Optimal Pose Guided Redirected Walking with Pose Score Precomputation

Sen-Zhe Xu*
Tsinghua University
BIMSA

Tian Lv†
Tsinghua University

Guangrong He‡
Tsinghua University

Chia-Hao Chen§
Tsinghua University

Fang-Lue Zhang¶
Victoria University of Wellington

Song-Hai Zhang‖
Tsinghua University
BNRist

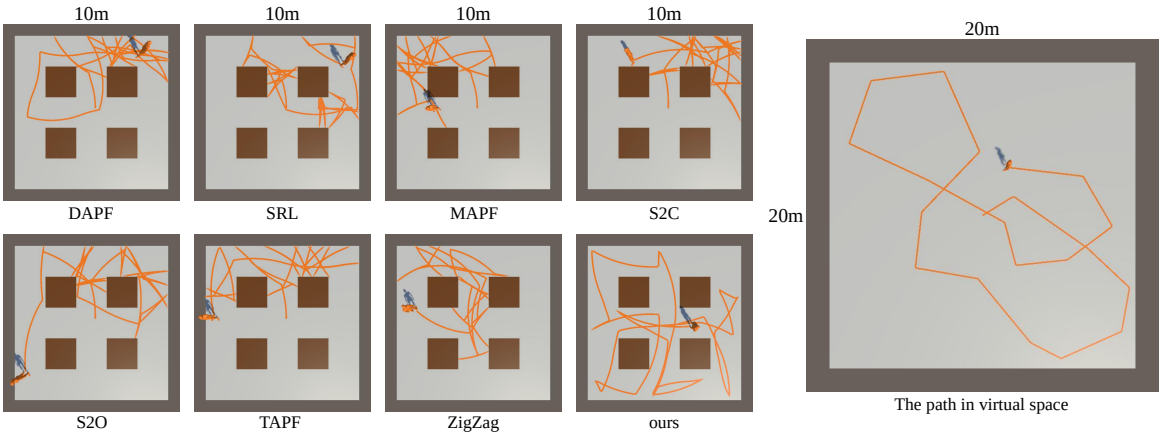

Figure 1: The physical trajectories of a user using different RDW controllers while walking along the same virtual path. The physical space is $10m \times 10m$ with obstacles, while the virtual space is $20m \times 20m$. By redirecting the user to the most favorable reachable position and orientation in the physical space, our method can better use physical space and trigger fewer resets.

Abstract

Redirected walking (RDW) aims to reduce the collisions in the physical space for VR applications. However, most of the previous RDW methods do not consider future possibilities of collisions after imperceptibly redirecting users. In this paper, we combine the subtle RDW methods and reset strategy in our method design and propose a novel solution for RDW that can make better use of physical space and trigger fewer resets. The key idea of our method is to discretize the representation of possible user positions and orientations by a series of standard poses and rate them based on the possibilities of hitting obstacles of their reachable poses. A transfer path algorithm is proposed to measure the accessibility among standard poses and is used to support the calculation of the scores of standard poses. Using our method, the user can be redirected imperceptibly to the optimal pose with the best score among all the reachable poses from the user's current pose during walking. Experiments demonstrate that our method outperforms state-of-the-art methods in various environment sizes and obstacle layouts.

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

*e-mail: xsz@mail.tsinghua.edu.cn
†e-mail: lvt18@mails.tsinghua.edu.cn
‡e-mail: hgr18@mails.tsinghua.edu.cn
§e-mail: accplusjh@gmail.com
¶e-mail: fanglue.zhang@vuw.ac.nz
‖e-mail: shz@tsinghua.edu.cn (corresponding author)

1 Introduction

Redirected walking (RDW) technology enables users to explore a large virtual environment within a limited physical space. The subtle RDW strategies imperceptibly fine-tune users' travel direction and speed when they are walking in the virtual space to help users avoid obstacles in the physical space. Once a user hits an obstacle, the overt [24] RDW technology will reset their orientation and position. A good reset strategy will also help the user turn to an optimal direction to avoid hitting obstacles again in the future. The subtle RDW algorithm and the reset strategy collaboratively reduce the number of collisions between users and physical obstacles in virtual applications. They both have been attracting researchers' attention in recent years.

However, most previous studies treat the subtle RDW and the reset strategy as two separate problems. The widely used subtle RDW methods are mainly based on heuristic ideas, such as steer-to-center (S2C), steer-to-orbit (S2O). Recently reinforcement learning-based methods such as steer-to-optimal-target (S2OT) [17] have been introduced to address the issue. However, they do not consider resetting the direction when a user hits an obstacle. In turn, the commonly used reset strategies, such as reset-to-center (R2C), two-one-turn, do not consider the influence of steering either. The recent work based on artificial potential functions (APF) [2] proposed several reset strategies in the combination of steering methods, such as reset-to-gradient (S2G) and modified reset-to-center (MR2C) [27]. Nevertheless, they only use the gradients of the user's current pose to decide the following direction, without considering how to transit to the optimal future pose.

This work aims to provide a new RDW method that can better use physical space and simultaneously reduce the number of resets needed during redirected walking. To achieve the goal, we need to estimate the possibility of hitting an obstacle in the future user

position and orientation after steering them. However, users' poses could have infinite possibilities since the space is continuous, making it an extremely complex problem. Therefore, we simplify it by defining several appropriate user positions and orientations in the physical space and always propose redirecting users to those favorite poses. Accordingly, the description of user poses, including position and orientation, is converted to a discretized form. We only need to measure the safety of limited numbers of standard poses. Besides, to help users explore the entire environment with redirected walking, we propose an algorithm to approach a smooth transfer path from one standard pose. It helps the precomputation of the pose scores by checking the distance between the pose and all the accessible poses and whether a collision will happen when traveling between them.

This paper presents and evaluates our approach to redirect the users to walk on a series of discrete standard poses in the physical space to reduce collisions. The method first steers the user to a standard pose imperceptibly, then gives each standard pose a score to measure its safety, and redirect the user to the most favorable pose that can be reached based on their intention and current pose. The method also acts as a unified controller for steering and reset since the scoring system can recommend optimal targets for reset when needed and steer the user. As a result, the user can be steered to the optimal pose while traveling and be reset to the optimal orientation when the steering is not possible. The decisions for steering and resetting are made together, reducing the number of resets in joint efforts. The main contributions of the paper are as follows:

- This paper proposes a novel RDW method that takes steering and reset into consideration simultaneously and evaluates it by simulation experiments.
- This paper proposes a discrete pose representation model and a transfer path algorithm to precisely redirect the user to the desired poses under the curvature gain's constraints.
- This paper proposes a pose safety value-based RDW strategy to quantitatively redirect the user to the optimal pose for a smaller number of resets.

2 Related Work

2.1 Redirected Walking

Razzaque firstly raised the concept of redirected walking in 2001 [18]. Early methods include using scaled translation [28] and augmenting motion in the user's intended direction [8]. Suma *et al.* [24] divides the RDW techniques into two categories, the *subtle* methods, and *overt* methods, depending on whether the adjustment of movement is perceptible. Nilsson *et al.* [13] further reviewed the RDW methods and divides the methods into four finer-grained categories: **scripted**, **reactive**, **predictive**, and **resetting**. The first three correspond to the *subtle* methods, while the last one is *overt*, which will be specifically discuss in subsection 2.2.

Scripted methods steer the user under the assumption that the users are walking along a pre-designed virtual path planned by the system developers in advance. Changing blindness [25] and introducing impossible overlapping spaces [26] are two influential strategies of scripted methods. However, scripted controllers need to be customized for different virtual environments and perform poorly if the user does not follow the pre-determined virtual path.

Reactive methods do not rely on the assumptions or predictions about the user's future path. They only rely on the user's current status and previous movements to work widely in various virtual and physical environments. Razzaque *et al.* [17] proposed three general reactive methods, namely, steer-to-center (S2C), steer-to-orbit (S2O), and steer-to-multiple-targets (S2MT). S2C constantly redirects the user to the center of the physical space. S2O redirects the user to a circle around the physical center. S2MT redirects the user to one of the pre-defined physical waypoints. Hodgson *et al.* [7] added a new strategy named steer-to-multiple+center, which includes the physical center to the pre-defined waypoints in S2MT. They also showed that S2C outperforms other reactive algorithms in most scenarios, while S2O has advantages when the user walks along a long straight virtual path. Azmandian *et al.* [1] further reinforced Hodgson's conclusion by comparing several reactive algorithms in different physical environment sizes and aspect ratios. In recent efforts, Chen *et al.* [4] described an RDW algorithm for irregularly shaped and dynamic physical environments. Lee *et al.* [9] proposed steer-to-optimal-target (S2OT) for RDW. Different from S2MT, they estimate optimal steering target through reinforcement learning. Chang *et al.* [3] and Strauss *et al.* [23] also proposed methods to redirect the users using reinforcement learning. Thomas *et al.* [27] proposed an RDW method based on the potential artificial fields. Bachmann *et al.* [2] made the APF method allowing multiple users to walk in the same tracking space. Messinger *et al.* [11] improved the APF method for irregular tracking space.

Predictive methods incorporate the user's future movements prediction into the redirection. Predictive controllers can be effective since they tend to predict the future information of the system, but their performance will decrease if the prediction is not accurate. Zmuda *et al.* [31] presented a planning algorithm called Fully Optimized Redirected Walking for Constrained Environments (FORCE), which steers the user depending on the probabilistic prediction of the user's virtual path through a known virtual environment. Nescher *et al.* [12] also proposed a method named Model Predictive Control Redirection technique (MPCRed), which is a planning framework for determining the best redirection by analyzing the geometry of the virtual environment (VE). Recently Dong *et al.* [5] improved the artificial potential field methods by applying the future dynamic state predictions.

Some other methods choose to alter the user's path to avoid collisions. Simeone *et al.* [20] put virtual objects to represent physical obstacles to avoid the users walking into obstacles but found that users were likely to interact with the added virtual objects on the contrary. They later replaced the obstacles with surfaces that people would not typically walk onto, such as water or lava, and found that the user's path was successfully altered [19]. Sra *et al.* [22] implemented a method to procedurally generate virtual environments that match the walkable area of the physical environment. These methods changed the layout of the virtual space, which may cause semantic disharmony, and need to be customized for different virtual and physical pairs.

Our proposed method is reactive and does not rely on the customization of the VE and physical environment (PE). We make no physical and virtual space requirements, which means the users can walk freely anywhere in VE. It increases the difficulty of the problem but also increases the versatility of the method.

2.2 Reset Strategies

Though *subtle* RDW tries to minimize the collisions, users still have a chance to hit the boundaries or obstacles. Therefore, *overt* techniques like a reset need to be used to ensure the users' safety. The reset technique pauses the virtual experience and then changes the user's orientation and position to favorable. The reset maneuver reduces users' immersive experiences and should be used as few as possible.

The most common reset strategy is two-one-turn, proposed by Williams *et al.* [29]. It notifies the users to turn around while doubling the users' rotation in VE. So a 360° virtual turn yields just a 180° physical turn, making the user turn to the direction they came physically while keeping the virtual state unchanged. Researchers have studied softer ways to achieve reset to reduce the breaks in immersion. Peck *et al.* [14–16] proposed to use distractors in VE to let users focus on while being reorientated.

Some methods have been proposed to reduce the collisions to make the user reorientation in a more favorable direction. Reset-to-center(R2C) is a general method, which always resets the user's

direction to the center of the physical space. Other reset techniques may be specific to the RDW controller. Thomas *et al.* [27] introduced three improved reset strategies, including reset-to-gradient(R2G), modified reset to center (MR2C), and step-forward reset to gradient(SFR2G), which works in collaboration with the potential field controllers [2, 27]. Other reorientation techniques such as creating a portal [6] or deploying a narrator inside the virtual environment [30] also help the user adjust their traveling direction.

In general, the reset technique is only used when RDW techniques fail to prevent collisions. However, our method considers resetting and steering simultaneously. In our method, reset can be applied without encountering obstacles, making the user be reset in advance to avoid collisions in the future.

3 METHOD

In order to reduce the frequency of user reset, we need to consider the possibility of encountering obstacles for each location and orientation when redirecting users in the physical space. However, the physical space is continuous, making it difficult to evaluate the value for an infinite number of states. Therefore, we propose to use discrete standard poses to express the user's position and orientation. We then find a transfer path to guide the walking between each pair of poses, which is also used for rating each pose by checking all its accessible poses. With the scores of each standard pose, we just need to redirect the user to the best reachable standard pose, i.e., the standard pose that is the least likely to encounter obstacles to reduce future collisions.

3.1 The Discrete Descriptor of the Physical Space

Floor tiles inspire the representation of positions in real rooms. We assume that square floor tiles with side length δ are tiled on the ground in the physical space. The corners of the tiles are $T = \{t_1, t_2, \ldots, t_n\}$ as the *standard positions* in physical space. The nearest *standard position* then represents a user's location. The orientation of a user is also discretized by m angles, forming a set of orientations $\Omega = \{\omega_1, \omega_2, \ldots, \omega_m\}$, called *standard orientations*. The closest *standard orientation* represents the user's orientation.

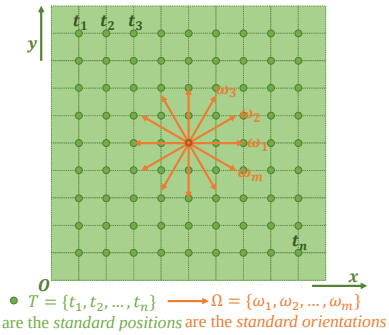

Figure 2: The discrete descriptor of the physical space.

The states containing a *standard position* and a *standard orientation* are called *standard poses* $P = \{(t, \omega) \,|\, t \in T, \omega \in \Omega\}$. On the one hand, the standard poses can describe the user's pose under a tolerable error range. On the other hand, they have a limited quantity that is easy to analyze. We stipulate that the user can only walk on the standard poses to prevent accumulation errors. This stipulation is unseen by the users. If the user is currently not on the standard poses, we will insensibly redirect their walking to a certain standard pose. The advantage of the discrete representation is that we can estimate the probability of encountering obstacles for each pose and evaluate it as an explicit score.

3.2 Transfer Path Between Standard Poses

Given a start point and an end point and their tangent directions, we must design a smooth path to connect them to make the user walk from the current pose to the desired standard pose. Though countless curves meet such constraints (for example, a spline curve can connect two points with prescribed tangent directions smoothly), the curvature is hard to control to satisfy the gain threshold.

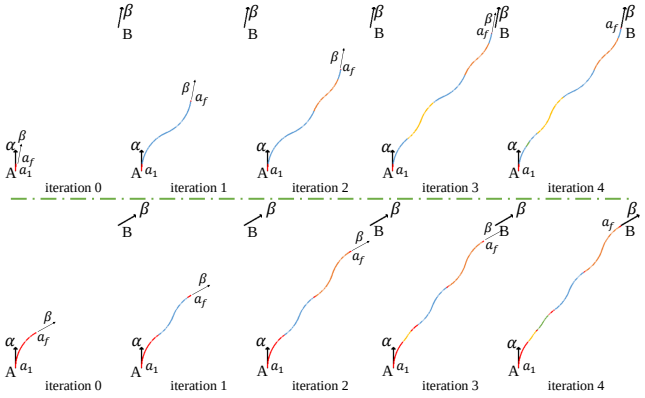

Figure 3: Two examples of the path transfer algorithm. The paths are found by continuously inserting 'S'-turns to reduce the residual gap to the target. Different colors are used to illustrate the inserted 'S'-turns in each iteration. In the figure, the curvature radius is 1m, and the distance of AB is 4.3m.

In this subsection, we propose to look for the transfer path from one pose to another under the constraints of curvature gain threshold. Suppose the starting position is A and the ending position is B. The required tangent directions of A and B are α and β, respectively. Our goal is to find a smooth path, i.e. an isometric point sequence $C = \langle a_1, a_2, \ldots, a_l \rangle$ in computational representation, to connect A and B. The distance between any two adjacent points in C is a fixed constant Δs, representing the distance traveled by the user in a frame time Δt. The tangent vector of a_i is defined as $\overrightarrow{a_{i-1}a_i}$, and the tangent direction of the first point a_1 is set to the user's current orientation. We use $\theta(a_i)$ to represent its tangent direction, and $\boldsymbol{\theta}(a_i)$ as the unit vector along $\theta(a_i)$. Since the curve C is smooth and its maximum curvature is limited by the gain threshold, the tangent direction of points in C must further satisfy a curvature constrain. So, we are going to find:

$$
\begin{aligned}
& C = \langle a_1, a_2, \ldots, a_l \rangle && (1) \\
& \text{s.t.} \quad a_1 = A, \theta(a_1) = \alpha, a_l = B, \theta(a_l) = \beta \\
& \qquad |\overrightarrow{a_{i-1}a_i}| = \Delta s \quad (1 < i \le l) \\
& \qquad \frac{\Delta s}{|\theta(a_i) - \theta(a_{i-1})|} = R \quad (1 < i \le l)
\end{aligned}
$$

Where R is the minimum radius of curvature allowed by the curvature gain threshold. In order to maximize the use of physical space, we set the radius of curvature of every point on C to be equal to R. We construct such C in an iterative manner. Firstly we assume that $\alpha \neq \beta$. We fix the starting point and the entrance tangent vector of C to satisfy: $a_1 = A$ and $\theta(a_1) = \alpha$, as shown in Fig. 3. As curve C grows from the starting point with the iterations, the index value of the last point l will also increase accordingly. Our idea is to preferentially satisfy $\theta(a_l) = \beta$ in each iteration, and then gradually adjust a_l to approach B through the iterations. For doing this, we first construct a circular arc:

$$
\left(x - A_x - R\cos\left(\alpha \pm \frac{\pi}{2}\right)\right)^2 + \left(y - A_y - R\sin\left(\alpha \pm \frac{\pi}{2}\right)\right)^2 = R^2 \tag{2}
$$

We sample points at every interval Δs on the arc to form C_0. The sampling starts from A and stops when a point with tangent β on the arc appears. As the arc from A can turn either left or right, we choose the inferior arc of the two arcs for sampling. If $\alpha = \beta$, this

inferior arc cannot be found, then we form C_0 with only a single point A: $C_0 = \langle A \rangle$. C_0 can ensure its starting point and entrance tangent direction is exactly A and α, and its exit tangent direction is β, as shown in Fig. 3(interation 0). But there is a residual distance between the end of C_0 and B. We then gradually reduce this residual distance through iterations by inserting 'S'-turns. Considering the relationship between a point and the tangent vector of its previous point, we group the points into two categories:

$$g(a_i) = \begin{cases} 1, & \text{if} \quad \boldsymbol{\theta}(a_{i-1}) \times \boldsymbol{\theta}(a_i) \cdot \boldsymbol{z} > 0 \\ -1, & \text{if} \quad \boldsymbol{\theta}(a_{i-1}) \times \boldsymbol{\theta}(a_i) \cdot \boldsymbol{z} < 0 \end{cases} \tag{3}$$

$\boldsymbol{z}$ is the normal vector of the floor, pointing up. From (3), $g(a_i) = 1$ if a_i is located on the left of its previous point's tangent direction, and we name it a left-turning point. Similarly, a_i is a right-turning point when $g(a_i) = -1$ and it is on the right side of the previous point's tangent direction. Since the tangent change between two adjacent points in C is fixed to $|\theta(a_i) - \theta(a_{i-1})| = \Delta s / R$, the tangent direction of each point can be obtained from its turning category and the tangent direction of its previous point:

$$\theta(a_i) = \theta(a_{i-1}) - g(a_i)\frac{\Delta s}{R} \tag{4}$$

Therefore, for any C, when inserted with the same number of left-turning and right-turning points, the tangent direction of the end point will keep unchanged, while the end point's position may shift. Using this principle, we can gradually reduce the residual gap for C by inserting proper 'S'-turns.

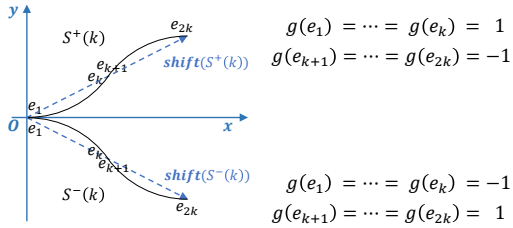

Figure 4: A pair of conjugate 'S'-turns which start from the origin and face the positive x-axis.

An 'S'-turn is a point sequence sampled from two spliced arcs and is composed of a series of left-turning points followed by the same number of right-turning points, making the curve 'S'-shaped. To be merged into C, 'S'-turns satisfy the interval and curvature constraints . So a piece of 'S'-turn' curve can be uniquely defined by the number of turning points $2k$:

$$S(k) = \langle e_1, \ldots, e_k, e_{k+1}, \ldots, e_{2k} \rangle \tag{5}$$

where the first k points turn left and the following k points turn right. If a pair of 'S'-turns with the same lengths has the opposite turnings orders, their shapes will just be flipped, as shown in Fig. 4. We use the superscripts '$\pm$' on $S(k)$ to distinguish these conjugate situations. Suppose a pair of conjugate 'S'-turns is starting from the origin and the set-off direction is along the positive x-axis (i.e., the starting orientation is 0°), the displacements of its tails relative to its head are:

$$\boldsymbol{shift}\left(S^{\pm}(k)\right) = \left[2R\sin\frac{k\Delta s}{R}, \pm 2R\left(1 - \cos\frac{k\Delta s}{R}\right)\right] \quad (1 \le k \le K) \tag{6}$$

K is the considered length limit of the 'S'-turns. Note that if the set-off direction of the 'S'-turn is not along the positive x-axis, and the set-off direction is an arbitrary orientation ω, the displacement vector will also rotate accordingly by multiplying by the rotation matrix:

$$\boldsymbol{shift}_{\omega}\left(S^{\pm}(k)\right) = \begin{bmatrix} \cos\omega & -\sin\omega \\ \sin\omega & \cos\omega \end{bmatrix} \boldsymbol{shift}\left(S^{\pm}(k)\right) \tag{7}$$

In the i-th iteration, we are going to find an optimal insertion position a_τ as well as an optimal shape of 'S'-turn for the current path C_{i-1}, so that when inserting the 'S'-turn to C_{i-1} at the position of a_τ, the end of the new path C_i is closer to B. When inserting the 'S'-turn, a_τ is considered as the previous point of e_1 in the definition of its tangent direction and turning category. We hope that the displacement direction caused by the inserted 'S'-turn is as close as possible to the direction of the residual gap, so the desired a_τ and k are:

$$a_\tau, k = \underset{a_\tau \in C_{i-1}, 1 \le k \le K}{\arg\min} \left| \varphi\left(\overrightarrow{a_l B}\right) - \theta(a_\tau) - \varphi\left(\boldsymbol{shift}_{\theta(a_\tau)}\left(S^{\pm}(k)\right)\right) \right| \tag{8}$$

$\varphi(\cdot)$ represents the argument of a vector. (8) is solved by enumeration or the Nelder–Mead method [21] in the variable domain. After finding a_τ and k, C_i is constructed by splitting C_{i-1} at a_τ and inserting the corresponding 'S'-turn. As the turning category of each point is known, the new positions and tangent directions of the points after a_τ can be inferred and updated.

If in a certain iteration n, the new residual distance is larger than the current one, we then stop iterating and use C_{i-1} as our final C. We denote the found path as $C(A, \alpha, B, \beta)$. We next consider the scale of $C(A, \alpha, B, \beta)$'s residual distance. We accept the path if the residual distance is smaller than a threshold ε.

3.3 Score Estimation of Standard Poses

To find out the most favorable pose to redirect the user to, we set a score for each standard pose to indicate the level of safety when a user sets off from it. Note that the safety of a pose is affected by the safety of the accessible poses from itself. So this can be solved as a dynamic programming problem. We need to figure out the accessibility of the standard poses in the physical environment for score estimation.

We enumerate all the pairs of standard poses and use the transfer path algorithm in the subsection 3.2 to calculate the curvature-gain-satisfied path between the pose pairs. Denote a pair of poses as $(U_t, U_\omega) \in P$ and $(V_t, V_\omega) \in P$, if the path $C(U_t, U_\omega, V_t, V_\omega)$ does not exist, (V_t, V_ω) is labeled as not accessible by (U_t, U_ω). Otherwise, we should verify whether the path exceeds the walkable space. We traverse the points of $C(U_t, U_\omega, V_t, V_\omega)$ in a binary search manner and use the ray casting algorithm to determine whether the point is in an obstacle or outside the room. If all the path points are not in obstacles or outside the room, we deem that (V_t, V_ω) is accessible by (U_t, U_ω). The accessibility between them is recorded on a map for future inquiries.

We now estimate the scores of poses for avoiding resets. It is a recursive process. The score of a pose is calculated based on the scores of its accessible poses, including the poses accessible after applying a reset. The estimated score of pose (U_t, U_ω) after round i is denoted as $Q_i(U_t, U_\omega)$. The estimation is based on the following considerations:

1. The score of (U_t, U_ω) is based on other poses that can be transferred to.

2. The farther the user can go from (U_t, U_ω), the safer (U_t, U_ω) is. Therefore, if the user can reach a pose (V_t, V_ω) from (U_t, U_ω), we should use the distance $|U_t V_t|$ as a reward for this path.

3. Future reset times need to be minimized. So if the user can reach a pose (V_t, V_ω) from (U_t, U_ω) only by being reset first, we should give a penalty η to this path.

Therefore, in the i-th round of the score estimation of (U_t, U_ω), we firstly check out all the poses that can be accessed from $(U_t, *)$, where $*$ represents all *standard orientations* in Ω. We denote the set of the found poses as $Access(U_t, *)$. When $*$ is just equal to U_ω, the user can reach the found pose without resetting; otherwise, a user needs to be reset to the direction represented by $*$, and then walk to the corresponding pose. For all the pose in $Access(U_t, *)$,

we use their current scores to estimate $Q_i(U_t, U_\omega)$, considering the path reward and reset penalty:

$$Q_i(U_t, U_\omega) = \max_{(V_t, V_\omega) \in Access(U_t, *)} (|U_t V_t| + \lambda Q_{i-1}(V_t, V_\omega) - reset \cdot \eta) \tag{9}$$

$\lambda \in (0,1)$ is an attenuation coefficient of the potential next pose. It decays the next poses' influence to make the score estimation converge. $reset \in \{0,1\}$ is a binary judgment flag of reset. If the pose transfer demands a reset action, a penalty η will cost to the score reward.

The scores of all standard poses are set to zeros at the beginningIn the i-th round of estimation, we update the score estimates for all poses $(U_t, U_\omega) \in P$ using equation (9). The score estimation is carried out round by round until the estimated score converges.

3.4 The Inference Phase

In the inference phase of the method, the RDW controller is working based on the user's displacement and rotation within the time of each frame, Δt. The controller redirects the user from the current pose to the most favorable standard pose when the user walks forward. To achieve that, the controller firstly finds the closest standard pose of the user's current physical pose and also uses this standard pose to represent the user's current pose. Although the representation pose may have a deviation from the users' actual pose, we omit the impact of this small error on the controller's decision-making.

The controller then finds the accessible pose set of the standard pose and uses the equation (9) to select the target pose with the best rewards to go. The scores of the candidate target poses are obtained by looking up the score table that has been iteratively estimated. As the user's actual pose may have a deviation from the representation pose, the controller then uses the transfer path algorithm in subsection 3.2 to verify the path existence from the actual pose to the selected target pose. Since the actual pose is very close to the representation standard pose, the path will exist in most cases. If the path does not exist, the controller then continues to find the sub-optimal target pose until the path to the target standard pose exists. The controller uses the transfer path to steer the user and uses the maximum curvature gain to follow the transfer path as he walks forward in the virtual space. If the user keeps moving forward, the user will be redirected precisely to the selected target standard pose. Moreover, if the user rotates or reaches the end of the path, the controller repeats the above process and steers the user using the recalculated new transfer path.

When the user moves forward, the translation gain is determined by the pose scores of the user's left-hand and right-hand orientations. We consider two tentative poses, of which the positions are the same as the user's current position and the orientations are the user's current orientation plus 90° or minus 90°. We also use the scores of the closest standard poses as their scores. If their scores are both *small*, it indicates that the user might be walking along a narrow aisle. In that case, we use the maximum translation gain to speed up the user, making them unlikely to stop in a narrow aisle where resets may easily happen. In other cases, the translation gain is set to 1.0.

When the user rotates, the rotation gain is determined by the scores of the current user pose and the following standard pose along the rotation direction. If the current score is *large*, we apply the minimum rotation gain to slow down the rotation. If the current score is *small*, we apply the maximum rotation gain to speed up the rotation. Otherwise, we look at the score of the following standard pose along the rotation direction. If the score of the future pose is *large*, we speed up the rotation, and if the next pose has a *small* score, we slow the rotation down. In other cases, the rotation gain is set to 1.0. Then we can give the user a greater chance of staying in a favorable pose after an arbitrary rotation.

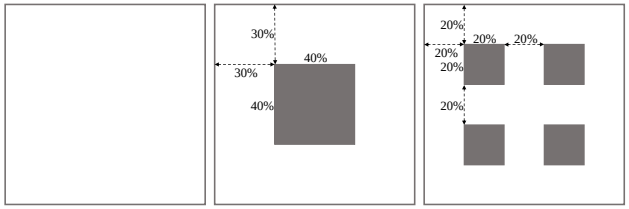

Figure 5: Environment layouts used for testing. A is a control environment with no physical obstacles. B is a moderately complex environment with an obstacle in the middle. C is a highly complex environment with scattered multiple obstacles.

Finally, we consider a score to be *big* if it exceeds the score of two-thirds of the standard poses, and we consider a score to be *small* if it is lower than two-thirds of the standard poses.

4 Evaluation

4.1 Implementation Details

In our experiment, the spacing between *standard positions* δ is set to 0.5m. This length is slightly larger than the width of an adult's shoulder ($34cm \sim 38cm$), which is considered sufficient for describing the user's position in the room. The number of *standard orientations* m is set to 12, which means that the angle between two adjacent *standard orientations* is 30°. The smallest radius of curvature allowed in the path R is set to 7.5m, which is a commonly employed threshold [1, 7, 27]. We use $K = 90$ as the maximum considered length of the 'S'-turns. The residual distance threshold ε for path acceptance is set to 0.1m, a tolerable error compared to the human body scale. The walking position error does not accumulate since our RDW framework will always redirect users to a new *standard position*. In the score estimation stage, we use $\lambda = 0.999$ as the attenuation coefficient to amplify the impact of future reset penalties on the current decision. The scale of the reset penalty η is determined by the size of the physical space and we use the area of the physical space as the value of η. In the inference phase, the translation gain is from 0.86 to 1.26, and the rotation gain is 0.8 to 1.49. We use the same gains thresholds for the other methods to compare. For the fast calculation of the pose accessibility, if it has been verified that $(V_t, V_\omega) \in P$ is accessible from $(U_t, U_\omega) \in P$, and $\overline{U_\omega} \in \Omega$, $\overline{V_\omega} \in \Omega$, where $\overline{*}$ is the reverse orientation of $*$, we directly recognize that $\left(U_t, \overline{U_\omega}\right)$ is accessible from $\left(V_t, \overline{V_\omega}\right)$ at the same time. This technique can speed up the calculations.

4.2 Environment Layouts

We evaluate the RDW algorithms using three physical environment layouts: environments A, B, and C, as shown in Fig. 5. Environment A is a free environment with no obstacles, used as a control layout. Environment B is a moderately complex environment. It has a single obstacle centered along both x boundary and y boundary, with a side length of 40% of the environment side length. Environment B is a representative sample of the environments with an obstacle in the middle. Environment C is a highly complex environment. It has four evenly distributed obstacles with a side length of 20% of the environment, making the aisles extremely narrow. C is a representative sample of the environments with scattered multiple obstacles. All layouts are square with a side length of 10m or 20m. To make our method adaptable to various environments, we have no requirements on the shape of obstacles in the virtual space, and the users can walk freely in the virtual space. It increases the difficulty of the test because the virtual waypoints can be freely and randomly distributed without any prior constraints.

4.3 Simulation Design

In order to evaluate the algorithm and eliminate the influence of randomness, plenty of tests are often needed for the RDW algorithms. Simulation can avoid the enormous cost of live user studies, and it is common to test the RDW algorithm. We thus evaluate our RDW controller using simulation experiments. The settings of our simulation design are mostly consistent with that of Thomas *et al.* [27]. The simulated user in the virtual space initially stands on the first waypoint. Every time the user reaches a waypoint, he will turn around to face the next waypoint and then walk straight to the next waypoint. This process continues until the user reaches the last waypoint. The user's initial position in the physical space is set at the center of the environment and facing towards the lower Y boundary. The initial position outside the obstacle is on the Y-axis for the physical space with obstacles in the center of the environment. The RDW controller receives the user's displacement and rotation in the virtual space in each frame and redirects the user's physical movement. If the user needs to be reset, we rotate the user in the physical space with a reflex angle (greater than 180°) to the desired direction, meanwhile rotating the user by 360° in the virtual space. The worst case of this strategy is equivalent to two-one-turn, a common reset technique in the RDW literature, where the user rotates 360° in the virtual space and turns back in the physical space.

The simulated user in the virtual space translates at a constant speed of 1m/s and rotate at a constant rate of $\pi/2$ rad/s. Our simulation runs at a frame rate of 30 FPS on a laptop with an Intel Core i7-8750H CPU 2.20GHz and 32GB of RAM.

5 Experiments and Results

We compare our method with several state-of-the-art RDW methods, including Steer-to-Center(S2C) [7], Steer-to-Orbit(S2O) [7], Zigzag [17], Thomas *et al.*'s APF (TAPF) [27], Messinger *et al.*'s APF (MAPF) [11], Dynamic APF (DAPF) [5] and the steer-by-reinforcement learning method (SRL) [23]. We have briefly introduced the above methods in Section 2. The experiments were conducted with the help of OpenRDW [10].

5.1 Experiments with Random Paths

We first test our method with the other methods using randomly generated virtual paths. We use the same method as in [1] to generate random virtual user trajectories. Each waypoint is generated with a random interval from 2m to 6m from the previous waypoint with a random angle within $-\pi$ and π between the directions of the neighboring waypoints. We test the algorithms with 100 virtual paths in every physical environment layout, each containing 100 randomly generated virtual waypoints. The same 100 virtual paths were applied to test the comparison algorithms. We analyze the *number of resets* and *average virtual distance between resets* under the three environments. A significance level of $\alpha = 0.05$ is applied for all the tests.

5.1.1 Environment A

We firstly analyze the performance of all the methods in environment A, under the condition of 10m and 20m side lengths. For each side length, a method has 100 observations of the *number of resets* and *average virtual distance between resets*, so we use Kolmogorov-Smirnov test to figure out whether the results are normally distributed. The test denied the assumption that the results follow normal distribution for almost all the methods, so we use the non-parametric technique Kruskal-Wallis H test to compare these data. We show the distributions of the observations for all the methods using the box plots in Fig. 6 and report the detailed medians(Mdn) and the inter-quartile ranges(IQR) in the supplementary.

We can see that our method has a relatively advantageous *number of resets* and *average virtual distance between resets* among the majority of the methods. Kruskal-Wallis H test verified that the results of the methods have statistically significant differences, in the *number of resets* in 10m ($H(7) = 304.571, p < .001$), *number of resets* in 20m ($H(7) = 492.382, p < .001$), *average virtual distance between resets* in 10m ($H(7) = 302.779, p < .001$) and 20m ($H(7) = 488.760, p < .001$). So we further conduct post hoc pairwise comparisons using Mann-Whitney U test to find out if there are any differences between our method and other methods. The p values are adjusted by the Bonferroni method. The post hoc comparisons results are reported in the supplementary. The post hoc comparisons show that our method has statistically significant differences with DAPF, SRL, MAPF, S2C, and S2O. From the box plots in Fig. 6 we can find that our method is superior to these methods. From Fig. 6 we can also find that TAPF, Zigzag, and our method have similar median values, and the Zigzag slightly outperforms our method in the 10m layout in terms of the median value. However, the Mann-Whitney U test finds no significant difference between our method and the two methods.

5.1.2 Environment B

For environment B, the Kolmogorov-Smirnov test indicated that most of the results are not normally distributed for both the two metrics in both the 10m and 20m spaces. So the non-parametric techniques are applied to analyze these results. The distributions of the *number of resets* and *average virtual distance between resets* are shown in Fig. 6, and we report the detailed medians(Mdn) and the inter-quartile ranges(IQR) values in the supplementary.

For the 10m size layout, the Kruskal-Wallis H test finds that not all methods have a same *number of resets* distribution ($H(7) = 293.712, P < .001$) and *average virtual distance between resets* distribution ($H(7) = 308.9, P < .001$). The further post hoc pairwise comparisons using the Mann-Whitney U test revealed that our method has a relatively smaller *number of resets* and a relatively longer *average virtual distance between resets* than most of the methods. But the comparisons find no significant difference between our method and the S2O in *number of resets* ($U = 3805.5, p = 0.097$) and *average virtual distance between resets* ($U = 4035.0, p = 0.515$). The p values are adjusted by the Bonferroni method. The details of the post hoc pairwise comparisons between our method and the comparison methods are shown in the supplementary.

The analysis of the 20m size layout by Kruskal-Wallis H test also reject the assumption that the distributions of *number of resets* ($H(7) = 571.447, P < .001$) and *average virtual distance between resets* ($H(7) = 574.805, P < .001$) are the same for all the methods. Mann-Whitney U test used for post hoc comparisons shows that the differences between our method and the most comparison methods are significant. In contrast, the MAPF method has no significant difference from our method in this case.

5.1.3 Environment C

For environment C, Kolmogorov-Smirnov tests show that not all results follow a normal distribution. So we use non-parametric techniques to analyze these data. The result distributions are shown in Fig. 6, and the median(Mdn) and inter-quartile range(IQR) values of each group are reported in the supplementary. From the figure, we can see that our method has the best performance of both the *number of resets* and *average virtual distance between resets*.

Kruskal-Wallis H test is applied to analyze the distribution difference of the results. We find that the distributions of *number of resets* in each group are significantly different in the 10m environment, $H(7) = 471.009, P < .001$, as well as the 20m environment, $H(7) = 450.248, P < .001$. We then use the Mann-Whitney U test for post hoc comparisons. The Bonferroni correction is used to correct the p values. The post hoc comparisons found that our method outperforms all comparison methods in *number of resets* in both

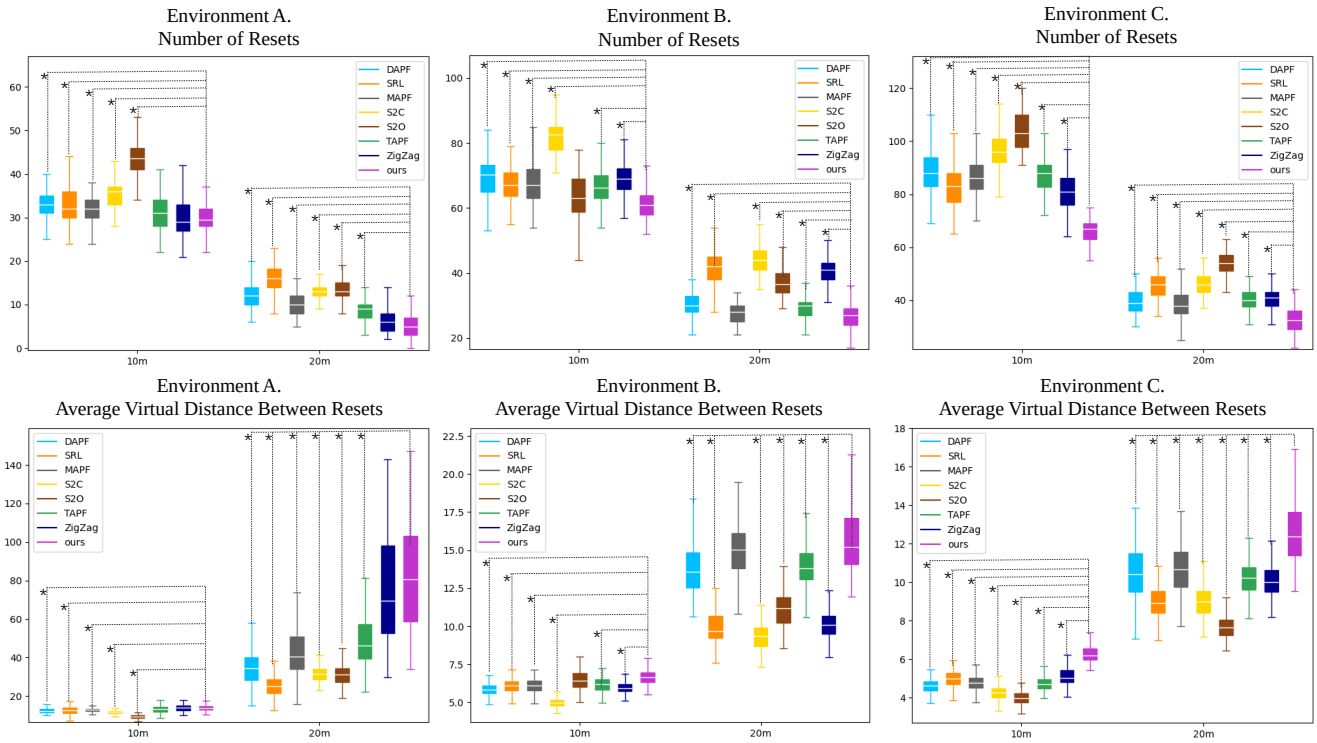

Figure 6: The box plots of the results of experiments with random paths. The distribution of the outliers is not included in the whiskers. The dotted lines with an asterisk (*) indicate the significant difference from our method ($p < 0.05$)

10m and 20m environments, and the differences are statistically significant. The comparison details are reported in the supplementary.

The situation is similar when analyzing the *average virtual distance between resets*. Kruskal-Wallis H test shows that the methods have a different distribution of *average virtual distance between resets*, whether in the 10m space ($H(7) = 474.924, P < .001$) or the 20m space ($H(7) = 457.695, P < .001$). The post hoc comparisons confirmed that our method has a longer *average distance between resets* and the difference with all the comparison methods is statistically significant.

5.1.4 Discussion

The results of the *number of resets* and *average virtual distance between resets* in environments A, B, and C demonstrate that our method surpasses the previous methods, especially when the layouts have more obstacles and smaller spaces. In the most complex environment (C), our method outperforms the other methods and shows more significant advantages in the 10m space than the 20m space. The reason is that our method pre-calculates the score of the specified poses and precisely guides the user to the most favorable reachable pose every time. This mechanism helps the user avoid the obstacles effectively and better utilize the physical space. In contrast, the other methods only infer the control action from the current user pose information or use heuristic targets as references. Thus they do not ensure the target pose to be the optimal reachable pose from the current state. Moreover, some comparison methods do not combine the reset strategy with steering. They only rely on basic reset strategies such as two-one-turn, making it difficult for them to choose a good reset direction when trapped by obstacles. In Environment B, with fewer obstacles, our method also outperforms the other methods. However, the maximum advantage is not as much as that in Environment C. We note that the effect of S2O is significantly improved compared to other methods, especially in the 10m layout. The reason is that there is only an obstacle in the center, the orbit used for steering the user in S2O will bypass the obstacle. On the contrary, the effect of S2C is significantly reduced because it always redirects the user to the center of the environment where the obstable is placed in this environment. It demonstrates that the results of S2C and S2O are highly affected by the specific physical environment. Instead, our method uses the precomputed pose scores and the steering path algorithm to direct the user to bypass the obstacles in a deterministic way, and changes of the obstacle layout will not interfere with our approach. Therefore, our method is more generalizable. Environment A is a controlled environment with no obstacles, where we find that the difference among the result distributions of different methods is not that large. It may be because the methods tend to take similar actions in such a simple situation. Our method is at the forefront among the methods, and there is no significant difference between our method and the other methods in Environment A.

5.2 Experiments with Long Straight Paths

We further test the methods with long straight virtual walking, which is challenging for RDW controllers because it means the virtual space to be infinitely large. Here, the user's autonomous turning is no longer available to fold their path in physical space, so the RDW controller can only adjust their orientation to better use the physical space. To test the performance of different methods under the long straight virtual path, we make the simulated user walk along a straight path in the virtual space with a fixed length of 400m, and use different RDW controllers to redirect the user in physical space. The same environments A, B, and C described in subsection 4.2 are used for this experiment. The user's initial position and orientation in the physical space are randomly generated out of the obstacles. Each method is tested 100 times, and the same 100 initial poses are used for all the methods in the same layout. Since the virtual path

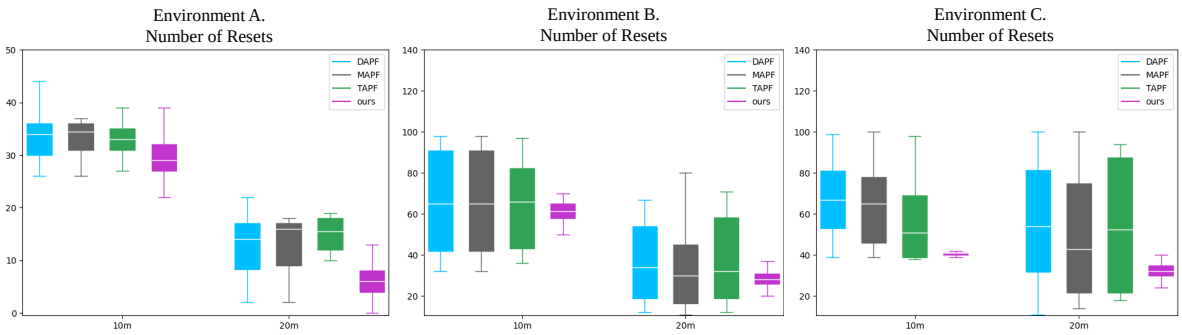

Figure 7: The box plots of the results of experiments with long straight paths. The distribution of the outliers is not included in the whiskers.

length is fixed, the *average virtual distance between resets* is linked to the *number of resets*. We only analyze the *number of resets* for the methods.

Since some of the methods to compare with do not take the reset direction into account, our experiment finds that these methods often get many resets. As users would not change their heading when walking along long straight paths, their performance depends heavily on their initial poses. The frequently resetting after the user appeared around the scene corners made us believe that it is pointless to compare the distribution with the inconsistent results of some methods. Thus we only compare the distributions with the APF based methods, which can reset according to the gradients to get rid of obstacles.

Fig. 7 shows the box plots of the *number of resets* of the different methods in the three environments. The medians and inter-quartile ranges of the *number of resets* are further reported in the supplementary. Our method has shown a relatively small *number of resets* when walking on long straight virtual paths, and the distributions of the *number of resets* between our method and the other methods are notably different. The longer the box-plot is, the more the starting pose influences the method's effect*number of resets*. Thus we can conclude that the performance of our method is essentially unaffected by the initial poses in various environments and sizes. Our method uses pre-calculated scores to guide the user's direction, accurately steering and resetting the user to the optimal reachable pose. So that no matter where the user appears initially, our method can quickly help him get away from the blocks with obstacles. We also find that when traveling along the long straight path in Environment C, resets are fewer than in Environment B. It is because the big obstacle stops the user from walking across the environment while walking in environment B. The users must navigate the obstacle, which causes a rise in the number of resets. Although environment C is the most complicated, the space between obstacles makes it easier for our method to move the user. Our transfer path algorithm can help find the optimal way to redirect the user to pass by the obstacles.

6 Limitation

There are several limitations of our method. Firstly, our method is not suitable for dynamic physical spaces. Compared with other reactive methods like S2C, our method requires preprocessing and precomputing the pose scores. The precomputation time is affected by the number of standard poses and the size of the physical space. Therefore, our method is incapable of adapting to dynamically changing physical spaces, since our method cannot guarantee that the precomputation is done in real-time for unpredictable physical space changes. Our method will be less effective if the user frequently changes their orientations. When they walk in a straight line in the virtual space, our method pre-calculates his future path to an optimal target in the physical space and steers the user to walk along that path with the curvature gains. It is favorable for walking in a long straight line in the virtual space. However, if the user constantly changes their walking direction, our method will have to recalculate the optimal target and the transfer path frequently. It will increase the computation cost and cannot ensure that the user reaches the original optimal target every time. Another limitation is that we only evaluated the algorithm using the simulation experiments. Although we can make an enormous number of tests, longer testing paths, and freely designed environments in the simulated environment, a full user study is important to test the real user experience and find the potential shortcomings that have not been found in our experiments.

7 Conclusion and Future Work

This paper brings a new RDW solution, which uses the pre-computed scores of several standard poses to redirect the users to the most favorable pose every time. Our method is based on the idea that users should always be redirected to the optimal target during walking. We represent user pose discretely and figure out the score indicating how unlikely the user is to hit obstacles for each standard pose. The RDW controller uses the scores to determine the optimal reachable target. A transfer path algorithm is proposed to steer the user from an arbitrary pose to the specified standard pose concerning curvature gain threshold, imperceptibly allowing the user to reach the desired pose. Experiments show that our method outperforms the state-of-the-art methods in various environment layouts and sizes, especially in the small environment with complicated obstacles. The test of the long straight virtual paths shows that our method has an advantage in walking on long straight paths, and the performance of our method is not affected by the initial pose.

Our work can be improved in the future. Our method does not consider the RDW of multiple users simultaneously. In future work, our work can be extended to support redirected multiple users by dynamically modifying the pose scores when other users obstruct the path. In addition, the current *standard positions* are distributed on a square grid, which might not be the best. Different grid shapes for the *standard positions*, such as the regular hexagonal grid, may increase the number of reachable standard poses while maintaining the same *standard position* density. We also plan to find a more effective discrete representation for standard poses in future work.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Project Numbers 61772298, 62132012), Research Grant of Beijing Higher Institution Engineering Research Center, Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology, and by a grant from the 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*, pp. 93–100, 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(5):2022–2031, 2019.

[3] Y. Chang, K. Matsumoto, T. Narumi, T. Tanikawa, and M. Hirose. Redirection controller using reinforcement learning. *arXiv preprint arXiv:1909.09505*, 2019.

[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 '18). Workshop on Everyday Virtual Reality*, 2018.

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

[6] S. Freitag, D. Rausch, and T. Kuhlen. Reorientation in virtual environments using interactive portals. In *2014 IEEE symposium on 3D user interfaces (3DUI)*, pp. 119–122. IEEE, 2014.

[7] 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, 2013.

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

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

[10] 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. IEEE, 2021.

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

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

[13] N. C. Nilsson, T. Peck, G. Bruder, E. Hodgson, S. Serafin, M. Whitton, F. Steinicke, and E. S. Rosenberg. 15 years of research on redirected walking in immersive virtual environments. *IEEE computer graphics and applications*, 38(2):44–56, 2018.

[14] 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(3):383–394, 2009.

[15] 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. In *2010 IEEE Virtual Reality Conference (VR)*, pp. 35–38. IEEE, 2010.

[16] T. C. Peck, H. Fuchs, and M. C. Whitton. An evaluation of navigational ability comparing redirected free exploration with distractors to walking-in-place and joystick locomotio interfaces. In *2011 IEEE Virtual Reality Conference*, pp. 55–62. IEEE, 2011.

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

[18] S. Razzaque, Z. Kohn, and M. Whitton. Redirected walking, eurographics 2001 proc., 2001. *On-line available: http://www. cs. unc. edu/eve/pubs. html*, 2001.

[19] A. L. Simeone, I. Mavridou, and W. Powell. Altering user movement behaviour in virtual environments. *IEEE transactions on visualization and computer graphics*, 23(4):1312–1321, 2017.

[20] A. L. Simeone, E. Velloso, and H. Gellersen. Substitutional reality: Using the physical environment to design virtual reality experiences. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 3307–3316, 2015.

[21] S. Singer and J. Nelder. Nelder-mead algorithm. *Scholarpedia*, 4(7):2928, 2009.

[22] M. Sra, S. Garrido-Jurado, C. Schmandt, and P. Maes. Procedurally generated virtual reality from 3d reconstructed physical space. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*, pp. 191–200, 2016.

[23] 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(5):1955–1963, 2020.

[24] E. A. Suma, G. Bruder, F. Steinicke, D. M. Krum, and M. Bolas. A taxonomy for deploying redirection techniques in immersive virtual environments. In *2012 IEEE Virtual Reality Workshops (VRW)*, pp. 43–46. IEEE, 2012.

[25] E. A. Suma, S. Clark, D. Krum, S. Finkelstein, M. Bolas, and Z. Warte. Leveraging change blindness for redirection in virtual environments. In *2011 IEEE Virtual Reality Conference*, pp. 159–166. IEEE, 2011.

[26] E. A. Suma, Z. Lipps, S. Finkelstein, D. M. Krum, and M. Bolas. Impossible spaces: Maximizing natural walking in virtual environments with self-overlapping architecture. *IEEE Transactions on Visualization and Computer Graphics*, 18(4):555–564, 2012.

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

[28] B. Williams, G. Narasimham, T. P. McNamara, T. H. Carr, J. J. Rieser, and B. Bodenheimer. Updating orientation in large virtual environments using scaled translational gain. In *Proceedings of the 3rd Symposium on Applied Perception in Graphics and Visualization*, pp. 21–28, 2006.

[29] B. Williams, G. Narasimham, B. Rump, T. P. McNamara, T. H. Carr, J. Rieser, and B. Bodenheimer. Exploring large virtual environments with an hmd when physical space is limited. In *Proceedings of the 4th symposium on Applied perception in graphics and visualization*, pp. 41–48, 2007.

[30] R. Yu, Z. Duer, T. Ogle, D. A. Bowman, T. Tucker, D. Hicks, D. Choi, Z. Bush, H. Ngo, P. Nguyen, et al. Experiencing an invisible world war i battlefield through narrative-driven redirected walking in virtual reality. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 313–319. IEEE, 2018.

[31] 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(11):1872–1884, 2013.