Redirected Walking Based on Historical User Walking Data

Cheng-Wei Fan*
Tsinghua University

Sen-Zhe Xu†
Tsinghua University

Peng Yu‡
Tsinghua University

Fang-Lue Zhang§
Victoria University of Wellingtong

Song-Hai Zhang¶
Tsinghua University, BNRist

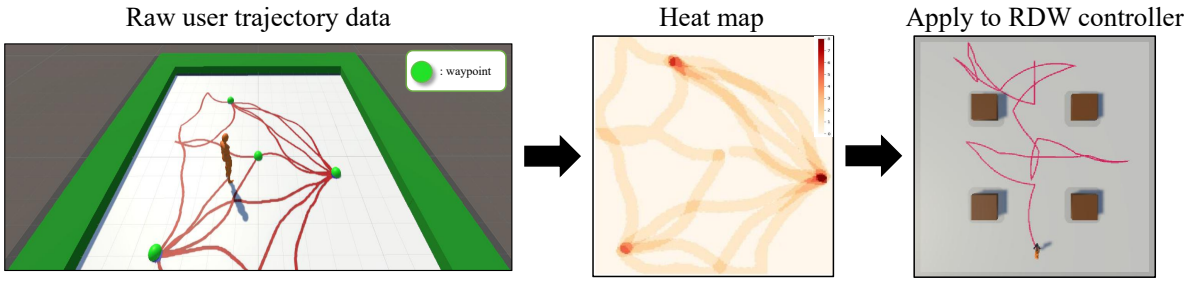


Figure 1: We propose a novel redirected walking method based on historical user walking data. We process the raw user trajectories into a heat map and build a graph-based discrete representation. In the left image, the green balls are the waypoints of the preset simulation paths. Note that although the waypoints are not included in the raw data, they can be used to compare with the waypoints calculated from the heat map. By combining the compact user walking data with RDW controller, we develop a more effective redirected walking algorithm.

Abstract

With redirected walking (RDW) technology, people can explore large virtual worlds in smaller physical spaces. RDW controls the trajectory of the user's walking in the physical space through subtle adjustments, so as to minimize the collision between the user and the physical space. Previous predictive algorithms place constraints on the user's path according to the spatial layouts of the virtual environment and work well when applicable, while reactive algorithms are more general for scenarios involving free exploration or unconstrained movements. However, even in relatively free environments, we can predict the user's walking to a certain extent by analyzing the user's historical walking data, which can help the decision-making of reactive algorithms. This paper proposes a novel RDW method that improves the effect of real-time unrestricted RDW by analyzing and utilizing the user's historical walking data. In this method, the physical space is discretized by considering the user's location and orientation in the physical space. Using the weighted directed graph obtained from the user's historical walking data, we dynamically update the scores of different reachable poses in the physical space during the user's walking. We rank the scores and choose the optimal target position and orientation to guide the user to the best pose. Since simulation experiments have been shown to be effective in many previous RDW studies, we also provide a method to simulate user walking trajectories and generate a dataset. Experiments show that our method outperforms multiple state-of-the-art methods in various environments of different sizes and spatial layouts.

Index Terms: Computing methodologies—Computer graphics—Graphics systems and interfaces—Virtual reality

*e-mail: fcw20@mails.tsinghua.edu.cn
†e-mail: xsz@tsinghua.edu.cn
‡e-mail: anicca97@163.com
§e-mail: fanglue.zhang@vuw.ac.nz
¶e-mail: shz@tsinghua.edu.cn (corresponding author)

1 Introduction

The grand aim of Immersive Virtual Environments (IVE) research is to synthesize sensory information that leads to a high degree of perceived realism of virtual environments. Locomotion, as a fundamental and universal activity performed during interaction within IVEs, will be a frequent everyday activity for more and more people in the coming era of the Metaverse. Early IVE researches performed locomotion with interaction devices, such as joysticks, wands, and mouses. Nevertheless, using these devices to initiate self-motion in virtual environments (VE) provides unnatural input and feedback to the body. Traveling in IVEs by real walking has been considered the most intuitive way to drive virtual motion and improve the sense of presence [25] [31]. To map the often larger virtual world to the physical space, stand-still devices were developed to create a partially realistic walking sensation. However, the haptics of walking in VE is still not fully resolved due to the lack of realism compared to real walking. For maintaining an illusion of walking freely in VE at a low cost, Redirection Walking (RDW) [24] compresses a large virtual world into a significantly smaller physical space, allowing the user to experience a more realistic walk with imperceptible redirection or speed changes in their physical environments (PE).

Despite all the recent advancements, effectively redirecting walking for VR environments with different spatial layouts and complexities still remains challenging due to the following reasons: 1. The infinite possibilities of potential paths to redirect users. Since the extremely large search space brings a huge difficulty in finding the globally optimal redirection solution, reactive methods were proposed to guide users with manually designed constraints, such as physical targets (Steer-to-Center and Steer-to-Multiple-Targets) or a specific circular trajectory (Steer-to-Orbit). However, these methods perform worse than the predictive methods when possible future movements of users can be estimated [18,40], as the manually given rules are not guaranteed to make the best decision when guiding users. 2. The limited generalizability of the existing methods across different VR scenes and physical spaces. Although previous predictive methods work well in their specific environments, their strong assumption on the spatial layouts makes their methods hard to generalize to other scenes.

In this research, we aim to fill that gap by developing a redirection solution that is applicable to various environments and can provide high-quality free walking based on predicting the properties of the future paths. We found that the historical walking data of users in a VE can be a reliable and accessible source for future path analysis and prediction, which is general for all environments regardless of their spatial layouts and complexities. More importantly, the historical walking data for different environments provide essential guidance for the prediction method to fit various environment structures.

We are the first to investigate how to represent and analyze the historical walking data for future walking redirection. We use a discretized representation of the physical space to encode the user poses, including their position and orientation, enabling path estimation. We evaluate all the discrete physical poses and assign a score to represent the priority of each pose by its reachable pose number and the longest distance it could reach. To reduce the errors from discretization, we guide the users to the discrete poses to the greatest extent. We also make a similar discretization for the virtual space. To properly analyze users' historical walking data, irrelevant information, such as individual walking habits, needs to be eliminated. We expand each recorded virtual walking route to a band to cover more nearby poses. Then all the historical walking routes can be summarized in a walking heatmap in the virtual space. Furthermore, a weighted directed graph is built up using the poses covered by the heatmap to help the guidance of users.

Our RDW algorithm is built upon the weighted graph representing the historical walking data. Once a user finishes turning or resetting, we will find a target pose in physical space and guide the user to it by considering the distances and weights of the poses and whether they could cause resets. When the user has to reset on the way from their current pose to the next pose, we increase the weights of these reachable poses. When a reset can be avoided if the user walks to some reachable poses when arriving at the virtual destination, we will increase the weights of those poses. At the same time, we also adopt a different reset strategy compared with previous approaches. Common reset strategies, such as reset-to-center (R2C) and two-one-turn, are versatile but not necessarily effective because they can not make further optimization according to the physical layout. Instead, our algorithm considers all the orientations of the current boundary position and turns the user to the best orientation when a reset happens. It also reduces the interruptions in the user's walking experience.

This paper presents the following contributions to the RDW research:

- A novel RDW algorithm based on the analysis of users' historical walking data. Our algorithm has a better generalizability than predictive algorithms and is more effective than reactive algorithms in various VR applications where users can freely explore.
- A dataset containing simulated user's historical walking data.
- An algorithm to convert the user's historical walking data into a walking heatmap and encode it into a weighted directed graph.
- A user study and extensive simulation experiments comparing the performance of our RDW algorithm with other state-of-the-art RDW algorithms.

2 Related Work

Razzaque [24] is generally credited as the person who built the RDW technology. When the magnitude of the conflict does not exceed the perceptual threshold, human vision dominates vestibular sensation, thereby enabling the user to be unknowingly redirected to walk along a virtual path separate from their physical path. In Razzaque's method, the operation of inducing the user's straight path in the virtual scene into a circular arc path by rotating the virtual scene is called curvature gain. The operation of changing the user's rotation angle in the real space by rotating the virtual scene is called rotation gain. When the user walks in the virtual scene, panning the virtual scene along the user's movement direction or the opposite direction can make him not aware that the distance he walks in the real space has changed. This operation is called translation gain [12].

2.1 Perceptual Thresholds

For the authenticity of the user experience, redirected walking needs to limit the operation of the system to situations where the user does not perceive abnormality. Steinicke et al. [27] showed that in order not to be detected by ordinary users, the rotation gain needs to be greater than 0.67 and less than 1.24, the translation gain needs to be greater than 0.86 and less than 1.26, and the curvature gain needs to have a radius of at least 22.03 meters. Waller et al. [32] have achieved a turning radius of 7.5 meters, but the detection rate is low. The study by Grechkin et al. [8] found that data from both experiments by the constant stimulus method and the Green's maximum likelihood procedure did not show that the curvature gain detection threshold was affected by the presence of translation gain. Neth et al. [19] demonstrated a correlation between a user's physical speed and curvature gain detection, i.e., as users walked faster, their curvature gain detection threshold was lower. Hutton et al. [11] focused on individual differences in tolerance and susceptibility to redirection and proposed a method to quickly and accurately calculate the rotation gain threshold for a single user. Williams et al. [34] demonstrated that perception threshold is influenced by gender, visual field, and confounding factors, finding a strong correlation between simulator disease score and threshold gain. There has also been some work trying to apply rotation and curvature gains beyond the usual perceptual limits by introducing rotation during blinking or saccade movements [14, 20, 29].

2.2 Predictive RDW

Predictive algorithms incorporate the user's future motion predictions into redirection. Zmuda et al. [40] identified collision-free paths by using the shape of the tracked area and a map of obstacles and multi-step probabilistic predictions of the user's virtual path through a known virtual environment. Nescher et al. [18] proposed a method called Model Predictive Controlled Redirection Technique (MPCRed), which is able to dynamically select suitable redirection techniques and control parameters such as their strength. This study formulates the problem of guiding users in small physical rooms using redirection techniques as an optimal control problem. This allowed them to apply efficient probabilistic programming algorithms to maximize the free walking experience. Their proposed algorithm uses a map of the virtual environment to continuously determine the best redirection technique that must be applied next. Recently Dong et al. [7] proposed a new strategy for multi-user redirected walking using a dynamic artificial potential field, which generates repulsive forces to guide users away from obstacles and other users, and uses gravity to guide users to open or unobstructed spaces. In this approach, users are not only repelled by walls, but also by other users and their future states. Where predictive algorithms are applicable, they have been shown to significantly outperform reactive algorithms.

2.3 Reactive RDW

Razzaque et al. [24] proposed three steering strategy algorithms: Steer-to-Center (S2C); Steer-to-Orbit (S2O); Steer-to-Multiple-Targets (S2MT). These three algorithms respectively guide the user to the center of the real space by using the redirection gain operation, guide the user's walking trajectory into a specific circular trajectory, and guide the user to multiple designated target points in the real

space in turn. Hodgson et al. [9] extended the Steer-to-Multiple-Targets algorithm to the Steer-to-Mutiple+Center algorithm, and designed experiments to compare the pros and cons of the four guided strategy algorithms. They found that, in their case, the S2C algorithm performed best among the four algorithms. In another experiment by Hodgson et al. [10], the virtual scene where the user roams is a road scene, and the roaming direction is constrained to a certain extent. In this case, the S2O algorithm performs better than the S2C.

Azmandian et al. [1] studied how the total area and size of the tracking space affects the performance of S2C and S2O algorithms. Chen et al. [4] described an RDW algorithm for non-ideal physical environments. Lee et al. [15] proposed Steer-to-Optimal-Target (S2OT) for redirected walking. It estimates the optimal steering target through reinforcement learning, specifically using a technique called Deep Q-Learning. There were also works by Chang et al. [3] and Strauss et al. [28] using reinforcement learning to redirect users. Thomas et al. [30] proposed an artificial potential field based RDW method (APF). Bachmann et al. [2] made an APF method that allows multiple users to walk in the same tracking space. Messinger et al. [17] improved the APF method for irregular tracking spaces. Xu et al. [37] quantitatively redirected the user to discrete best-reachable poses by simultaneously considering steering and resetting.

In addition to reducing physical collisions by only considering the layout of the PE, some recent RDW methods also focus on improving the user experience in the VE. Recently Chen et al. [5] and Wang et al. [33] proposed to use reinforcement learning to align the virtual environment with the physical environment in order to provide users with passive haptics while walking. Xu et al. [36] proposed a point-of-interest-aware RDW strategy to guarantee that users will not trigger resets at important locations in the VE while reducing the number of resets. Recently an RDW solution for cloud-based multiplayer VR scenarios has also been proposed [35], aiming to ensure that remote users located in different PEs have the same number of resets for fairer online gaming and reduce the number of resets for all users.

2.4 Reset Strategy

Even with RDW techniques, collision-free motion in large virtual environments cannot be guaranteed when the tracking area is small [19]. For example, a user may perform a turn towards a wall while approaching that wall. How to handle situations where users encounter physical boundaries needs to be considered. Williams et al. [26] proposed several methods for resetting users and examine users' perceptions of each reset technique. The results show the pros and cons of the freeze backup, freeze turn, and two-turn methods. Some methods [19, 21, 22] achieve the function of reset by diverting the user's attention, such as the use of visual and auditory distractors. Cirio et al. [6] presented the user with a visual barrier to indicate where the boundaries are.

There are some methods that not only consider the user's reset experience, but also consider the impact of the reset orientation on the user's next walking. Reset to center (R2C) is a general method that always resets the user's orientation to the center of the physical space. Thomas et al. [30] introduced three improved reset strategies, including reset to gradient (R2G), modified reset to center (MR2C), and step-forward reset to gradient (SFR2G), which addressed the case where the R2C strategy fails when an obstacle is located between the user and the center of the physical environment. Zhang et al. [38] proposed a method using an optimization technique to calculate the optimal reset direction of each obstacle boundary endpoint. They also proposed an one-step out-of-place resetting strategy, which avoids subsequent resets by introducing appropriate displacements during the reset process [39].

3 METHOD

The historical walking data is valuable for decision-making in different stages of our redirection system. Because there is no available system that records detailed walking data of users, we generate historical walking data of users through simulation experiments. However, the walking data could be messy and noisy, which need to be further processed to get a more compact and intuitive representation. Here, a weighted directed graph is employed to provide a clear representation of walking data. We will introduce the generation of the simulation data and the algorithm built upon our compact graph-based representation.

3.1 Generation of the Dataset

Our method is designed to minimize disruptions during VR experiences where a user can freely explore the environment by leveraging historical walking data. However, there is no off-the-shelf solution that can accurately and conveniently record the detailed walking data of users. Therefore, we propose to simulate the user's walking to obtain the required large amount of historical data. To make our method applicable in real use cases, we generate the simulated walking data with some assumptions based on our observations of VR users' walking in the preliminary experiment. The assumptions are:

- During the exploration process, the users have a large probability of moving toward the destination in the virtual scene.
- When moving toward the destination, the user will have a certain probability of being attracted by other destinations.
- Users are likely to stay at a destination for a while before moving on to the next destination.
- The user‘s walking trajectory is not fully straight. Instead, the walking trajectories can contain some random patterns.

The above four assumptions are our main rules to guide the generation of our user walking data. We consider the assumption that users often walk towards certain destinations based on the following reason: In a real VR application where users can freely walk, the user often has a specific task involving moving to a destination in the virtual space. VR explorations where users only walk aimlessly in a large space are very rare. Therefore, in this work, we focus on the scenarios where the users can experience, rather than an environment that probably is only useful for programmers and developers.

Since users could have different walking styles and it is impossible to enumerate all of the styles when simulating the walking data. Instead, we use the following way to generate various user walking data: We divide the user's walking trajectory into different temporal sections. In each section, the user may adopt three different walking modes, namely walking straight forward, walking clockwise and counterclockwise along an arc with a fixed radius. In our work, we set the radius of the arc to $3m$. Among the three modes, advancing along the arc can be more specific by defining advancing to the left front and advancing to the right front. The path length of each temporal section is consistent, and various walking trajectories can be simulated according to the different arrangements and combinations of these three modes. This is not sufficient for walking data generation, because this only ensures that different trajectories can be generated, but does not let the users mostly move toward the destination in the virtual scene. Therefore, we adopt the method of dividing the trajectory into sections to generate the path between two destinations each time and apply a coordinate transformation on the generated path to ensure the path follows our assumptions. Finally, the paths between different destinations are stitched together to generate smooth user walking trajectories. In order to satisfy the second assumption, we randomly add a random interference to make

a small probability of changing the user's current destination. Here, we can use the location where the interference event occurs as a temporary destination. We stitch the front and back paths together through the temporary destination.

Now we briefly explain the different effects of the three different modes on the generated trajectories. We denote the set of destinations as $D = \{d_1, d_2, \ldots, d_m\}$. Each destination in a set containing the location (d_{i_x}, d_{i_y}). Since we record the user's position and orientation information of each temporal section, the user's walking data is actually represented by a sequence $P =< p_0, p_1, p_2, \ldots, p_n >$. Each pose p_i in the sequence contains the user's position (x_i, y_i) and orientation θ_k. Here, the orientation of the starting point where (x_0, y_0) is $(0,0)$ is the x-direction. Suppose the user is walking at speed v, and the time slice we take is Δt. Then the user will walk $\Delta s = v \times \Delta t$ in the temporal section. We assume that the user is currently in pose $p_k = (x_k, y_k, \theta_k)$. For the case of moving along a straight line, the user's next gesture $p_{k+1} = (x_{k+1}, y_{k+1}, \theta_{k+1})$ can be expressed as:

$$\theta_{k+1} = \theta_k \tag{1}$$

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} + R(\theta_k) \times \begin{pmatrix} \Delta s \\ 0 \end{pmatrix} \tag{2}$$

$$R(\theta_i) = \begin{pmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{pmatrix} \tag{3}$$

Where $R(\theta_i)$ is the rotation matrix of the vector. It makes the generated user walking path smooth. Since the user moves in a straight line during this temporal section, the user's orientation remains the same as before. For the case of advancing along an arc with a fixed radius, the user's position and orientation will change, and the user's next pose $p_k(x_{k+1}, y_{k+1}, \theta_{k+1})$ can be expressed as:

$$\theta_{k+1} = \theta_k + \Delta\theta, \quad \Delta\theta = \Delta s / r \times h(k) \tag{4}$$

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} + R(\theta_k) \times \begin{pmatrix} 2r\sin\frac{\Delta\theta}{2}\cos\Delta\theta \\ 2r\sin\frac{\Delta\theta}{2}\sin\Delta\theta \end{pmatrix} \tag{5}$$

$$h(k) = \begin{cases} 1, & \text{user turns to left} \\ -1, & \text{user turns to right} \end{cases} \tag{6}$$

where $h(k)$ is the function that indicates whether the user goes along a left or right arc. We calculate the distance between the user's current location and starting location after each temporal section. The generation will stop when the cumulated distance equals the distance between the two destinations that need to be generated. We limit the user's offset angle to ensure that the user's distance from the initial position increases after each time slice. We require that the angle θ_i of each pose satisfies the following condition:

$$|\theta_i - \theta_0| < \frac{\pi}{2} \quad or \quad |2\pi - |\theta_i - \theta_0|| < \frac{\pi}{2} \tag{7}$$

When the angles of all poses satisfy the above condition, we can ensure that walking along the generated path is moving further away from the initial point and closer to the destination. However, just meeting the condition about the distance is not sufficient. We also need to do a 2D transformation on the generated path to make it connected with other paths. Suppose that what is currently generated is the user's path from the destination d_u to the destination d_v. Then, according to the above algorithm, we can first generate $P =< p_0, p_1, \ldots, p_n >$ to satisfy:

$$(x_n - x_0)^2 + (y_n - y_0)^2 \geq (d_{v_x} - d_{u_x})^2 + (d_{v_y} - d_{u_y})^2 \tag{8}$$

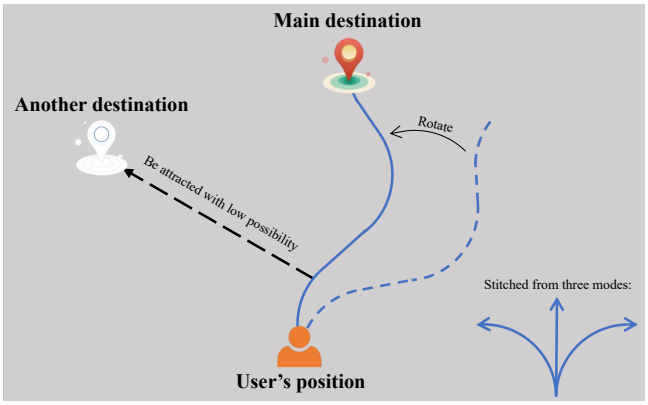


Figure 2: The generation of simulated walking trajectories.

When the temporal section is small enough, we can consider that equation holds when the condition is satisfied. Then we only need to perform a rigid transformation on each point p_i in the sequence P to make the p_0 and d_u coincide and the p_n and d_v coincide. After the rigid transformation, the sequence $P' =< p'_1, p'_2, \ldots, p'_n >$ is added to the entire user path. The sub-paths are generated one by one according to the same algorithm. The rigid transformation changes each point in the sequence as follows:

$$\alpha = \arctan\frac{d_{v_y} - d_{u_y}}{d_{v_x} - d_{u_x}} + \frac{\pi}{2}g(i) - \theta_n \tag{9}$$

$$\begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \begin{pmatrix} d_{u_x} \\ d_{u_y} \end{pmatrix} + R(\alpha) \times \begin{pmatrix} x_i \\ y_i \end{pmatrix} \tag{10}$$

The process of trajectory simulation is shown in the Fig. 2. After investigating the number of places that users mainly go to in different virtual scenes, we set the number of destinations in the dataset to $5-10$. For each scene, we generated $500-1000$ pieces of historical user walking data. When generating, the temporal section is set to $1/30s$, and the user's speed is assumed to be $1m/s$. In order to restore the user's trajectory in the virtual space, our simulated path will randomly stay at the destination for a certain period. We also provide probabilities for selecting different walking modes (straight, clockwise, or counterclockwise), and different historical user walking data can be generated by changing these probability values.The time taken to generate the simulation dataset depends on the total length of the paths and the time dispersion. For a virtual environment with given parameters (speed, walking mode probability, etc.), we set the temporal section to 1/30s in order to generate a simple video of the paths. At any moment during runtime, the next pose can be calculated in constant time. Also, since each pose is rigidly transformed at most once, the time complexity of the algorithm is of the same order as the output. When the total length of the path is 10000 meters, the computation time is 0.5-1.0 seconds. When generating simulation paths for a new virtual environment, we can complete the generation in a short time without pre-loading any simulation path data.

3.2 Encode from Historical Trajectory Data

The raw historical trajectory data is messy and noisy, thus, is difficult to use. If the data is from the real world, it could also contain useless information, such as the user's walking habits. As seen from Fig. 3, some people walk very straight, some walk along arcs, and some walk tremblingly. We also found that even just walking between the same targets a and b, the walking trajectories of different people will be different. As shown in the Fig. 3, they seem to be doing different things. These situations will significantly affect the redirection of

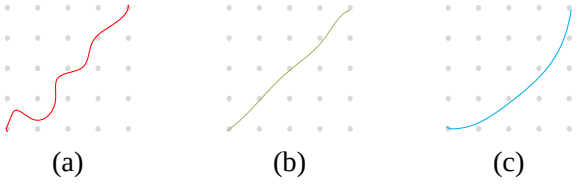


Figure 3: Walking trajectories of different users. (a) A user walking tremblingly. (b) A user walking straight. (c) A user walking along arcs.

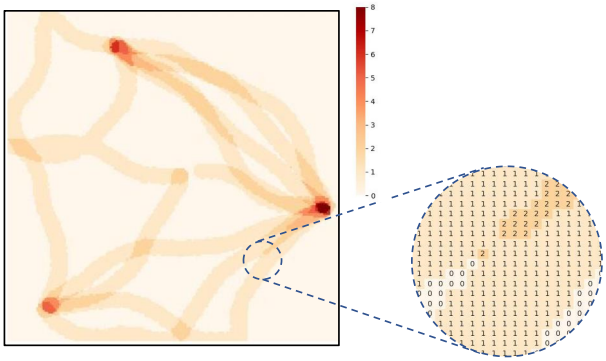


Figure 4: A heatmap generated from the walking data.

our method. Thus, we need to eliminate meaningless information as much as possible.

First, we discretized the virtual space with an interval of $0.1m$ to obtain a discrete representation of user positions in the virtual space, making the calculation possible. The user will stop at certain positions in the virtual space, which are usually interaction points or their destinations in the virtual space. We first obtain the destinations according to the historical data of the user's stay. Given these destinations, we divide the historical walking data of each user by the destinations to obtain sub-paths and the two endpoints of each sub-path. If it is the first/last sub-path, its two endpoints will include the starting/ending point of the entire user's walking trajectory. For each discrete point, we record the count of times the user passes them. When the minimum distance between one sub-path and a discrete point is less than 0.3 meters, we consider that the sub-path passes that point. Summarizing all the users' walking data, we can get an overall heatmap that records each discrete point's count of being passed. In Fig. 4, we visualize the heatmap where we use different colors to represent different counts.

With the heatmap of historical walking data, it becomes intuitive to extract representative paths. All we need to do is to find the path with the darkest color. First, we define that a point can be connected to eight adjacent points, and a path is a set of connected points from one waypoint to another. The same point can only be passed at most once by one path. We define $P(a,b)$ as the set of all paths starting at a and ending at b. Now we need to find a *representative path* from a to b. Specifically, we need the set of connected 2D points $C' = \{c'_1, c'_2, \ldots, c'_n\}$ of the sub-path between the two endpoints to satisfy the following conditions:

$$pathValue(C_i) = \min_{c_j \in C_i} value(c_j), \quad C_i = \{c_1, c_2, \ldots, c_j, \ldots\} \tag{11}$$

$$pathValue(C') \geq pathValue(C_i), \quad \forall C_i \in P(a,b) \tag{12}$$

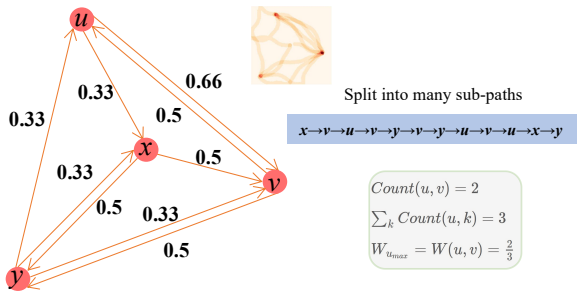


Figure 5: An example of a processed weighted directed graph. Directed edges with a weight of 0 are hidden.

where C_i is any path, c_j is a discrete point in the path and the *value* function allows us to obtain the times of that discrete point being passed. If there are multiple eligible paths, we take any one of them as a representative path.

We observed that the probabilities of the users walking from one waypoint to other waypoints are significantly different. Taking this into account in our algorithm will obviously improve the performance of our algorithm. We define here a weighted directed complete graph G_w whose nodes are all the waypoints. The edge weights of G_w can be obtained directly from the preprocessed historical user walking data as follows, where $W(u,v)$ refers to the weight of the edge from u to v, and $Count(u,v)$ refers to the number of paths from u to v, equivalent to $|Path(u,v)|$. $W_{u_{max}}$ is the maximum value of the weights of the outgoing edges starting from u.

$$W(u,v) = \frac{Count(u,v)}{\sum_k Count(u,k)} \tag{13}$$

$$W_{u_{max}} = \max W(u,k), \quad k \neq u \tag{14}$$

Now we get a directed graph from the historical walking data. Fig. 5 is an example of a weighted directed graph.

3.3 RDW based on the Graph

Before discussing how to use weighted directed graphs, we first introduce how to discretize the user's position and orientation in physical space, which will help us quickly calculate from the infinite possibilities. The spacing between discrete locations is set to $0.5m$, a length slightly larger than the width of a human shoulder, enough to describe the user's position in the room. We uniformly sample twelve angles to discretize orientations such that the angle between two adjacent orientations is $\pi/6$.

Given the weighted directed graph, we can obtain the distance between the user's position on a path and their current destination and get the probability of the user going in different directions when reaching a waypoint. We design an effective redirection algorithm to utilize the above two kinds of information. Previous predictive algorithms could work in similar situations, but since there are no obstacles or virtual boundaries to limit the user's walking paths and the user has a possibility to go beyond the expected path, an effective algorithm needs to consider the state of users to work well for an environment that a user can freely explore. Therefore, compared with the algorithm of selecting several walking modes and searching in a certain depth, which is usually used by the predictive methods, we use the greedy algorithm to avoid being limited to certain types of walking patterns without the constraints imposed by the searching depth. Our method is able to make redirection with the consideration of all positions and orientations that can be reached from the current position and orientation with gain constraints.

We categorize the user's walking states into the following types and apply corresponding strategies for guiding their walking:

On the path between two waypoints Our algorithm defines two kinds of user situations when they are on the path between two waypoints: 1. If the user has to reset at least once when walking from the current discrete pose to the next waypoint (that is, to reach the next waypoint in the virtual space, the user's path in the physical space will be folded for continuing walking), we intend to select the target pose based on the distances to these reachable poses. 2. When the user can walk to the next waypoint without a reset, we will use a higher probability of choosing them as the next destination. To achieve that, we give it an extra score that makes it easier to stand out than other poses.

We use the method of evaluating discrete points in physical space in the work of Xu et al. [37], which contains the following steps: 1. Build a discrete representation to describe the user's poses, including their positions and orientations. The discrete poses are called *standard poses*. 2. Connect two standard poses with a path that holds a tangent continuity at the entry and exit points and has a fixed curvature to satisfy the gain threshold. 3. Evaluate the standard poses using a function $Q(x,y,\theta)$. The calculation of $Q(x,y,\theta)$ for a certain standard pose (x,y,θ) is based on the estimation of all the standard poses it can reach and the distance between (x,y,θ) and each of the standard poses. The distance is used as a reward. Finally, the evaluation function is used to find the optimal standard pose to guide the user to redirect to it. Assuming the user is currently at position (u,v) facing β and has a path length s_{rest} to the next destination in virtual space, the score of each reachable pose (x_i,y_i,θ_i) is calculated as follows:

$$S(x_i,y_i,\theta_i) = Q(x_i,y_i,\theta_i) + G(x_i,y_i,\theta_i) \tag{15}$$

$$G(x_i,y_i,\theta_i) = \begin{cases} 1000 & ,Dis(x_i,y_i,\theta_i,u,v) \in [0.86\times s_{rest}, 1.26\times s_{rest}] \\ 0 & ,Dis(x_i,y_i,\theta_i,u,v) \notin [0.86\times s_{rest}, 1.26\times s_{rest}] \end{cases} \tag{16}$$

$$S' = \max S(x_i,y_i,\theta_i) \tag{17}$$

where $Q(x_i,y_i,\theta_i)$ is the score of the physical space in Xu et al. [37]'s work, and $Dis(x_i,y_i,\theta_i,u,v)$ refers to the path length from pose (x_i,y_i,θ_i) to position (u,v), which can be calculated by the previous path connection algorithm. The value of G is decided by comparing s_{rest} with $dis(x,y,\theta,u,v)$, which determines whether the user can reach the pose (x_i,y_i,θ_i) without reset. If the user can reach it, we will add 1000 as a reward. Otherwise, it means that the user needs a reset before arrival and the score will be the same as Q, which is determined to some extent by the distance. We use S' to denote the highest score of all the evaluated poses. When guiding the user, we will select its corresponding physical pose as the target pose.

At a waypoint and making a turn When the user is at a waypoint u and starts to turn, we try to find the rotation gains to make all the possible virtual orientations of the poses starting from the current position correspond to the orientation with the highest score in the physical space. Since we already know all the waypoint information, we can use this to adjust the gain. More specifically, We adjust the rotation gain to 1 when there is a waypoint within 30° range in front of the user, to $1-0.2\times W(u,v)/W_{u_{max}}$ when there is a waypoint v within 10° in front of the user, and to 1.49 when there is no waypoint within 30° in front of the user.

Outside the path of the graph When the user is outside the weighted directed graph in the virtual space, we can only use some of the information provided by the graph. According to the user's current orientation, we can get the possible destinations in front of the user and consider the impact of these points when selecting a path to guide the user in physical space.

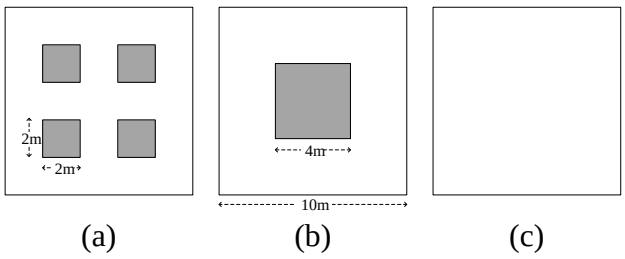


Figure 6: Layouts of our physical spaces. (a) Four obstacles, (b) One obstacle, and (c) No obstacles.

We can take each of these waypoints in the user's current orientation as the next waypoint, set s_{rest} to the euclidean distance between the current position and that waypoint, and compute the G function for each pose as in the first case. The difference is that the calculation is done for each waypoint, and we greedily add up the values of these G functions to find the optimal pose and guide the user there.

4 Evaluation

We conduct extensive simulation experiments to compare our method with previous methods. We also design and conduct a user experiment to demonstrate the effectiveness of our method in real applications.

4.1 Simulation Experiment

4.1.1 Experiment Environment

A significant advantage of simulation experiment is that a substantial number of experiments with a large diversity of their physical spaces can be conducted simultaneously, which is impossible for conventional user experiments. Therefore, simulation experiments have been widely used to evaluate a method in previous related researches. We tested the performance of the algorithm in three different physical environments. We set the size of the three physical spaces to $10\times 10m$. Their main difference is in the distribution of obstacles in space. We placed four obstacles of the same size $2m\times 2m$ in the first physical space, as shown in Fig. 6 (a). These four obstacles divide the physical space into criss-cross paths. We placed a $4\times 4m$ obstacle in the middle of the second physical space, as shown in Fig. 6 (b), which will make the user only move around the physical space. In the third physical space, we do not use any obstacles, as shown in Fig. 6 (c), and the user can walk freely in this space. In the first and the third environments, we set the user's initial position as $(5.25,5.25)$. We set the user's initial position in the second environment at $(1.25,1.25)$. The initial orientation of the user is the negative direction along the y-axis.

The size of the virtual space we use is $20m\times 20m$, which users can explore freely. We set the user's initial position in the virtual space as the center of the virtual space and the initial orientation of the user as the negative direction of the y-axis. The user's walk speed is a constant speed of $1m/s$. The turning speed of a user is set as a fixed angular velocity of $\pi/2$ rad/s after reaching the waypoint until facing the next waypoint. In each test, we simulated 100 virtual waypoints.

The test data are generated based on the same rules applied to the generation of the user's historical walking data, making the test more similar to real applications. According to our survey, we simulated and generated historical user walking data for 10 different

Table 1: The mean of reset times of different predictive algorithms in the test virtual scenes.

Environment	Method	$S_0(7)$	$S_1(5)$	$S_2(8)$	$S_3(6)$	$S_4(10)$	$S_5(6)$	$S_6(6)$	$S_7(8)$	$S_8(8)$	$S_9(9)$	Total
no obstacle	FORCE	143.5	164.4	137.2	92.3	264.0	159.2	228.0	217.1	144.3	409.4	195.57
	MPCred	82.7	104.8	93.5	72.2	81.4	74.9	83.2	93.5	80.3	98.3	86.48
	OUR	**75.1**	**89.3**	**80.0**	**64.1**	**75.3**	**67.7**	**73.2**	**82.0**	**73.5**	**84.0**	**76.42**
one obstacle	FORCE	143.0	177.7	181.4	221.3	211.3	198.1	338.0	187.0	143.6	254.7	206.52
	MPCred	117.3	139.9	127.3	96	120	108.4	114.8	132	111.9	127.3	119.49
	OUR	**84.1**	**94.2**	**86.5**	**76.5**	**84.2**	**76.9**	**83.5**	**92.7**	**78.0**	**88.1**	**84.47**
four obstacles	FORCE	145.1	173.5	204.2	290.9	275.9	207.1	281.6	240.6	195.8	177.8	219.25
	MPCred	124.2	147.6	133.7	99	124	110.9	121.2	136.8	120.3	138.1	125.58
	OUR	**98.5**	**115.8**	**109.4**	**85.2**	**98.3**	**83.1**	**99.9**	**111.4**	**94.2**	**110.3**	**100.61**

Table 2: The mean of reset times of different reactive algorithms in the test virtual scenes.

Environment	Method	$S_0(7)$	$S_1(5)$	$S_2(8)$	$S_3(6)$	$S_4(10)$	$S_5(6)$	$S_6(6)$	$S_7(8)$	$S_8(8)$	$S_9(9)$	Total
no obstacle	S2C	91.2	115.5	105.1	84.2	90.0	82.3	95.0	126.7	111.2	115.2	101.64
	S2O	119.4	109.8	129.5	78.1	97.9	82.5	97.2	121.9	108.4	142.4	108.71
	SRL	107.1	100.1	96.9	74.8	96.0	81.1	93.4	102.1	94.2	103.8	94.89
	ZIGZAG	209.9	312.8	245.5	342.0	182.3	157.1	163.7	195.2	261.2	154.9	231.08
	APF	**61.2**	**75.4**	**70.8**	**44.3**	**66.3**	**53.3**	**62.3**	**75.7**	**62.6**	**73.0**	**64.49**
	XU2022	80.0	101.1	89.3	66.4	82.4	69.5	77.0	93.8	79.9	87.3	82.67
	OUR	75.1	89.3	80.0	64.1	75.3	67.7	73.2	82.0	73.5	84.0	76.42
one obstacle	S2C	198.7	223.3	213.9	192.6	246.2	202.3	196.7	213.5	186.0	217.0	208.94
	S2O	129.4	151.2	133.3	119.9	125.0	112.8	147.5	132.8	124.3	150.0	132.62
	SRL	163.6	204.8	173.9	160.1	166.2	153.1	169.7	192.1	166.9	178.6	173.41
	ZIGZAG	262.9	241.5	284.4	250.0	197.5	204.5	359.6	300.2	319.2	257.1	269.36
	APF	192.1	272.2	251.0	174.7	207.6	171.3	208.0	223.8	206.0	233.2	213.99
	XU2022	104.7	123.6	111.5	95.6	106.1	98.0	106.0	115.2	101.0	116.3	107.80
	OUR	**84.1**	**94.2**	**86.5**	**76.5**	**84.2**	**76.9**	**83.5**	**92.7**	**78.0**	**88.1**	**84.47**
four obstacles	S2C	237.5	296.4	281.9	204.6	239.6	212.3	269.2	267.2	269.1	265.1	253.42
	S2O	269.8	311.0	288.8	236.0	262.6	229.5	270.9	318.4	259.1	291.9	276.18
	SRL	250.3	250.7	236.1	193.6	209.7	173.7	206.9	231.4	204.3	225.7	218.30
	ZIGZAG	263.1	357.3	250.3	254.4	276.4	322.9	241.0	264.6	299.0	278.5	278.87
	APF	208.2	242.8	220.8	190.9	207.2	188.7	207.6	231.6	207.5	227.4	213.27
	XU2022	120.7	152.6	133.9	104.5	125.7	108.3	124.5	136.4	117.4	137.6	126.16
	OUR	**98.5**	**115.8**	**109.4**	**85.2**	**98.3**	**83.1**	**99.9**	**111.4**	**94.2**	**110.3**	**100.61**

virtual environments, and the number of destinations in each virtual environment was set to between 5 and 10.

We simulate the scenario where the user can walk completely freely in the virtual space by removing all the obstacles in the virtual space of our test environment. Some related researches use obstacles and boundaries in their virtual space, sometimes leading to a virtual space shape where their method is advantageous during testing. However, the scenes of the real use cases can be significantly different in terms of their shapes and obstacle layouts. Therefore, we choose not to insert any obstacles to limit their walking. Instead, we require them to walk to the next waypoint to encourage them to walk in a certain direction and not limit their walking paths. This approach can easily evaluate the performance of the algorithm in different environments. For each virtual scene, we generated 10 virtual paths for testing. That means each algorithm will be tested by 100 times in each physical space. An algorithm will be tested 300 times in different virtual environments and physical spaces. The simulations were run on a Windows computer with an Intel(R) Core(TM) i5-4200H CPU and 8GB RAM.

4.1.2 Comparison Experiments with Predictive Algorithms

Our approach shares many similarities with predictive algorithms. Predictive algorithms also consider the user's possible future movements in their prediction. However, most of the predictive algorithms only predict the walking paths of the user by analyzing the layout of obstacles or the boundaries of the virtual space. For example, the FORCE method [40] assumes that the virtual space is a virtual store with aisles. The virtual space is divided into long, narrow horizontal/vertical aisles by the shelves. A user clearly has several possible directions when they are in the middle of an aisle or a cross. The MPCred method [18] proposed by Nescher et al. also uses a similar furniture shop as the experimental environment. Prediction algorithms can achieve good performance when directing users in those regular environments. However, they cannot represent most types of virtual layouts in real applications. When the user is able to explore the virtual environment freely, they are not able to provide effective guidance. On the contrary, our method relies on the historical data regardless of the specific spatial layouts in the virtual space, thus can be adopted to a free space. Since our algorithm also predicts the user's future movement, the input of our algorithm can also become the input of FORCE. We test FORCE, MPCred, and our algorithm in ten virtual spaces with three different physical spaces. Considering that the interruptions caused by encountering obstacles greatly affect a user's experience quality, we use the number of resets for a user as the metric of the performance of each algorithm. The test results are shown in Table 1. Each of the 10 virtual environments has a different number of destinations. We denote the i-th virtual scene with j destinations as $S_i(j)$.

The search approach adopted in FORCE is time-consuming, especially when the search depth is extremely large if the user can freely explore the virtual space. In theory, the paths connecting different destinations form fully-connected graphs, and the complexity of the search algorithm is exponential. When implementing FORCE, we considered 5 different curvature gains and 3 different rotation gains. We did not consider the translation gain because the number of different combinations of the translation and the curvature gains would be too large. We set the search depth to 2. Nevertheless, FORCE was still the slowest among all the tested algorithms and ran

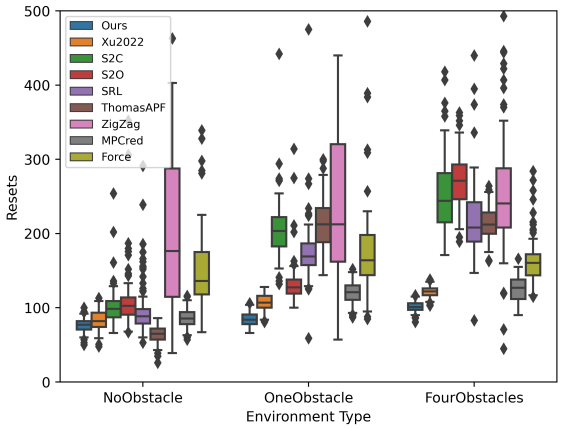


Figure 7: The box plots of the experiment results. The outliers are not included in the whiskers.

even slower when there were more destinations. MPCred produces better results than FORCE and most reactive methods, but it is still worse than our result, demonstrating that the conventional predictive algorithms do not fit the virtual space that can be freely explored.

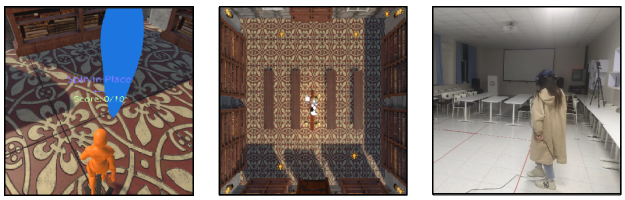
4.1.3 Comparison Experiments with Reactive Algorithms

Reactive algorithms often work better in virtual environments where users can freely explore. Heuristic-based methods are generally better in real-time. We realize the test of Xu et al.'s work in 2022 [37] through simulation experiments. Tested Steer-to-Center (S2C) [9], Steer-to-Orbit (S2O) [9], Zigzag [23], Thomas etc. al.'s APF [30] and the steer-by-reinforcement learning method (SRL) by using OpenRDW [16] platform. These algorithms are tested in each physical space with a total of 100 paths in ten virtual scenes. The results of the test are shown in Table 2. When using the OpenRDW to test, if the user stays in one place for too long or the number of resets is too high, we will directly end the test and declare the test invalid. In the tests of all algorithms, S2C has a total of 5 invalid tests, S2O has a total of 1 invalid tests, SRL has a total of 15 invalid tests, ZigZag has a total of 46 invalid tests, and APF has no invalid tests.

The results show that APF performs best in scenes without obstacles, followed by our algorithm. For scenes with one obstacle in the center of physical space, our algorithm performs significantly better than other algorithms with the fewest resets in all virtual scenes. The method of Xu et al. also performs well in this scenario. In addition, S2O performs better than other algorithms, which is consistent with the characteristics of the algorithm. Our algorithm and Xu et al.'s algorithm far outperform others in the scenario of evenly distributing 4 obstacles in the physical space. Our algorithm slightly outperforms Xu et al.'s algorithm in all 10 scenarios. S2O, which performs well when there is one obstacle in the center of the physical space, performs almost as badly as ZigZag when the distribution of obstacles in the physical space does not match the algorithm characteristics. ZigZag performs the worst in all physical spaces. The box plots of the results of experiments are shown in Fig. 7.

4.1.4 Discussion

From the results of the above experiments, we can see that our algorithm has excellent results compared with the predictive or reactive algorithms. The predictive algorithm FORCE does outperform most reactive algorithms in many scenarios. However, as the number of destinations in the physical space increases, FORCE takes more and more time and resources. The limits of predictive algorithms make



(a) (b) (c)

Figure 8: User experiment environments. (a) Third-person view of the virtual scene and user. (b) Top view of the virtual environment. (c) Photo of the physical environment.

people pay more attention to reactive algorithms. There are many types of reactive algorithms, heuristic-based, reinforcement learning based, etc. Reactive algorithms are not limited by virtual space and usually have better real-time performance. Specific algorithms in specific scenarios may have better performance. For example, S2O performs well when the obstacle is in the center of the physical space, and APF achieves the best results when there is no obstacle in the physical space. These algorithms are excellent, take into account various properties in physical space and use various gains well. However, another point that these algorithms rarely consider is the interaction between the user and the virtual space. With the interaction, users have various experiences in the virtual space; the interaction itself is also a kind of information that can be used. Our algorithm considers the historical user walking data and implements the RDW algorithm on this basis. It has an excellent effect as predictive algorithm and can be used generally in various virtual spaces. Our algorithm's results are only slightly inferior to APF when there are no obstacles in the physical space. Our algorithm outperforms other algorithms in scenes with obstacles. In many cases, it even owns half of the number of resets compared to most algorithms. APF performed well in a physical space without obstructions and had $2-3$ times as many resets as our algorithm in the other two tests.

4.2 User Study

4.2.1 Experiment Design

We evaluate our method through real user experiments. Since predictive methods are not applicable for our environments, we choose representative reaction algorithms, S2C [9] and APF (Thomas APF) [30] to compare with our method. Our virtual environment is a 10m*10m square room containing a crystal to indicate the destination. Each subject wore an HTC VIVE while walking within a 4m*4m square tracking space with a 0.5m wide buffer to prevent collision.

The subjects were initially located in the center of the virtual and physical space. Before the experiment began, subjects took a preliminary test in the virtual environment to reduce discomfort during the formal experiment. They performed three walking experiments with the same sequence of waypoints in all three experiments, testing S2C, APF, and our method, respectively. We used a crystal to mark the target position in the virtual space. When the user touched the crystal, the crystal representing the current target would disappear and the next target crystal would appear at a different position. The user's task was to go through 10 waypoints. It should be noted that we did not specify the paths for the user to follow when walking towards the next waypoint.

We recorded the number of resets and the distance traveled in the virtual space. The total reset number can roughly reflect the interruption of the user's VR experience. However, since our experiment does not strictly limit the user's walking path, just checking the reset

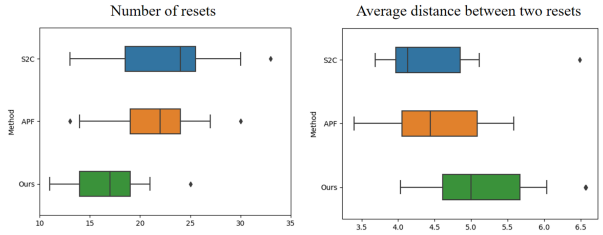


Figure 9: The distribution of the total number of resets and the average distance between two resets.

number is not sufficient to evaluate their experience. Therefore, we add the average distance the user walked in the virtual space between two successive resets as another indicator for their experience quality. Generally speaking, the longer this distance is, the better the user's VR experience is.

4.2.2 Result

We collected information from 15 subjects whose ages were distributed from 18 to 27 years old (9 females and 6 males). 10 of them stated they had VR experiences. The distribution of the total number of resets and the average distance traveled by the user between resets is shown in Fig. 9.

We used the Shapiro-Wilk test to check the total number of resets. All 3 groups of data followed normal distribution ($p = 0.780, 0.978, 0.944 > 0.05$). In the Levene's test, the 3 groups of data show homoscedasticity ($p = 0.156 > 0.05$). Therefore, we performed an analysis of variance using one-way ANOVA and the statistical results were significant, with differences between methods in the total number of resets ($F = 5.672, p = 0.007 < 0.01$). The t-test showed that our method had fewer total resets than the S2C algorithm ($t(14) = -4.059, p < 0.01$) and the APF algorithm ($t(14) = -4.236, p < 0.01$). The Shapiro-Wilk test shows that the mean walking distance of users does not follow the normal distribution ($p < 0.05$). In the Kruskal-Wallis test, the results show that there is a significant difference between different methods in terms of the the average user walking distance. The average user walking distance of our method is greater than the S2C algorithm ($p < 0.05$) and greater than the APF algorithm ($p < 0.05$).

The physical space without obstacles has been used in both simulation and user experiments. It can be seen that the APF algorithm have different results in the two experiments. A possible reason is that the configurations of the two experiments are not exactly the same: we used a 10m*10m physical environment in the simulation experiment, but a 4m*4m physical environment in the user experiment due to the length limitation of the cables. The surrounding walls naturally form a barrier, which makes the space available for free exploration much smaller than the barrier-free environment in the simulation experiment, and is more like a new environment configuration than the barrier-free environment.

The results of user experiments show that our method is more effective than previous methods. Our method successfully reduces the number of resets using historical data and increases the continuity of the immersive experiences in VR applications. Moreover, our method has no restrictions on the virtual environment, and users can explore freely in it. In our experiments, although we do not specify the path for the user between two targets, subjects tend to walk along paths that are similar to the simulated paths we produced, which means that the they do not stay outside the weighted directed graph for a long time, allowing us to effectively guide the user to the optimal physical space pose in most of the time.

Since a single set of experiments required subjects to walk about 50m, which is too long for some people and may cause VR sickness such as nausea or vertigo, we prepared a motion sickness SSQ [13] table for the subjects. The results showed that most people did not have adverse reactions during the test, and a small number of people had mild vertigo. In the conversation, we learned that the vertigo might be caused by some fast rotations when resets happen.

5 Conclusion And Future Work

We design a novel RDW method based on historical user walking data. We encode the historical user walking data to a heatmap, and generate a weighted directed graph from the heatmap for a discretized representation. We dynamically update the scores of different reachable poses in the physical space during the user's walking. We rank the scores and choose the optimal target position and orientation to guide the user to the best pose. Every time the user turns or finishes resetting, we find a target pose in physical space and guide the user to get there. In our experiments, our method significantly reduces the interruptions in the user's walking experience, outperforming previous predictive algorithms when the physical scenes have obstacles. Moreover, our algorithm has a wide range of application scenarios as reactive algorithms but has a much better performance.

Our algorithm has some limitations. Because our algorithm is based on historical user walking data, it does not perform better than other algorithms without that data. Secondly, our method cannot be used in dynamic environments with variable obstacles, as our algorithm requires prior information about the physical environment. Thirdly, since there is no off-the-shelf solution that can accurately and conveniently record the detailed walking data of users, we need to use a manual simulation of the walking data. Although we can generate all possible paths by changing parameters, the generated data is not as sophisticated as the real walking data, which may have an impact on the accuracy of our simulation-based approach. At the same time, although we have a dense heat map, only the distance of the user to the next destination or whether the user is heading to the next destination is considered in the heat map. In the future, we will encode more accurate user walking information in our graph-based representation to provide a more precise guidance.

Acknowledgments

This work was supported by the Natural Science Foundation of China (Project Number 62132012), and Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology. Fang-Lue Zhang was supported by Marsden Fund Council managed by Royal Society of New Zealand (No. MFP-20-VUW-180).

References

[1] M. Azmandian, T. Grechkin, M. T. Bolas, and E. A. Suma. Physical space requirements for redirected walking: How size and shape affect performance. In *ICAT-EGVE*, 2015.

[2] E. R. Bachmann, E. Hodgson, C. Hoffbauer, and J. Messinger. Multi-user redirected walking and resetting using artificial potential fields. *IEEE Transactions on Visualization and Computer Graphics*, 25:2022–2031, 2019.

[3] Y. Chang, K. Matsumoto, T. Narumi, T. Tanikawa, and M. Hirose. Redirection controller using reinforcement learning. *IEEE Access*, 9:145083–145097, 2021.

[4] H. Chen, S. Chen, and E. S. Rosenberg. Redirected walking strategies in irregularly shaped and dynamic physical environments. In *25th IEEE Conference on Virtual Reality and 3D User Interfaces (VR 2018). Workshop on Everyday Virtual Reality*, 2018.

[5] Z.-Y. Chen, Y.-J. Li, M. Wang, F. Steinicke, and Q. Zhao. A reinforcement learning approach to redirected walking with passive haptic feedback. In *2021 IEEE International Symposium on mixed and augmented reality (ISMAR)*, pp. 184–192. IEEE, 2021.

[6] G. Cirio, M. Marchal, T. Regia-Corte, and A. Lécuyer. The magic barrier tape: a novel metaphor for infinite navigation in virtual worlds with a restricted walking workspace. In *VRST '09*, 2009.

[7] T. Dong, X. Chen, Y. Song, W. Ying, and J. Fan. Dynamic artificial potential fields for multi-user redirected walking. *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 146–154, 2020.

[8] T. Grechkin, J. Thomas, M. Azmandian, M. T. Bolas, and E. A. Suma. Revisiting detection thresholds for redirected walking: combining translation and curvature gains. *Proceedings of the ACM Symposium on Applied Perception*, 2016.

[9] E. Hodgson and E. R. Bachmann. Comparing four approaches to generalized redirected walking: Simulation and live user data. *IEEE Transactions on Visualization and Computer Graphics*, 19:634–643, 2013.

[10] E. Hodgson, E. R. Bachmann, and T. Thrash. Performance of redirected walking algorithms in a constrained virtual world. *IEEE Transactions on Visualization and Computer Graphics*, 20:579–587, 2014.

[11] C. Hutton, S. Ziccardi, J. Medina, and E. S. Rosenberg. Individualized calibration of rotation gain thresholds for redirected walking. In *ICAT-EGVE*, 2018.

[12] V. Interrante, B. Ries, and L. Anderson. Seven league boots: A new metaphor for augmented locomotion through moderately large scale immersive virtual environments. *2007 IEEE Symposium on 3D User Interfaces*, pp. –, 2007.

[13] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *International Journal of Aviation Psychology*, 3(3):203–220, 1993.

[14] E. Langbehn, F. Steinicke, M. Lappe, G. F. Welch, and G. Bruder. In the blink of an eye: Leveraging blink-induced suppression for imperceptible position and orientation redirection in virtual reality. *ACM Trans. Graph.*, 37(4), 2018.

[15] D.-Y. Lee, Y.-H. Cho, and I.-K. Lee. Real-time optimal planning for redirected walking using deep q-learning. *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 63–71, 2019.

[16] Y.-J. Li, M. Wang, F. Steinicke, and Q. Zhao. Openrdw: A redirected walking library and benchmark with multi-user, learning-based functionalities and state-of-the-art algorithms. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 21–30, 2021.

[17] J. Messinger, E. Hodgson, and E. R. Bachmann. Effects of tracking area shape and size on artificial potential field redirected walking. *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 72–80, 2019.

[18] T. Nescher, Y.-Y. Huang, and A. M. Kunz. Planning redirection techniques for optimal free walking experience using model predictive control. *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 111–118, 2014.

[19] C. T. Neth, J. L. Souman, D. Engel, U. Kloos, H. H. Bülthoff, and B. J. Mohler. Velocity-dependent dynamic curvature gain for redirected walking. *2011 IEEE Virtual Reality Conference*, pp. 151–158, 2011.

[20] A. Nguyen, M. Inhelder, and A. M. Kunz. Discrete rotation during eye-blink. In *AVR*, 2018.

[21] T. C. Peck, H. Fuchs, and M. C. Whitton. Evaluation of reorientation techniques and distractors for walking in large virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 15:383–394, 2009.

[22] T. C. Peck, H. Fuchs, and M. C. Whitton. Improved redirection with distractors: A large-scale-real-walking locomotion interface and its effect on navigation in virtual environments. *2010 IEEE Virtual Reality Conference (VR)*, pp. 35–38, 2010.

[23] S. Razzaque. *Redirected Walking*. PhD thesis, University of North Carolina at Chapel Hill, USA, 2005.

[24] S. Razzaque, Z. W. Kohn, and M. C. Whitton. Redirected walking. In *Eurographics*, 2001.

[25] R. A. Ruddle. The effect of translational and rotational body-based information on navigation. In *Human walking in virtual environments*, pp. 99–112. Springer, 2013.

[26] B. W. Sanders, G. Narasimham, B. Rump, T. P. McNamara, T. H. Carr, J. J. Rieser, and B. Bodenheimer. Exploring large virtual environments with an hmd when physical space is limited. *Proceedings of the 4th symposium on Applied perception in graphics and visualization*, 2007.

[27] 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*, 16:17–27, 2010.

[28] R. R. Strauss, R. Ramanujan, A. Becker, and T. C. Peck. A steering algorithm for redirected walking using reinforcement learning. *IEEE Transactions on Visualization and Computer Graphics*, 26:1955–1963, 2020.

[29] Q. Sun, A. Patney, L. yi Wei, O. Shapira, J. Lu, P. Asente, S. Zhu, M. McGuire, D. P. Luebke, and A. E. Kaufman. Towards virtual reality infinite walking. *ACM Transactions on Graphics (TOG)*, 37:1 – 13, 2018.

[30] J. Thomas and E. S. Rosenberg. A general reactive algorithm for redirected walking using artificial potential functions. *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 56–62, 2019.

[31] M. Usoh, K. Arthur, M. C. Whitton, R. Bastos, A. Steed, M. Slater, and F. P. Brooks. Walking>walking-in-place>flying, in virtual environments. In *SIGGRAPH '99*, 1999.

[32] D. Waller, E. R. Bachmann, E. Hodgson, and A. C. Beall. The hive: A huge immersive virtual environment for research in spatial cognition. *Behavior Research Methods*, 39:835–843, 2007.

[33] M. Wang, Z.-Y. Chen, W.-C. Cai, and F. Steinicke. Transferable virtual-physical environmental alignment with redirected walking. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[34] N. L. Williams and T. C. Peck. Estimation of rotation gain thresholds considering fov, gender, and distractors. *IEEE Transactions on Visualization and Computer Graphics*, 25:3158–3168, 2019.

[35] S.-Z. Xu, J.-H. Liu, M. Wang, F.-L. Zhang, and S.-H. Zhang. Multi-user redirected walking in separate physical spaces for online vr scenarios. *arXiv preprint arXiv:2210.05356*, 2022.

[36] S.-Z. Xu, T.-Q. Liu, J.-H. Liu, S. Zollmann, and S.-H. Zhang. Making resets away from targets: Poi aware redirected walking. *IEEE Transactions on Visualization and Computer Graphics*, 28(11):3778–3787, 2022.

[37] S.-Z. Xu, T. Lv, G. He, C.-H. Chen, F.-L. Zhang, and S.-H. Zhang. Optimal target guided redirected walking with pose score precomputation. In *2022 IEEE conference on virtual reality and 3D user interfaces (VR)*. IEEE, 2022.

[38] S.-H. Zhang, C.-H. Chen, Z. Fu, Y. Yang, and S.-M. Hu. Adaptive optimization algorithm for resetting techniques in obstacle-ridden environments. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[39] S.-H. Zhang, C.-H. Chen, and S. Zollmann. One-step out-of-place resetting for redirected walking in vr. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[40] M. A. Zmuda, J. L. Wonser, E. R. Bachmann, and E. Hodgson. Optimizing constrained-environment redirected walking instructions using search techniques. *IEEE Transactions on Visualization and Computer Graphics*, 19:1872–1884, 2013.