Adaptive Optimization Algorithm for Resetting Techniques in Obstacle-Ridden Environments

Song-Hai Zhang, *Member, IEEE*, Chia-Hao Chen, Fu Zheng, Yong-Liang Yang, and Shi-Min Hu, *Senior Member, IEEE*

Abstract—Redirected Walking (RDW) algorithms aim to impose several types of gains on users immersed in Virtual Reality and distort their walking paths in the real world, thus enabling them to explore a larger space. Since collision with physical boundaries is inevitable, a reset strategy needs to be provided to allow users to reset when they hit the boundary. However, most reset strategies are based on simple heuristics by choosing a seemingly suitable solution, which may not perform well in practice. In this article, we propose a novel optimization-based reset algorithm adaptive to different RDW algorithms. Inspired by the approach of finite element analysis, our algorithm splits the boundary of the physical world by a set of endpoints. Each endpoint is assigned a reset vector to represent the optimized reset direction when hitting the boundary. The reset vectors on the edge will be determined by the interpolation between two neighbouring endpoints. We conduct simulation-based experiments for three RDW algorithms with commonly used reset algorithms to compare with. The results demonstrate that the proposed algorithm significantly reduces the number of resets.

Index Terms—Redirected walking, resetting, adaptive optimization, obstacle-ridden area, redirection

1 Introduction

Virtual reality (VR) has been developing very fast in the recent years as witnessed by the emergence of many revolutionary devices such as Oculus Rift and HTC Vive series. Its immersive experiences have enabled a number of emerging VR applications in games, training, healthcare, E-commerce, etc., where the user is often required to navigate in a virtual world. In practice, the dimension of the virtual space usually does not match that of the real space. Then how to explore a relatively large (even infinite) virtual space in a relatively small space in the real world is a critical problem.

To address the above problem, several redirected walking (RDW) algorithms have been proposed. The basic idea is to distort the user's walking path in the real world to avoid hitting physical boundary without being noticed in the virtual navigation. The representative algorithms include Steer-to-Center (S2C), Steer-to-Orbit (S2O), Steer-to-Multiple-Targets (S2MT), APF-RDW (Redirected walking based on Artificial Potential Field), etc. Even so, the collision with physical boundaries is inevitable, especially when the user's navigation path is long or/and the layout in the real space is complicated. Hence it is desirable to accompany the RDW algorithm with one or more reset strategies. Such a strategy can help users to reset to some ideal direction when hitting the boundary, allowing them to walk for a relatively long distance before the next collision.

However, most reset algorithms are only based on simple heuristics such as Reset-to-Center (R2C) and Reset-to-Gradient (R2G). There is a lot of room for optimization to further reduce the number of collisions. Despite some search-based optimization has been used for RDW algorithms such as MPCRed [1] and FORCE [2], the optimization of the reset strategy is yet to be explored to the best of our knowledge. In this paper, we propose a novel reset algorithm to optimize the reset direction adaptive to a given RDW algorithm. The algorithm works by dividing the boundary of obstacles and the physical environment with some endpoints. The reset direction at each endpoint is optimized instead of heuristically determined. The reset direction of the point in-between two neighbouring endpoints is determined by their linear interpolation. To quantitatively measure the quality of reset, we perform virtual navigation simulation to achieve the expected resets with provided reset directions. This is done by sampling abundant walking paths and simulating the resets accordingly. The reset directions are iteratively updated to minimize the resets. To evaluate our algorithm, we also conduct a variety of simulation-based experiments using various RDW algorithms under different environments. For realistic consideration, we demonstrate a bedroom and a living room with a similar layout in our experiments in Fig. 1 (The left is a 3d scene, the right is a 2d orthogonal plan). The results show

- *Song-Hai Zhang and Shi-Min Hu are with the Department of Computer Science and Technology, Beijing National Research Center for Information Science and Technology (BNRist), Tsinghua University, Beijing 100190, China. E-mail: shz@tsinghua.edu.cn, gynyfyt2012@hotmail.com.*
- *Chia-Hao Chen and Fu Zheng are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100190, China. E-mail: shimin@tsinghua.edu.cn, accplusjh@gmail.com.*
- *Yong-Liang Yang is with the Department of Computer Science, University of Bath, BA2 7AY Bath, U.K. E-mail: y.yang@cs.bath.ac.uk.*

Manuscript received 20 Dec. 2020; revised 9 Nov. 2021; accepted 22 Dec. 2021. Date of publication 4 Jan. 2022; date of current version 28 Feb. 2023.
This work was supported in part by National Key Technology R&D Program under Grant 2017YFB1002604, in part by the National Natural Science Foundation of China under Grants 61521002 and 62132012, and in part by the Research Grant of Beijing Higher Institution Engineering Research Center, and Tsinghua–Tencent Joint Laboratory for Internet Innovation Technology. The work of Yong-Liang Yang was supported by RCUK Grant CAMERA (EP/M023281/1, EP/T022523/1), and a gift from Adobe.
(Corresponding author: Song-Hai Zhang.)
Recommended for acceptance by D. A. Bowman.
Digital Object Identifier no. 10.1109/TVCG.2021.3139990

Physical layout similar to test environment 2

Physical layout similar to test environment 1 and 3

Fig. 1. Common physical environments within an obstacle-ridden area.

that our reset strategy largely improves the heuristical reset, and can easily adapt to the given RDW algorithm in obstacle-ridden areas with obstacles.

Our work makes two major contributions. 1) We present a novel optimization-based RDW reset strategy that significantly outperforms heuristical ones. 2) Our simulation-based reset optimization is adaptive to any RDW algorithm, and applicable for any obstacle-ridden area with arbitrarily shaped obstacles.

2 Related Work

2.1 Redirected Walking

Redirected walking (RDW) methods attempt to differ the virtual movement from the physical one, such that the users are able to explore a larger space. This is normally achieved by steering the users to some ideal point, or repelling them from the physical boundary. Three types of gains (rotation, translation, curvature) play leading roles when manipulating users' walking paths. Razzaque[3] proposed several generic RDW algorithms, including Steer-to-Center (S2C), Steer-to-Multiple-Target (S2MT), and Steer-to-Orbit (S2O). After that, much research related to these algorithms have been conducted. Hodgson and Bachmann [4] studied 4 different RDW algorithms (include also Steer-to-Multiple+Center which combines S2C and S2MT) with simulation-based and live user experiments. They discovered that S2C outperforms the other three algorithms on average, while S2O leads to better results than S2C when users perform long, straight-line navigation. Note that S2C referred in [5] is a modified version of Razzaque's one, aiming to overcome the deficiency when the angle between travel and center directions is too large.

With the advances of machine learning, some emerging technologies, especially reinforcement learning, have also been brought into this field. Lee *et al.* [6] proposed Steer-to-Optimal-Target (S2OT) to decide the best target among the candidates to steer to. By applying Deep-Q Learning, the score that indicates the optimal target is selected. Later, an extended algorithm called Multiuser-Steer-to-Optimal-Target (MS2OT) [7] was proposed. This is a multi-user version of S2OT. An important pre-reset action and other improvement measures were considered here. Strauss *et al.* [8] also used deep reinforcement learning in RDW. By employing Proximal Policy Optimization (PPO), a deep neural network is trained to directly estimate the rotation, translation, and curvature gains to transform a virtual environment, given a user's position and orientation in the tracked space. Another work by Chang *et al.* [9] presented a new redirection controller by PPO, and applied a new RDW algorithm (advanced center-based translation gain) and a new reset method (Turn-to-Furthest) for experimental comparison.

As the theoretical foundation of our simulation experiments, some cognitive and perceptual thresholds on three types of gains when influencing human walking have been thoroughly studied. Steinicke *et al.* [10] investigated three types of gains, and the thresholds of not noticing these gains. They showed that the rotation gain should be between 0.67 and 1.24, translation gain should be between 0.86 and 1.26, and curvature radius should be no less than 22m. Grechkin *et al.* [11] found no remarkable influence on curvature gain when combining with translation gain, and proposed a smaller curvature radius. Neth *et al.* [12] gave a more specific conclusion on how curvature gain should be decided by walking velocity.

Our RDW reset optimization can be coupled with any given RDW algorithm. It further improves user experience in virtual navigation with largely reduced resets.

2.2 Resetting

Although there are some mechanisms enabling the user to walk along a collision-free path by aligning virtual and physical environment [13], [14], [15], the reset step is inevitable if we do not constrain and control the user's physical locomotion for reactive and predictive RDW, since the user's walking length can be very long compared with the dimension of the real physical space. Generally, reset is always considered together with RDW algorithms. Williams *et al.* [16] proposed 3 reset strategies, including Freeze - Backup, Freeze - Turn, and 2:1 - Turn. The Freeze - Turn and 2:1 - Turn are the most commonly used for early reset algorithms. For example, Freitag *et al.* [17] proposed to enable the user to create portals to reach the target far away without much reorientation. Bachmann *et al.* [18] employed a modified version of 2:1 - Turn in the research of multi-user redirected walking. But after all, the above reset strategies pay more attention to maintaining the immersive experience rather than reducing the collision. And the related reset algorithms are preliminary attempts, and only focus on collision with the boundary.

More recent reset algorithms are developed to fit the corresponding RDW algorithms. For example, Reset-to-Center (R2C) always resets the user's direction to the center of the physical environment, which coincides with Steer-to-Center. Reset-to-Gradient (R2G) was proposed for APF-RDW, and used in [18], [19]. Thomas *et al.* [20] presented three new reset algorithms for a new APF-RDW algorithm named Push/Pull Reactive (P2R), including MR2C (a modified version of Reset-to-Center), R2G, and SFR2G (a modified version of Reset-to-Gradient). Besides, the reset step implicitly breaks the continuity of the user's walk in the virtual space, resulting in break-in-presence (BIP) effect. Peck *et al.*[21] used distractors in real applications to maintain natural locomotion. Sra *et al.*[22] integrated similar attractors into redirection to make them imperceptible. Cools *et al.* [23]

investigated different interactivity types and their effects on users. Our reset optimization enhances existing methods by using their resets as an initialization, and further optimizing reset directions based on walking simulation, resulting in a large decrease of resets. Note that in practical applications, the Freeze - Backup and the modified version of 2:1 - Turn are also viable for our algorithm, depending on user experience.

2.3 Obstacles in the Real Environment

In many situations, the shape of the physical environment is not as ideal (e.g., a circle or rectangle) as we thought. Some RDW algorithms work well only when the area is regular. Treating the walkable area as a whole, the ideal steering point might be the center of gravity (assuming that each point in the area has the same density), or the geometric center. However, it becomes much harder to choose a steering point when there are unorganized obstacles in the area (common for indoor environments), let alone multiple steering points for Steer-to-Multiple-Target.

Fajen *et al.* [24] built a dynamic system in order to predict human route selection to avoid obstacles while reaching a goal. The angular acceleration in the system depends on the angle and distance to both obstacles and the target. Chen *et al.* proposed two approaches to achieve obstacle avoidance. Steer-to-Farthest [25] attempts to steer the user towards one of the farthest physical boundary points from the user's current position. It depends on a cost function related to the distance from the user to the boundary, and the angle between the head orientation and the vector from the user to the obstacle. The other one [26] is a planning algorithm that uses dynamic obstacles to direct the user away from the boundary of the irregularly shaped physical environment. Valentini *et al.* [27] proposed an approach to reconstruct the real obstacles in the virtual environment to maintain the awareness of the real environment.

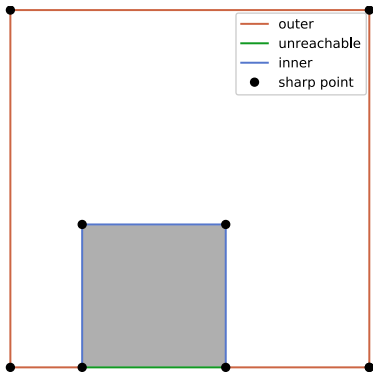
The physical environments in our experiments are general obstacle-ridden areas. We design layouts with obstacles of varying shapes to demonstrate the performance of our algorithm.

3 Adaptive Reset Optimization Algorithm

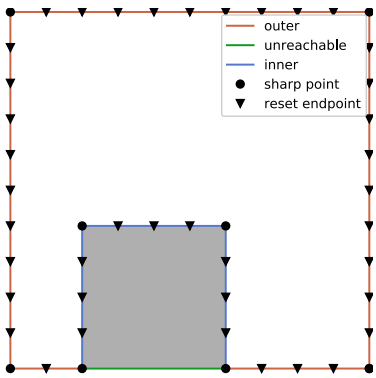
Unlike previous reset algorithms that rely on simple heuristics, we propose a novel optimization-based reset algorithm that complements existing RDW algorithms. Our algorithm manifests its adaptability in two aspects:

- It could be steered with different RDW algorithms on demand, even adapt to different walking behaviors of different users (that is, use the characteristics of specific users' walking data as a basis for optimization).
- It can handle physical boundary of any shape, and does not make any assumption on the convexity/concavity of the boundary of the environment or the obstacle.

In the following, we will elaborate on the optimization procedure of the algorithm, then introduce the hyper-parameters involved in the algorithm, and discuss the details of setting these hyper-parameters.



(a) Outer boundary, inner boundary, and unreachable boundary, with sharp points at the corners.



(b) The discretization after adding reset endpoints.

Fig. 2. The physical environment in an obstacle-ridden area is represented in 2D (a), and is discretized for simulation-based reset optimization (b).

3.1 Algorithm Description

Following prior work on redirected walking, we also reduce the problem to 2D for simplicity (see Fig. 2). Suppose the real space is represented by a number of boundaries, including an outer environment boundary and several inner obstacle boundaries. Each boundary comprises a set of inter-connected *smooth edges*, where neighbouring smooth edges meet at *sharp points*. Note that any point on a smooth edge has a continuous tangent except for the sharp point (as shown in Fig. 2a).

The ultimate goal for a reset algorithm is to specify a reset direction when hitting a boundary point. We will conduct massive simulated walking to optimize the resetting. Since the resetting in RDW causes a decrease in the sense of immersion, it is reasonable to use the number of resets as an evaluation metric for final reset directions. It can be done on-the-fly for heuristical reset where simple rules can be applied such as Reset-To-Center (just point to the center). However, for optimization-based approach, this is not feasible as the optimization is more expensive. Given the physical boundaries are practically fixed in real applications, we can pre-compute the reset direction and assign it accordingly when needed. It can be seen that optimizing the entire boundary point set is computationally prohibitive given the physically-based nature of the problem. Thus we propose to solve a discrete problem that is in line with finite element analysis [28] as widely used for physical simulation.

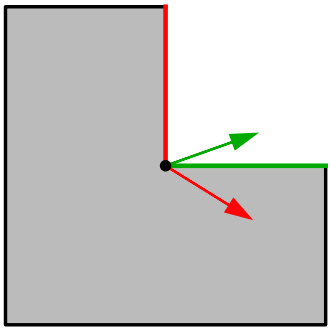


Fig. 3. The special rare case of boundary collision at concave sharp point.

As shown in Fig. 2b, we perform a standard 2D discretization in our implementation. It splits each smooth edge into several segments by adding evenly distributed intermediate points if necessary. After discretization, the length of each segment is less than a pre-defined edge interval l. Both the initial sharp points and the newly added intermediate points are called *endpoints*. In practice, we only optimize the reset direction at each endpoint. For the reset direction at any other boundary point, it is linearly interpolated from two neighbouring endpoints.

Let $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m\}$ denote the set of m endpoints. Their reset directions are represented as $\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$, where $\mathbf{v}_i = (x_i, y_i)$ is the reset vector at $\mathbf{p}_i$. The optimization is formulated as

$$\begin{aligned} \min_{\mathcal{V}} \quad & F(\mathcal{V}) \\ s.t. \quad & \angle(\mathbf{v}_i, \mathbf{t}_i) \in (0, \pi), \end{aligned} \tag{1}$$

where F is the objective function measuring the quality of the reset $\mathcal{V}$, $\mathbf{t}_i$ is the tangent vector at $\mathbf{p}_i$, and the angle $\angle(\mathbf{v}_i, \mathbf{t}_i)$ is measured in a local coordinate frame formed by the tangent $\mathbf{t}_i$ and normal $\mathbf{n}_i$.

Remarks.Note that we use the tangent $\mathbf{t}_i$ at the endpoint to constrain the reset direction (within an angle of π as in Eqn. (1)), thus avoid boundary collision. If an endpoint is a sharp point, it might have several reset vectors, depending on how many line segments intersect at that point. In practice, as shown in Fig. 3, there are two reset vectors at one point. These two vectors belong to different edges as indicated by their colors.

During the optimization, we optimize the two reset vectors with respect to their corresponding tangents. This better addresses the shape context of an individual smooth edge and its surrounding space, so as the interpolation along the edge. If a vector (such as the red one in Fig. 3) points inside the obstacle, this might trigger a new collision immediately after a reset, leading to consecutive collisions. There are practical solutions for this issue. For instance, we can choose a point $\mathbf{p}$ very close to the endpoint but not on the boundary, then simulate user movement with a short distance from this point in the reset direction. If it hits the boundary (meaning this direction is not appropriate), we continue to change the reset vector at this endpoint until the user no longer hits the boundary. We can obtain $\mathbf{p}$ as $\mathbf{p} = \alpha\mathbf{N} + \beta(\mathbf{M} - \mathbf{e})$, where $\mathbf{N}$ is the normal vector of this boundary on $\mathbf{e}$, $\mathbf{M}$ is the coordinate of the midpoint of this boundary, $\mathbf{e}$ is the endpoint, α and β are small values. But most conveniently, this issue can also be solved by a post-processing step to change the reset direction after optimization.

The key to the optimization in Eqn. (1) is how to define the objective function F and how to optimize $\mathcal{V}$ accordingly. For a good reset algorithm, it should provide reset directions that can largely reduce the number of resets. Hence we define F as the expected resets during the virtual navigation. However, F can only be estimated but not determined as the user's navigation paths can be very flexible in the virtual space. Therefore, we utilize a simulation-based approach to statistically evaluate F. More specifically, we sample s points in the virtual world and simulate the navigation path by connecting these points in the sampling order, by which we obtain a long walking distance. Then based on the given RDW algorithm and the current reset configuration $\mathcal{V}$, we can simulate the number of resets for each path. And we repeat this process for r times to simulate a distribution of navigation paths and their corresponding resets. Finally, we take the average resets as the expected F.

To optimize $\mathcal{V}$ based on F, the conventional way is to apply gradient-based numerical optimization. As the evaluation of F is based on statistical simulation, it is infeasible to compute gradient or any higher-order derivatives. Thus we employ a stochastic optimization approach similar to simulated annealing [29]. Starting from an initial reset configuration $\mathcal{V}_0$, we iteratively update the current configuration through a stochastic process as follows. We select k reset directions to update for each iteration. The potential update is performed by randomly adjusting the reset direction within a certain range ϵ. Then we perform simulation and evaluate F to estimate the quality of the update. If the simulation value is less than the current (minimum) resets by a threshold T_{update}, then we update $\mathcal{V}$ and also the minimum resets. After a specific number (1,000 by default) of iterations, we achieve the optimized reset $\mathcal{V}^*$. The optimization procedure is structured in Algorithms 1 and 2. The setup of hyper-parameters will be detailed in the next subsection.

Note that to initialize the optimization, we need to provide an initial reset configuration $\mathcal{V}_0$ to start the simulation. The simplest way to do so is to randomly sample reset directions in the feasible space, i.e., $\angle(\mathbf{v}_i, \mathbf{t}_i) \in (0, \pi)$. However, random initialization is very likely to make our optimization fall into a local minimum. It makes more sense to make the initialization adaptive to the specific RDW algorithm. This makes sense as reset algorithms are always coupled with RDW algorithms. For instance, Reset-To-Center reset can be employed to initialize reset optimization for Steer-To-Center RDW algorithm. And our optimization can further reduce the number of resets from the initialization (see Section 4).

3.2 Hyper-Parameters in the Optimization

This subsection elaborates the details on setting up the hyper-parameters in the optimization.

- *Threshold of updating parameters* (T_{update}): First, we take the number of resets from the first round of simulation as the minimum objective function value F_{min}. In each iteration when we simulate the virtual navigation, if the current number of resets F is less

than the current minimum, then we may make updates according to T_{update}. More specifically, if the decay of the objective is greater than the threshold, we update F_{min} with F. A larger T_{update} increases the confidence that the number of (expected) resets decreases after the update. On the other hand, a smaller threshold makes the update easier, but the confidence of reducing resets decreases. This threshold is set to 2 in our experiments.

- *Sampled points* (N_S): There is obviously a linear relationship between sampled points and resets. The more points we sample, the more resets take place, which makes T_{update} more reachable. However, merely increasing the number of sampled points does not make the experiment easier. This is because all sampled points are from the same normal population. After they are added, the variance accumulates. To reduce the variance, we have to repeat the simulated path many times, and take the average as expected resets. In each simulated path, we sample 1000 points.
- *Repetitions of walking* (R_W): In practice, we need to repeat the virtual navigation many times to reduce the reset variance and make a good expectation of resets. The variance will decrease quadratically as the number of repetitions increases. This value is set to 500.
- *Edge interval* (I_E): Although linear interpolation is used to calculate the reset direction at an intermediate point, the optimal reset direction between two reset endpoints may not behave linearly (especially when the edge between the two endpoints is not a straight line but a curve). Reducing the edge interval can improve the accuracy of the result, because we can approximate a nonlinear interpolation through piecewise linear interpolation. However the computational cost will increase. For the ease of processing, l is set to 1.
- *Number of reset directions updated* (N_{para}): Each time we randomly choose one or more reset directions to update. Updating one parameter at a time tends not to be enough to generate sufficient gradients. That is, the number of reduced resets will not exceed the threshold T_{update}. Thus we need to update more parameters to make an adequate decay. As the optimization goes on, this becomes harder as the current reset configuration is close to optimum. As a result, the number of updates needs to grow. In our experiment, the initial k is set to 3, and it increases by 1 for every 400 iterations.
- *Search step* (ϵ): Like the gradient descent method, we need to control the search step size for each iteration. In our implementation, the search step is the maximum value of the random angle change of each reset direction. In the experiment, this value is dynamically set to $\max(10, 55 - 5 \cdot N_{para})$.
- *Perceptual thresholds*: The translation gain is greater than 0.86, less than 1.26. The rotation gain is greater than 0.67, less than 1.24, which are the same as suggested in [10]. The radius of curvature is 7.5m and approximately equals to 1/0.13, where 0.13 is the curvature gain suggested in [12], corresponding to an average user walking speed of 1.4m/s[30].

Algorithm 1. Adaptive Reset Optimization

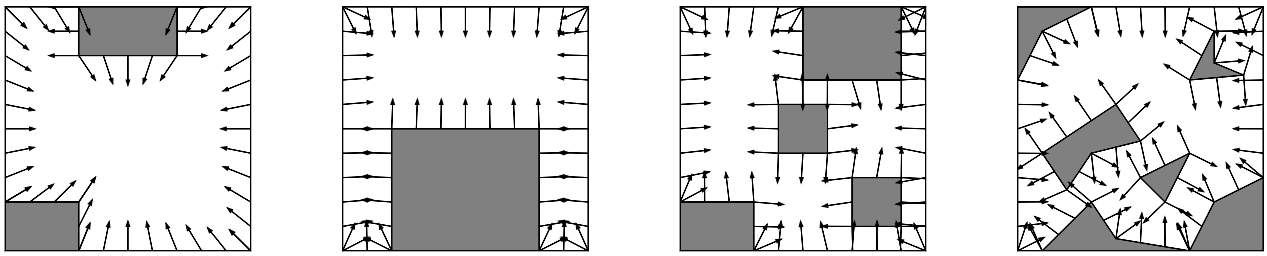
Require: A set of boundary smooth edges $\mathcal{E}$ of physical environment
function ResetOpt($\mathcal{E}$)
  split $\mathcal{E}$ by a set of endpoints $\mathcal{P}$
  $\mathcal{V}_0 = \emptyset$
  for each reset direction $\mathbf{v}$ at $\mathbf{p} \in \mathcal{E}$ **do**
    $\mathbf{v}$ = Reset_Strategy($\mathbf{p}$)
    $\mathcal{V}_0 = \mathcal{V}_0 \bigcup \{\mathbf{v}\}$
  end for
  F_{min} = GetAverageResets($\mathcal{V}_0$)
  for $i \leftarrow 1$ to R_W **do**
    for a random $\mathcal{V}_k$, s.t.$\mathcal{V}_k \subseteq \mathcal{V}$ and $|\mathcal{V}_k| = k$ **do**
      for $\mathbf{v} \in \mathcal{V}_k$ **do**
        $\mathbf{v}$. Rotate(Random(-$\epsilon/2, \epsilon/2$))
      end for
      F = GetAverageResets($\mathcal{V}$)
      if $F < F_{min} - T_{update}$ **then**
        $F_{min} \leftarrow F$
      else
        Revoke the change in $\mathcal{V}_k$
      end if
    end for
  end for
  return $\mathcal{V}$
end Function

Algorithm 2. Get the Average Resets

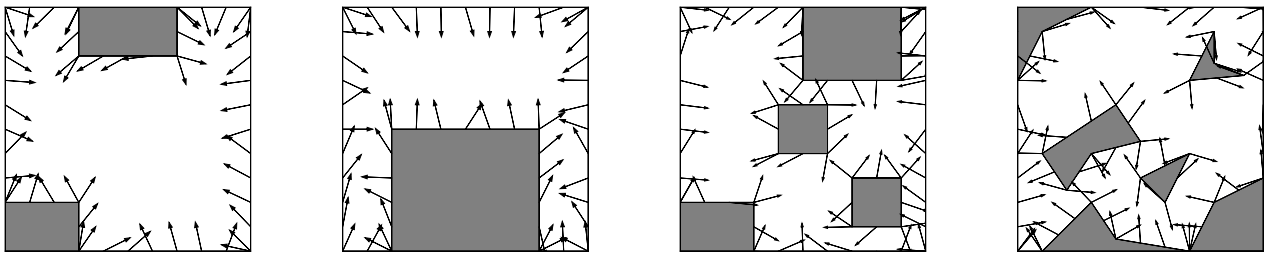
Require: A set of reset direction $\mathcal{V}$
function GetAverageResets($\mathcal{V}$)
  Average $\leftarrow 0$
  for $i \leftarrow 1$ to R_W **do**
    Average $\leftarrow$ Average + RandomWalking($\mathcal{V}$)
  end for
  return Average / R_W
end Function
function RandomWalking($\mathcal{V}$)
  Initalize the Simulated user $\mathbf{U}$
  for $i \leftarrow 1$ to N_S **do**
    Sample a point $\mathbf{p}$ in virtual Environment
    NextDir $\leftarrow \mathbf{p} - \mathbf{U}$.position
    $\mathbf{U}$.TurnTowards(NextDir/||NextDir||)
    while $\mathbf{U}$.position $\neq \mathbf{p}$ **do**
      $\mathbf{U}$.StepTo($\mathbf{p}$)
      if $\mathbf{U}$ Collide With Boundary on $\mathbf{c}$ **then**
        $\mathbf{p}_1, \mathbf{p}_2 \leftarrow$2 Nearest endpoints
        $\mathbf{v}_1, \mathbf{v}_2 \leftarrow$Reset direction of $\mathbf{p}_1, \mathbf{p}_2$ in $\mathcal{V}$
        $L \leftarrow \sqrt{\|\mathbf{c} - \mathbf{p}_1\|^2 + \|\mathbf{c} - \mathbf{p}_2\|^2}^{-1}$
        $\mathbf{v}' \leftarrow \|\mathbf{c} - \mathbf{p}_1\|\mathbf{v}_2 + \|\mathbf{c} - \mathbf{p}_2\|\mathbf{v}_1$
        $\mathbf{U}$.TurnTowards($L\mathbf{v}'$)
      end if
    end while
  end for
end Function

4 Evaluation

To evaluate the effectiveness of the algorithm, we conduct a variety of simulation-based experiments based on widely-used RDW algorithms: No-Steering(NS), Steer-to-Center(S2C),



(a-1) Environment 1: Reset-to-Center for Steer-to-Center (b-1) Environment 2: Reset-to-Gradient for P2R (c-1) Environment 3: Reset-to-Center for No-Steering (d-1) Environment 4: Perpendicular reset for No-Steering



(a-2) Environment 1, after optimization, using S2C+R2C (b-2) Environment 2, after optimization, using P2R+R2G (c-2) Environment 3, after optimization, using NS+SFR2G (d-2) Environment 4, after optimization, using P2R+SFR2G

Fig. 4. The 4 physical environments with obstacles, the initial reset vectors(-1) and vectors after optimization(-2) at the boundary reset endpoints. Result of each environment(-2) demonstrates the vectors optimized from one of the combination of selected redirected algorithm and reset algorithm.

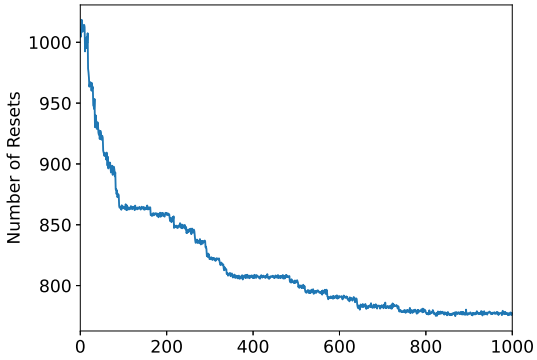
and Push/Pull Reactive. Following [20], MR2C, R2G, and SFR2G are used in the experiment as the initial state of our optimization. Since the T2F algorithm is not as linear as other algorithms, there may be a sudden change in the position of the farthest point at some points. Therefore, it does not hold good interpolation properties(and our algorithm works by interpolating between two points to represent an algorithm so that it can be optimized), so we are not planning to add T2F in our experiments. But it is still valuable for research, and there will be a more suitable optimization method for it in the future.

Note that SFR2G is different from a regular in-place method [31], [32], which usually decides the reset direction right on the spot that the user encounters a collision. Instead, it takes the point a few steps away in the direction of the current gradient and resets the user in the gradient direction of that position. In order to optimize it by our algorithm, we make the following adjustment. We only optimize the direction of gradient direction on the point hitting an obstacle in SFR2G. After taking a few steps in that direction, the user finally selects the gradient direction of the achieved point as the reset direction. There are cases where we are not able to take a few steps when performing SFR2G (i.e., because the user would collide with another boundary when following the direction of the reset vector). In this case, SFR2G will degenerate into R2G, that is, reset the direction without selecting another point.

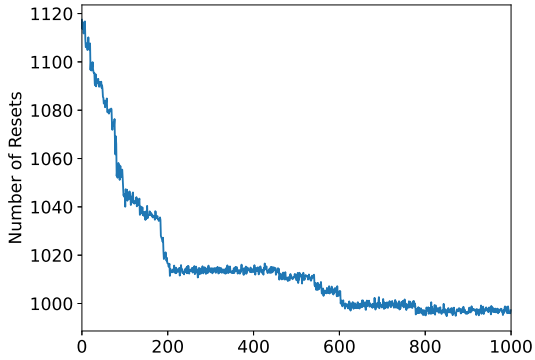
In addition, as we test our algorithm in environments with different layouts composed of obstacles with different shapes, we may want to see how the size of virtual space affects the resets. The effect of space size was studied in [33], as well as those irregular space shapes such as trapezoid- or L-shaped areas. Usually the size of the virtual environment is larger than that of the physical environment, however the extent to which the virtual space is larger than the physical space also affects the final result. As such, we conduct experiments on 20m×20m and 40m×40m virtual spaces.

To highlight the impact of obstacles for optimization, four different layouts with obstacles are designed to test different RDW algorithms (see Figs. 4a-1, 4b-1, 4c-1, 4d-1). And our testing layouts with reduced walkable space are more challenging compared with practical cases. The physical environment is a 10m×10m rectangular space, while the virtual environments are 20m×20m and 40m×40m obstacle-free rectangular spaces with fixed borders. No assumption on navigation is made here, as in practice the user may step to anywhere that is open in the virtual space. Each environment has its representative characteristics in the result , and its difference from other environments is highlighted with a variety of RDW and resetting algorithms. To visualize the optimization effect, we chose an algorithm configuration for each environment, and demonstrate the reset vectors after optimization (Figs. 4a-2, 4b-2, 4c-2, and d-2) together with the number of resets during optimization according to the number of iterations (Fig. 5). Moreover, we perform each RDW method for each environment, the results are summarized in Table 1, 2, 3, and 4, including the average resets before (R_{before}) and after (R_{after}) optimization, and the reduction rate ($R_{\text{reduction}}$): $R_{\text{reduction}} = 1 - R_{\text{after}}/R_{\text{before}}$

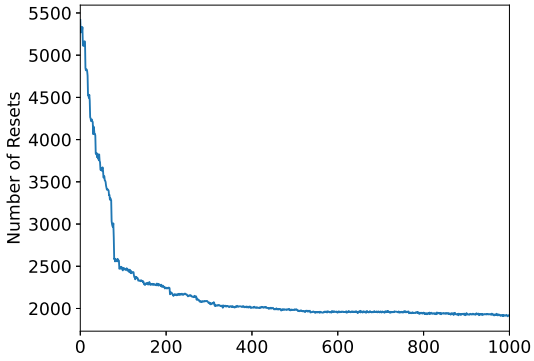
We first employ our reset optimization on R2C, R2G for two reactive RDW algorithms S2C, APF-RDW, respectively. Two extra experiments are conducted for No-steering, where R2C reset is applied for one experiment, and all reset vectors are perpendicular to the boundary for the other experiment. For all experiments, we first assign a reset vector at each endpoint according to the original reset strategy (see Fig. 4). Then we perform 1,000 iterations for S2C, P2R, and No-Steering in two environments with different obstacle layouts. For each simulation round of a given reset configuration, we



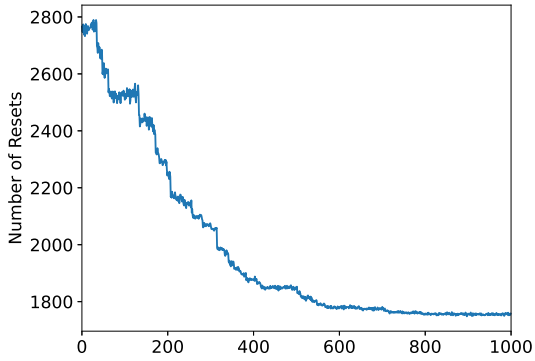
(a) Resets during optimization in Environment 1, applying the algorithmic combination of S2C and R2C



(b) Resets during optimization in Environment 2, applying the algorithmic combination of P2R and R2G



(c) Resets during optimization in Environment 3, applying the algorithmic combination of NS and SFR2G



(d) Resets during optimization in Environment 4, applying the algorithmic combination of P2R and SFR2G

Fig. 5. The convergence plot of resets during optimization for the test environments. X-axis represents number of optimizations, Y-axis represents the average resets(under 500 simulated walking) under current optimizations. Different environments are applied with different RDW and Reset methods.

TABLE 1
The Results for Each RDW Algorithm and Reset Method of Environment 1, the Average Resets of 500 Simulations Before(columns 3-5) and After(columns 6-8) Optimizations, and the Percentage of Resets Reduced

Virtual Size	RDW Alg.	Resets Before			Resets After			Reduction Rate		
		MR2C	R2G	SFR2G	MR2C	R2G	SFR2G	MR2C	R2G	SFR2G
20x20	NS	1193.1	1275.3	1186.6	1089.8	1103.7	1166.2	8.66%	13.46%	1.72%
	S2C	1010.1	854.8	844.7	777.2	773.1	844.9	23.06%	9.56%	0.00%
	P2R	1113.0	1100.5	1098.9	921.8	902.0	1094.8	17.18%	17.49%	0.52%
40x40	NS	2274.1	2453.1	2260.0	1957.7	1951.4	2150.0	13.91%	20.45%	4.87%
	S2C	2300.9	1810.3	1785.0	1487.4	1462.0	1752.8	35.36%	19.24%	1.80%
	P2R	2406.9	2272.5	2288.6	1621.5	1838.7	2210.1	32.63%	19.09%	3.43%

record the mean resets to form the convergence plot (see Fig. 5). Since our optimization method is more closed to a reactive RDW (through a large number of simulations to reduce average resets), we are yet to apply our method on predictive or learning-based RDW. But it is important to point out that our method is capable to cope with those algorithms.

4.1 Experiment Design

4.1.1 Reset Optimization for S2C

In our first environment, there is a 2m×4m rectangular obstacles on the top middle and a 2m×3m rectangular obstacles on the bottom left of the physical environment(see Fig. 4a-1). This environment is almost obstacle-free in the middle part. As mentioned in [4], S2O outperforms S2C only for long, straight-line navigation. Otherwise S2C always performs better. In fact, we can only walk along a virtual straight line that is no more than three times the length of the physical space at most. Besides, in most cases, the path length between two consecutive waypoints is far less than that distance. Therefore, we choose S2C as our first RDW algorithm to optimize instead of S2O or other algorithms.

We conduct Steer-to-Center for redirected walking, with MR2C as a heuristic start for our navigation simulation. In

TABLE 2
The Results for Each RDW Algorithm and Reset Method of Environment 2, the Average Resets of 500 Simulations Before(columns 3-5) and After(columns 6-8) Optimizations, and the Percentage of Resets Reduced

Virtual Size	RDW Alg.	Resets Before			Resets After			Reduction Rate		
		MR2C	R2G	SFR2G	MR2C	R2G	SFR2G	MR2C	R2G	SFR2G
20x20	NS	1583.8	2417.4	2196.6	1279.7	1323.8	2003.3	19.20%	45.24%	8.80%
	S2C	1430.2	1331.4	1282.8	998.9	1005.1	1224.0	30.16%	24.51%	4.58%
	P2R	1094.3	1117.4	1147.2	987.1	997.2	1134.1	9.80%	10.76%	1.14%
40x40	NS	3116.2	5176.6	4445.2	2267.2	2346.0	3905.5	27.24%	54.68%	12.14%
	S2C	3101.9	2673.4	2531.2	1875.4	1814.2	2242.8	39.54%	32.14%	11.39%
	P2R	2140.6	2129.9	2287.8	1895.8	1856.9	2201.5	11.44%	12.82%	3.77%

TABLE 3
The Results for Each RDW Algorithm and Reset Method of Environment 3, the Average Resets of 500 Simulations Before(columns 3-5) and After(columns 6-8) Optimizations, and the Percentage of Resets Reduced

Virtual Size	RDW Alg.	Resets Before			Resets After			Reduction Rate		
		MR2C	R2G	SFR2G	MR2C	R2G	SFR2G	MR2C	R2G	SFR2G
20x20	NS	3059.0	5331.1	5417.2	1464.0	1550.0	1919.9	52.14%	70.93%	64.56%
	S2C	3708.8	3646.1	4293.3	1717.9	1594.2	1735.8	53.68%	56.28%	59.57%
	P2R	1637.6	2733.5	2895.8	1468.7	1571.9	1785.9	10.31%	42.49%	38.33%
40x40	NS	6271.5	6066.2	14173.0	2698.5	2840.5	3025.8	56.97%	53.17%	78.65%
	S2C	5506.7	6973.6	10603.8	3176.0	2863.6	3156.2	42.32%	58.94%	70.24%
	P2R	3178.4	5264.3	5743.7	2611.1	2745.9	3578.2	17.85%	47.84%	37.70%

TABLE 4
The Results for Each RDW Algorithm and Reset Method of Environment 4, the Average Resets of 500 Simulations Before(columns 3-5) and After(columns 6-8) Optimizations, and the Percentage of Resets Reduced

Virtual Size	RDW Alg.	Resets Before			Resets After			Reduction Rate		
		MR2C	R2G	SFR2G	MR2C	R2G	SFR2G	MR2C	R2G	SFR2G
20x20	NS	4306.9	3326.1	2938.2	1629.6	1622.2	1930.0	62.16%	51.23%	34.31%
	S2C	3734.4	3209.0	3836.7	1750.9	1709.5	1853.4	53.11%	46.73%	51.69%
	P2R	2570.3	2525.7	2766.2	1613.3	1639.2	1757.9	37.23%	35.10%	36.45%
40x40	NS	7569.1	9014.8	6840.1	2903.3	2903.9	3360.6	61.64%	67.79%	50.87%
	S2C	11243.8	8004.8	10345.3	3505.4	3416.3	3483.0	68.82%	57.32%	66.33%
	P2R	5635.1	5169.8	6271.1	2827.29	3020.4	3422.8	49.83%	41.58%	45.42%

this physical environment, when the user collides with some points on the boundary of obstacles, it is not possible to reset to the center because the way from the point to the center is blocked by other obstacles. In order to deal with this situation, the reset vector assigned to each endpoint on the obstacle is perpendicular to it. When all the rectangular objects in the scene are orthogonal to space, this reset method is exactly the same as MR2C. For the reset at the endpoints on the environment boundary, suppose the reset endpoint is $\mathbf{p}_i$, the center of physical environment as $\mathbf{o}$, then the reset vector at this endpoint is $\mathbf{o} - \mathbf{p}_i$.

4.1.2 Reset Optimization for P2R

In the second environment, there is a 5m×6m rectangular obstacles on the bottom middle of the physical environment (Fig. 4b-1). Unlike Steer-to-Center, we apply Reset-to-Gradient to the redirected walking by artificial potential fields. The awkward situation of blocking center in Reset-to-Center will not happen here. Due to the repulsive force of the obstacle boundary, the reset vector always points out of the obstacle boundary.

Due to the characteristics of the artificial potential function, the closer to the boundary of the obstacle, the greater the force is received. Given the force is inversely proportional to the distance, we will have infinite force on the boundary of the obstacle. However, in practice, it is not the case that the center of the user hits the boundary because the user has a volume. In the simulation, we treat the user as a small point without the volume for simplicity. But we add a correction term δ to the repulsion function in the artificial potential fields to avoid infinite repulsion

$$U_{repulsive}(\mathbf{c}) = \sum_{O \in obstacles} \frac{1}{\|\mathbf{c} - \mathbf{p}_O\| + \delta},$$

where $\mathbf{c}$ denotes the current position of the user, $\mathbf{p}_O$ is its closest point on the obstacle O, δ determines the influence to

the user when close to one boundary. Here we let $\delta = 0.1$, which is comparable to the radius of a human head. The direction of the force is from the closest point on the obstacle to the user.

When the user collides with an obstacle, the composition of the forces from all obstacles will be almost perpendicular to the boundary, as shown in Fig. 4b-1. The problem comes when the user is restricted in a narrow space bounded by two opposite boundaries. Then colliding with one boundary will reset to the other boundary and vice versa, resulting in a bouncing situation between two boundaries with many resets.

4.1.3 *Reset Optimization For More Complicated Environment*

In order to evaluate our reset optimization as an independent algorithm in virtual navigation, we conduct experiments in two more environments (environment 3 as in Fig. 4c-1 and environment 4 as in Fig. 4d-1).

In environment 3, there are 4 rectangular obstacles in the area, including a 2m×3m rectangular obstacle at the bottom left, a 2m×2m square at the center, a 2m×2m square at the bottom right (1m away from both sides), and a 3m×4m rectangular obstacle at the top right (1m away from the right side). It can be observed that the area is approximately divided into three rectangular sub-areas by the obstacles. In environment 4, more complex and irregular boundaries are designed. It is difficult to identify the number and shape of sub-areas divided by obstacles in this case. And even being identified, it is hard to find their geometric centers like that of a rectangle. The remaining (compared with Sections 4.1.1 and 4.1.2) RDW method NS and reset method SFR2G are applied in environment 3, and the combination of P2R and SFR2G are used in environment 4 as we speculate that they are the most likely to achieve good results.

For environment 4, we design an additional comparative experiment to see how the edge interval affects the optimization. Besides the original experiments with parameter 1, edge intervals of 2 and of ∞ (which means every smooth edge only has 2 endpoints on both ends of the edge) are considered. As expected, a smaller edge interval should give better results at the end, because the smaller the edge interval, the more accurate the optimization. And we also expect the larger edge interval can optimize faster.

4.2 Results

In this sub-section, we perform statistical analysis on the pre-optimization and post-optimization results of each environment and the algorithms they use. First, we conduct normality analysis, and then use appropriate methods for comparisons based on the analysis results. In addition, even if a small part of the samples does not obey normal distribution, the average value can actually reflect the performance of the algorithm. This can also be seen in comparative tests (T-test and Mann-Whitney U-test), and the p-value obtained by the final statistical analysis is very small. In the following discussion, we default that the data in the table conform to the statistical conclusions. On this basis, we discuss the configuration and optimization effects of each algorithm.

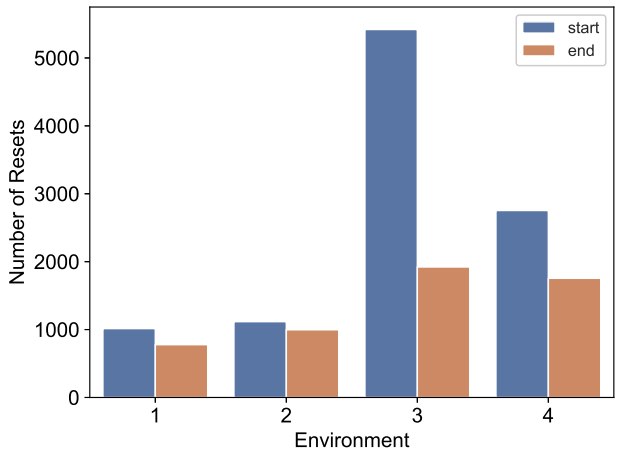


Fig. 6. The bar plot of resets before and after optimization from Environment 1 to Environment 4 respectively.

The optimization result of environment 1 is shown in Fig. 5a. It can be seen that the number of resets decreases as the optimization progresses. Fig. 4a-2 shows the reset vectors at the endpoints after optimization, and Fig. 6 shows a bar plot of the number of resets before (Mdn = 979.00, IQR = 99, M = 1014.21, STD = 105.37) and after (Mdn = 778, IQR = 28, M = 777.60, STD = 21.33) the optimization. We conducted Kolmogorov-Smirnov tests for 500 times of user walking simulation, the data after optimization obeys a normal distribution ($D = 0.0309, p = 0.714 > 0.05$), but the data before optimization do not ($D = 0.185, p < 0.001$), so we use Mann-Whitney U-test and find out there is a significant difference between the two sets of data (U≈0.0,p≪0.001).

The optimization result of environment 2 is shown in Fig. 5b. Similarly, the number of resets decreases dramatically after optimization. Fig. 4b-2 shows the reset vectors at the endpoints after optimization, and Fig. 6 shows a bar plot of the number of resets before (Mdn = 1113.5, IQR = 43.25, M = 1116.39, STD = 35.94) and after (Mdn = 997, IQR = 31.25, M = 996.93, STD = 23.99) the optimization. We conducted Kolmogorov-Smirnov tests for 500 times of user walking simulation before ($D = 0.043, p = 0.290 > 0.05$) and after ($D = 0.037, p = 0.500 > 0.05$) optimization, results show that they both obey the normal distribution, thus we conduct T-test for them ($t(499) = 61.80, p \ll 0.001$).

The optimization result of environment 3 is shown in Fig. 5c. It largely reduces the number of resets. Fig. 4c-2 shows the reset vectors at the endpoints after optimization, and Fig. 6 shows a bar plot of the number of resets before (Mdn = 5414, IQR = 836, M = 5418.93, STD = 625.61) and after (Mdn = 1909, IQR = 138.25, M = 1921.82, STD = 102.29) the optimization. We conducted Kolmogorov-Smirnov tests for 500 times of simulated user walking before ($D = 0.032, p = 0.645 > 0.05$) and after ($D = 0.054, 0.096 > 0.05$) optimization, results show that they both obey the normal distribution, thus we conduct T-test for them ($t(499) = 123.35, p \approx 0$).

The optimization of environment 4 is also effective as shown in Fig. 5d. Fig. 4d-2 shows the reset vectors at the endpoints after optimization. Fig. 6a shows a bar plot of the number of resets before (Mdn = 2712, IQR = 371.25, M = 2753.33, STD = 300.03) and after (Mdn = 1748.5, IQR = 54, M = 1755.21, STD = 69.64) the optimization. We conducted

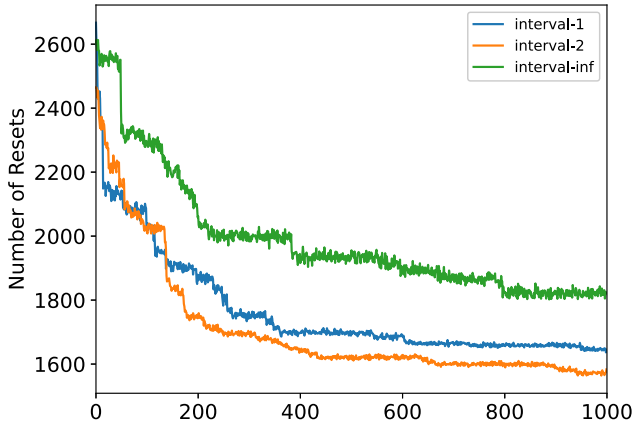


Fig. 7. The convergence plot of resets with different edge intervals (1, 2, and ∞) for Environment 4. X-axis represents the number of iterations during the optimization, Y-axis represents the average resets (over 500 walking simulations).

Kolmogorov-Smirnov tests for 500 times of simulated user walking before ($D = 0.063, p = 0.035 < 0.05$) and after ($D = 0.162, p = \ll 0.05$) optimization, results show that they both disobey the normal distribution, thus we conduct Mann-Whitney U-test and find out there is a significant difference between the two sets of data(U=215.0,p$\ll$0.001).. Results of additional experiment are in Fig. 7, the interval of ∞ got the worst performance compared with 1m and 2m, but 2m is actually better than 1m.

4.3 Discussion

The results show that the initial reset strategy has a lot of room for optimization. The number of resets largely reduces after optimization. By judging the resultant reset vectors, the differences from initial reset vectors are quite noticeable. Here we discuss the result for each environment in detail.

4.3.1 Discussion of Environment 1

From the result of environment 1(Fig. 4a-1), it turns out that all reset vectors point out of the obstacle boundaries. This is also true for duplicated reset vectors at sharp points. Although some of them originally point into the obstacle, they are all free of collision after optimization. The reset vectors on the environment boundary still point to the middle of the space, but not the exact center.

From Table 1 we can see that the optimization effect is not as good as other environments. The layout of this environment is nearly obstacle-free, and the middle part of the environment is relatively open, thus MR2C is nearly the best collision handling solution for all RDW algorithms. In addition, it can be seen that although SFR2G is better than the other two methods at first, there is not much room for optimization. Although the other two methods are not comparable to SFR2G at first, our method can make them exceed SFR2G. Also, Table 1 demonstrates that the virtual environment with a larger size brings more reduction rate, and in the following table we can find this conclusion holds for most cases.

4.3.2 Discussion of Environment 2

As shown in the results of environment 2(Fig. 4b-2), the empty space is narrow on the left and right of the area. Initially, the reset vectors at the endpoints therein are almost perpendicular to the boundary, except for the points near the corner. After optimization, these vectors point to the outside of this area with more open space.

In addition, the second reset endpoint from the bottom up on the right boundary of the obstacle does not point out of the narrow space. Instead, it points to the bottom of that space. The following formula on expected resets helps us to understand where this result comes from

$$E_{resets}[\mathbf{v}|\mathbf{p}] = \iint_{(x,y)\in L} G(x,y)P(x,y)\mathrm{d}x\mathrm{d}y,$$

where $\mathbf{v}|\mathbf{p}$ indicates resetting to the direction of vector $\mathbf{v}$ when the user collides with the boundary at point $\mathbf{p}$, L denotes the physical boundary the user may collide with, $G(x,y)$ represents the expectation of resets at the point (x,y), and $P(x,y)$ is the probability density of the next collision with (x,y). $E_{resets}[\mathbf{v}|\mathbf{p}]$ denotes the expected value of the number of resets when applying vector $\mathbf{v}$ on point $\mathbf{p}$ during the entire walking simulation (while other vectors of reset endpoints are fixed).

Take the aforementioned endpoint as an example. After colliding with this endpoint and resetting, the user may collide with the bottom edge shortly afterwards. Then the user's orientation is almost reset to upright, and could walk into an open space. Assume the vector points to the upper left, then it bears a high possibility to collide with the left boundary of the environment. In both cases, it has to go through one collision to enter the open space. And even after hitting the left boundary, it may collide with the left edge of the obstacle again. So for this endpoint, the current reset direction is not globally optimal, i.e., the optimization leads to a local minimum. Thus it is reasonable to update multiple parameters (reset vectors) at once, because doing so is helpful to jump out of the local minimum. But after optimization, the reset vectors could be like those in the bottom right space. They are not optimal in a greedy view due to a foreseeable collision, but are good resets in a long run statistically.

Table 2 also demonstrates that the optimization performs better when altering the virtual environment size from 20m$\times$20m to 40m$\times$40m. As the obstacle is larger, the space is more irregular than environment 1, the overall reduction rate is greater. Similar to Table 1, SFR2G does not exhibit a clear advantage in the initial and final resets, while our optimization on SFR2G is still effective.

4.3.3 Discussion of Environment 3 and 4

For environment 3, it may have some kind of weird reset vectors. In environment 1, all vectors pointing inside the obstacle are finally optimized to point outside the obstacle. In contrast, some vectors are optimized to pointing inside the obstacles while it originally pointing outside (Fig. 4c-2). But it is reasonable for SFR2G. When the user cannot walk forward during SFR2G, it is equivalent to R2G. If a reset vector points inside the obstacle, it means our optimization tends to apply R2G for resetting and face the direction of gradient descent. By observing Table 3, we can find R2G always performs better than SFR2G, and this is consistent with our inference. Also we can find a special case that for the combination of MR2C

and S2C, the reduction rate actually decreases when using the larger virtual environment. Therefore, the increase of virtual space does not always have a positive effect on the optimization space of the algorithm, which needs to be estimated based on the combination of RDW algorithms.

Similar conclusions with environment 3 hold as the results of environment 4. It can be seen from the final reset vectors (Fig. 4d-2) that the optimization results guide the user to the largest empty area in the upper half of the physical layout, and there is also some reset vectors point inside the obstacles. The explanation of this phenomenon is the same as that mentioned when discussing environment 3. The reduction rates of virtual size 40m×40m are larger than those of 20m×20m in Table 4, except MR2C with NS, whose reduction rate is higher than 60% when the virtual size is 20m×20m. Note that SFR2G outperforms another 2 reset methods in this environment on NS, and is only slightly worse than MR2C and R2G on P2R. Though SFR2G does not have the advantage over other methods after optimization (same as the previous environment), it still makes promising improvement. In general, the final result of SFR2G optimization will generally be slightly worse, which is also to be expected, because the final reset direction of SFR2G depends on the surrounding gradient, and the gradient around a certain boundary point generally only covers a small range of the angle, while the other two methods can be arbitrarily optimized within the range of 180 degrees.

The additional experiments (see Fig. 7) verify our understanding of the edge interval. Large edge interval results in worse performance since it cannot precisely control the reset vector at each position. But for two similar intervals, the larger one optimizes longer length of the boundary each time, in this case, the optimization speed increase effect brought by the length is more significant. And this can explain why the interval of 2m is better than 1m. Moreover, there are other factors that affect the optimization potential, such as it does not always work well that using the same fixed interval, the position of reset endpoints might be of great influence to the result. All these factors should be considered in further study.

5 User Study

5.1 Experiment Design

To evaluate how our optimization works in practice, we also conducted a user study with the help of human participants. The users were asked to walk in a 5x5m^2 area with a 2x2.5m^2 obstacle at the bottom center of the area in our meeting room, as shown in Fig. 8a. The real obstacle is a little smaller than the simulated one, and the real boundary is larger than the simulated one, such that the user could receive a prompt before hitting the real boundary. This can help to turn the user around to avoid real collision and possible injury. We used HTC VIVE HMD, which is suitable for our experiment in a space of this size. The virtual space size was $20 \times 20m^2$. The object (a blue capsule, as shown in Fig. 8b) was randomly generated in the virtual space. The user was required to move towards the object until touching it. Then the object disappeared and emerged in a new position at least 5m away from the current position. We used P2R as the RDW method for our irregularly shaped environment. In the user test, we compared 4 types of reset methods, including 1) our method after optimization(OPT); 2) Reset to Perpendicular(R2P) which re-orients the user to the direction that is perpendicular to the boundary; 3) the mix of R2P and R2C (since R2C does not work on every boundary, those boundaries that cannot apply R2C were replaced by R2P, and this is the same as MR2C); and 4) R2G which is the most commonly used in APF-RDW. SFR2G will not be applied here because the size of the physical environment may be too constrained to perform it. We performed 4 trials on each reset strategy. In each trial, the user was required to keep moving and touching the generated object until encountering 10 resets. Then we recorded the user's walking distance during the trial. The values of various gains are the same as those used in the simulation experiment. Different reset algorithms use the same parameters except for different reset behaviors. In order to reduce the experimental deviation caused by mutual influence between these 4 trails, the order of experiments for these reset algorithms is random.

(a) Our physical environment, the rectangular obstacle is desks with computer controlling our VR equipment.

(b) Our virtual environment. There will always be a blue capsule generate in the environment guiding user movement towards it.

Fig. 8. Physical(a) and virtual(b) environment in the user experiment.

We implemented a redirected walking interface in the VR environment. Whenever the user collides with the physical boundary, the perspective of the user is set frozen (Freeze-Turn). There are some texts and arrows to help the user adjust the direction in order to face the direction that a certain reset strategy provides. However, the user does not collide with the real object since the walkable area is larger than the simulated area, and the user is still able to walk outside the boundary. Only if the user is inside the simulated area and facing the correct direction, the perspective is activated. There are 12 participants included, composed of 8 females and 4 males, and about half of them have the experience of using virtual reality devices. They learned about this user study through the student work group and department group in the communication software, and volunteered to participate.

5.2 Result

Fig. 9 shows one of our walking paths using our OPT reset strategy. The physical path may look shorter because we doubled the user's walking distance in the virtual environment. The results of each reset method of their walking distance are shown in Fig. 10. We conducted Kolmogorov-Smirnov Test for 4 groups of results. Each of them fit the normal distribution ($D = 0.173, 0.142, 0.137, 0.190$, $p = 0.806$,

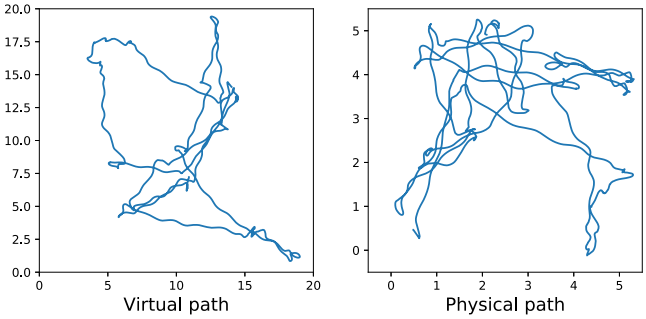


Fig. 9. Example simulated path. There are some parts outside the tracking physical area(5x5m^2) because the walkable area is larger than the simulated area to avoid hard collision.

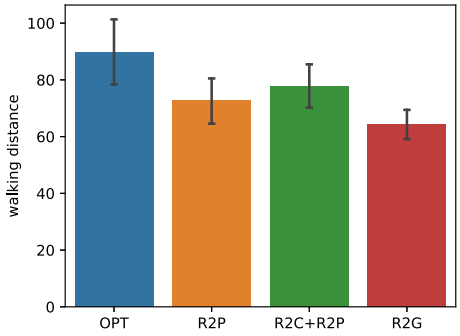


Fig. 10. The result of user study by measuring walking distance in 10 resets.

$0.938, 0.954, 0.709 > 0.05$). Then the ANOVA test shows there is a significant difference between the four reset methods($F(3,44) = 5.49$, $p < 0.005$). T-test shows our optimization method works better than R2P($t(11)=2.27$, $p < 0.02$), has advantages over the mix of R2C and R2P($t(11)=1.59$, $p < 0.07$), and significantly outperform R2G($t(11)=3.76$, $p < 0.001$). We also ask our subjects to fill Simulator Sickness Questionnaire(SSQ)[34] before and after the experiment, and four methods reached similar scores between 3.5 to 4.5, they all obey the normal distribution ($D = 0.215, 0.275, 0.250, 0.258, 0.250, p > 0.27$), ANOVA test shows no difference ($F(3,44)=0.13$, $p > 0.9$) between the sickness by 4 reset strategies.

5.3 Discussion

The user study proves the capability of reducing resets of our optimization algorithm. We notice the mix of R2C and R2P method work better than using R2P or R2G only. But in many situations, we cannot easily tell whether the mix of certain methods works well in which environment. And even it works well, our algorithm still possesses an advantage. In the aspect of user experience, our method should have similar performance with other methods because we uniformly use the Freeze-Turn method. However, the experience may also be determined by the resets, more resets brings more vertigo. Since our methods reduce the resets, it should provide less sickness during long term walking.

6 Conclusion and Future Work

In this work, we present a novel optimization algorithm for redirected walking (RDW) reset in irregular physical space with obstacles. Inspired by finite element analysis, we discretize the environment and obstacle boundaries, and optimize reset directions at discrete endpoints while interpolating directions elsewhere. The optimization is based on virtual navigation simulation in a stochastic process, and is adaptive to a given RDW algorithm. We conduct several experiments with different environment layouts and RDW algorithms. The results show that our optimization-based reset can significantly reduce resets in different scenarios, thus largely benefit the existing RDW algorithms. The results also highlight the choice of different combinations of RDW and resetting techniques exert a strong influence on the result during finite iterations of optimization. Theoretically, the in-place method should have the same optimal solution for reset vectors, however, due to the stochastic nature of the optimization process and the difference in optimization speed, different algorithms will bring differences in the results.

In the future, we would like to further improve the performance of our algorithm. The simulation-based optimization can be very fast. It takes 2-12 seconds to perform one optimization when $R_W = 500$ in a 20x20m^2 virtual area with a single 3GHz CPU, depending on which RDW algorithm is used for simulation. Since repetitions of walking are highly parallelizable (we also implemented the multi-core version on CPU, reducing 38% to 62% of the time with 2 and 4 cores), our algorithm can be transferred into modern GPU with numerous cores, significantly reducing the running time. We are also interested in studying how to further accelerate the simulation based on a pre-trained simulation regressor [35].

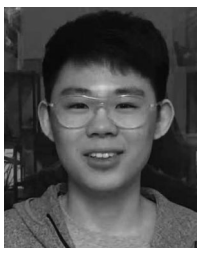
References

[1] T. Nescher, Y. Huang, and A. Kunz, "Planning redirection techniques for optimal free walking experience using model predictive control," in *Proc. IEEE Symp. 3D User Interfaces*, 2014, pp. 111–118.
[2] M. A. Zmuda, J. L. Wonser, E. R. Bachmann, and E. Hodgson, "Optimizing constrained-environment redirected walking instructions using search techniques," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 11, pp. 1872–1884, Nov. 2013.
[3] S. Razzaque, "Redirected walking," Ph.D. dissertation, Univ. North Carolina at Chapel Hill, USA, 2005. [Online]. Available: https://techreports.cs.unc.edu/papers/05–018.pdf
[4] E. Hodgson and E. R. Bachmann, "Comparing four approaches to generalized redirected walking: Simulation and live user data," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 4, pp. 634–643, Apr. 2013.
[5] E. Hodgson, E. Bachmann, and D. Waller, "Steering immersed users of virtual environments: Assessing the potential for spatial interference," *ACM Trans. Appl. Percept.*, vol. 8, pp. 1–22, 2011.
[6] D.-Y. Lee, Y.-H. Cho, and I.-K. Lee, "Real-time optimal planning for redirected walking using deep Q-learning," in *Proc. IEEE Conf. Virt. Reality 3D User Interfaces*, 2019, pp. 63–71.
[7] 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 *Proc. IEEE Conf. Virt. Reality 3D User Interfaces*, 2020, pp. 155–163.
[8] R. R. Strauss, R. Ramanujan, A. Becker, and T. C. Peck, "A steering algorithm for redirected walking using reinforcement learning," *IEEE Trans. Vis. Comput. Graphics*, vol. 26, no. 5, pp. 1955–1963, May 2020.
[9] Y. Chang, K. Matsumoto, T. Narumi, T. Tanikawa, and M. Hirose, "Redirection controller using reinforcement learning," *IEEE Access*, vol. 9, pp. 145 083–145 097, 2021.
[10] F. Steinicke, G. Bruder, J. Jerald, H. Frenz, and M. Lappe, "Estimation of detection thresholds for redirected walking techniques," *IEEE Trans. Vis. Comput. Graphics*, vol. 16, no. 1, pp. 17–21, Jan./Feb. 2010.
[11] T. Grechkin, J. Thomas, M. Azmandian, M. Bolas, and E. A. Suma, "Revisiting detection thresholds for redirected walking: Combining translation and curvature gains," in *Proc. ACM Symp. Appl. Percept.*, 2016, pp. 113–120.

[12] C. T. Neth, J. L. Souman, D. W. Engel, U. Kloos, H. H. Bulthoff, and B. J. Mohler, "Velocity-dependent dynamic curvature gain for redirected walking," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 7, pp. 1041–1052, Jul. 2012.
[13] R. Yu *et al.*, "Experiencing an invisible world war I battlefield through narrative-driven redirected walking in virtual reality," in *Proc. IEEE Conf. Virt. Reality 3D User Interfaces*, 2018, pp. 313–319.
[14] E. Langbehn, P. Lubos, G. Bruder, and F. Steinicke, "Application of redirected walking in room-scale VR," in *Proc. IEEE Virt. Reality*, 2017, pp. 449–450.
[15] E. Langbehn and F. Steinicke, "Space walk: A combination of subtle redirected walking techniques integrated with gameplay and narration," in *Proc. ACM SIGGRAPH Emerg. Technol.*, 2019, pp. 1–2.
[16] B. Williams *et al.*, "Exploring large virtual environments with an HMD when physical space is limited," in *Proc. 4th Symp. Appl. Percept. Graph. Vis.*, 2007, pp. 41–48.
[17] S. Freitag, D. Rausch, and T. Kuhlen, "Reorientation in virtual environments using interactive portals," in *Proc. IEEE Symp. 3D User Interfaces*, 2014, pp. 119–122.
[18] E. R. Bachmann, E. Hodgson, C. Hoffbauer, and J. Messinger, "Multi-user redirected walking and resetting using artificial potential fields," *IEEE Trans. Vis. Comput. Graphics*, vol. 25, no. 5, pp. 2022–2031, May 2019.
[19] T. Dong, X. Chen, Y. Song, W. Ying, and J. Fan, "Dynamic artificial potential fields for multi-user redirected walking," in *Proc. IEEE Conf. Virt. Reality 3D User Interfaces*, 2020, pp. 146–154.
[20] J. Thomas and E. S. Rosenberg, "A general reactive algorithm for redirected walking using artificial potential functions," in *Proc. 26th IEEE Conf. Virt. Reality 3D User Interfaces*, 2019, pp. 56–62.
[21] T. C. Peck, H. Fuchs, and M. C. Whitton, "Evaluation of reorientation techniques and distractors for walking in large virtual environments," *IEEE Trans. Vis. Comput. Graphics*, vol. 15, no. 3, pp. 383–394, May/Jun. 2009.
[22] M. Sra, X. Xu, A. Mottelson, and P. Maes, "VMotion: Designing a seamless walking experience in VR," in *Proc. Designing Interactive Syst. Conf.*, 2018, pp. 59–70.
[23] R. Cools and A. L. Simeone, "Investigating the effect of distractor interactivity for redirected walking in virtual reality," in *Proc. Symp. Spatial User Interaction*, 2019, pp. 1–5.
[24] B. R. Fajen and W. H. Warren, "Behavioral dynamics of steering, obstacle avoidance, and route selection," *J. Exp. Psychol.: Hum. Percept. Perform.*, vol. 29, no. 2, pp. 343–362, 2003.
[25] H. Chen, S. Chen, and E. S. Rosenberg, "Redirected walking strategies in irregularly shaped and dynamic physical environments," in *Proc. IEEE VR Workshop Everyday Virt. Reality*, 2018.
[26] H. Chen, S. Chen, and E. S. Rosenberg, "Redirected walking in irregularly shaped physical environments with dynamic obstacles," in *Proc. IEEE Conf. Virt. Reality 3D User Interfaces*, 2018, pp. 523–524.
[27] I. Valentini, G. Ballestin, C. Bassano, F. Solari, and M. Chessa, "Improving obstacle awareness to enhance interaction in virtual reality," in *Proc. IEEE Conf. Virt. Reality 3D User Interfaces*, 2020, pp. 44–52.
[28] A. Hrennikoff, "Solution of problems of elasticity by the framework method," *J. Appl. Mechanics*, vol. 8, no. 4, pp. 169–175, 1941.
[29] S. Kirkpatrick, C. D. G. Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[30] N. Sekiya, "The invariant relationship between step length and step rate during free walking," *J. Hum. Movement Stud.*, vol. 30, no. 6, pp. 241–257, 1996.
[31] J. N. Templeman, P. S. Denbrook, and L. E. Sibert, "Virtual locomotion: Walking in place through virtual environments," *Presence*, vol. 8, no. 6, pp. 598–617, 1999.
[32] M. Usoh *et al.*, "Walkingwalking-in-placeflying, in virtual environments," in *Proc. 26th Annu. Conf. Comput. Graph. Interactive Techn.*, 1999, pp. 359–364.
[33] J. Messinger, E. Hodgson, and E. R. Bachmann, "Effects of tracking area shape and size on artificial potential field redirected walking," in *Proc. IEEE Conf. Virt. Reality 3D User Interfaces*, 2019, pp. 72–80.
[34] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal, "Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness," *Int. J. Aviation Psychol.*, vol. 3, no. 3, pp. 203–220, 1993.
[35] T. Feng, L.-F. Yu, S.-K. Yeung, K. Yin, and K. Zhou, "Crowd-driven mid-scale layout design," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 132–1, 2016.

Song-Hai Zhang (Member, IEEE) received the PhD degree of computer science and technology from Tsinghua University, Beijing, China, in 2007. He is currently an associate professor in the Department of Computer Science and Technology, Tsinghua University,China. His research interests include computer graphics, virtual reality, and image/video processing.

Chia-Hao Chen received the BS degree of computer science and technology from Nanjing University, Nanjing, China, in 2020. He is currently working toward the master's degree in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include computer graphics, media analysis, and computer vision.



Fu Zheng is currently working toward the undergraduate degree in Xinya College, Tsinghua University, China. He majors in computer science and technology, with a second majoy in digital entertainment design. He will work as a video game designer after graduation.

Yong-Liang Yang received the BS and PhD degrees of computer science from Tsinghua University, China. He is a senior lecturer in the Department of Computer Science, University of Bath, U.K. His research interests are broadly in visual computing and interactive techniques.

Shi-Min Hu (Senior Member, IEEE) received the PhD degree from Zhejiang University, China, in 1996. He is currently a professor in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include digital geometry processing, video processing, rendering, computer animation, and computer-aided geometric design. He has published more than 100 papers in journals and refereed conferences. He is editor-in-chief of the *Computational Visual Media (Springer)*, and on editorial board of several journals, including the *Computer Aided Design (Elsevier)* and the *Computers & Graphics (Elsevier)*.