee that at every i_p-th level of the tree there will be no more than $c_c = c_r c_p$ nodes.

2.3 Leveraging Architecture

We define a *tour* of a virtual world as a sequence of waypoints in virtual world space manually defined by the developer. An advantage of this manual approach is that architectural limitations are taken into consideration, and it is unnecessary to computationally verify that the path does not pass through walls in the virtual environment. For each room in the virtual environment, we define hard waypoints (points that must be visited) and soft waypoints (optional points), and illegal pairs (waypoint pairs that cannot be reached in succession due to path obstruction). We now construct all sub-paths beginning from the room entrance waypoint and ending with the exit waypoint. Each subpath must contain all hard waypoints, a subset of soft waypoints, and no illegal pairs. To incorporate these sub-paths in the search, we create multiple instances of the graph search and merge the results once the sub-paths converge via the same classification technique.



(a) Path in virtual world (b) Path in real world

Figure 3: Optimized path found in the virtual world and its corresponding path in the real world (tracking area).

3 Discussion and Conclusion

In this paper, we describe a theoretical framework that defines a walking path through a virtual environment and calculates the parameters for applying redirected walking based on a utility function that minimizes potential breaks in presence from overt reorientations. While we have yet to perform formal studies, we have conducted informal tests using a virtual model of a large real world building shown in Figure 3. In doing so, we have observed that combining individual redirection techniques (rotation, curvature, and translation gains) affect each other's behavior in non-trivial ways. For example, the experienced curvature radius is affected by rotation gain, and is no longer strictly dependent on the curvature gain parameter alone. Furthermore, it may drop below the threshold defined in [3], which may cause the redirection to be noticeable. Additionally, to avoid manual defining of waypoints, we can explore automatic path generation techniques. The framework as implemented currently functions as an offline search algorithm and does not dynamically correct for unexpected user behavior. Our broader goal is to extend this work as part of a system that can dynamically optimize virtual and physical paths, while correcting for deviation during run time. Such a solution could also support non-static environments.

References

[1] E. Hodgson and E. Bachmann. Comparing four approaches to generalized redirected walking: simulation and live user data. *IEEE Transactions on Visualization and Computer Graphics*, 19(4):634–643, Apr. 2013.

[2] S. Razzaque, Z. Kohn, and M. C. Whitton. Redirected Walking. In *Eurographics (Short Presentation)*, 2001.

[3] F. Steinicke, G. Bruder, J. Jerald, H. Frenz, and M. Lappe. Estimation of detection thresholds for redirected walking techniques. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):17–27, 2010.

[4] J. Su. Motion Compression for Telepresence Locomotion. *Presence: Teleoperators and Virtual Environments*, 16(4):385–398, Aug. 2007.