IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 7, 2020, accepted September 22, 2020, date of publication October 5, 2020, date of current version October 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3027985

Mapping Various Large Virtual Spaces to Small Real Spaces: A Novel Redirected Walking Method for Immersive VR Navigation

HUIYU LI[1] AND LINWEI FAN[2,3]

[1]School of Computer Science and Technology, Shandong University, Jinan 250101, China
[2]School of Computer Science and Technology, Shandong University of Finance and Economics, Jinan 250014, China
[3]Shandong Co-Innovation Center of Future Intelligent Computing, Yantai 264005, China

Corresponding author: Huiyu Li (huiyl91@163.com)

This work was supported in part by the National Nature Science Foundation of China under Grant 62002200, Grant 61472227, and Grant 61802229; in part by the Shandong Co-Innovation Center of Future Intelligent Computing (Shandong 2011 Project); and in part by the Natural Science Foundation of Shandong Province under Grant ZR2018BF007.

ABSTRACT Real-walking techniques such as redirected walking can provide users with a highly immersive presence in virtual environments. However, different space sizes or boundary shapes of virtual and real spaces limit the performance of real-walking techniques. This article proposes a novel redirected walking method supporting users continuously navigating in various large virtual spaces by real-walking in small real spaces. Herein, we present a Voronoi-based method to generate paths called the skeleton graph consisting of navigable paths and way-points. To map the skeleton graph of the virtual space to the real space, we propose a static graph mapping method adopting relocation and curvature adjustment. Additionally, a global optimization with boundary constraints is applied to minimize the total curvature of all paths, resulting in the complete convergence of the mapped paths in real space. By applying both virtual and mapped skeleton graphs in the roaming, our method provides users with a continuous walking experience without any interruptions. The experimental results show that unlike the existing methods that are only effective for specific virtual spaces, our method enables effective real-walking roaming in various large virtual spaces, that have polygon-shaped floor plans, and successfully reduces the number of collisions and perceptual distortion in the virtual scenes.

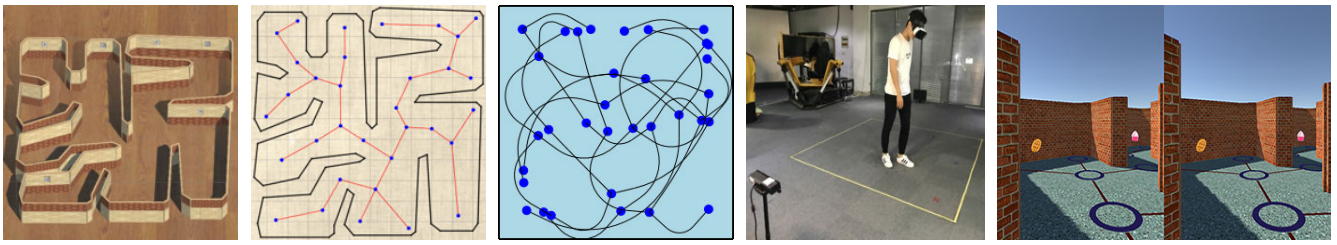
INDEX TERMS Virtual reality, redirected walking, head-mounted display, real-walking, navigation.

I. INTRODUCTION

With the rapid development of virtual reality (VR) in both hardware and software, users can now freely navigate through virtual spaces. Real-walking with VR devices, such as projected rooms (CAVEs) and head-mounted displays (HMDs), has been shown to provide a more natural user experience, in terms of presence and immersion, than walking-in-place and joystick-based locomotion [1], [2]. For example, [3], [4] allow users to physically walk in a well-designed virtual space that shares the same size and shape as the real space. However, real-walking requires sufficiently large real spaces, that almost always have different (usually smaller) sizes and shapes from the corresponding virtual spaces. Thus, proper mapping between the two spaces is necessary to offer a believable presence in the virtual space and feasible navigation in the real space. Solving this problem remains an important issue in VR navigation.

To allow users to walk through virtual spaces larger than the real space, a variety of techniques, such as space manipulation [5]–[7], planar mapping [8], [9] and redirected walking [10]–[12], have been proposed to preserve the feeling of moving naturally through virtual spaces while simultaneously keeping the user physically constrained within the real space. Among these, the space manipulation technique [6] adopts self-overlapping virtual architectural layouts to effectively compress large interior virtual spaces into smaller real spaces. However, this technique is not sufficiently general to provide navigation guidance because it is only suitable for a specific kind of virtual space (e.g. rooms or corridors). In the planar

The associate editor coordinating the review of this manuscript and approving it for publication was Luigi De Russis.



(a) (b) (c) (d) (e)

FIGURE 1. Overview of the proposed redirected walking method: the user perceives to walk along the skeleton graph (b) in virtual space (9.4 *m* × 9.5 *m*) (a) while real walking along the mapped skeleton graph (c) in real space (2.5 *m* × 2.5 *m*) (d). Herein, one snapshot of the user's HMD view when the user navigates in virtual space as shown in (e).

mapping technique [8], a smooth mapping from a given virtual space to a custom real space is computed. Although this technique avoids breaking up the flow of the user experience, the mapping procedure of this technique may strongly distort the geometric structure of the mapped virtual space, reducing both visual and locomotion fidelity to users.

Compared with space manipulation and planar mapping techniques, redirected walking [11] builds upon the principle that the brain considers visual cues to dominate the natural movement of the human body. Therefore, this technique introduces subtle differences between motions in the real world and what is perceived in the virtual world. Since redirected walking only manipulates the user's motion state (position and rotation), it can be applied to more kinds of virtual spaces, while providing undistorted visual artifacts of the virtual space. However, redirected walking does not provide general methods to map between a given pair of virtual and real spaces. In addition, to prevent users from colliding with the boundaries of the real space, redirected walking often interrupts users and requires them to reset their rotation.

Therefore, we propose a novel redirected walking method supporting continuous user navigation in various large virtual spaces by real walking in a given small real space. In this article, we represent both virtual and real spaces as floor plans, and adopt an HMD as the VR device to allow free navigation. The virtual spaces under consideration in this work are those for which the floor plans satisfy simple polygonal shapes (polygons without inner holes) such as indoor or polygonal constrained areas of outdoor virtual spaces. Our method consists of the pre-computing stage and the real-time updating stage. For a given virtual space, our method automatically generates the skeleton graph of the virtual space, including way-points and straight line paths, and calculates the mapped skeleton graph by mapping the virtual skeleton graph to the real space, in the pre-computing stage. By applying position inverse mapping and proper rotation redirection during the real-time updating stage, our method effectively guides the users to navigate with the mapped skeleton graph in the real space while perceiving walking along the virtual skeleton graph in the virtual space. Fig. 1 shows an overview of our system.

Both skeleton graph generation and skeleton graph mapping have been studied in redirected walking methods. However, our redirected walking method has several innovative features that differ from the prior art. Previous skeleton graph generation of redirected walking methods [13], [14] only adopt the navigation mesh or Voronoi diagram that contain redundant way-points and paths for VR navigation, as the skeleton graph of the virtual space. By contrast, our method further optimizes the Voronoi diagram of the virtual space by density-based spatial clustering and hierarchical Voronoi diagram strategy in order to reduce the redundancy of the generated skeleton graph. Previous redirected walking methods that support skeleton graph mapping such as FORCE [15] and MPCRed [16] dynamically choose and weight different redirection techniques to steer users to navigate along a series of given paths and way-points in the virtual space. However, these existing methods apply reset techniques [17] to reorienting users away from the boundaries of the real space, resulting in frequent interruptions. By contrast, our method maps the entire virtual skeleton graph to the real space in the pre-computing stage, thus establishing the mapping relations between virtual and real spaces. In this manner, our method only applies curvature gains (a continuous rotation applied while the user is walking forwards) [18] and does not adopt the reset technique, providing users with a continuously immersive experience.

In summary, the main contributions of this article include the following:

(1) We propose a novel redirected walking method that allows users to navigate in various large virtual spaces by real walking in small real spaces. This method is suitable in many different virtual spaces that are satisfied simple polygonal shapes;

(2) A static graph mapping method using global optimization is proposed to map all of the way-points and paths of the skeleton graph from the virtual space to the real space by relocation and appropriate curvature adjustment. It provides users with a continuous walking experience, making the navigation very smooth;

(3) To increase robustness and usability, our method extends all of the way-points and paths of the skeleton graphs in both virtual and real spaces. In this manner, our method

can allow users to appropriately leave the skeleton graph in navigation, expanding the reachable areas of the virtual space.

The remainder of this article is organized as follows: Section II reviews the literature related to real-walking techniques, Section III describes our novel redirected walking method supporting users navigating in various virtual spaces by walking in the small real space, Section IV shows the results from a user study, and Section V summarizes the contributions of this work in addition to proposing the limitations and future enhancements.

II. RELATED WORK

A. REAL WALKING

Typical immersive VR systems include walking-in-place and real-walking [1]. Previous studies [1], [19] prove that real-walking provides a more immersive experience compared to walking-in-place [20]–[22] or other indirect navigation methods. To support real walking in smaller real spaces to experience the larger virtual spaces, many methods have been proposed such as redirected walking [10], space manipulation [6] and planar mapping [8]. Below we review the first category, i.e., redirected walking technique.

B. REDIRECTED WALKING

The redirected walking (RDW) technique is originally proposed by Razzaque *et al.* [10]. Generally, this technique steers users walking along a pre-defined virtual path through the application of imperceptible locomotion gains to the user's natural motion, while in fact users are moving around within the tracked-space. According to [23], [24], different types of locomotion gains have been generalized, including rotation gain, curvature gain, and translation gain. Furthermore, Steinicke *et al.* evaluated detection thresholds for different locomotion gains [25], [26]. By combining different locomotion gains, a variety of RDW techniques have been proposed in recent years. According to [27], there are three categories for RDW techniques:

1) REACTIVE RDW

The main principle of reactive RDW [11] technique is to use subtle continuous manipulations to guide users toward a certain physical place or pattern regardless of a user's future virtual travel. In general, prior works [11], [28]–[30] described three generalized methods: steer-to-center (S2C), which focuses on guiding users towards the center of real space; steer-to-multiple-waypoints (S2W), which always guides users to a set of points; and steer-to-orbit (S2O), which guides users along a circular path around the center. Although these methods are flexible to redirect users in the virtual space, they cannot guarantee that a user remains safely within the real space. Therefore, the reorientation reset technique [12], [17], which effectively prevents users from colliding with a nearby real space boundary, is generally adopted in reactive RDW technique to keep users inside the real space. Compared with other RDWs, the reactive RDW works effectively in any real space. However, this technique usually ignores user's future virtual motions and causes frequent interruptions in presence, causing negative impacts.

2) PREDICTIVE RDW

The predictive RDW technique analyzes the virtual and real spaces to predict users' future motions to manipulate redirection. Some predictive RDW techniques [18], [31], [32] use the characteristics of human walking behavior to determine the steering strategies. Others [15], [16] adopt planning method to select proper locomotion gains, to dynamically calculate optimal physical locations, which are used to steer the user toward, by a series of pre-computed navigable paths and decision points. The FORCE method [15] employs probabilistic planning and adopts a terminal state evaluation function to determine feasible gains that have to be used. The MPCRed technique [16] treats different redirected walking techniques, including basic locomotion gains and resetting technique, as actions and provides state update function for these actions to optimize space and minimize costs. However, both FORCE and MPCRed use a manually defined skeleton graph to properly plan the redirection, while the applied resetting technique of the two methods may cause breaks in presence.

3) RESETTING RDW

To prevent the user from leaving the real space, the resetting RDW technique [12], [17] is applied to reorient users away from a nearby real space boundary. The resetting RDW technique requires users to continuously rotate in-place to readjust users towards the center of real space. To reset the users' orientation, Williams *et al.* developed several approaches [12]: the overt 2:1 turn, freeze-turn, and visual distractors. These studies prove that resetting RDW can work effectively in any size real space. However, frequent interruptions inevitably increase as the size of the real space diminishes, seriously affecting user experience.

III. METHODOLOGY

Generally, the proposed method provides users with a skeleton graph that is automatically generated in the virtual space, and allows them to navigate through the mapped skeleton graph in real space. With proper mapping, the method can effectively keep users inside the real space, and cause less perceptual distortion during navigation. Our method is split into two stages: the **pre-computing stage** and the **real-time updating stage**.

The procedure of our method is shown in Fig. 2. In the **pre-computing stage**, given the 2D floor plans for the virtual space S_v and real space S_r, our method first computes a simplified Voronoi skeleton graph of S_v (see Section III-A). Then, a static graph mapping method is implemented to map the skeleton graph (the user's paths) from S_v to S_r (see Section III-B). Moreover, our method expands the waypoints and paths of both skeleton graphs to extend the reachable areas of the virtual space. Fig. 3 illustrates several pre-computing results for different inputs. In the **real-time**

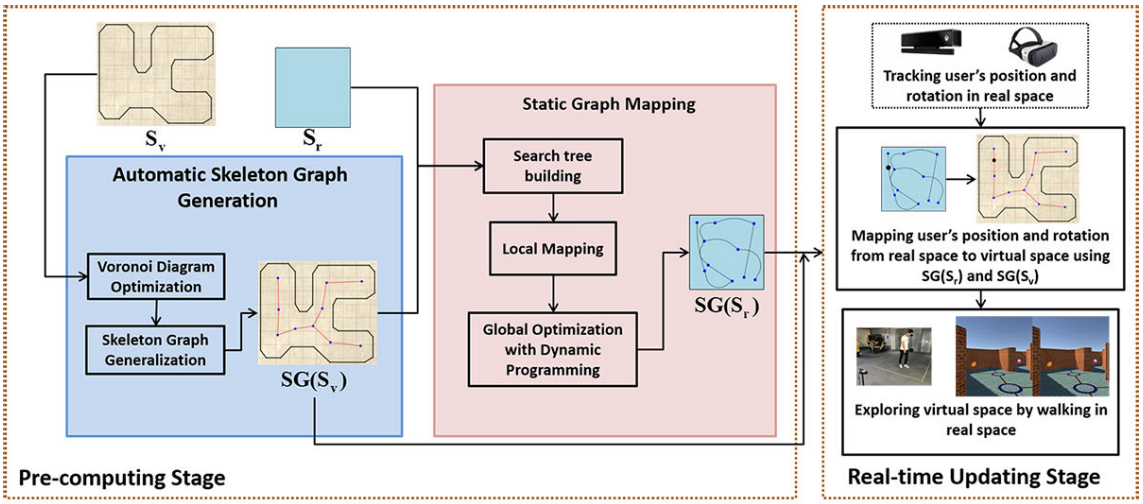


FIGURE 2. The procedure of our method. In the pre-computing stage, our method first computes a simplified Voronoi skeleton graph of the virtual space. Then, a static graph mapping method is conducted to map the skeleton graph from the virtual space to real space. In the real-time updating stage, the user's virtual position and rotation are dynamically calculated by the tracked information of the user in the real space.

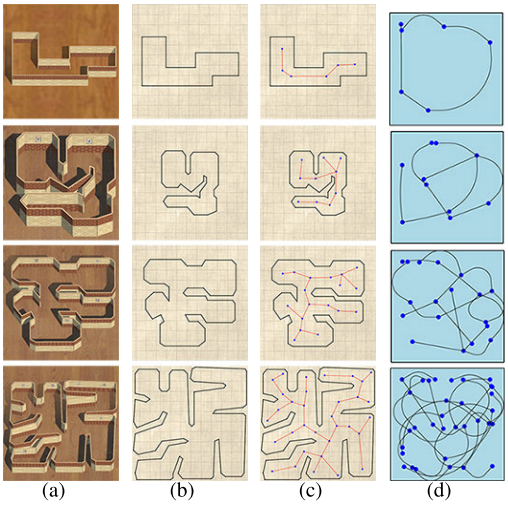


FIGURE 3. Pre-compute examples: (a) Virtual spaces with different sizes (from top to bottom: 6.6 *m* × 3.4 *m*, 4.8 *m* × 5.3 *m*, 7.2 *m* × 7.6 *m*, and 9.4 *m* × 9.5 *m*); (b) 2D floor plans extracted from the virtual spaces; (c) The skeleton graphs which are automatically generated in virtual spaces; (d) The static graph mapping results of skeleton graphs in real spaces (2.5 *m* × 2.5 *m*).

updating stage, the user's virtual position and rotation are determined by the tracked information in real space using position inverse mapping and proper rotation redirection (see Section III-C). To improve real-time performance, most computations including automatic skeleton graph generation and static graph mapping are conducted in the pre-computing stage.

A. AUTOMATIC SKELETON GRAPH GENERATION

In this step, our method automatically generates a skeleton graph that contains way-points and paths to represent the skeleton of S_v. An overview of the graph generation method is shown in Fig. 4.

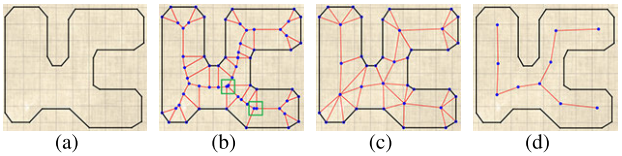


FIGURE 4. Overview of the graph generation method: (a) Input S_v; (b) Voronoi diagram of S_v; (c) The approximate Voronoi diagram generated by the original Voronoi diagram using a clustering method; (d) 2-VD as the result of the graph generation method.

1) INPUTS

Our method takes S_v as input to generate the skeleton graph. To simplify the computation, the representations of S_v are planar 2D polygonal shapes. In practice, these shapes are extracted from floor plans or design figures of virtual spaces, as shown in Fig. 3(b).

2) VORONOI DIAGRAM OPTIMIZATION

Since a Voronoi diagram (VD) has the characteristics of a maximum clearance circle and is often used for collision-free path planning [33], we adopt VD to generate the skeleton graph (SG) for a given S_v. Each Voronoi edge of VD(S_v) can represent the path for navigation and each vertex can be treated as the way-point for the user to select the next path. However, the Voronoi vertices of VD(S_v) can remain locally intensive, as shown in the green box of Fig. 4(b), unnecessarily increasing computational cost in the subsequent process. To overcome this problem, we simplify VD(S_v) with a clustering method to reduce the number of Voronoi vertices.

Inspired by the prior density-based spatial clustering method [34], our clustering method computes the local density ρ_i for each Voronoi vertex v_i to facilitate analytical computation of clustering. The quantity ρ_i is defined as

$$\rho_i = \sum_j \chi(d_{ij} - d_c) \tag{1}$$

where v_j is the vertex that is directly connected to v_i, d_{ij} is the Euclidean distance between vertices v_i and v_j and d_c is the cutoff distance. The value of d_c is related to the radius of the way-point that is expanded to a circle region in Section III-B3. The function $\chi(x)$ satisfies:

$$\chi(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases} \tag{2}$$

Since we aim to reduce the vertex density of VD(S_v), an iteration strategy is adopted to replace the clustered vertex sets with single vertex, as shown in Fig. 5. For each iteration, we first extract the vertex v_{max} that has the maximum ρ_{max} and its relevant clustering vertex set V_{max}. By searching the edges from VD, we obtain an edge set E_{max} for which the edges satisfy that the connected vertices are all covered by V_{max}. Finally, we replace the set V_{max} with the clustering center v_{max} and remove E_{max} from VD(S_v). To maintain the original topology of VD(S_v), the edges that connect to V_{max} are reconnected to v_{max}. All vertices of VD(S_v) are processed iteratively, until there are no $\rho_i > 1$. The clustering result is shown in Fig. 4(c).

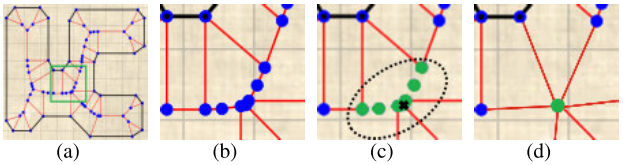


FIGURE 5. The procedure of clustering method: (a) Voronoi diagram of S_v; (b) The enlarged part of green box; (c) The cluster point set (green points) and the clustering center (black cross); (d) Local clustering result.

3) SKELETON GRAPH GENERALIZATION

After the clustering process, VD(S_v) can be simplified to an approximate Voronoi diagram AVD(S_v). Although the density of AVD(S_v) is simplified, unnecessary edges that are redundant for the skeleton graph remain. To further simplify AVD(S_v), we adopt a hierarchical Voronoi diagram strategy proposed by [35]. In our method, the hierarchical Voronoi diagram HVD(S_v) consists of two diagrams: a first level Voronoi diagram 1-VD(S_v) and a second level Voronoi diagram 2-VD(S_v). The 1-VD(S_v) is represented by AVD(S_v) and 2-VD(S_v) is calculated by simplifying 1-VD(S_v). To build 2-VD(S_v), the Voronoi vertex with the value of 1 and its associated Voronoi edges are deleted in 1-VD(S_v) until the degree-1 Voronoi vertices are all removed. Finally, we take 2-VD(S_v) as the skeleton graph SG(S_v) (please refer to Fig. 4(d)).

B. STATIC GRAPH MAPPING

This step focuses on mapping the skeleton graph SG(S_v) of the virtual space to an optimized skeleton graph SG(S_r) of the real space. As pointed out by [28], the user who is directed along an arc path within real space has the potential to explore a longer straight-line path in virtual space. Therefore, compared with straight line paths, arc paths are easier to converge in real space. In this article, to fold SG(S_v) into S_r, all paths (indicated by straight line segments) are transformed into arc paths by adjusting the curvature, while all way-points are relocated. The mapping procedure can be described as a search strategy that progressively maps all of the paths and way-points of SG(S_v). For a given SG(S_v), we first build a search tree to establish the search order of all paths and way-points. Initially, the root way-point is directly translated to an initial position of S_r. Then, the leaf paths and associated leaf way-points are simultaneously mapped to S_r using a local mapping method. Following the similar step, the leaf paths and leaf way-points of a searched way-point can be mapped. The procedure is performed iteratively until all paths and way-points are properly mapped. During the mapping procedure, a global optimization with boundary constraints is applied to minimize the total curvature of all paths and realize complete convergence in real space. Fig. 6 illustrates the mapping procedure of a given SG(S_v).

1) SEARCH TREE

Given the starting way-point of SG(S_v) (i.e., the user's start position in S_v), a search tree that begins with the starting way-point is constructed by the DFS algorithm [36]. In the search tree, each node corresponds to a way-point of SG(S_v), and each edge indicates a path of SG(S_v). Therefore, according to the search tree, we can determine a certain search order of all of the way-points and paths (see Fig. 6(b)).

2) LOCAL MAPPING

During the mapping procedure, the mapping results of a path and its terminal way-point depend on the starting way-point of the path. Considering the mapping shown in Fig. 7, assuming that way-point x_{v0} is properly mapped to the real way-point x_{r0}, we now map the leaf way-point x_{ve} and the leaf path between x_{v0} and x_{ve} to the real way-point x_{re} and the real path between x_{r0} and x_{re}. The vector v_v defines the magnitude and direction of the leaf path. Way-point x_{te} and vector v_t are the results of directly translating x_{ve} and v_v to x_{r0}. Clearly, v_t defines the motion path when a user locates on x_{r0} and walks along the direction of v_v. Hence, the point x_t lies on v_t and can be calculated by a linear function:

$$x_t = sv_t + x_{r0}, \quad s = \frac{|x_v - x_{v0}|}{|v_v|} \tag{3}$$

where $x_v \in v_v$, while x_{v0} and x_{r0} are the starting points of v_v and v_t. Since the local mapping method maps a straight line path to an arc path, v_t is used to compute the real path.

To maintain the continuity of SG(S_r), the real path shares the same starting point with v_t, and its length l equals $|v_t|$. In addition, the start tangent vector of the real path shares the same direction with v_t. The vector $v_\perp$ is perpendicular to v_t. Considering the transformation between v_t and the real path shown in Fig. 7(b), for point x_t in v_t, the mapped point x_r is given by

$$x_r = m + n + x_{r0} \tag{4}$$

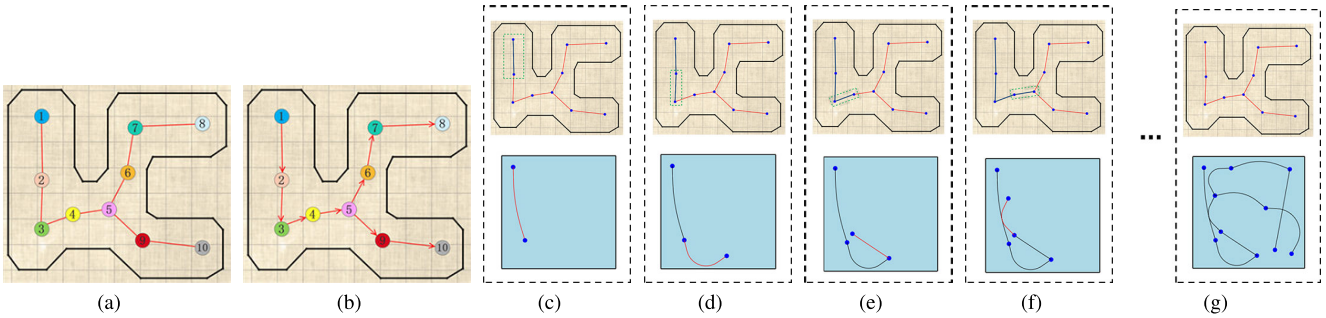


FIGURE 6. Overview of the static graph mapping method: (a) Input SG(S_v); (b) The search order of the paths and way-points determined by the search tree in SG(S_v); (c)-(f) iteratively map paths and way-points by a local mapping method following the search order; (g) All paths and way-points are mapped to S_r and optimized by a global optimization with boundary constraint.

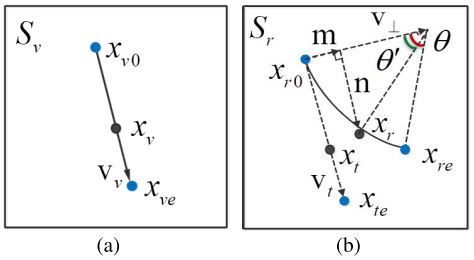


FIGURE 7. Local mapping procedure between S_v and S_r. The straight line path in SG(S_v) is mapped to an arc path in SG(S_r).

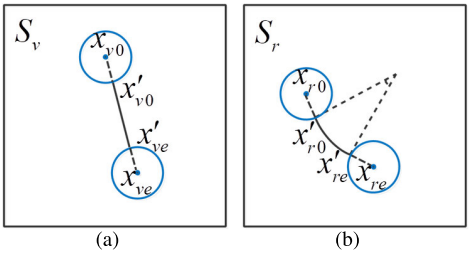


FIGURE 8. The procedure of way-point extension in both S_v and S_r.

where $\mathbf{m}$, $\mathbf{n}$ are two affine vectors. Given the central angle θ of the real path $\overset{\frown}{x_{r0}x_{re}}$ (see Fig. 7(b)), the radius r can be easily calculated as $r = l/\theta$. Using Equation 3, the current central angle θ' of $\overset{\frown}{x_{r0}x_r}$ can be defined as $\theta' = s\theta$. We then compute $\mathbf{m}$, $\mathbf{n}$:

$$m = (r - r\cos\theta')\frac{v_\perp}{|v_\perp|}, \quad n = r\sin\theta'\frac{v_t}{|v_t|} \tag{5}$$

By applying $\mathbf{m}$,$\mathbf{n}$ to Equation 4, the mapping relation between x_v and x_r can be formulated as

$$\begin{aligned} x_r &= (r - r\cos(s\theta))\frac{v_\perp}{|v_\perp|} + r\sin(s\theta)\frac{v_t}{|v_t|} + x_{r0} \\ s &= \frac{|x_v - x_{v0}|}{|v_v|} \end{aligned} \tag{6}$$

Since the path mapping is completed, the associated leaf way-point is located at the end of the real path. To provide suitable θ for each real path, a global optimization will be introduced in the following subsection.

3) WAY-POINT EXTENSION

In the current mapping procedure, the way-points are represented as single points. Thus, it is difficult for a user to know whether he has succeeded in reaching a certain way-point during navigation. This issue significantly decreases the robustness of our method. Therefore, we adopt an effective method to solve this issue by redefining the way-points as circular regions. For both SG(S_v) and SG(S_r), each way-point is replaced by a circular region with fixed radius r_e. With the increase in r_e, the circular region is more likely to intersect the boundary. Therefore, a suitable value is highly necessary.

Due the extension of the way-points, some changes to the local mapping method are necessary as shown in Fig. 8. For SG(S_v), the leaf path is reset to start with x'_{v0} and end with x'_{ve}, where x'_{v0} and x'_{ve} are the two intersections of the original line segment between $\odot x_{v0}$ and $\odot x_{ve}$. The same transformations are applied to the mapped way-point and the arc path in SG(S_r). With the extension of the way-points, users can freely walk around a way-point and easily reach the desired way-point.

4) GLOBAL OPTIMIZATION WITH BOUNDARY CONSTRAINT

The procedure for correctly mapping SG(S_v) to SG(S_r) is explained above. However, a part of SG(S_r) may be located outside S_r. To keep all points x_r of SG(S_r) inside S_r, a boundary constraint is provided for our mapping method. Since S_r is constructed by a rectangular convex hull, we use a set of straight line functions $\{L_j\}$ to indicate S_r. Let o_c be the center of S_r; x_r is guaranteed to be inside S_r as long as x_r and o_c are placed on the same sides of all L_j. Therefore, a series of linear constraints $\left((x_r\ L_j)^T\ (o_c\ L_j)\right) > 0$ are applied during the mapping procedure. The goal of the optimization is to minimize an energy function that captures the undesirable outcomes [37] and satisfies the boundary constraint. According to the mapping procedure, we first fix an initial position u_h of the real space S_r and a root way-point w_k of SG(S_v) to generate the basic energy function. Let θ_{p_i} be the central angle of an arc path $\overset{\frown}{p_i}$, where $\overset{\frown}{p_i} \in P$, P is the set of all arc paths of SG(S_r). Previous research [25] has revealed that a lower

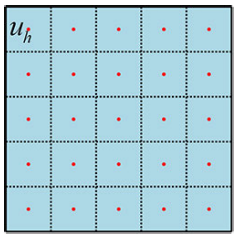


FIGURE 9. Sample initial positions for the real space in our experiment (2.5 m × 2.5 m). Each red dot indicates a possible initial position u_h.

redirection rate indicates less perceptual distortion. However, the arc path with a large θ_{p_i} causes a large redirection rate. Therefore, the basic energy function with constraint can be summarized as:

$$E(u_h) = \min \sum_{i}^{N} \theta_{p_i} \quad s.t. \ \left(x_r \, L_j\right)^T \left(o_c \, L_j\right) > 0 \tag{7}$$

However, $E(u_h)$ may vary as u_h changes. Thus, we perform stratified sampling with each stratum containing 4% of the area of S_r to generate a set of all possible initial positions U. A sampling example is shown in Fig. 9. The optimal basic energy function can be written as follows:

$$E_{opt}(w_k) = \min_{u_h \in U} \{E(u_h)\} \tag{8}$$

Different search trees can be created by choosing different way-points as roots of SG(S_v). Thus, SG(S_v) obtains different optimization energies with different search orders. Let W be the set of all way-points in SG(S_v). The final optimization function becomes

$$E_{final} = \min_{w_k \in W} \left\{E_{opt}(w_k)\right\} \tag{9}$$

By combining all of these optimizations and constraints with the mapping procedure, an optimal skeleton graph of S_r can be calculated from an input skeleton graph of S_v.

C. REAL-TIME UPDATING

In the pre-computing stage, the method generates the skeleton graph of the virtual space and maps the skeleton graph to the real space, as described in Sections III-A - III-B. However, an update from the current user's position and rotation in the real space to virtual space is necessary for VR navigation. To enable users to correctly navigate with the skeleton graph in virtual space, several procedures that dynamically adjust the user's position and rotation are applied and described bellow.

1) POSITION INVERSE MAPPING

Given the current user position $x_r(t)$ at time step t, our goal is to calculate the corresponding virtual position $x_v(t)$. Let Δt and $x_r(t - \Delta t)$ be the average frame rate and the last user position for each frame, respectively. Since user motion is continuous, the mapping of $x_r(t)$ is determined by $x_r(t - \Delta t)$. The user's real position satisfies the following three cases: (1). If $x_r(t - \Delta t)$ and $x_r(t)$ are both located on a way-point of SG(S_r), the mapping position $x_v(t)$ can be calculated by directly translating $x_r(t)$ to the corresponding way-point of SG(S_v). (2). If $x_r(t - \Delta t)$ is located on a way-point of SG(S_r) while $x_r(t)$ is outside, the path that $x_r(t)$ belongs to is selected by determining the distance between each associated path and $x_r(t)$. (3). If $x_r(t-\Delta t)$ and $x_r(t)$ are both located on the path of SG(S_r), we use the inverse function of Equation 6 to calculate the inverse mapping position $x_v(t)$.

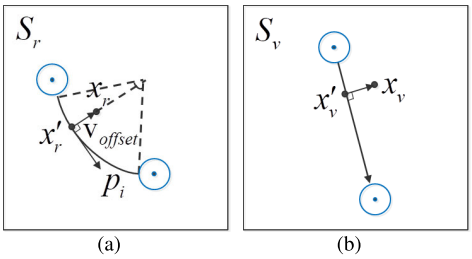


FIGURE 10. The procedure of path extension. Using offset vectors, the corresponding position on the virtual path can be calculated from the current real position.

2) PATH EXTENSION

We provide the position inverse mapping to calculate the virtual position in S_v. However, this method may be inefficient if the user deviates from the real paths. To solve this problem, we apply a path extension for each real path as shown in Fig. 10. Let $\overset{\frown}{p_i}$ be the real path that the user walks on, and x_r is the current user position in S_r. Noticing that the user has left $\overset{\frown}{p_i}$, we calculate an ideal position x'_r that lies on $\overset{\frown}{p_i}$. At the same time, x'_r is collinear with x_r and the center of the circle to which $\overset{\frown}{p_i}$ belongs. Thus, the vector $\mathbf{v}_{offset} = x_r - x'_r$ indicates the offset between x_r and x'_r. Using the inverse of Equation 6, x'_r and $\mathbf{v}_{offset}$ can be transformed to the corresponding virtual path of $\overset{\frown}{p_i}$ to generate the inverse mapping position of x_r.

3) ROTATION REDIRECTION

Paths in SG(S_v) are represented by straight line segments but are indicated by arc segments in SG(S_r). Since we have established the mapping relation between virtual and real skeleton graphs, appropriate redirection gains are applied to user's real rotation $\delta_r(t)$ in S_r. Different from the reactive RDW technique [11], [28] and some predictive RDW [15], [16], we only add subtle continuous angle changes called curvature gains [18] to correct the user's virtual rotation $\delta_v(t)$ when the user is walking along SG(S_r). The curvature gains depend on the tangent vectors that correspond to the current real position.

In summary, the entire procedure of the proposed method is summarized in Algorithm 1.

IV. EVALUATION

A user study is conducted to evaluate the performance and usability of our method.

Algorithm 1 A Novel Redirected Walking Method for Immersive VR Navigation

Require: Virtual space S_v, Real space S_r
Ensure: Virtual position $x_v(t)$ and rotation $\delta_v\ (t)$

// **Stage 1: Pre-computing**
// **Automatic Skeleton Graph Generation**
Calculate Voronoi diagram VD(S_v);
Calculate AVD(S_v) using the clustering method;
Calculate 2-VD(S_v) using the hierarchical Voronoi diagram strategy;
SG(S_v) $\Leftarrow$ 2-VD(S_v);
// **Static Graph Mapping**
Build the search tree of SG(S_v) using DFS;
Map all way-points and paths of SG(S_v) to S_r using the local mapping method;
Apply the way-point extension method to all mapped way-points in S_r;
Calculate SG(S_r) by adopting the global optimization with boundary constraint;
// **Stage 2: Real-time Updating**
Obtain the current real position $x_r(t)$ and rotation $\delta_r\ (t)$ by tracking devices;
if $x_r(t)$ locates on SG(S_r) **then**
    Calculate the virtual position $x_v(t)$ by inverse mapping $x_r(t)$;
else {$x_r(t)$ deviates from SG(S_r)}
    Calculate the virtual position $x_v(t)$ by the path extension method;
end if
Calculate the virtual rotation $\delta_v\ (t)$ by applying subtle curvature gain to real rotation $\delta_r\ (t)$;
return $x_v(t)$, $\delta_v\ (t)$;

A. PARTICIPANTS

A total of 14 participants (9 male and 5 female) are recruited to participate in the user study. Their ages range from 24 to 29 ($M = 26.14$, $SD = 1.61$). Among these participants, 3 participants have no prior experience with HMDs while the rest have some basic knowledge of HMDs.

B. EXPERIMENTAL DESIGN

To evaluate the performance of our method, a comparative study is designed using a within-subjects design including three conditions: our method, steer-to-center (S2C) and steer-to-orbit (S2O). These three methods are investigated in three virtual spaces with different sizes: *small*, *medium*, and *large*. The dimensions of all three virtual spaces that are illustrated in Fig. 11 are 6.5 $m \times 5.8\ m$, 7.7 $m \times 8.3\ m$, and 9.4 $m \times 9.5\ m$, respectively. The dependent variables include *navigation time*, *collision numbers* and *redirection rate*. During the experiment, we counterbalance the presentation order of the different methods.

Inspired by [38], we adopt a post-interview-based method to investigate the usability of our method after the experiment.

FIGURE 11. Virtual spaces in different sizes: (a) Small (6.5 $m \times 5.8\ m$); (b) Medium (7.7 $m \times 8.3\ m$); (c) Large (9.4 $m \times 9.5\ m$).

We evaluate the simulator sickness caused by each method in different virtual space sizes.

C. IMPLEMENTATION

As shown in Fig. 1(d), the real space is fixed in a 2.5 $m \times 2.5\ m$ rectangular area without any internal obstacles and a Kinect V2 is used to track the participant's real position. To perceive the virtual space, the participant wears a cardboard HMD that is composed of a cardboard and a smart phone, and the phone's gyroscope sensor provides the participant's rotations for our system. Within the WLAN, the Kinect and the smart phone are connected to a PC that is used to process the position and rotation data. We adopt the Unity3D engine to render the virtual spaces. Due to the limitation of real space size (2.5 $m \times 2.5\ m$), we adopt an experience value (0.25m) to set the radius of all of the way-points in both virtual and real skeleton graphs for the way-point extension. For the comparison methods, S2C and S2O are realized in the experiment as proposed by [29], [30].

D. PROCEDURE AND TASK

To accustom the participants with each method, the participants are asked to navigate in a simple training virtual space using the above three methods before starting the formal experiment. When participants finish their training, the formal experiment begins.

In the formal experiment, all of the virtual spaces are assigned to similar search tasks. For each virtual space, the search task is to find all of the target objects that are preset in the virtual space. The target objects of each search task are located on fixed positions and appropriately scaled to be convenient for searching. To avoid repeating the search for a certain target, the target object is treated as "searched" and becomes invisible when participants move close to it. The search task is performed until all of the target objects are searched. For each search task, the number of targets is 8. During the experiment, S2C and S2O adopt reorientation resets if the participants move close to the boundaries of the real space. When starting the reorientation reset, the system displays a warning tip that is represented as a UI element in the upper right of the participants' visions. The warning tip remains visible until the reorientation reset is completed. In our method, a collision may occur if the participant deviates too far away from the paths. To guide the participants back to the paths, the system displays the same warning tip for

the participant who leaves the path and possibly collides with the boundaries of the real space. The warning tip becomes invisible when the participant returns back to current path.

Each participant is required to complete each search task using all three methods in a random order. During the experiment, we ask the participants to complete the tasks as fast as possible. After completing a search task by each method, the participant is required to immediately answer the simulator sickness questionnaire (SSQ) [39].

E. MEASURES

The measures used to investigate the performance include: *navigation time*, *collision numbers* and *redirection rate*. The *navigation time* is a major factor for investigating the efficiency of these methods. It is measured by the time for a participant to finish one search task with the assigned method. The *collision numbers* is used to evaluate the effectiveness of the method. This factor is measured by the numbers of the reorientation reset in S2C and S2O. For our method, we use the numbers of warning tips that occur to measure this factor. The *redirection rate* is the factor to estimate the perceptual distortion. The user feels more perceptual distortion as this factor increases. This factor is defined as the mean of the absolute redirection rotation per second for a single navigation through virtual space. For the usability of our method, we evaluate the simulator sickness. The simulator sickness is measured by the mean SSQ scores of each method in different virtual spaces at the end of experiment.

F. RESULTS

1) NAVIGATION TIME

Fig. 12 shows the mean navigation time of each method in different virtual spaces. We conduct Paired-samples T tests to investigate the differences between our method and other two methods. In the *small* virtual space, the mean navigation time of our method is significantly lower than S2C ($t = 12.39, p < 0.01$) and S2O ($t = 12.11, p < 0.01$). Compared with S2C (194.25±16.56 *s*) and S2O (189.71±14.25 *s*), our method (107.85±13.25 *s*) shows better efficiency. For the *medium* virtual space, our method (159.79±14.56 *s*) provides smaller navigation time in comparison with S2C (303.97±13.95 *s*) and S2O (291.66±15.57 *s*). The T test reveals that our method significantly lowers the navigation time comparing with S2C ($t = 20.78, p < 0.01$) and S2O ($t = 21.79, p < 0.01$). For the *large* virtual space, S2C (488.03±26.67 *s*) and S2O (472.01±31.76 *s*) provide larger navigation time than our method (289.14±23.26 *s*). Therefore, our method can significantly reduce mean navigation time (for S2C, $t = 21.53$, $p < 0.01$; for S2O, $t = 16.09, p < 0.01$).

2) COLLISION NUMBERS

Fig. 13 shows the mean collision numbers of each method in different virtual spaces. In the *small* virtual space, the collision numbers of our method (1.93±0.73) is smaller than S2C (11.07±1.21) and S2O (12.21±1.48). The results of T test

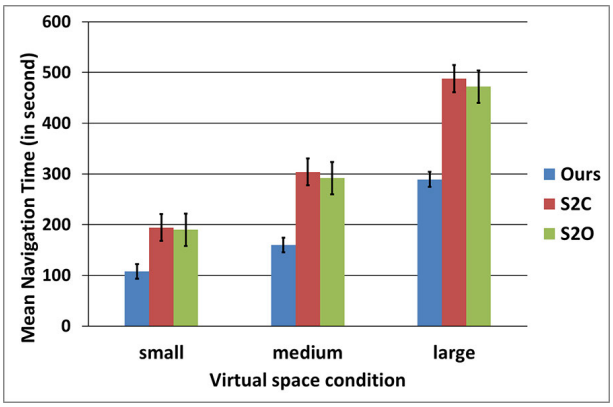


FIGURE 12. The mean navigation time of each method in different virtual spaces.

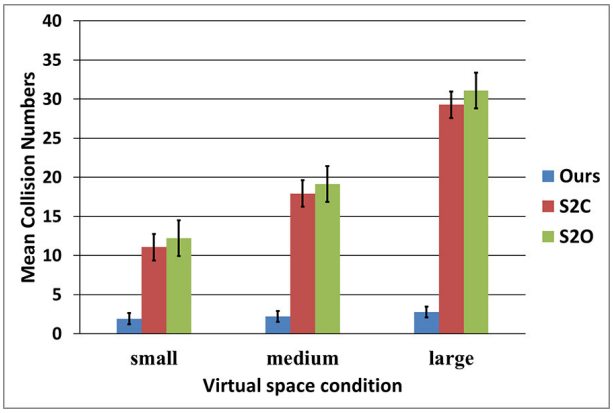


FIGURE 13. The mean collision numbers of each method in different virtual spaces.

show that our method significantly reduces mean collision numbers comparing with S2C ($t = 25.58, p < 0.01$) and S2O ($t = 25.02, p < 0.01$). For the *medium* virtual space, our method (2.21±0.69) provides smaller collision numbers than S2C (17.93±1.69) and S2O (19.14±2.28). The results of T tests reveal that our method performs better in term of mean collision numbers than S2C ($t = 28.08, p < 0.01$), and S2O ($t = 21.89, p < 0.01$). For the *large* virtual space, S2C (29.28±2.43) and S2O (31.07±2.99) lead to higher collision numbers than our method (2.8±0.70). Therefore, our method produces fewer collisions than S2C ($t = 34.32, p < 0.01$) and S2O ($t = 32.17, p < 0.01$).

3) REDIRECTION RATE

Fig. 14 shows the mean redirection rate of each method in different virtual spaces. For the *small* virtual space, our method (5.44±0.65 deg/s) reduce the mean redirection rate compared to S2C (7.59±0.69 deg/s) and S2O (7.26±0.59 deg/s). The results of T tests reveal significant differences between our method and S2C ($t = 8.55, p < 0.01$), S2O ($t = 8.94$, $p < 0.01$). In the *medium* virtual space, the mean redirection rate: our method (6.91 deg/s ± 0.62), S2C (9.79 deg/s ± 1.01) and S2O (8.94 deg/s ± 0.99). The results of T test

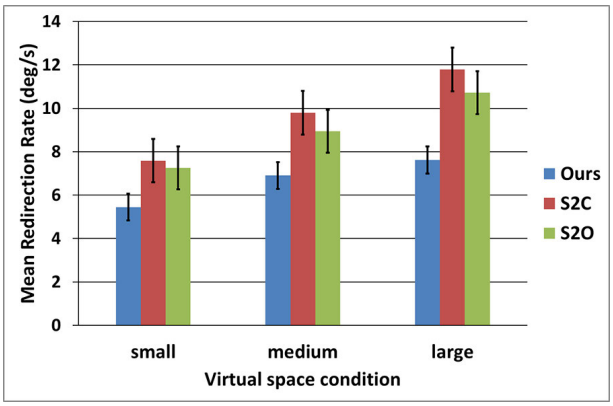


FIGURE 14. The mean redirection rate of each method in different virtual spaces.

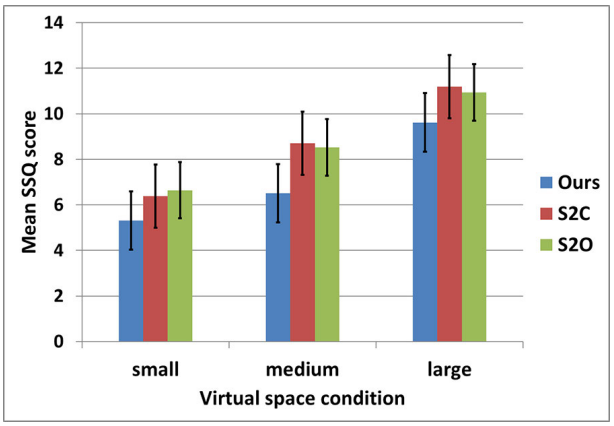


FIGURE 15. Mean SSQ score of our method in different virtual spaces.

show that our method significant reduces redirection rate compared with S2C ($t = 8.20$, $p < 0.001$) and S2O ($t = 6.71$, $p < 0.01$). The mean redirection rate of our method (7.62 ± 0.61 deg/s) is smaller than S2C (11.79 ± 1.42 deg/s) and S2O (10.72 ± 0.96 deg/s) in the *large* virtual space. We find that our method can significantly reduce the redirection rate compared to S2C ($t = 8.94$, $p < 0.01$) and S2O ($t = 7.93$, $p < 0.01$).

4) USABILITY

For simulator sickness, Fig. 15 shows the mean SSQ score of each method in different virtual spaces. In *small* virtual space, the mean SSQ score of our method (5.31 ± 3.83) is slightly smaller than that of S2C (6.38 ± 3.69) and S2O (6.66 ± 3.41). The results of a Paired-samples T test show that participants experience significant increases in simulator sickness from S2C ($t = 3.23$, $p < 0.01$) and S2O ($t = 2.32$, $p = 0.02 < 0.05$) compared to our method. With regard to *medium* virtual space, our method (6.51 ± 4.60) provides a smaller mean SSQ score than S2C (8.71 ± 3.40) and S2O (8.53 ± 3.59). The T test results reveal that our method significantly reduces the simulator sickness compared with S2C ($t = 2.29$, $p = 0.02 < 0.05$) and S2O ($t = 1.89$, $p = 0.04 < 0.05$). For *large* virtual space, both S2C (11.18 ± 3.98) and S2O (10.93 ± 3.37) lead to a larger mean SSQ score than our method (9.62 ± 3.89). By performing the Paired-samples T test, the results show that our method significantly reduces the simulator sickness compared to S2C ($t = 2.72$, $p = 0.01 < 0.05$) and S2O ($t = 1.92$, $p = 0.04 < 0.05$).

G. DISCUSSION

To estimate the performance of our method, two RDW methods namely S2C and S2O are compared to our method in the experiment. Summarizing the results, our method produces significant improvements. For a given virtual space that satisfies a simple polygonal shape, participants take less time to finish a search task using our method. By contrast, the participants spend more time on the resets when they are using S2C and S2O. Furthermore, the results prove that compared with S2C and S2O, our method can effectively prevent participants from colliding with the real space boundaries. With increasing size of the virtual space, the number of collision for S2C and S2O increases significantly. By contrast, the results of our methods show a slight increase in the number of collisions with increasing virtual space size, indicating that our method is more effective. Moreover, we find that our method provides the minimal redirection rate among these three methods. The main reason for this is that almost any head rotation can cause redirections in both S2C and S2O. By contrast, our method only applies the redirections on the paths. Therefore, it can be summarized that users perceive less perceptual distortion using in our method than when S2C and S2O are used. Moreover, we measure the SSQ score in different virtual spaces to understand the simulator sickness that can occur in the three methods. The result shows that compared to using S2C and S2O participants experience less simulator sickness adopting our method in each virtual space condition. In addition to the simulator sickness, the human perception of redirected walking methods on spatial cognition and performance is an important work. Investigating the effect of our method on spatial cognition and performance is left for the future work.

A significant advantage of our method is that the skeleton graphs of both the virtual and real spaces are pre-computed, effectively keeping users walking inside the real space. This advantage is proven in the evaluation. Since our method has no resets that cause breaks in the presence, another advantage of our method is that it provides users with a continuous walking experience. By contrast, S2C and S2O frequently produce resets to keep the participants away from the boundaries of the real space.

V. CONCLUSION

In this article, we present a novel redirected walking method that supports continuous real-walking experience in a small real space while perceiving various large virtual spaces with an HMD. By using the skeleton graph generation method, our method can provide suitable skeleton graphs for various virtual spaces that have simple polygon-shaped floor plans. Based on the static graph mapping method, our system

TABLE 1. Performance measurement for the global optimization of the static graph mapping (pre-computing stage).

measure / virtual spaces	size	way-points	iterations	time cost
small	$6.5m \times 5.8m$	11	960	$2.05s$
medium	$7.7m \times 8.3m$	21	5769	$21.31s$
large	$9.4m \times 9.5m$	33	44654	$135.34s$

can establish the mapping relation between virtual and real skeleton graphs. During VR navigation, the proposed system adopts the mapping relation to effectively guide users to walk along the mapped skeleton graph while perceiving to navigate along the virtual skeleton graph without any interruptions. The proposed method is evaluated within a user study and compared with the existing RDW methods. The results indicate that our method has the potential to effectively avoid collisions with the real space boundaries and significantly reduces perceptual distortion during VR navigation.

Limitations and future work. Since skeleton graphs can provide proper paths for the virtual spaces that are under consideration in our method, our work provides an efficient Voronoi-based skeleton graph generation method. To expand the reachable areas of the virtual space, we extend all way-points and paths of both virtual and real skeleton graphs in an appropriate manner. Thus, our method allows users to appropriately leave the skeleton graph in the navigation, providing users with a certain extent of free walking. However, a limitation of our method is that the pre-computed paths cannot provide the user with completely free walking in the virtual space. In the future, we expect to improve the static mapping method to increase the extent of free walking in our method.

Our method adopts global optimization to iteratively improve the mapping results. Therefore, another limitation of our method is that the global optimization increases the computational cost of the pre-computing stage. In our method, the global optimization requires $O(n^2)$ time for the iteration procedure and is primarily affected by the number of the way-points in the virtual skeleton graph. Table 1 provides performance measurements of the global optimization by testing three virtual spaces that are illustrated in Section IV-B. The results indicate that the number of iteration and processing time significantly increase with increasing number of way-points. In future study, we expect to achieve further improvements for the global optimization to reduce the processing time of the pre-computing stage.

Furthermore, the proposed method currently supports a single user walking in the virtual space. Previous work [40] shows the potential to allow multiple users perceiving different virtual spaces while moving in a common real space. Considering a fixed real space, our method can provide different real paths for users who are exploring different virtual scenes. By adopting reasonable path arrangement strategy [40], it is possible to allow multi-users to walk along different real paths which are generated by our method in a real space. Therefore, our next effort is to realize a multi-user VR experience in a single real space. In addition, we would like enhance our method to further support the virtual spaces with complex polygonal shapes (polygons with inner holes).

REFERENCES

[1] M. Usoh, K. Arthur, M. C. Whitton, B. Rui, A. Steed, M. Slater, and F. P. Brooks, "Walking > walking-in-place > flying, in virtual environments," in *Proc. 26th Annu. Conf. Comput. Graph. Interact. Techn.*, 1999, pp. 359–364.

[2] B. G. Witmer and M. J. Singer, "Measuring presence in virtual environments: A presence questionnaire," *Presence, Teleoperators Virtual Environ.*, vol. 7, no. 3, pp. 225–240, Jun. 1998.

[3] W. Gai, C. Yang, Y. Bian, C. Shen, X. Meng, L. Wang, J. Liu, M. Dong, C. Niu, and C. Lin, "Supporting easy physical-to-virtual creation of mobile VR maze games: A new genre," in *Proc. CHI Conf. Hum. Factors Comput. Syst.*, May 2017, pp. 5016–5028.

[4] A. L. Simeone, E. Velloso, and H. Gellersen, "Substitutional reality: Using the physical environment to design virtual reality experiences," in *Proc. 33rd Annu. ACM Conf. Hum. Factors Comput. Syst. CHI*, 2015, pp. 3307–3316.

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

[6] E. A. Suma, Z. Lipps, S. Finkelstein, D. M. Krum, and M. Bolas, "Impossible spaces: Maximizing natural walking in virtual environments with self-overlapping architecture," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 4, pp. 555–564, Mar. 2012.

[7] K. Vasylevska, H. Kaufmann, M. Bolas, and E. A. Suma, "Flexible spaces: Dynamic layout generation for infinite walking in virtual environments," in *Proc. IEEE Symp. 3D User Interface (3DUI)*, Mar. 2013, pp. 39–42.

[8] Q. Sun, L. Y. Wei, and A. Kaufman, "Mapping virtual and physical reality," *Acm Trans. Graph.*, vol. 35, no. 4, pp. 1–12, 2016.

[9] Z.-C. Dong, X.-M. Fu, C. Zhang, K. Wu, and L. Liu, "Smooth assembled mappings for large-scale real walking," *ACM Trans. Graph.*, vol. 36, no. 6, pp. 1–13, Nov. 2017.

[10] S. Razzaque, Z. Kohn, and M. C. Whitton, "Redirected walking," in *Proc. Eurographics*, 2001, pp. 105–106.

[11] S. Razzaque, "Redirected walking," Ph.D. dissertation, Dept. Comput. Sci., Univ. North Carolina, Chapel Hill, NC, USA, 2005.

[12] B. Williams, G. Narasimham, B. Rump, T. P. McNamara, T. H. Carr, J. Rieser, and B. Bodenheimer, "Exploring large virtual environments with an HMD when physical space is limited," in *Proc. 4th Symp. Appl. Perception Graph. Visualizat. APGV*, 2007, pp. 41–48.

[13] M. Azmandian, T. Grechkin, M. Bolas, and E. Suma, "Automated path prediction for redirected walking using navigation meshes," in *Proc. IEEE Symp. 3D User Interface (3DUI)*, Mar. 2016, pp. 63–66.

[14] M. Zank and A. Kunz, "Optimized graph extraction and locomotion prediction for redirected walking," in *Proc. IEEE Symp. 3D User Interface (3DUI)*, Mar. 2017, pp. 120–129.

[15] 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.

[16] T. Nescher, Y.-Y. Huang, and A. Kunz, "Planning redirection techniques for optimal free walking experience using model predictive control," in *Proc. IEEE Symp. 3D User Interface (3DUI)*, Mar. 2014, pp. 111–118.

[17] T. C. Peck, H. Fuchs, and M. C. Whitton, "The design and evaluation of a large-scale real-walking locomotion interface," *IEEE Trans. Vis. Comput. Graphics*, vol. 18, no. 7, pp. 1053–1067, Jul. 2012.

[18] C. T. Neth, J. L. Souman, D. Engel, U. Kloos, H. H. Bulthoff, and B. J. Mohler, "Velocity-dependent dynamic curvature gain for redirected walking," in *Proc. IEEE Virtual Reality Conf.*, Mar. 2011, pp. 151–158.

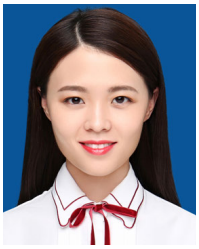
[19] R. A. Ruddle and S. Lessels, "The benefits of using a walking interface to navigate virtual environments," *ACM Trans. Comput.-Hum. Interact.*, vol. 16, no. 1, pp. 1–18, Apr. 2009.

[20] J. L. Souman, P. R. Giordano, M. Schwaiger, I. Frissen, H. Ulbrich, A. D. Luca, and M. O. Ernst, "Cyberwalk: Enabling unconstrained omnidirectional walking through virtual environments," *Acm Trans. Appl. Perception*, vol. 8, no. 4, pp. 1–22, 2008.

[21] H. Iwata, H. Yano, and H. Tomioka, "Powered shoes," in *Proc. ACM SIGGRAPH Emerg. Technol. SIGGRAPH*, 2006, p. 28.

[22] M. Schwaiger, T. Thíźmmel, and H. Ulbrich, "Cyberwalk: Implementation of a ball bearing platform for humans," in *Int. Conf. Hum.-Comput. Interact., Interact. Platforms Techn.*, 2007, pp. 926–935.
[23] F. Steinicke, G. Bruder, L. Kohli, J. Jerald, and K. Hinrichs, "Taxonomy and implementation of redirection techniques for ubiquitous passive haptic feedback," in *Proc. Int. Conf. Cyberworlds*, Sep. 2008, pp. 217–223.
[24] E. A. Suma, G. Bruder, F. Steinicke, D. M. Krum, and M. Bolas, "A taxonomy for deploying redirection techniques in immersive virtual environments," in *Proc. IEEE Virtual Reality Workshops*, Costa Mesa, CA, USA, 2012, pp. 43–46.
[25] F. Steinicke, G. Bruder, J. Jerald, H. Frenz, and M. Lappe, "Analyses of human sensitivity to redirected walking," in *Proc. ACM Symp. Virtual Reality Softw. Technol. VRST*, 2008, pp. 149–156.
[26] 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. 17–27, Jan. 2010.
[27] 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 Comput. Graph. Appl.*, vol. 38, no. 2, pp. 44–56, Mar. 2018.
[28] T. Field and P. Vamplew, "Generalised algorithms for redirected walking in virtual environments," in *Proc. Int. Conf. Artif. Intell. Sci. Technol.* Hobart, TAS, Australia: Univ. Tasmania, 2004, pp. 21–25.
[29] E. Hodgson and E. 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.
[30] E. Hodgson, E. Bachmann, and T. Thrash, "Performance of redirected walking algorithms in a constrained virtual world," *IEEE Trans. Vis. Comput. Graphics*, vol. 20, no. 4, pp. 579–584, 2014.
[31] M. Azmandian, T. Grechkin, M. Bolas, and E. Suma, "The redirected walking toolkit: A unified development platform for exploring large virtual environments," in *Proc. IEEE 2nd Workshop Everyday Virtual Reality (WEVR)*, Mar. 2016, pp. 9–14.
[32] M. Zank and A. Kunz, "Where are you going? Using human locomotion models for target estimation," *Vis. Comput.*, vol. 32, no. 10, pp. 1–13, 2016.
[33] A. Sud, E. Andersen, S. Curtis, M. C. Lin, and D. Manocha, "Real-time path planning for virtual agents in dynamic environments," *IEEE Trans. Vis. Comput. Graphics*, vol. 14, pp. 91–98, 2007.
[34] R. Alex and L. Alessandro, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, p. 1492, 2014.
[35] X. Wang, C. Yang, J. Wang, and X. Meng, "Hierarchical Voronoi diagram-based path planning among polygonal obstacles for 3D virtual worlds," in *Proc. IEEE Int. Symp. VR Innov.*, Mar. 2011, pp. 175–181.
[36] X. Pan, L. Wu, and Z. Hu, "A privacy protection algorithm based on network Voronoi graph over road networks," *J. Comput. Res. Develop.*, vol. 52, no. 12, pp. 2750–2763, 2015.
[37] D. P. Bertsekas, *Dynamic Programming and Optimal Control*. Nashua, NH, USA: Athena Scientific, 2000.
[38] D. A. Bowman, J. L. Gabbard, and D. Hix, *A Survey of Usability Evaluation in Virtual Environments: Classification and Comparison of Methods*. Cambridge, MA, USA: MIT Press, 2002.
[39] 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, Jul. 1993.
[40] S. Marwecki, M. Brehm, L. Wagner, L.-P. Cheng, F. Mueller, and P. Baudisch, "VirtualSpace–overloading physical space with multiple virtual reality users," in *Proc. CHI Conf. Hum. Factors Comput. Syst. CHI*, 2018, p. 241.

HUIYU LI received the B.S. degree in software from Shandong University, Jinan, China, in 2014, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology. His research interests include virtual reality and human–computer interaction.



LINWEI FAN received the Ph.D. degree from Shandong University, Jinan, China, in 2019. She is currently an Associate Professor with the School of Computer Science and Technology, Shandong University of Finance and Economics, and a member of the Shandong Provincial Key Laboratory of Digital Media Technology. Her research interests include computer graphics and image processing.