Efficiently Bypassing Obstacles to Reach Specified Locations: A Curvature Gain-Based Redirected Walking Strategy

Sen-Zhe Xu
Tsinghua University
xsz@tsinghua.edu.cn

Jia-Hong Liu
Tsinghua University
liujiaho23@mails.tsinghua.edu.cn

Ge Yu
Technology and Engineering Center for Space Utilization, Chinese Academy of Sciences
yuge@csu.ac.cn

Song-Hai Zhang*
Tsinghua University
shz@tsinghua.edu.cn

Abstract

In this paper, we propose a novel Redirected Walking (RDW) strategy that enables users to navigate to specified physical locations while avoiding obstacles in virtual reality (VR) walking experiences using curvature gain. Our method is capable of computing a range of paths to the specified target under the constraint of a curvature gain threshold. To enhance user immersion, the proposed strategy minimizes the frequency of changes in the steering direction of curvature gain, requiring no more than one adjustment before reaching the target. We introduce a descriptor to represent the computed paths and our approach is analytical, providing accurate solutions in a controlled time, and is adaptable to various space and obstacle layouts and target specifications. By exploiting the continuity of obstacle edges, our method quickly and accurately excludes occluded paths by considering only the endpoints of obstacle edges. We introduce applications such as *maximum walkable distance estimation* and *reachable area estimation* with our method to improve the performance of RDW controllers. We conducted extensive simulation experiments in both common single-player RDW scenarios and a novel multiplayer online RDW scenario with various physical space layouts to evaluate the effectiveness of our approach. Experimental results demonstrate that our approach significantly reduces the number of resets compared to state-of-the-art methods. We also conducted a live user study to evaluate our method and compare it with state-of-the-art methods, and the results also verified the effectiveness of our method in terms of user experience and number of resets.

***Keywords:** Redirected walking, curvature gain, obstacle avoidance, path descriptor*

*Song-Hai Zhang is the corresponding author.

1. Introduction

In recent years, Virtual Reality (VR) has undergone rapid development thanks to breakthroughs in hardware and software technology. This has transformed the way humans interact with computers and others, offering broad application prospects in various fields such as socializing, education, entertainment, training, and creation. However, there are still many challenges in providing users with an immersive locomotion experience when they wish to move in the VR environment. For example, real walking is considered the most natural way for locomotion in virtual reality, but the physical space of the user is often smaller than the virtual space, which forces users to frequently reset their position or orientation to continue their virtual journey.

Redirected Walking (RDW) [30, 40, 13, 23] is a technique that reduces user collisions in limited physical spaces by remapping the movements between virtual and physical spaces. It mainly takes advantage of the imprecision in human proprioception and vision, which is known as gains, to subtly manipulate the virtual environment. *Curvature gain* is one of the most frequently used gain types in the RDW algorithms. When the user walks straight to a target in the virtual space, curvature gain [36] can be applied to deflect the user's physical path by slowly rotating the virtual environment in every frame, enabling the user to follow a circular arc in the physical space.

In VR applications, users typically move in a straight line to reach virtual objects, locations or points of interests (POIs) rather than walking in curves, which gives RDW algorithms ample opportunities to apply curvature gains [1, 2, 44, 26, 10, 39, 45, 46]. By applying different magnitudes and directions of curvature gains, it's conceivable that the physical position of the user upon completing a segment of virtual straight walking will also differ. There-

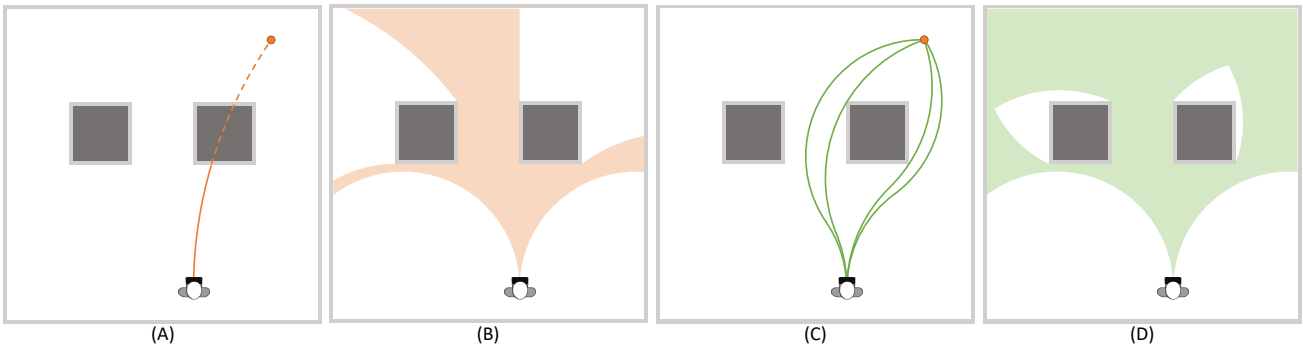

Figure 1: Redirecting the user to a specified target while bypassing obstacles in the physical space with curvature gain. (A) Redirecting the user with a trivial steering strategy to the specified target, where the path is easily blocked by obstacles. (B) With the trivial steering strategy, the area a user can reach (in orange) is rather limited. (C) Our method can find a series of collision-free paths to a specified target while satisfying the curvature gain threshold, and our paths only need to change the steering direction at most once before reaching the target. (D) Our method can steer the user to a wider area in the physical space (in green), thereby enhancing RDW controller design and assisting in redirecting the user to favorable positions to reduce the need for resets.

fore, the key question we aim to address here is, when the user walks straight in the virtual space, **whether it is possible to steer the user to a specified position in the physical space without encountering any obstacles with curvature gain?** Achieving this goal brings considerable benefits for designing RDW toolkit, such as we can designate a spacious position or a reset-friendly position as the user's physical destination, thus maximizing the user's walkable range after turning around or resetting.

Many influential RDW methods reduce collisions by operating on curvature gains. For example, heuristic-based method such as steer-to-center (S2C) [32] uses curvature gain to steer the user to the space center; artificial potential function based methods [44, 26, 4, 10] use curvature gain to push users away from obstacles; reinforcement learning based methods [21, 22, 39, 7] predict the gains and targets for steering the user; alignment-based methods [43, 45, 46] steer the user in the direction that the physical space and the virtual space are aligned, etc. However, these works can only decide the curvature gain based on the current state of the user, and cannot guarantee that the user will reach or pass through a certain physical position in the subsequent walking. As the physical location of arrival is not controllable, it is also impossible to specify the future location of the user in the physical space. To the best of our knowledge, there is currently no general RDW method to steer the user to specified positions while bypassing obstacles in complex physical spaces with curvature gain.

One trivial method to steer the user to a specified position is to steer the user with an arc path that passes through both the starting position and the specified position, while the tangent of the arc at the starting position is exactly the user's starting orientation, as shown in Fig. 1(A). However, this approach yields only a single path to the destination, which can be easily obstructed by obstacles in a complex physical environment. Additionally, the user's accessible area is also significantly limited with this trivial steering strategy (Fig. 1(B)).

In this paper, we propose a general redirected walking technique to steer the user to a specified physical position with curvature gain while bypassing all obstacles in a complex physical space. To minimize any discomfort that may arise from changes in curvature gain, we use a constant magnitude of curvature gain to steer the user, and only allow the direction of the curvature gain to flip once before the user reaches the specified position. We developed a method for computing the blocked path using the obstacle point, and by only computing paths occluded by certain endpoints on an edge, we can accurately exclude a batch of occluded paths that are blocked by obstacle edges. The paths found by our method are not singular, and our method can find a series of collision-free paths to one specified target while satisfying the curvature gain threshold, allowing for more selectivity in steering the user to the given target. Our approach is entirely analytical, avoiding iterative optimization methods or search strategies, so the results of our method are accurate and the computation time of our method is only linearly related to the number of obstacle edges.

Our method is general and can be used with different obstacle layouts and targets. The generality of our approach enables it to be applied to design variety of RDW toolkit with great effectiveness. For instance, our method could be used to efficiently estimate the maximum walkable distance from a given pose, or the reachable physical area for a given virtual distance. Through these applications, we can quantify the spaciousness and resetting benefit of physical

positions and select the best reachable physical position to steer the user.

We evaluated our method with both simulation experiments and live user study. For the simulation experiments, we use both a common single-user RDW task and a more challenging multi-user online RDW task [48] to test our method. The latter requires synchronous resetting of all users for an interactive and fair VR game environment, which poses a greater challenge for reducing the number of resets. We test our method with a variety of complex environment layout combinations. We also conducted live user studies to verify the experience of our method both objectively and subjectively. Our method is able to find a longer walkable way for the user and steer the user to a wider physical area, and the experimental results show that our method can make all users have a significant smaller number of resets than state-of-the-art methods, and cause less sickness increase.

The main contributions of our work are as follows:

- We propose a general RDW method of steering the user to a specified target while bypassing obstacles, which can be used for many applications.
- For better user immersion, our method uses a constant magnitude of curvature gain to steer the user and strategically limits the direction of the curvature gain to only flip once before the user reaches the specified position.
- All processes of our method are analytical, and our method can get a series of collision-free paths to one specified target in a controlled time.
- We conducted comprehensive evaluations including simulation experiments and live user studies to validate the effectiveness of our proposed method.

2. Related Works

2.1. Research on Curvature Gain and Associated Gains

Razzaque *et al*. [33] proposed to manipulate the user's walking trajectory by constantly rotating the virtual scene about the user while the user is walking in 2001, which was actually an operation of the curvature gain. Steinicke *et al*. [35, 36] later named this locomotion gain as Curvature Gain and defined its value as the inverse of the physical path radius (the curvature of the physical path).

A number of studies have been conducted to determine the detection threshold of curvature gain. Steinicke *et al*. [36] suggested that the radius of curvature of the curvature gain should not be less than 22 meters. Hodgson *et al*. [15] proposed to use 7.5 meters as the minimum radius of curvature gain. Grechkin *et al*. [14] found that the estimated minimum curvature radius can be 6.41m for the no translation gain condition, and can be even lower when retesting the users. Currently, a significant number of works follow the value recommended by Hodgson *et al*., taking 7.5 meters as the minimum radius of curvature gain.

The above works show that, the detection threshold for curvature gain isn't set in stone. Nguyen *et al*. [28] investigated the effect of environment size on curvature gain thresholds with the 2-alternative forced choice method. They also investigated the role of gender on curvature redirection thresholds [29], and found that women have higher curvature redirection thresholds than men. Bölling *et al*. [6] tested users' adaptation to long-term exposure to curvature gain and found that prolonged exposure to curvature gain led to a rise in the detection threshold. Matsumoto *et al*. [25] deceive users about path curvature by making users touch a curved wall while displayed a straight wall in the virtual space, and found that haptic cues can significantly affect the detection threshold for curvature gain.

Curvature gain is often used in combination with translation gain [35, 18, 17] and rotation gain [35, 31, 47], but their effects are distinct. Translation gain only scales the path length and does not change the path shape, whereas rotation gain only works when the user turns around in place. Since the curvature gain can modify the user's trajectory while the user is walking, many RDW strategies employ curvature gains to steer the user and avoid collisions with obstacles. In addition to considering straight walking, recently there are also some studies extending the concept of curvature gain to bending gains [19, 34], taking into account the user's bending movement in virtual environments.

Besides curvature gain, there exist other techniques to manipulate the mapping between the physical space and the virtual space, such as change blindness [41] and saccade blindness [20, 27, 42]. Nevertheless, curvature gain does not need to change the appearance and content of the virtual environment, nor does it require any additional equipment or sensors. Therefore, curvature gain is still a popular and common way for redirecting users.

2.2. Redirected Walking Strategies using curvature gains

Razzaque *et al*. [32] propose three heuristic strategies to steer users with curvature gain: steer-to-center (S2C), steer-to-orbit (S2O) and steer-to-multiple-targets (S2MT). They try to steer the user to the center of the space, a predefined circular orbit and a set of predefined waypoints, respectively. Hodgson *et al*. [15] integrate S2C into S2MT to steer the user back to the center faster than S2MT.

As the principle of the heuristic strategies is fixed, they tend not to handle the pysical space with complex obstacles well. Recently Artificial Potential Functions (APF) based methods [44, 26, 4, 10] have been proposed to reduce collisions in arbitrary layouts. Taking the work of Thomas *et*

al. [44] as an example, they generate a potential field with an artificial potential function in the physical space, and push the user away from obstacles or pull the user towards the open space according to the potential field. They also introduce three reset strategies based on the potential field.

The reinforcement learning-based methods [21, 22, 39, 7] are another kind of strategies to steer users. Deep neural networks can be used to recommend targets for steering the users [21, 22], or directly predict locomotion gains [39, 7].

Based on the consideration that the user will not actively encounter obstacles in the virtual space, the alignment-based strategies [43, 45, 46] are also proposed to steer the user. They steer the user to the direction where obstacles in the physical space are more aligned with that in the virtual space.

There are some other RDW strategies with curvature gains designed specifically for dynamic environments and multiplayer. Chen *et al*. [8] proposed two greedy strategies, namely steer-to-farthest and trapezoidal road map, to adapt RDW to irregularly shaped, dynamic physical environments. Additionally, they introduced various techniques for planning algorithms to support irregularly shaped and dynamic physical environments. Bachmann *et al*. [5] and Azmandian *et al*. [2] explored how to steer two users at the same time. Dong *et al*. [12] presented a method for redirecting three users in a shared space. Dong *et al*. [11] further proposed a density-based RDW to steer multiple users and avoid collisions among users. Xu *et al*. [48] proposed a solution for multi-user online VR games. They make the resets of different users in different physical spaces to occur simultaneously, thereby improving the interactivity and fairness of online VR games.

However, the above methods lack the ability to steer the user to a specified position with the curvature gain. Controllably steering the user to specified positions can enhance the authenticity of passive haptic feedback and improve their interaction with the surrounding physical environment [37, 38, 43, 9]. Although Xu *et al*. [48] plan paths to specified positions based on predefined circles, it lacks flexibility and suffers from obstacles. Recently Xu *et al*. [49] proposed a method to steer the user to the specified physical position and orientation using curvature gain. However, their method can only generate a single path through iterative optimization and cannot account for obstacles. Furthermore, their method may change the curvature gain direction multiple times during walking, increasing the risk of user discomfort.

3. Bypassing Obstacles with Curvature gain

3.1. Paths to a specified position and its representation

Suppose the user's starting position in the physical space is (x_s, y_s) and the starting physical orientation is θ_s. For the convenience of expression, we turn to the local coordinate system for observation, where we take (x_s, y_s) as the origin and θ_s as the positive direction of the x-axis. In order to do this, we only need to transform any point (x_{glb}, y_{glb}) in the world coordinate system according to the following relationship, and then we can get its coordinates (x_{loc}, y_{loc}) in the local coordinate system:

$$\begin{bmatrix} x_{loc} \\ y_{loc} \end{bmatrix} = \begin{bmatrix} \cos\theta_s & \sin\theta_s \\ -\sin\theta_s & \cos\theta_s \end{bmatrix} \begin{bmatrix} x_{glb} - x_s \\ y_{glb} - y_s \end{bmatrix} \tag{1}$$

In the local coordinate system, the user's starting position becomes $(0,0)$, and the user's starting orientation becomes the positive x-axis. Unless otherwise specified, in the following we use the local coordinate system for analysis.

Now our goal is to steer the user to a specified physical position (x_e, y_e) with the curvature gain when the user walks straight in the virtual space. As it often requires an unrealistic walking length to steer the user to the back side of their starting position due to the gain threshold limitation, this goal only makes sense when (x_e, y_e) is on the front side of the user, i.e. $x_e \geq 0$. For practical consideration, we don't try to steer the user to any target behind them and determine (x_e, y_e) is unreachable when $x_e < 0$.

To achieve our goal, a trivial way is to steer the user with an arc that passes through both the starting position $(0,0)$ and the specified destination (x_e, y_e), and the tangent of the arc at the starting position is exactly the user's starting orientation, as shown in Fig. 2(A). We denote this path as $P_{mid}(x_e, y_e)$. It is easy to find out that the center of arc $P_{mid}(x_e, y_e)$ is located at:

$$\left(0, \frac{x_e^2 + y_e^2}{2y_e}\right) \tag{2}$$

However, $P_{mid}(x_e, y_e)$ is not the only possible path to reach (x_e, y_e). Imagine that there happens to be an obstacle on $P_{mid}(x_e, y_e)$, it is just unwise to mark (x_e, y_e) as unreachable since we can still utilize the open space around $P_{mid}(x_e, y_e)$ to bypass the obstacle. To bypass the obstacle, the curvature gain of the path must not be set in stone. Just as other methods like S2C or APF constantly adjust the steering direction to steer the user to the center of space or away from obstacles, we should also appropriately adjust the steering direction to bypass the obstacle and reach (x_e, y_e).

Different from the previous methods, we prioritize user immersion by minimizing changes in steering direction, allowing only a single alteration per path to the specified position. Furthermore, we do not vary the magnitude of the curvature gain in the path. Under this constraint, we can still obtain two types of paths by reducing the path radius based on $P_{mid}(x_e, y_e)$ and changing the steering direction at an appropriate position. The two types of paths are "turning

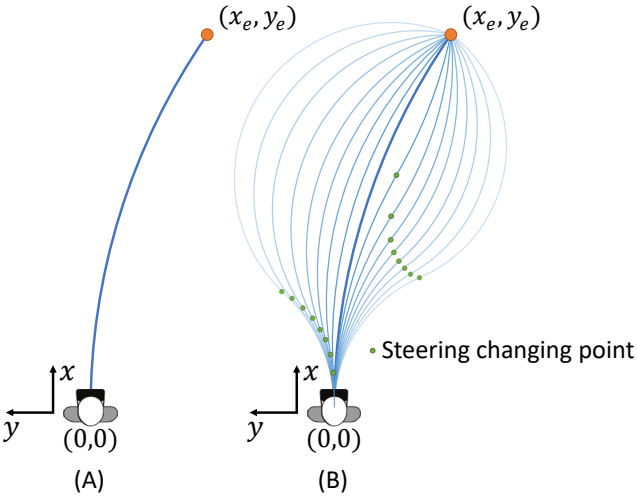

Figure 2: The potential paths to a given position. By steering the user left then right or right then left with different curvature gains, we can make the user reach the target with only one change in steering direction.

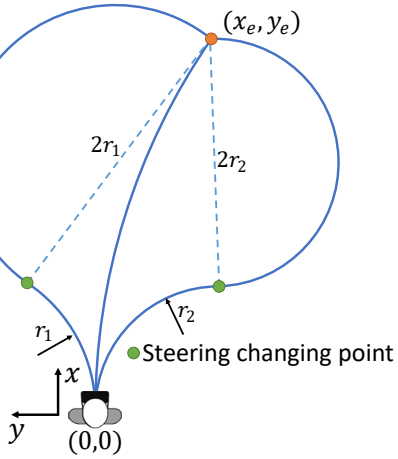

Figure 3: The path with the largest and smallest (on both sides) radius of curvature to the target. The centered path is the path with the largest radius. Let's denote the minimum path radii on the left and right sides as r_1 and r_2, and by analysis we know that they make the distance from the steering changing point to the target exactly twice their own.

left then right" paths and "turning right then left" paths. As the radius of curvature continue to decrease, we can get a series of non-intersecting paths distributed on the left and right sides of $P_{mid}(x_e, y_e)$, as shown in Fig. 2(B).

Since these paths have different radii of curvature and different steering direction orders, we need to find a simple descriptor to represent them. Since the first stage of the path must be tangent to the x-axis at $(0,0)$ as the user's staring orientation is given, the center of the first path arc must lie on the y-axis. In turn, given the y-coordinate of the first arc center, the first stage of the path and further the whole path to (x_e, y_e) can be uniquely determined. Therefore, **we use the y-coordinate of the center of the first arc as a path's descriptor**, and denote the path corresponding to the descriptor value f as $P_f(x_e, y_e)$.

For the path $P_f(x_e, y_e)$, its radius of curvature is $|f|$, and the center of the first path stage is $(0, f)$. In order to make the path reach (x_e, y_e), we can derive that the center of the second arc should be:

$$\begin{cases} x_{II} = \left(\frac{1}{2} + \frac{3f^2}{2D^2}\right) x_e + \frac{f}{|f|}\left(\frac{\sqrt{\Delta}}{2D^2}\right)(y_e - f) & (f \neq 0) \\ y_{II} = \left(\frac{1}{2} + \frac{3f^2}{2D^2}\right)(y_e - f) - \frac{f}{|f|}\left(\frac{\sqrt{\Delta}}{2D^2}\right) x_e + f & (f \neq 0) \end{cases} \tag{3}$$

Where $D^2 = {x_e}^2 + (y_e - f)^2$ and $\Delta = \left(D^2 - f^2\right)\left(9f^2 - D^2\right)$. The steering changing point of the path $P_f(x_e, y_e)$ is actually the midpoint of the centers of the two arcs, which we denote as $T_f(x_e, y_e)$:

$$T_f(x_e, y_e) = \left(\frac{x_{II}}{2}, \frac{y_{II} + f}{2}\right) \tag{4}$$

To obtain all the potential paths to (x_e, y_e), we need to find out the feasible range for descriptor f. **On the one hand**, to avoid the path radius being too large that the path cannot reach (x_e, y_e), the path radius $|f|$ should not exceed the radius of $P_{mid}(x_e, y_e)$. **On the other hand**, to avoid the path radius being too small that (x_e, y_e) exceeds the path's maximum travel range, the path radius $|f|$ should be no smaller than a radius r that makes the distance from (x_e, y_e) to the steering changing point reaches $2r$, as shown in Fig.3. If the radius continues to decrease, the distance from the steering changing point to (x_e, y_e) will exceed the second arc's diameter, making (x_e, y_e) unreachable. Based on the above inference, we can derive that the feasible f for reaching (x_e, y_e) should lie in:

$$\begin{aligned} F(x_e, y_e) = &\left[-\frac{x_e^2 + y_e^2}{2|y_e|}, \frac{-y_e - \sqrt{8x_e^2 + 9y_e^2}}{8}\right] \\ &\cup \left[\frac{-y_e + \sqrt{8x_e^2 + 9y_e^2}}{8}, \frac{x_e^2 + y_e^2}{2|y_e|}\right] \end{aligned} \tag{5}$$

In fact, $F(x_e, y_e)$ is also the solution set for $\Delta > 0$ in Equation (3). It's worth noting that both $\pm\frac{x_e^2 + y_e^2}{2|y_e|}$ in Equation (5) describe the path $P_{mid}(x_e, y_e)$, the only difference is which of the first or second arc is interpreted as 0. Note that **due to the limitation of the *curvature gain threshold***, f should also be in $L = [-\infty, -R] \cup [R, \infty]$, where R is the

minimum radius allowed by the *curvature gain threshold*. Therefore, the final feasible range of descriptor f for all potential paths to (x_e, y_e) is:

$$\Gamma(x_e, y_e) = F(x_e, y_e) \cap L \tag{6}$$

As a result, we can choose any $f \in \Gamma(x_e, y_e)$ to obtain a path to (x_e, y_e). To make the user reach (x_e, y_e) along a certain path $P_f(x_e, y_e)$, the RDW controller only need to steer the user with the curvature gain $1/|f|$ to the side of $(0, f)$ point, and when the user reaches the steering changing point $T_f(x_e, y_e)$, steer the user to the opposite side until the user reaches the specified target. In order for the RDW controller to steer the user in the global coordinate system, any local coordinate (x_{loc}, y_{loc}) can be restored to its global coordinate by the following formula:

$$\begin{bmatrix} x_{glb} \\ y_{glb} \end{bmatrix} = \begin{bmatrix} \cos\theta_s & -\sin\theta_s \\ \sin\theta_s & \cos\theta_s \end{bmatrix} \begin{bmatrix} x_{loc} \\ y_{loc} \end{bmatrix} + \begin{bmatrix} x_s \\ y_s \end{bmatrix} \tag{7}$$

And if $\Gamma(x_e, y_e)$ is an empty set, we can conclude that there is no path to (x_e, y_e) due to the *curvature gain threshold*.

3.2. Computation of paths occluded by obstacles

Though we find all potential paths to the specified position (x_e, y_e), some paths may be occluded by obstacles if there are complex obstacles distributed in the space. A trivial way to screen out unoccluded paths is to sample some paths in $\Gamma(x_e, y_e)$ and judge whether they are occluded by obstacles. However this calculation is too slow for the RDW controller, and the unoccluded paths cannot be fully recalled either.

Obstacles are usually represented as polygons. For an edge of an obstacle, since it is continuous in space, the path sandwiched between two paths occluded by its endpoints must also be occluded. If we can accurately find the descriptors of the paths occluded by its two endpoints, we can quickly exclude a batch of paths occluded by the entire obstacle edge based on spatial continuity. So we first investigate the problem of which path in $\Gamma(x_e, y_e)$ will be blocked by a given obstacle point (x_{obs}, y_{obs}).

It is worth noting that since all paths in $\Gamma(x_e, y_e)$ are distributed in the heart-shaped region shown in Fig.3, if (x_{obs}, y_{obs}) lies outside this region, it does not occlude any paths in $\Gamma(x_e, y_e)$. Conversely, (x_{obs}, y_{obs}) will occlude only one path, since all paths in $\Gamma(x_e, y_e)$ do not intersect each other.

Suppose the descriptor of the blocked path is $\hat{f}$. Let's start by assuming that (x_{obs}, y_{obs}) blocks the **first arc** of $\hat{f}$, as shown in Fig. 4(A). Therefore, we can directly compute the descriptor $\hat{f}$ for the occluded path by substituting (x_{obs}, y_{obs}) into Equation (2): $\hat{f} = \frac{x_{obs}^2 + y_{obs}^2}{2y_{obs}}$. Then following Equation (3) and (4), we calculate the steering changing point $T_{\hat{f}}(x_e, y_e)$ of the path $\hat{f}$ to (x_e, y_e). Finally, we

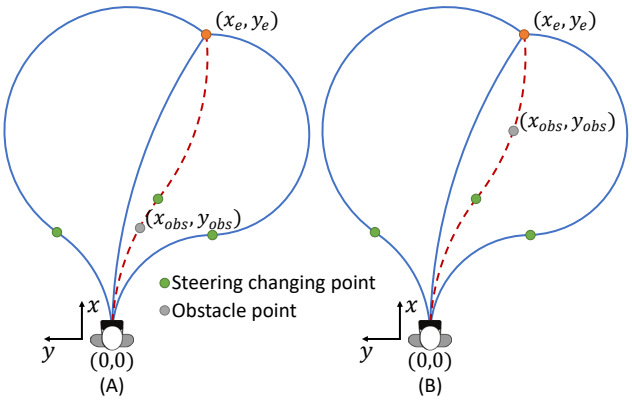

Figure 4: Two possible cases of an obstacle point blocking a path.

only need to verify whether the user will reach the obstacle before reaching the steering changing point, i.e. whether (x_{obs}, y_{obs}) is on the arc between $(0,0)$ and $T_{\hat{f}}(x_e, y_e)$, we can determine whether the path $\hat{f}$ is really blocked by the obstacle point (x_{obs}, y_{obs}). The order of the points on an arc can be quickly determined through the mixed product operation of vectors:

$$[x_{obs}, y_{obs}, 0]^{\mathrm{T}} \times \boldsymbol{T}_{\hat{f}}(x_e, y_e) \cdot [0, 0, Sgn(\hat{f})]^{\mathrm{T}} \tag{8}$$

Where $Sgn(\cdot)$ is the sign function. If (8)> 0, which means that (x_{obs}, y_{obs}) is closer to the starting position than the steering change point, $\hat{f}$ **is the desired occluded path**. Otherwise, we should continue to assume that (x_{obs}, y_{obs}) blocks the **second arc** of $\hat{f}$, as shown in Fig. 4(B). In this case, since the distance from the center of the second arc to (x_e, y_e) and (x_{obs}, y_{obs}) is $|\hat{f}|$, and the distance between the centers of the two arcs is $2|\hat{f}|$, we can formulate the following equations:

$$\begin{cases} (x_{II} - x_e)^2 + (y_{II} - y_e)^2 = \hat{f}^2 \\ (x_{II} - x_{obs})^2 + (y_{II} - y_{obs})^2 = \hat{f}^2 \\ (x_{II} - 0)^2 + (y_{II} - \hat{f})^2 = 4\hat{f}^2 \end{cases} \tag{9}$$

Where (x_{II}, y_{II}) is the center of the second arc of path $\hat{f}$. By eliminating x_{II} and y_{II}, we get a quartic equation with respect to $\hat{f}$. Here we directly give its solutions:

$$\begin{aligned} \hat{f} = \frac{-b}{4a} \mp \frac{1}{2}\sqrt{\frac{b^2}{4a^2} - \frac{2c}{3a} + \delta} \\ \pm \frac{1}{2}\sqrt{\frac{b^2}{2a^2} - \frac{4c}{3a} - \delta \mp \frac{\frac{-b^3}{a^3} + \frac{4bc}{a^2} - \frac{8d}{a}}{4\sqrt{\frac{b^2}{4a^2} - \frac{2c}{3a} + \delta}}} \end{aligned} \tag{10}$$

The two $\mp$ signs in (10) take both the upper and lower halves, so $\hat{f}$ has 4 solutions in total, where a, b, c, d, e are the

coefficients of the quartic equation:

$$\begin{cases} a = 2\left(y_e - y_{obs}\right)^2 \\ b = 2\left(\left(x_e - x_{obs}\right)^2 + \left(y_e - y_{obs}\right)^2\right)\left(y_e + y_{obs}\right) \\ \quad + 4\left(x_e y_{obs} - x_{obs} y_e\right)\left(x_e - x_{obs}\right) \\ c = \frac{1}{2}\left(x_e - x_{obs}\right)^4 + \frac{1}{2}\left(y_e - y_{obs}\right)^4 + \left(x_e^2 - x_{obs}^2\right)\left(y_e^2 - y_{obs}^2\right) \\ \quad - 2\left(x_e x_{obs} + y_e y_{obs}\right)\left(x_e - x_{obs}\right)^2 - 4 x_e x_{obs}\left(y_e - y_{obs}\right)^2 \\ d = -\left(\left(x_e - x_{obs}\right)^2 + \left(y_e - y_{obs}\right)^2\right) \\ \quad \times \left(\left(x_e^2 + y_e^2\right) y_{obs} + \left(x_{obs}^2 + y_{obs}^2\right) y_e\right) \\ e = \frac{1}{2}\left(x_e^2 + y_e^2\right)\left(x_{obs}^2 + y_{obs}^2\right)\left(\left(x_e - x_{obs}\right)^2 + \left(y_e - y_{obs}\right)^2\right) \end{cases} \tag{11}$$

And the intermediate factor δ can be calculated as follows:

$$\begin{cases} \delta = \dfrac{\sqrt[3]{2}\delta_1}{3a\sqrt[3]{\delta_2 + \sqrt{-4\delta_1^3 + \delta_2^2}}} + \dfrac{\sqrt[3]{\delta_2 + \sqrt{-4\delta_1^3 + \delta_2^2}}}{3\sqrt[3]{2}a} \\ \delta_1 = c^2 - 3bd + 12ae \\ \delta_2 = 2c^3 - 9bcd + 27ad^2 + 27b^2 e - 72ace \end{cases} \tag{12}$$

The reason $\hat{f}$ has four solutions is that (x_{II}, y_{II}) in Equation (9) is essentially the common intersection of three circles, and since the third circle has at most 4 intersections with the first two circles, by varying $\hat{f}$ each of them may become the common intersection. To find the only solution that truly represents the occluded path, we filter the 4 solutions based on three rules:

1. For path $\hat{f}$, the calculated steering changing points should be the same when (x_e, y_e) and (x_{obs}, y_{obs}) are substituted into Equation (3) and (4).
2. The calculated steering changing point cannot be behind the user (x coordinate should > 0).
3. The path $\hat{f}$ must go through (x_{obs}, y_{obs}) before reaching the target (x_e, y_e).

The above three rules help screen out the only truly occluded path among $\hat{f}$'s 4 solutions. After solving the problem of which path in $\Gamma(x_e, y_e)$ is blocked by an obstacle point, we can **quickly determine the set of paths blocked by an obstacle edge based on the endpoints of the obstacle edge**. As shown in Fig.5, for a given obstacle edge E, we first calculate its part E' that falls in the overall passable area, i.e. the heart-shaped region shown in Fig.3. As the overall passable area is bounded by 4 known arcs, E' can be quickly calculated by a line segment-arc intersection algorithm. **If E' does not exist, then E does not block any path in $\Gamma(x_e, y_e)$ at all**.

Otherwise, since the sign of the path descriptors on both sides of $P_{mid}(x_e, y_e)$ are different, we further divide E' into

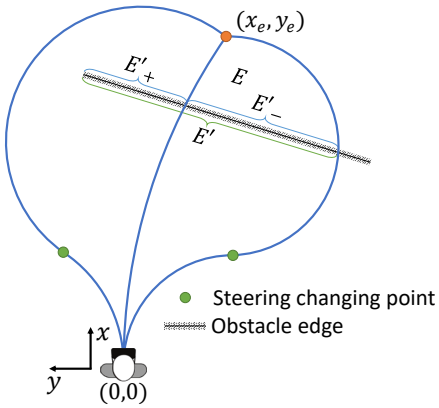

Figure 5: Illustration of the computation of paths occluded by an obstacle edge.

E'_- and E'_+ by $P_{mid}(x_e, y_e)$. Note that E'_- and E'_+ may not both exist. We process E'_- and E'_+ separately. Taking E'_+ as an example, we compute the two paths $\hat{f}_l$ and $\hat{f}_r$ occluded by its two endpoints. It is easy to know that $\hat{f}_l$ and $\hat{f}_r$ have the same signs. From the continuity of the edge, it can be seen that all the paths in $\left[\hat{f}_l, \hat{f}_r\right]$ **will all be occluded** and thus can be excluded from $\Gamma(x_e, y_e)$. Therefore, **by calculating only two occluded paths, we can exclude a batch of paths in one fell swoop**. In practice, in order to prevent E'_+ from being too long so that a few paths may enter E'_+ in the middle and then go out, we can divide E'_+ into sub-edges of smaller length (such as $1m$) to calculate their occlusion intervals separately, and take the union of them as the occlusion interval of E'_+. The union of the occlusion intervals of E'_- and E'_+ is the occlusion interval of E. Finally, by merging the occlusion intervals of all obstacle boundaries, we get all occluded paths in $\Gamma(x_e, y_e)$, and all the remaining feasible paths $I(x_e, y_e) \subseteq \Gamma(x_e, y_e)$.

As a result, we can choose any $f \in I(x_e, y_e)$ to obtain path to (x_e, y_e) without encountering obstacles. And if $I(x_e, y_e)$ is an empty set, we can conclude that there is no path to (x_e, y_e) due to the obstacles. Different from numerical optimization computations, all the above processes are analytical and the amount of computation is only linearly related to the number of obstacle edges, so we can get the exact path range to the specified target without encountering obstacles in a controlled time. Our method can work under various obstacle layouts with different specified targets.

4. Applications

Due to the generality of our method, by changing the query target and starting pose, our method can be applied to various meaningful RDW tasks. For instance, our method can be used for estimating the user's *maximum walkable distance* and the *reachable areas* from a given pose. These tasks are helpful for the RDW controller to maximize the

user's walking distance, evaluate the spatial properties of physical positions and steer the user to better positions. And these tasks could be quite tricky when there are complex obstacles in the space for RDW methods that cannot cope with obstacles when planning a path. Below we describe in detail how our method works in these intractable tasks.

4.1. Maximum walkable distance from a given pose

When a user starts from a given pose and walks straight in the virtual space, how far the physical space allows him to go is a spatial property of the starting pose. This property is useful for measuring how spacious a physical position is, and whether it is suitable for resetting users, etc.

Since the *maximum walkable distance* is caused by the restriction of the obstacles, the longest possible path from the starting pose must end at the edge of an obstacle. Since the layout of obstacles is unpredictable, finding this distance is actually an NP problem. However, our method can quickly estimate this distance. We sample some equally spaced points at the edge of the obstacle (as the method of Zhang *et al*. [50] does). Taking the input pose as the starting pose (x_s, y_s, θ_s), and taking the sampling points on the edge of the obstacle individually as the target point (x_e, y_e), we can calculate the feasible path set $I(x_e, y_e)$ without encountering obstacles to (x_e, y_e) respectively.

The length of each path in $I(x_e, y_e)$ can be derived from the sum of the lengths of its two arcs. It can be proved by the monotonicity and continuity of the path length that the descriptor of the longest path to (x_e, y_e) should be at the endpoints of a certain interval in $I(x_e, y_e)$. Therefore, by calculating the path lengths at the endpoints of the intervals in $I(x_e, y_e)$, we can get the maximum distance to $I(x_e, y_e)$ without encountering obstacles. And the maximum value of the maximum distance to all sampling points is the *maximum walkable distance* from (x_s, y_s, θ_s), denoted as $L(x_s, y_s, \theta_s)$.

In practice, in order to get $L(x_s, y_s, \theta_s)$ quickly, we can use the bounds of $\Gamma(x_e, y_e)$ to find the maximum path distance to the sampling points without considering obstacles first, and sort the sampling points by these distances. If the currently found *maximum walkable distance* is greater than the maximum path distance to the next sampling point without considering obstacles, the remaining sampling points can be skipped directly. So we can get $L(x_s, y_s, \theta_s)$ quickly without computing collision-free paths for all sampling points.

For a position (x_s, y_s), by sampling the directions, we can also calculate the average and maximum of the *maximum walkable distance* in all its directions. The larger the **average** *maximum walkable distance* is, the more spacious that position is; the larger the **maximum** *maximum walkable distance* is, the farther the user can go after being reset at that position.

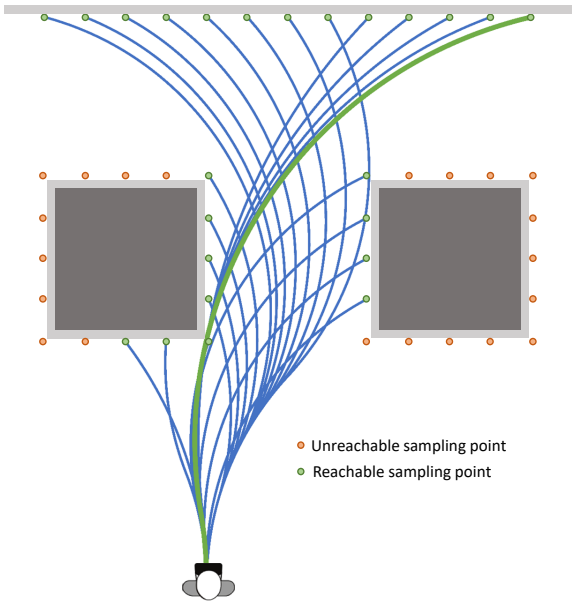

Figure 6: *Maximum walkable distance* estimation using our method. Our method finds all collision-free paths to each sampling point, and the figure shows the longest one to each sampling point. The path with the *maximum walkable distance* is marked in green. By pre-sorting the sampling points, our method can also obtain the *maximum walkable distance* without computing collision-free paths for all sampling points.

4.2. Reachable physical area for a given virtual distance

Now we know the quantitative measurement of how spacious a position is and how beneficial it is for resets. If we further know the user's reachable physical area for a given virtual distance, we can choose the optimal destination to steer the user.

For a given virtual distance l_v, its corresponding physical length is easy to find according to the translation gain:

$$S_{vir}(l_v) = \left[\frac{l_v}{g_t^{max}}, \frac{l_v}{g_t^{min}} \right] \tag{13}$$

Where g_t^{max} and g_t^{min} are the maximum and minimum thresholds of the translation gain. For a sampled point (x_e, y_e) in the physical space as target, we judge whether the path from the current pose to the specified target exists, that is, whether $I(x_e, y_e)$ is not empty.

If $I(x_e, y_e)$ is not empty, i.e. the overall passable area from the current pose to the sampling target (x_e, y_e) is not completely blocked, we find the longest and shortest paths in it. From the descriptors at the endpoints of the intervals in $I(x_e, y_e)$, we can easily find the path with longest length to (x_e, y_e). We denote the longest path length to (x_e, y_e) as s_{max}. And considering that the descriptor of the shortest

path may lie inside an interval in $I(x_e, y_e)$, it can be found in $I(x_e, y_e)$ by gradient descent searching. However, since the minimum length is very close to the length of $P_{mid}(x_e, y_e)$, it can be estimated from the descriptors at the endpoints of the intervals too. We denote the found shortest path length to (x_e, y_e) as s_{min}. So the length of the paths to (x_e, y_e) should lie in $S_{phy}(x_e, y_e) = [s_{min}, s_{max}]$.

If $S_{phy}(x_e, y_e) \cap S_{vir} \neq \emptyset$, it means that (x_e, y_e) is reachable at the given virtual distance l_v. By varying the translation gain, we can obtain different paths to (x_e, y_e) with different curvature gain. To get the user to a specific target (x_e, y_e) at the given virtual distance, we choose a specific path length $s \in S_{phy}(x_e, y_e) \cap S_{vir}$, then the desired translation gain g_t can be easily derived: $g_t = l_v / s$. In turn, we are also able to find a path of length s in $I(x_e, y_e)$, which can be achieved by binary search in $I(x_e, y_e)$. For the stability of the position reachability, we choose s that makes g_t close to 1.0, which is helpful to make (x_e, y_e) stably reachable in case l_v is floating, as we can shift the translation gain to compensate for the corresponding physical length. In addition, a g_t close to 1.0 also contributes to user comfort.

As the obstacle layout is unpredictable, the reachable physical area of distance $S_{vir}(l_v)$ is also an NP problem. So we take a series of uniform sampling points in the physical space as targets, and we can estimate the reachable physical area at the given virtual distance by verifying the reachability of the sampling positions.

Since the sampling positions can be pre-determined and their spaciousness and reset benefit can be pre-computed, we can steer the user to the optimal reachable sampling position based on the virtual distance l_v. In principle, when the user is about to reach the virtual target, we steer the user to a spacious position so that the user can turn around; otherwise, steer the user to a position that is conducive to reset so that a larger distance can be obtained after reset.

5. Evaluation and Performance

5.1. Evaluation Task

Above we presented our RDW method for steering users to a specified physical position while bypassing obstacles, and some of its meaningful applications. We first evaluate our algorithm using a conventional single-user RDW scenario. Further, to test the usability of the method and its derived applications, we apply them to a comprehensive real-world RDW task, i.e., multi-user online RDW.

Different from redirecting a single user or multiple users in a shared physical space, in the multi-player online VR games, multiple users are located in off-site physical spaces. If there is no unified management of the users' reset, users in the online virtual space will frequently reset asynchronously, which has a great impact on online games. Imagine that in the online virtual space, sometimes user A gets stuck due to reset, and sometimes user B gets stuck due to reset, which will greatly reduce the interactivity and playability of the game. A good solution is to always reset users together, but how to minimize the number of common resets for all users is a key issue.

Xu *et al*. [48] has proposed a promising solution to this problem. The key idea is that, for users who cannot reach the virtual target before encountering an obstacle, find the user with the smallest *maximum walkable distance* and steer other users to the position that is most favorable for reset in that user's walking time. However, their method only relies on a series of inscribed circles to plan potential paths generated by curvature gains and cannot avoid obstacles in complex spaces, thus its flexibility and performance are limited.

We follow the method of Xu *et al*. [48] to implement a new multi-user online RDW controller, and the difference is that we implement the estimation of the *maximum walkable distance* and the *reachable area for a given virtual distance* with our method. We compare our method with the method of Xu *et al*. [48] and several other state-of-the-art RDW methods. We believe that applying the method to a comprehensive real-world task is more effective for validating the method and its derived applications.

5.2. Implementation Details and Evaluation Designs

In our implementation, in order to estimate the *maximum walkable distance* for a given pose with our method, we set the sampling distance on the obstacle boundary to $0.5m$. Similarly, we also set the sampling density of the physical space to $0.5m$ to estimate the *reachable physical area* for a given virtual distance with our method. The minimum radius allowed by the curvature gain threshold is set to $R = 7.5m$, and the thresholds of minimum and maximum translation gain are set to $g_t^{min} = 0.86$ and $g_t^{max} = 1.26$, respectively [1, 5, 15, 44, 22, 4, 26, 10, 11, 45, 46]. Our method runs in real-time on a PC with Intel Core i7-9750H CPU and 8GB of RAM.

Simulation is widely recognized as an effective way to evaluate RDW methods [5, 1, 2, 44, 26, 10, 39, 45, 46] as it facilitates extensive experiments in various configurations and environments to eliminate the effect of randomness on evaluation results. Azmandian *et al*. [3] demonstrates that simulations can conservatively estimate the average performance of real users and is empirically valid for RDW methods evaluation. So we use extensive simulation experiments to test our method.

As with a number of previous methods [1, 44, 45, 21], we set the user's velocity in virtual space to $1m/s$, and the rotation rate to $\pi/2$ rad$/s$ to simulate each user's movement in the common virtual space. The target point of each user in the online virtual space is randomly generated at a random angle ranging from $-\pi$ to π, and at a distance between $2m$ and $6m$ from the previous target. The target points of

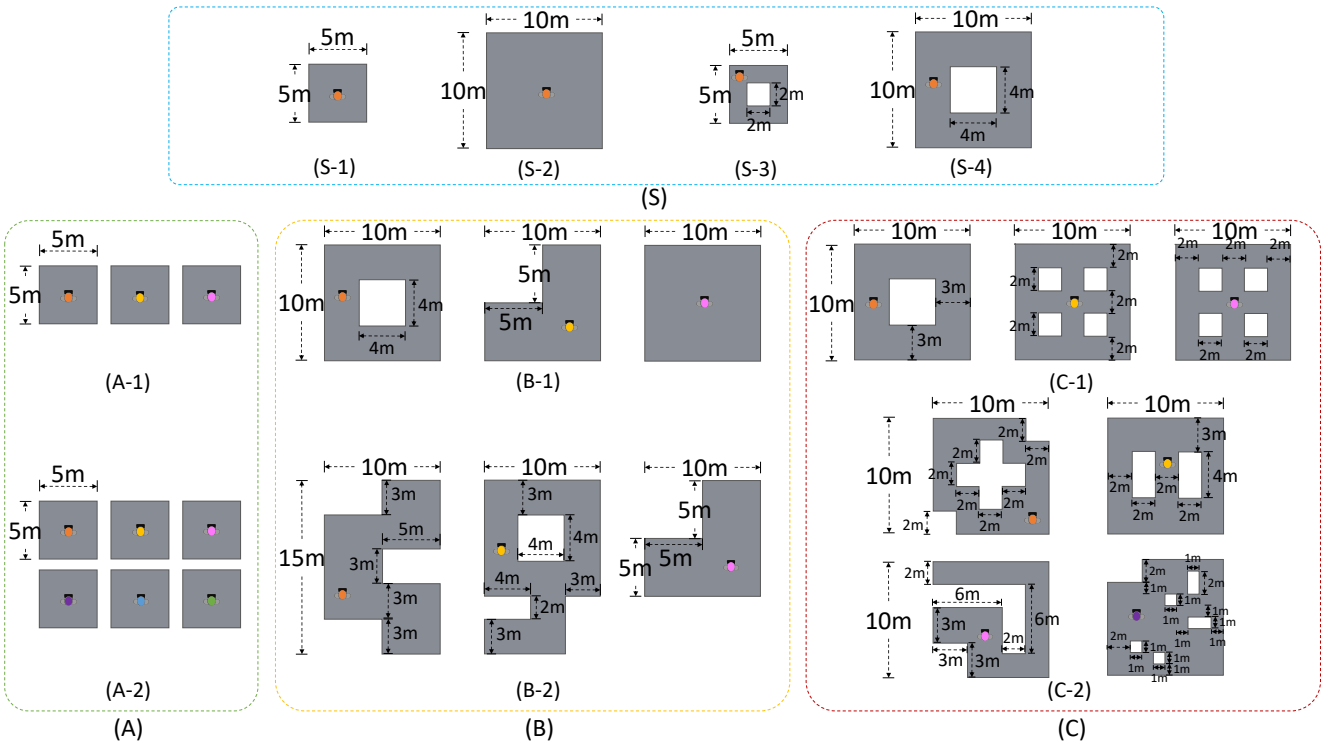

Figure 7: The physical environment layouts used for evaluation.

different users are different.

For the single-user RDW task, we evaluate our method across 4 distinct physical environments, as depicted in Fig. 7 (S). To facilitate comparison among these varying environments, we set these environments based on the presence or absence of obstacles and different sizes.

For the multi-user online RDW task, we test our method using 6 combinations of physical environments with complex obstacles, and their layouts are shown in Fig. 7 (A)˜(C). We classify these test environment layouts into 3 levels according to the complexity of obstacles. In level (A), each simulated user is located in an identical square space of size $5m \times 5m$ **without obstacles**. To evaluate the impact of the number of users on the number of resets, we tested 3 and 6 users using these identical physical spaces, as shown in Fig. 7(A-1) and (A-2). In level (B), the simulated users are located in different physical spaces with **complex obstacles**. Some of these spaces are non-convex and some have obstacles in the middle. To evaluate our method in different types of spaces, we also tested our method with two different combinations of obstacle layouts of level (B), as shown in Fig. 7(B-1) and (B-2). In level (C), the simulated users are located in different physical spaces with **very complex obstacles**. There are multiple obstacles in most physical spaces, and the distribution and shape of obstacles are complex and irregular. We also used two different combinations of obstacle layouts for testing, as shown in Fig. 7(C-1) and (C-2). Level (C) represents extreme testing situations.

We compare our method with several state-of-the-art methods of different types, namely Thomas *et al.*'s APF (TAPF) [44], steer-by-reinforcement learning (SRL) [39], steer-to-center (S2C) [15] and steer-to-orbit (S2O) [15]. As our multi-user online RDW task comes from the task of Xu *et al.* [48], we also compare our method with Xu *et al.*'s multi-user online RDW method [48]. It's worth noting that their method can also be applied in the single-user RDW scenario as a special case of off-site multi-user. For the reset orientation of the methods, both our method and Xu *et al.*'s method [48] adopt the reset strategy proposed by Xu *et al.* [48]. The other methods use the step-forward reset to gradient (SFR2G) strategy [44], where the gradient is calculated by TAPF. This reset strategy was demonstrated to outperform the other two reset strategies in TAPF.

We simulate each method 50 times for each environmental configuration. In each time of the simulation, we let each simulated user walk $400m$ in virtual space. Since the progress of different users may be different, the experiment ends when all users have finished walking. The OpenRDW [24] platform is used to help implement the simulation.

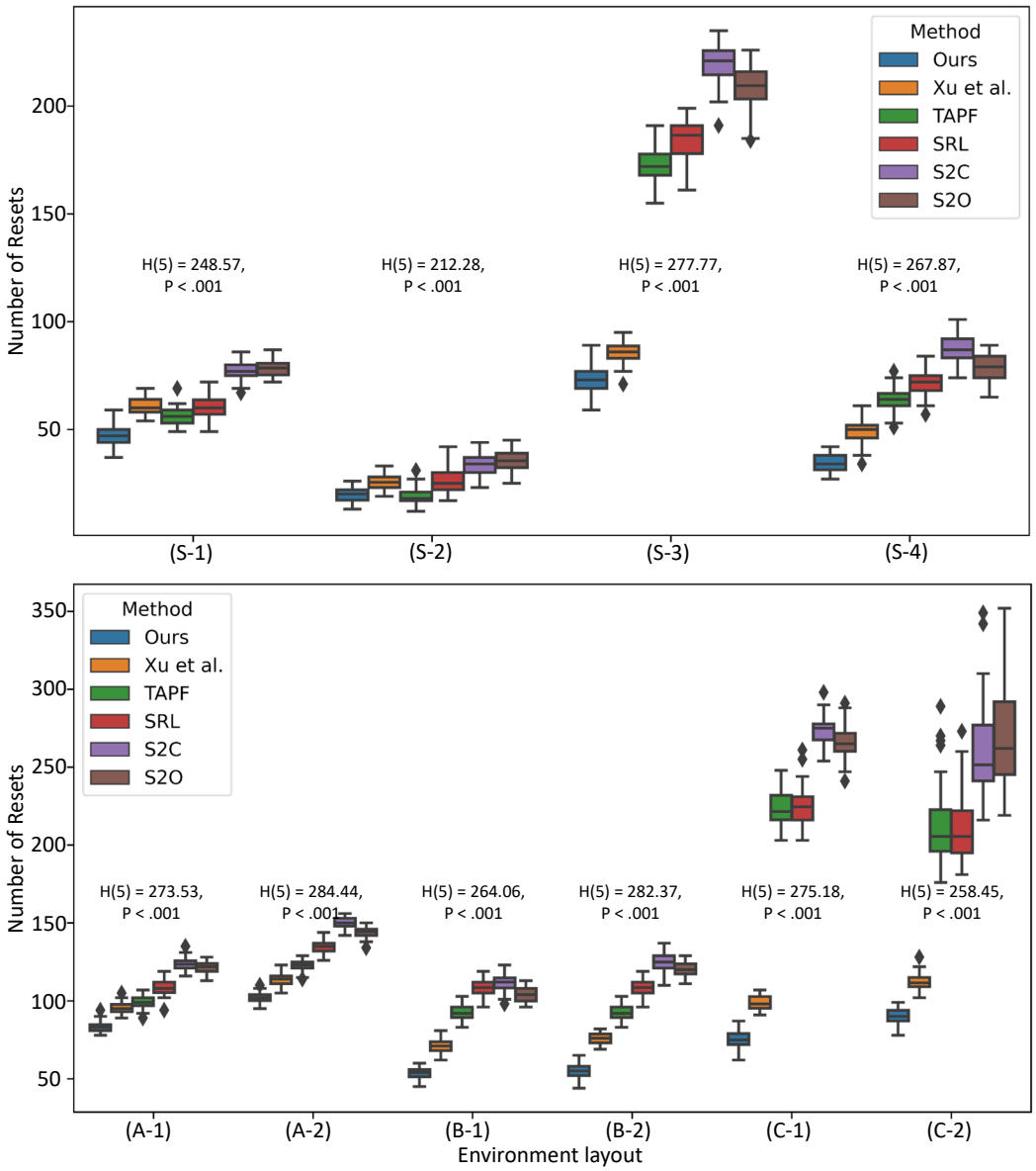

Figure 8: The box-plots of the number of resets under different environment layout configurations in **single-user** and **multi-user online** RDW task.

5.3. Results and Discussion

The distributions of the number of resets for all methods across different environmental configurations in both RDW tasks are shown in the box-plots in Fig.8. The results of all methods for single-user RDW task are shown in the upper portion of Fig.8, while the results for the multi-user online RDW task are shown in the lower portion. Since the sizes and layouts of physical spaces vary greatly under different environment configurations, the distribution of number of resets in different environment configurations is also very different. However, from the box-plots in Fig.8, we observe that our method generally performs better than all the other methods under most of the environment layouts. We can also find that the method of Xu *et al.* [48] also has noticeably lower numbers of resets than other comparison methods.

We further analyze the number of resets with the Kruskal-Wallis H test. Kruskal-Wallis H test on the experiment results suggests that there is indeed a statistical difference in the distribution of the number of resets for the 6 methods under each environment layout configuration. Except in environment layout (S-2), the post hoc pair-

wise comparison by Mann-Whitney U test finds satisfactory significance between our method and all other methods ($p < .001$). Moreover, the method of Xu *et al*. [48] also has statistical differences in performance compared to other methods ($p < .001$) in the majority of the environment layouts. These results effectively verify our observations. In environment layout (S-2), our approach is slightly weaker than TAPF, but the difference in results was not statistically significant. This may be because (S-2) is not only large in size, but also lacks obstacles. As our method's strength lies in its ability to utilize curvature gain to redirect users around obstacles, it performs better in environments with more complex obstacles, while its advantages may not be fully realized in obstacle-free environments. TAPF is one of the most excellent general RDW algorithms which employs a strategy of pushing users away from obstacles and attracting them towards open spaces. That's why our method's median is slightly weaker than TAPF only in the large, obstacle-free layout (S-2) (yet the difference isn't statistically significant). But our method outperforms the comparison methods in all the other experimental layouts.

In the environment layouts of level (A), the simulated users are located in regular physical spaces with identical shapes and sizes. The physical space is relatively small in these configurations, with only a size of $5m \times 5m$. Because these spaces are identical and have no obstacles in side, the performance difference of all methods turns out to be not very large. We find that TAPF generally performs better than the other three comparison methods SRL, S2C and S2O. This may be because TAPF uses its own reset strategy SFR2G, while the other three methods can only use TAPF's reset strategy. By comparing the number of resets in (A-1) and (A-2), we find that the number of resets does increase as the number of users increases. This is in line with intuition, because in a multi-user online VR game, the more users, the greater the probability that someone will trigger a reset at a certain moment. Although there are no obstacles, our method can generate more flexible paths and longer walking distances through the curvature gain, so method still has the most satisfying performance among the experimented methods.

In the environment layouts of level (B), the physical spaces are irregularly shaped with complex obstacles inside, and users are located in different physical layouts. As can be seen from Fig.8, although the larger physical space area makes the number of resets decreases to a certain degree, the difference in the number of resets for each method becomes more evident. Since we adopt the strategy of Xu *et al*. [48] to maximize the walking distance of the user with the shortest *maximum walkable distance* and steer other users to selected positions, our method and Xu *et al*. [48] significantly outperform the other methods. However, as our method could steer the user to bypass obstacles and allow the user to reach a wider spatial extent, our method further has a significant advantage over the method of Xu *et al*. [48].

In the environment layouts of level (C), the obstacles in physical spaces are even more numerous and irregular than those in level (B), which poses a greater challenge to the RDW controllers. We can see from the box-plots in Fig.8 that the performance of TAPF, SRL, ZigZag drops drastically under environment layouts at level (C). This is largely because these methods are not designed for the multi-user online VR scenarios, as they can only reduce the number of resets for each user individually, but cannot take into account the synergy between resets for all users. Our method and the method of Xu *et al*. [48] still maintain satisfactory results, which shows that the two methods both adapt well to the multi-user online VR scenarios. However, our method is not originally designed for this kind of scenario, but due to the generality of our method and its derived applications, it can achieve good results in this novel scenario. Since our method can steer the user to bypass obstacles and reach the desired position, our method shows better performance than Xu *et al*. [48] in these complex spaces.

6. User Study

6.1. Setup

In addition to the extensive simulations mentioned above, we design a user study to further investigate the practicality of our method. We do not target the multi-user online scenario, for the following two reasons: First, we have already shown that our method can be implemented into a multi-user online RDW controller that significantly outperforms several state-of-the-art methods in the simulations. Second, conducting experiments in this scenario requires more instructions and collaboration between users, which may lead to more interference factors. Therefore, a single-user RDW scenario is applied to efficiently examine our method.

We borrow several configurations directly from the simulation experiments to set up the user study: First, we continue to use the OpenRDW [24] platform and maintain the configurations including the redirection gains (described in Section 5.2). Second, the task for the user remains the same as in the simulations. As first proposed by Azmandian *et al*. [1], the user is supposed to turn and walk to the prompted waypoint. Third, the implementations for all controllers are the same.

The physical space layout used for the user study is shown in the Fig. 9 (a) and (b). Here, we employ an irregular shaped space layout with a square-shaped obstacle in the middle as our tracking space. However, for the user's safety, we did not actually place an real obstacle inside the experimental site, but instead we only used logical boundaries

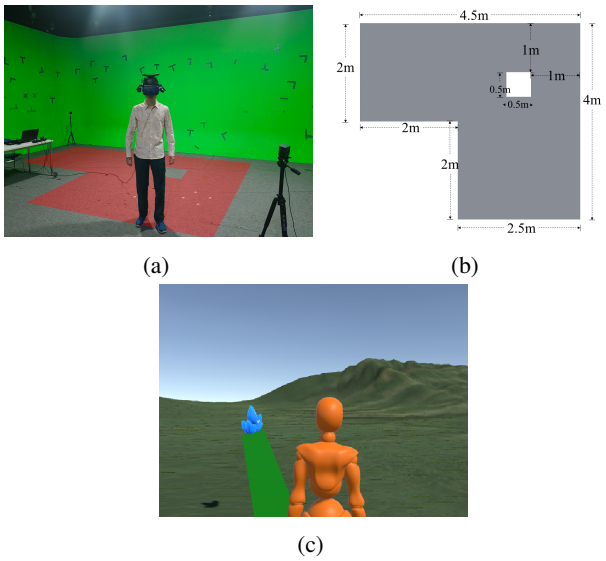

Figure 9: The environment configuration for the user study. The user walks in an irregular physical tracking environment with an invisible obstacle inside. The virtual environment is an open terrain where a blue crystal acts as the waypoint. (a): The real photo of the experiment site. (b): The physical space layout used for the user study. (c): The virtual environment scene.

to represent an imaginary obstacle. We ensure that a reset is triggered whenever the user reaches the real boundary or touches the invisible obstacle. As illustrated in Fig. 9(c), the corresponding virtual environment is an open terrain large enough for free exploration. A blue crystal floating in the air acts as the target point (waypoint) for the user, while a green line always stretches from the user to the crystal. Each time the user reaches the crystal, a new crystal is randomly generated as the new waypoint. We instruct the user to rotate in place and follow the green line to complete the task.

We choose TAPF [44] mentioned in 5.2 as the representative for the comparing methods. The main reason is that TAPF performs the best in the simple single-user environments such as (S-1) and (S-2) in Fig. 7 from the results of simulation experiments. We did not test other methods examined in the simulation experiments since the method Xu *et al*. proposed [48] is targeting and optimized for the multi-user online scenario, and the remaining methods are generally outperformed by TAPF.

We use Vive Pro 2 headset for users' VR experience. In each trial, the user is first instructed to put on the headset and adapt to the virtual environment. Then he/she walks for thirty meters virtually before unequiping and resting. We invite the user to fill out the Simulator Sickness Questionnaire (SSQ) [16] after each trial. Each user walks four trials, with two using our controller and two using the TAPF controller, respectively. The whole procedure takes about fifteen minutes for each user.

Table 1: The means and standard deviations of the number of resets and total SSQ score for both methods in the user study. The standard deviations are presented inside the brackets.

	Ours	**TAPF**
Number of resets	9.00 (1.07)	9.75 (1.34)
Total SSQ score	1.80 (2.27)	2.95 (3.47)

For evaluation, we record the total number of resets during each user walk, and also use the number of resets as the primary metric for the user study to verify the effectiveness of the methods. Besides, we also examine the SSQ to analyze user discomfort caused by the RDW manipulations. The SSQ score was computed from all 16 individual SSQ items and served as a secondary metric.

6.2. Results and Discussion

We recruited twenty participants from the campus to participate in the user study. The participants have an average age of around 22, including 7 females and 13 males. Therefore, a total of forty pairs of data is obtained, with each pair containing the number of resets and the total SSQ scores for both methods. We calculate the means and the standard deviations for both metrics, as listed in Table 1.

We plot the distribution of the number of resets in figure 10. In general, our controller results in fewer resets. Beside the means in Table 1, the median number of reset is 9 and 10 for our method and TAPF, respectively. From the results we notice that our method is more stable since TAPF tends to have poor performance sometimes. This corresponds with our observation during the experiment that users sometimes get stuck around the invisible obstacle when testing TAPF. Compared to TAPF, our method can adapt to irregular spaces and guide the user to bypass small obstacles better. A Kruskal-Wallis H test shows that the number of resets of the two methods has a significant difference ($p = 0.010$). Therefore, we conclude that our method outperforms TAPF in reducing the number of resets under the experiment task.

As for the total SSQ scores, our controller is rated with a smaller score than TAPF, indicating that our method causes less user discomfort. TAPF alters the curvature gain direction dynamically according to the user orientation and gradient, but our method ensures the user's curvature gain changes at most once before reaching the specified location while bypassing all the obstacles. This may be the reason why the SSQ score of TAPF is larger. We notice that the scores have significant personal differences among the participants (the results are in high standard deviations).

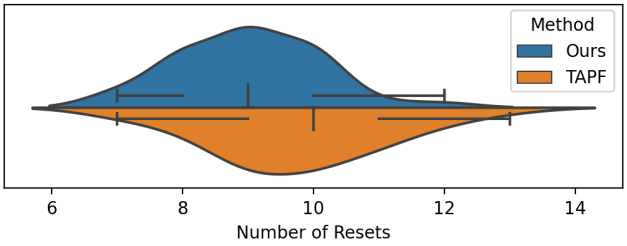

Figure 10: The distribution-plot of the number of resets for our method and TAPF in the user study.

Nevertheless, the SSQ scores are generally small since each walk only lasts for a short time. Therefore, both controllers can be considered applicable in short-term virtual locomotion. Comparably, our method has more potential to suppress user discomfort by reducing gain manipulations.

7. Limitations

Since our method exploits the curvature gain to bypass obstacles, our method is in turn limited by the curvature gain threshold. This is also a common limitation of methods using curvature gain. Although curvature gain has many advantages, some new gain techniques, such as change blindness, can redirect the user to a greater extent. If these novel gains can be combined with curvature gains, it may be more efficient to bypass obstacles in small spaces.

Since the computation time of our method is linearly related to the number of obstacle edges, the computational efficiency will decrease if the obstacle has a too large number of edges, such as the obstacle consists of curved surfaces. In this case, we can use the envelope surface to reduce the complexity of the obstacle surface.

As outlined in Section 2.1, curvature gain is often used in combination with translation gain and rotation gain. However, in our work, our only consider how to use curvature gain to steer users bypassing obstacles to reach specified positions for now, but we do not delve into the application of translation and rotation gain. Incorporating translation and rotation gain strategies into our method may further decrease the number of resets of our method.

8. Conclusion

In this work, we propose a general RDW method that finds a series of paths to steer the user to a specified physical position with curvature gain while bypassing all obstacles in a complex physical space. While bypassing obstacles, our method changes the steering direction at most once before the user reaches the specified position, which can minimize discomfort caused by curvature gain changes. The computation process of our method is analytical, so our method can get accurate solutions in a controlled time. The computation time is only linearly related to the number of obstacle edges. On this basis, we propose two meaningful applications of our method, namely maximum walkable distance estimation and reachable physical area estimation. We test our method with single user and multi-user online RDW tasks using a variety of complex physical space layouts, and conduct real-person user experiments under irregular space layouts to validate our method. The experimental results show that our method significantly outperforms the state-of-the-art methods.

Acknowledgement

This work was supported by the National Key Research and Development Program of China (No. 2023YFF0905104), the Natural Science Foundation of China (No. 62132012), Beijing Municipal Science and Technology Project (No. Z221100007722001) and the Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

References

[1] M. Azmandian, T. Grechkin, M. Bolas, and E. Suma. Physical Space Requirements for Redirected Walking: How Size and Shape Affect Performance. In M. Imura, P. Figueroa, and B. Mohler, editors, *ICAT-EGVE 2015 - International Conference on Artificial Reality and Telexistence and Eurographics Symposium on Virtual Environments*. The Eurographics Association, 2015. 1, 9, 12

[2] M. Azmandian, T. Grechkin, and E. S. Rosenberg. An evaluation of strategies for two-user redirected walking in shared physical spaces. In *2017 IEEE Virtual Reality (VR)*, pages 91–98, 2017. 1, 4, 9

[3] M. Azmandian, R. Yahata, T. Grechkin, J. Thomas, and E. S. Rosenberg. Validating simulation-based evaluation of redirected walking systems. *IEEE Transactions on Visualization and Computer Graphics*, 28(5):2288–2298, 2022. 9

[4] 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. 2, 3, 9

[5] E. R. Bachmann, J. Holm, M. A. Zmuda, and E. Hodgson. Collision prediction and prevention in a simultaneous two-user immersive virtual environment. In *2013 IEEE Virtual Reality (VR)*, pages 89–90, 2013. 4, 9

[6] L. Bölling, N. Stein, F. Steinicke, and M. Lappe. Shrinking circles: Adaptation to increased curvature gain in redirected walking. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):2032–2039, 2019. 3

[7] Y. Chang, K. Matsumoto, T. Narumi, T. Tanikawa, and M. Hirose. Redirection controller using reinforcement learning. *IEEE Access*, 9:145083–145097, 2021. 2, 4

[8] H. Chen, S. Chen, and E. S. Rosenberg. Redirected walking in irregularly shaped physical environments with dynamic obstacles. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 523–524, 2018. 4

[9] 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)*, pages 184–192, 2021. 4

[10] 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)*, pages 146–154, 2020. 1, 2, 3, 9

[11] T. Dong, Y. Shen, T. Gao, and J. Fan. Dynamic density-based redirected walking towards multi-user virtual environments. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pages 626–634, 2021. 4, 9

[12] T. Dong, Y. Song, Y. Shen, and J. Fan. Simulation and evaluation of three-user redirected walking algorithm in shared physical spaces. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 894–895, 2019. 4

[13] L. Fan, H. Li, and M. Shi. Redirected walking for exploring immersive virtual spaces with hmd: A comprehensive review and recent advances. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2022. 1

[14] T. Grechkin, J. Thomas, M. Azmandian, M. Bolas, and E. Suma. Revisiting detection thresholds for redirected walking: Combining translation and curvature gains. In *Proceedings of the ACM Symposium on Applied Perception*, SAP '16, page 113–120, New York, NY, USA, 2016. Association for Computing Machinery. 3

[15] 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. 3, 9, 10

[16] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and L. Mg. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The International Journal of Aviation Psychology*, 3:203–220, 1993. 13

[17] D. Kim, J.-e. Shin, J. Lee, and W. Woo. Adjusting relative translation gains according to space size in redirected walking for mixed reality mutual space generation. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*, pages 653–660, 2021. 3

[18] L. Kruse, E. Langbehn, and F. Steinicke. I can see on my feet while walking: Sensitivity to translation gains with visible feet. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 305–312, 2018. 3

[19] E. Langbehn, P. Lubos, G. Bruder, and F. Steinicke. Bending the curve: Sensitivity to bending of curved paths and application in room-scale vr. *IEEE Transactions on Visualization and Computer Graphics*, 23(4):1389–1398, 2017. 3

[20] 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), jul 2018. 3

[21] 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)*, pages 63–71, 2019. 2, 4, 9

[22] D.-Y. Lee, Y.-H. Cho, D.-H. Min, and I.-K. Lee. Optimal planning for redirected walking based on reinforcement learning in multi-user environment with irregularly shaped physical space. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 155–163, 2020. 2, 4, 9

[23] Y.-J. Li, F. Steinicke, and M. Wang. A comprehensive review of redirected walking techniques: Taxonomy, methods, and future directions. *Journal of Computer Science and Technology*, 37(3):561–583, 2022. 1

[24] 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)*, pages 21–30, 2021. 10, 12

[25] K. Matsumoto, Y. Ban, T. Narumi, T. Tanikawa, and M. Hirose. Curvature manipulation techniques in redirection using haptic cues. In *2016 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 105–108, 2016. 3

[26] 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)*, pages 72–80, 2019. 1, 2, 3, 9

[27] A. Nguyen, M. Inhelder, and A. Kunz. Discrete rotation during eye-blink. In L. T. De Paolis and P. Bourdot, editors, *Augmented Reality, Virtual Reality, and Computer Graphics*, pages 183–189, Cham, 2018. Springer International Publishing. 3

[28] A. Nguyen, Y. Rothacher, A. Kunz, P. Brugger, and B. Lenggenhager. Effect of environment size on curvature redirected walking thresholds. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 645–646, 2018. 3

[29] A. Nguyen, Y. Rothacher, B. Lenggenhager, P. Brugger, and A. Kunz. Individual differences and impact of gender on curvature redirection thresholds. In *Proceedings of the 15th ACM Symposium on Applied Perception*, SAP '18, New York, NY, USA, 2018. Association for Computing Machinery. 3

[30] 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. 1

[31] A. Paludan, J. Elbaek, M. Mortensen, M. Zobbe, N. C. Nilsson, R. Nordahl, L. Reng, and S. Serafin. Disguising rotational gain for redirected walking in virtual reality: Effect of visual density. In *2016 IEEE Virtual Reality (VR)*, pages 259–260, 2016. 3

[32] S. Razzaque. *Redirected Walking*. PhD thesis, University of North Carolina at Chapel Hill, USA, 2005. 2, 3

[33] S. Razzaque, Z. Kohn, and M. C. Whitton. Redirected Walking. In *Eurographics 2001 - Short Presentations*. Eurographics Association, 2001. 3

[34] H. Sakono, K. Matsumoto, T. Narumi, and H. Kuzuoka. Redirected walking using continuous curvature manipulation. *IEEE Transactions on Visualization and Computer Graphics*, 27(11):4278–4288, 2021. 3

[35] F. Steinicke, G. Bruder, J. Jerald, H. Frenz, and M. Lappe. Analyses of human sensitivity to redirected walking. In *Proceedings of the 2008 ACM Symposium on Virtual Reality Software and Technology*, VRST '08, page 149–156, New York, NY, USA, 2008. Association for Computing Machinery. 3

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

[37] F. Steinicke, G. Bruder, L. Kohli, J. Jerald, and K. Hinrichs. Taxonomy and implementation of redirection techniques for ubiquitous passive haptic feedback. In *2008 International Conference on Cyberworlds*, pages 217–223, 2008. 4

[38] F. Steinicke, G. Bruder, T. Ropinski, and K. Hinrichs. Moving towards generally applicable redirected walking. In *Proceedings of the Virtual Reality International Conference (VRIC)*, pages 15–24, 2008. 4

[39] 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. 1, 2, 4, 9, 10

[40] 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)*, pages 43–46, 2012. 1

[41] 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*, pages 159–166, 2011. 3

[42] Q. Sun, A. Patney, L.-Y. Wei, O. Shapira, J. Lu, P. Asente, S. Zhu, M. Mcguire, D. Luebke, and A. Kaufman. Towards virtual reality infinite walking: Dynamic saccadic redirection. *ACM Trans. Graph.*, 37(4), jul 2018. 3

[43] J. Thomas, C. Hutton Pospick, and E. Suma Rosenberg. Towards physically interactive virtual environments: Reactive alignment with redirected walking. In *Proceedings of the 26th ACM Symposium on Virtual Reality Software and Technology*, VRST '20, New York, NY, USA, 2020. Association for Computing Machinery. 2, 4

[44] 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)*, pages 56–62, 2019. 1, 2, 3, 4, 9, 10, 13

[45] N. L. Williams, A. Bera, and D. Manocha. Arc: Alignment-based redirection controller for redirected walking in complex environments. *IEEE Transactions on Visualization and Computer Graphics*, 27(5):2535–2544, 2021. 1, 2, 4, 9

[46] N. L. Williams, A. Bera, and D. Manocha. Redirected walking in static and dynamic scenes using visibility polygons. *IEEE Transactions on Visualization and Computer Graphics*, 27(11):4267–4277, 2021. 1, 2, 4, 9

[47] 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(11):3158–3168, 2019. 3

[48] 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. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–11, 2023. 3, 4, 9, 10, 11, 12, 13

[49] S.-Z. Xu, T. Lv, G. He, C.-H. Chen, F.-L. Zhang, and S.-H. Zhang. Optimal pose guided redirected walking with pose score precomputation. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 655–663, 2022. 4

[50] S.-H. Zhang, C.-H. Chen, F. Zheng, Y.-L. Yang, and S.-M. Hu. Adaptive optimization algorithm for resetting techniques in obstacle-ridden environments. *IEEE Transactions on Visualization and Computer Graphics*, 29(4):2080–2092, 2023. 8