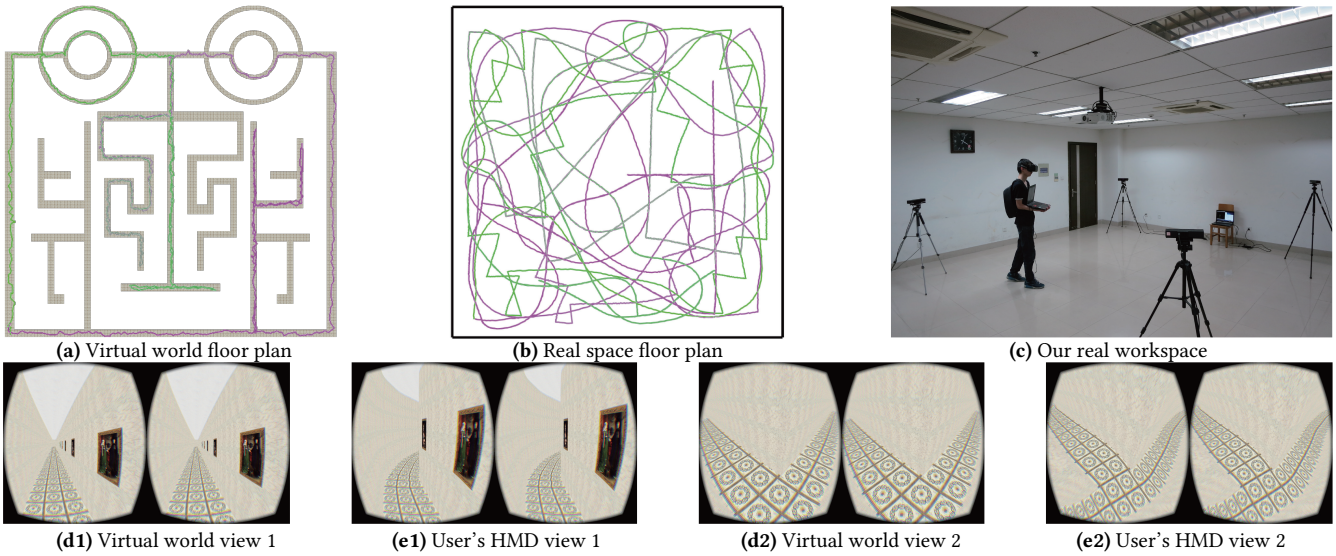
Smooth Assembled Mappings for Large-Scale Real Walking

ZHI-CHAO DONG, University of Science and Technology of China
XIAO-MING FU*, University of Science and Technology of China
CHI ZHANG, University of Science and Technology of China
KANG WU, University of Science and Technology of China
LIGANG LIU*, University of Science and Technology of China



(a) Virtual world floor plan **(b)** Real space floor plan **(c)** Our real workspace

(d1) Virtual world view 1 **(e1)** User's HMD view 1 **(d2)** Virtual world view 2 **(e2)** User's HMD view 2

Fig. 1. We propose a divide-and-conquer method to compute a map from a substantially large virtual scene, as shown in (a) ($24m \times 24m$), to a small real workspace (b) ($3.6m \times 3.6m$). (b) The real walking path of one participant to pass through the virtual scene in our real walking workspace. (c) The photo of our real walking workspace, which is used in all experiments in this paper. We build a low-cost tracking system with four Microsoft Kinect V2s, where each Kinect is placed at one corner of the workspace, to capture the users' position. The virtual world views and user's HMD views at two viewpoints are shown in the bottom row. Note that the virtual scene is about 44 times larger than the real workspace in area, and contains some loops and curve-shaped pathways.

Virtual reality applications prefer real walking to provide highly immersive presence than other locomotive methods. Mapping-based techniques are very effective for supporting real walking in small physical workspaces while exploring large virtual scenes. However, the existing methods for computing real walking maps suffer from poor quality due to distortion. In this paper, we present a novel divide-and-conquer method, called *Smooth Assembly Mapping* (SAM), to compute real walking maps with low isometric distortion for large-scale virtual scenes. First, the input virtual scene is decomposed into a set of smaller local patches. Then, a group of local patches is mapped together into a real workspace by minimizing a low isometric distortion energy with smoothness constraints between the adjacent patches. All local patches are mapped and assembled one by one to obtain a complete map. Finally, a global optimization is adopted to further reduce the distortion throughout the entire map. Our method easily handles teleportation technique by computing maps of individual regions and assembling them with teleporter conformity constraints. A large number of experiments, including formative user studies and comparisons, have shown that our method succeeds in generating high-quality real walking maps from large-scale virtual scenes to small real workspaces and is demonstrably superior to state-of-the-art methods.

CCS Concepts: • **Computing methodologies** → **Virtual Reality**;

Additional Key Words and Phrases: Virtual reality, real walking, virtual scene mapping, isometric distortion, teleportation

ACM Reference format:
Zhi-Chao Dong, Xiao-Ming Fu*, Chi Zhang, Kang Wu, and Ligang Liu*. 2017. Smooth Assembled Mappings for Large-Scale Real Walking. *ACM Trans. Graph.* 36, 6, Article 211 (November 2017), 13 pages.
https://doi.org/10.1145/3130800.3130893

*Corresponding authors: fuxm@ustc.edu.cn (Xiao-Ming Fu) and lgliu@ustc.edu.cn (Ligang Liu) .

0730-0301/2017/11-ART211 $15.00
https://doi.org/10.1145/3130800.3130893

1 INTRODUCTION

The rapid development of *virtual reality* (VR) in both hardware and software enables an active and dynamic ability for users to freely navigate through *virtual environments* (VEs). Real walking with VR devices, such as 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 [Usoh et al. 1999].

To allow users to walk through expansive VEs larger than the physical *real workspace* (RW), a variety of techniques, such as redirected walking, physical props, and space manipulation, etc., have been proposed to preserve the feeling of moving naturally through VEs while simultaneously keeping the user physically constrained within the RW [Cheng et al. 2015; Razzaque et al. 2001; Suma et al. 2012]. However, physical props and space manipulation are not general enough to provide navigation guidance within VEs, while redirected walking techniques often interrupt users and require them to reset their location with rotation gains. Although distractors are often adopted to mitigate these disruptions, they do not completely remedy the situation.

Recently, Sun et al. [2016] proposed a mapping-based method to compute a smooth mapping from the VE to the RW. This avoids breaking up the flow of the user experience and allows for smooth real walking within VEs. However, when applying the method of [Sun et al. 2016] on a large VE it may generate substantially distorted maps, resulting in unacceptable visual artifacts and infeasible walking navigation. This is because one global mapping from a large VE to a small RW inevitably yields serious distortions, thereby rendering it difficult to preserve distances and angles within a VE. Therefore, computing low distance distortion mappings for large-scale VEs remains a challenge.

Our method. To this end, we propose a novel approach, called *Smooth Assembled Mapping* (SAM), in order to map large-scale VEs into small RWs with low isometric distortion. Our method is based on the divide-and-conquer strategy. Instead of creating a global map between the whole VE and the RW, we create piecewise local mappings so that each one maps a subpart of the VE into the RW. This is achieved by decomposing the input VE into smaller local patches. Each patch is mapped into the RW by minimizing a low isometric distortion energy. A group of adjacent patches is then mapped and assembled together with smoothness constraints. After all patches are mapped, a global optimization is applied to further reduce the isometric distortion and conform to other constraints, including loop closure smoothness and interior obstacle avoidance. Our SAM method easily handles the teleportation technique by computing maps of individual regions and assembling them with the teleporter conformity constraints. In comparison with other state-of-the-art methods, our method generates maps from substantially large VEs into smaller RWs with low isometric distortion, thereby achieving better immersive experiences for VR applications. The feasibility and practicability have been validated by a number of experiments and formative user studies via a low-cost tracking system, which is implemented by four Kinect V2s.

2 RELATED WORK

Real walking. Typical immersive virtual reality systems include walking-in-place [Iwata et al. 2006; Schwaiger et al. 2007; Souman et al. 2008] and real walking [Usoh et al. 1999]. Real walking with head mounted displays (HMDs) provides more natural navigation and a more immersive experience compared to walking-in-place and other indirect methods of VR navigation [Usoh et al. 1999]. To support real walking in smaller RWs to experience the larger VEs, many techniques have been proposed in the past fifteen years, such as physical props [Cheng et al. 2015], space manipulation [Suma et al. 2011, 2012; Vasylevska and Kaufmann 2017; Vasylevska et al. 2013], redirected walking [Razzaque et al. 2001, 2002], and the mapping-based method [Sun et al. 2016]. Below we review the latter two categories, i.e., redirected walking and mapping based real walking.

Redirected walking. Redirected walking techniques build upon the principle that the brain considers visual cues to dominate the natural movement of the human body. Therefore, it is possible to introduce subtle differences between motions in the real world and what is perceived in the virtual world. Translation, rotation, and curvature gains are used and their thresholds are studied in [Steinicke et al. 2008]. Based on these gains, many techniques have been proposed, such as reactive methods [Field and Vamplew 2004; Hodgson and Bachmann 2013; Hodgson et al. 2011; Peck et al. 2012; Razzaque 2005; Williams et al. 2007], and predictive algorithms [Azmandian et al. 2014; Nescher et al. 2014; Zmuda et al. 2013]. However, these techniques cannot guarantee that a user remains safely within the tracked RW. Thus, a *reset* process, which uses rotation gain, is applied to prevent the user from leaving the RW [Peck et al. 2012; Williams et al. 2007]; however, this seriously affects the user experience. The most commonly used algorithm for reactive methods is Steer-To-Center (S2C), which focuses on steering the user towards the center of the real space. We compare with the S2C in Section 5.4.

Mapping based real walking. In a recent work, Sun *et al.* [2016] proposes a method of mapping between virtual and real scenes that lowers distortion. However, it may inevitably produce large distortion for mapping large scenes into relatively small real space. To tackle this problem, we propose a new method that computes a series of local mappings from parts of the virtual scene to the real space. Our method uses a low isometric distortion scheme and then assembles the parts into a smooth mapping.

Smooth planar mapping. There are a vast number of methods to compute smooth locally bijective planar maps with low distortion [Chen and Weber 2015; Fu et al. 2015; Levi and Weber 2016; Poranne and Lipman 2014; Sun et al. 2016]. A general method for generating these maps is proposed in [Poranne and Lipman 2014] by bounding the distortion on discrete sample points. [Fu et al. 2015; Sun et al. 2016] follows [Poranne and Lipman 2014] with different objective functions and optimization solvers. Different from existing works, we decompose the input domain, i.e., the virtual world floor plan, into small local patches and then map a small subset of patches into the target domain one by one with smoothness constraints. As inappropriate distortion bound may lead to infeasible results [Chen and Weber 2015; Levi and Weber 2016; Poranne and Lipman 2014; Sun et al. 2016], we adopt a low isometric distortion energy to compute the local maps, similar to [Fu et al. 2015].

3 PROBLEM AND FORMULATION

Inputs. We represent both VE and RW as planar floor plans, denoted as S^V and S^R, respectively. Generally, $S^V > S^R$ or even $S^V \gg S^R$. In practice, we consider the reachable region, which

is a network of navigation pathways in S^V. Without loss of generality, we also denote S^V as all covered reachable regions, which serves as the input VE domain.

3.1 Problem

Problem. The problem is to compute a smooth map $\mathbf{f} : S^V \to S^R$ that maps each point $(u, v) \in S^V$ to a point $(x, y) \in S^R$, according to some criteria and constraints.

Instead of computing a global map from S^V to S^R directly, which may result in substantially large distortion, we propose a *divide-and-conquer* strategy. First, S^V is decomposed into K small patches, i.e., $S^V = \cup_{k=1}^{K} P_k^V$. Each patch P_k^V can be individually mapped into S^R with little distortion by a local map $\mathbf{f}_k : P_k^V \to S^R$. Our goal is to smoothly assemble all local maps $\{\mathbf{f}_k\}_{k=1}^{K}$ into a global map $\mathbf{f} : S^V \to S^R$.

Challenges. Our SAM method is to stitch all local maps $\mathbf{f}_k$ together to obtain a smooth map $\mathbf{f}$. However, there are two challenges in achieving it. First, it is non-trivial to create small distance distortions when mapping large-scale VEs into small RWs. This is substantially critical in achieving positive user experiences when navigating VEs. Second, two adjacent local maps should be assembled in a smooth manner, that is, the navigation from one patch to another should be smooth enough that the user will not perceive the change.

Methodology. Inspired by the spline theory, we represent local maps as tensor Bézier maps (assuming that P_k^V are rectangular domains), which facilitates analytic computation of the smoothness between adjacent mappings, i.e., two adjacent maps can be smoothly assembled with linear constraints on the control points. Other constraints on the maps can be easily conducted as well.

3.2 Formulation

Mapping representation. We denote $P_k^V = [a_k, b_k] \times [c_k, d_k]$. The map $\mathbf{f}_k : P_k^V \to S^R$ is represented as a tensor Bézier patch as follows:

$$\mathbf{f}_k(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{c}_{i,j}^k B_i^n(u^*) B_j^m(v^*), (u, v) \in P_k^V, \tag{1}$$

where $\{B_i^n(\cdot)\}_{i=0}^n$ and $\{B_j^m(\cdot)\}_{i=0}^m$ are Bernstein polynomial basis functions of degrees, n and m respectively. $u^* = \frac{u-a_k}{b_k-a_k}$, $v^* = \frac{v-c_k}{d_k-c_k}$, and $\{\mathbf{c}_{i,j}^k\}_{i=0,j=0}^{n,m}$ are $(n+1) \times (m+1)$ control points, which are variables to define the map. To measure the quality of the map $\mathbf{f}_k$, we consider various costs of $\mathbf{f}_k$ on N_k points $\mathcal{Z}_k = \{\mathbf{z}_{k,l}\}_{l=1}^{N_k}$ evenly sampled in P_k^V [Poranne and Lipman 2014]. We set $n = m = 7$ to provide a trade-off of flexibility and scale for the variables and $N_k = 3600$ in our implementation.

Distortion cost. The distortion cost of $\mathbf{f}_k(\mathbf{z}_{k,l})$ is measured by the isometric energy as:

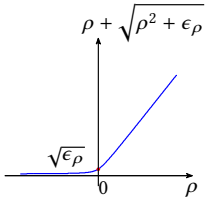
$$E_{k,l}^{dis} = \sum_{j=1}^{2} (\sigma_{k,l}^j)^2 + (\sigma_{k,l}^j)^{-2} = \|J_{k,l}\|_F^2 + \frac{\|J_{k,l}\|_F^2}{(\det J_{k,l})^2}, \tag{2}$$

where $J_{k,l}$ denotes the Jacobian of $\mathbf{f}_k$ at $\mathbf{z}_{k,l}$, $\sigma_{k,l}^1$, $\sigma_{k,l}^2$ are two singular values of $J_{k,l}$, and $\|\cdot\|_F$ denotes the Frobenius norm. If $\det J_{k,l} > 0$, i.e., $\mathbf{f}_k$ is locally bijective, $E_{k,l}^{dis}$ reaches the minimum when all singular values are equal to 1, and $E_{k,l}^{dis}$ goes to infinity when $\det J_{k,l}$ approaches zero, which prevents $\mathbf{f}_k$ from degeneracy and inversion.

Boundary cost. $\mathbf{f}_k(P_k^V)$ should lie completely inside S^R. We measure the boundary cost of $\mathbf{f}_k(\mathbf{z}_{k,l})$ as [Escobar et al. 2003]:

$$E_{k,l}^{bnd} = \sum_{j=1}^{4} \frac{2}{d_{k,l}^j + \sqrt{(d_{k,l}^j)^2 + \epsilon_d}}, \tag{3}$$

where $d_{k,l}^j$ is the signed distance from $\mathbf{f}_k(\mathbf{z}_{k,l})$ to the j^{th} boundary edge of S^R ($d_{k,l}^j > 0$ if $\mathbf{f}_k(\mathbf{z}_{k,l}) \in S^R$). ϵ_d is a small positive number that is used to penalize the negative distances. As shown in the right inset, when $d_{k,l}^j < 0$, the denominator becomes very small, thus making $E_{k,l}^{bnd}$ very large.

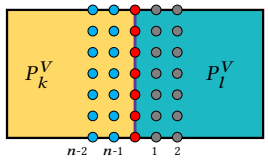


Overall cost. We have a cost function to measure the quality of the local map $\mathbf{f}_k$ from P_k^V to S^R as:

$$E(P_k^V, S^R) = \frac{1}{N_k} \sum_{l=1}^{N_k} (\omega_{dis} E_{k,l}^{dis} + \omega_{bnd} E_{k,l}^{bnd}), \tag{4}$$

where ω_{dis} and ω_{bnd} are weights for both terms ($\omega_{dis} = 5$ and $\omega_{bnd} = 1$ by default).

Smoothness constraints. onto another, we need to guarantee the smoothness between them. To verify this, we consider two adjacent patches P_k^V and P_l^V with a common boundary edge, as shown in the right inset. To guarantee the C^2 smoothness between $\mathbf{f}_k$ and $\mathbf{f}_l$, we have the following constraints on their control points:



$$\begin{aligned} \mathbf{c}_{n,j}^k &= \mathbf{c}_{0,j}^l, \\ (\mathbf{c}_{n,j}^k - \mathbf{c}_{n-1,j}^k)(b_k - a_k)^{-1} &= (\mathbf{c}_{1,j}^l - \mathbf{c}_{0,j}^l)(b_l - a_l)^{-1}, \\ (\mathbf{c}_{n,j}^k - 2\mathbf{c}_{n-1,j}^k + \mathbf{c}_{n-2,j}^k)(b_k - a_k)^{-2} &= (\mathbf{c}_{2,j}^l - 2\mathbf{c}_{1,j}^l + \mathbf{c}_{0,j}^l)(b_l - a_l)^{-2}, \end{aligned} \tag{5}$$

for $j = 0, 1, \cdots, m$. The constraints in Equation (5) are linear with respect to the control points. Thus, they can be represented as:

$$A\mathbf{C} = b, \tag{6}$$

where $\mathbf{C}$ is the vector of the control points.

4 ALGORITHM

4.1 Overview

Assuming that $\mathbf{f}_k$ on patch P_k^V has been obtained, we then assemble the map $\mathbf{f}_l$ on its adjacent patch P_l^V by minimizing (4) with constraints (6) one by one. However, it generally lacks sufficient freedom to fold over these local maps on each other and thus may result in large distortion. To allow more flexibility for folding over the local maps, we assemble a group of patches, called *super-patch*, at one time.

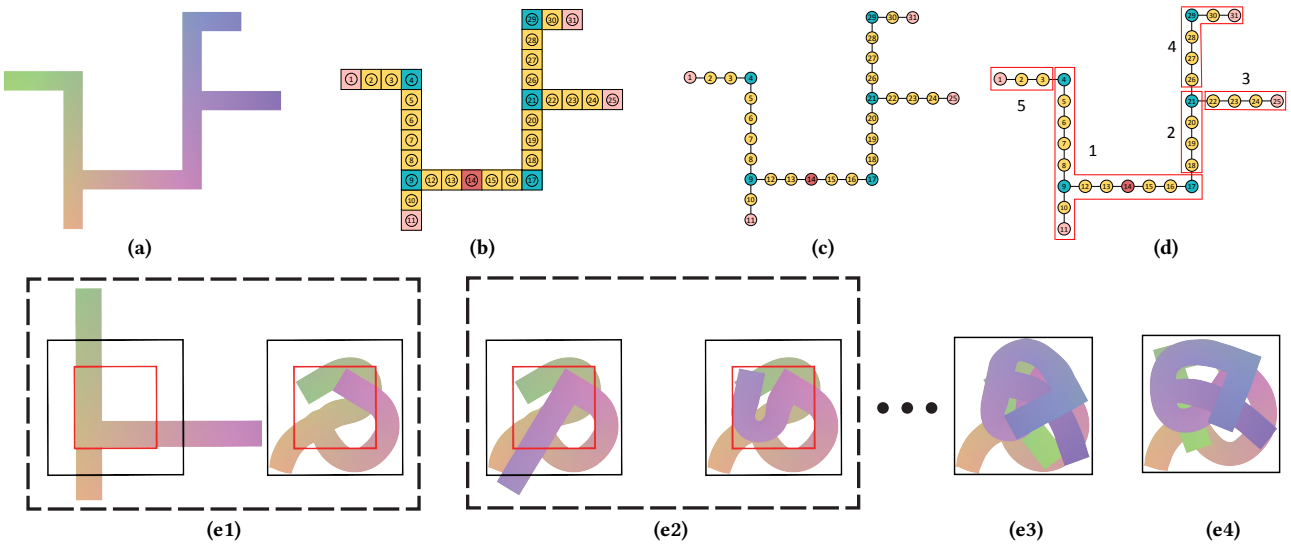


Fig. 2. Overview of our SAM method. (a) Input VE S^V (in gradient color which is a visualization for different regions); (b) S^V is decomposed into a series of primitive patches, which are classified into regular patches (yellow), leaf patches (pink), junction patches (blue), and pseudo-junction patches (red); (c) the dual graph $\mathcal{G}$ of the patches in (b); (d) $\mathcal{G}$ is partitioned into a set of trees (red polygons). Each tree represents a super-patch in S^V, which is a group of patches. (e1) Mapping the first super-patch into S^R (represented by the black rectangle). Left: the super-patch is initially placed; Right: then it is mapped into S^R with its junction-type leaf patches mapped into the safe region $\bar{S}^R$ (represented by the red rectangle). (e2) Mapping the second super-patch into S^R. Left: the super-patch is initially assembled by the smoothness constraint with (e1)-right; Right: then it is mapped into S^R with its junction-type leaf patches mapped into $\bar{S}^R$. (e3) All super-patches are mapped and assembled into S^R. (e4) A global optimization is performed on (e3) and the final mapping result is obtained. See the accompanying video for the live mapping process.

An overview of our SAM algorithm is shown in Fig. 2. Given an input S^V, we decompose it into a series of primitive patches. Then, the patches are grouped into a set of super-patches so that each super-patch consists of a few neighboring patches. Starting from one super-patch, we compute its map to S^R. Then, we compute and assemble the other maps one by one on its adjacent super-patches with smoothness constraints. This is done iteratively until all maps on super-patches have been computed and assembled. Finally, a global optimization with constraints, such as conforming loops and avoiding interior obstacles, is performed on the map.

4.2 Patches

Patches. Let us elaborate on our SAM method on S^V where all pathways are either horizontal or vertical with uniform width w (more general cases will be discussed later in Section 4.8). For each intersection of a horizontal pathway and a vertical pathway, we cover it a $w \times w$ rectangular patch (see the blue patches in Fig. 2 (b)). We call these patches *junction* patches. These junction patches partition the pathways into a series of straight road sections. For each road section, we decompose it into a sequence of $\lfloor L/w \rfloor$ equal-sized patches, called *regular* patches, for which L is the length of this road section ($\lfloor \cdot \rfloor$ denotes a floor function). Hence, S^V is entirely covered by a set of primitive rectangular patches $\{P_k^V\}_{k=1}^K$, where two adjacent patches share one common edge (Fig. 2 (b)).

Patch graph. We denote $\mathcal{G} = (V, E)$, where V is the set of nodes and E is the set of edges, as the dual graph of patches $\{P_k^V\}_{k=1}^K$. Each node of $\mathcal{G}$ corresponds to a patch of S^V and each edge of $\mathcal{G}$ that joins two nodes indicates that the two corresponding neighbor patches coincide at one common edge. We choose one node with the maximal valence as the *root* node v of $\mathcal{G}$. The nodes corresponding to the junction patches are called junction nodes. The other nodes with valence 2 and 1 are called regular and leaf nodes, respectively. The distance between two nodes is defined as the number of edges on the shortest path between them. See Fig. 2 (c).

4.3 Super-patches

Pseudo junction nodes. The junction nodes divide $\mathcal{G}$ into a set of edge sections. We denote γ as a length threshold ($\gamma = 3$ in our implementation). If an edge section contains $\Gamma > \gamma$ edges, it is then uniformly divided into $\lfloor \Gamma/\gamma \rfloor$ subsections; each subsection contains about γ edges (either γ or $\gamma + 1$). The common nodes of these new generated subsections are called *pseudo junction* nodes (see the red nodes in Fig. 2 (c)).

Super-patches. We partition the graph $\mathcal{G}$ into a set of non-overlapping trees, so that each of the trees represents one super-patches, as follows. Starting with v as the tree root node, we expand an as-large-as-possible tree $\mathcal{T}_v$. Each of of this tree's leaf nodes is either a leaf node of $\mathcal{G}$, or a junction node (either real or pseudo) of $\mathcal{G}$ with a distance to v within $[\gamma, 2\gamma]$. Note that we set a lower length bound γ for the path Γ from a junction-type leaf node to v to have enough flexibility for folding over patches of Γ into S^R. We set an upper length bound 2γ for Γ to avoid mapping too many patches at a time, which may result in large distortions. The nodes in $\mathcal{G}$ belonging to $\mathcal{T}_v$ are marked as 'visited'.

We regard each unvisited node v^* in $\mathcal{G}$, which is adjacent to the leaf nodes of $\mathcal{T}_v$, as the root node. Then, we extract a new tree $\mathcal{T}_{v^*}$ from the unvisited nodes in $\mathcal{G}$. The above process of tree extraction

is conducted iteratively until all of the nodes in $\mathcal{G}$ are visited. At the end, we have a partition of N_T trees for $\mathcal{G}$, i.e., $\mathcal{G} = \cup_{j=1}^{N_T} \mathcal{T}_j$. See the trees with different polygons in Fig. 2 (d).

4.4 Mapping assembly

Safe region in S^R. We define a subregion $\bar{S}^R$ of S^R, which is a scale of $\eta \in (0, 1)$ of S^R and centered in the middle of S^R ($\eta = 0.6$ by default). Since a junction-type leaf node of trees is connected to other trees, we expect that its corresponding patch is mapped into the safe region $\bar{S}^R$ so as to provide more space around its map for assembling the other maps.

Mapping of super-patches. For each super-patch $\mathcal{T}_j$, we denote the set of its junction-type leaf nodes as $\mathcal{T}_j^c$ and $\mathcal{T}_j^u = \mathcal{T}_j \setminus \mathcal{T}_j^c$. Then, we conduct the following energy minimization to map $\mathcal{T}_j$:

$$\begin{cases} \min\limits_{\{\mathbf{c}_{i,j}^k\}} \quad \sum\limits_{P_k^V \in \mathcal{T}_j^u} E(P_k^V, S^R) + \sum\limits_{P_k^V \in \mathcal{T}_j^c} E(P_k^V, \bar{S}^R), & \text{(7a)} \\ \text{s.t.} \quad A\mathbf{C} = b. & \text{(7b)} \end{cases}$$

Assembly of mappings. We compute mappings for all super-patches one by one in a width-first order using the mapping of a super-patch as constraints when computing the mapping of the adjacent super-patches. Suppose we have already obtained the mapping of the super-patch $\mathcal{T}_{j^*}$, and we are about to compute the mappings for all adjacent super-patches $\{\mathcal{T}_j, j \in J^*\}$ (J^* is a set of indices). Since the root patch of $\mathcal{T}_j$ is connected to $\mathcal{T}_{j^*}$, for each $j \in J^*$, we generate the map initialization by determining the first three rows or columns of the control points according to the smoothness constraints in Equation (5). Then, we uniformly extrapolate the other control points. To guarantee that there is no inversion in the extrapolation, we set a large value for the weight $\omega_{dis} = 100$ in Equation (4) for the patches in $\mathcal{T}_{j^*}^c$. Then, we apply the minimization Equation (7) to obtain the mapping for $\mathcal{T}_j$. Similarly, we compute the mappings for other super-patches iteratively until all super-patches are computed and assembled. The process of the map assembly is shown in the second row of Fig. 2.

4.5 Global optimization

After we compute and assemble the mappings for all of the super-patches, we do a global optimization on them to further reduce the distortion as:

$$\begin{cases} \min\limits_{\{\mathbf{c}_{i,j}^k\}} \quad \sum\limits_{k=1}^{K} E(P_k^V, S^R), & \text{(8a)} \\ \text{s.t.} \quad A\mathbf{C} = b. & \text{(8b)} \end{cases}$$

Constraints. There are some constraints to be considered in the global minimization. First, if there are loops of pathways in S^V, extra smoothness constraints on adjacent patches in the loops are added to Equation (8b). Second, for S^R with interior obstacles, the mappings should avoid overlapping the obstacles. Like [Sun et al. 2016], we resolve it by adding an interior obstacle penalized cost, which is represented by a 2D Gaussian-based barrier function for each interior obstacle, as in the cost function $E(P_k^V, S^R)$ in Equation (8a).

4.6 Teleportation

Our SAM method can easily handle teleportations in VR.

Scene decomposition via teleportation. Even large-scale, complex VEs can also be decomposed into multiple regions. Adjacent regions are connected via teleporters. The user can navigate from one region to another via these teleporters. This is achieved by computing a map of each region individually and then assembling all maps by adding the conformity constraints to the teleporters.

Suppose S^V is decomposed into N_r non-overlapping regions $\{Q_l^V\}_{l=1}^{N_r}$. We denote the teleporter connecting two adjacent regions $Q_{l_1}^V$ and $Q_{l_2}^V$ as T_{l_1,l_2}. We associate T_{l_1,l_2} with the patch $P_{k_1}^{V,l_1}$ (with control points $\mathbf{c}_{i,j}^{l_1,k_1}$) in $Q_{l_1}^V$ and the patch $P_{k_2}^{V,l_2}$ (with control points $\mathbf{c}_{i,j}^{l_2,k_2}$) in $Q_{l_2}^V$. Apparently, $P_{k_1}^{V,l_1}$ and $P_{k_2}^{V,l_2}$ are adjacent patches in S^V. As the teleporter, the patches $P_{k_1}^{V,l_1}$ and $P_{k_2}^{V,l_2}$ are expected to be mapped onto the same area in S^R. To this end we conduct the following constraints:

$$\mathbf{c}_{i,j}^{l_1,k_1} - \mathbf{c}_{i,j}^{l_2,k_2} = 0, \text{for all } i, j. \tag{9}$$

We define the teleportation cost at the teleporter T_{l_1,l_2} as:

$$E^{tel}(T_{l_1,l_2}) = \frac{1}{nm} \sum_{i=0}^{n} \sum_{j=0}^{m} \|\mathbf{c}_{i,j}^{l_1,k_1} - \mathbf{c}_{i,j}^{l_2,k_2}\|^2. \tag{10}$$

We denote T as the set of all teleporters. Note that two adjacent regions may have multiple teleporters; thus, we first compute N_t maps for each region individually, and then assemble them together by solving the following optimization:

$$\begin{cases} \min\limits_{\{\mathbf{c}_{i,j}^k\}} \quad \sum\limits_{l=1}^{N_r} \sum\limits_{P_k^{V,l} \in Q_l^V} E(P_k^{V,l}, S^R) + \omega_T \sum\limits_{T_{k,l} \in T} E^{tel}(T_{k,l}), & \text{(11a)} \\ \text{s.t.} \quad A\mathbf{C} = b, & \text{(11b)} \end{cases}$$

where the weight ω_T is set to 100 in our implementation.

Scene composition via teleportation. Our method can also easily implement teleportation for multiple input VEs. Specifically, we can compose the input VEs via some teleporters. First, we compute a map for each VE individually. Then, all maps are assembled by the optimization (11).

4.7 Numerical solver

We utilize the Newton's method to solve optimization (7), (8), or (11). We denote the Hessian matrix of the objective function as H_j, the gradient matrix as G_j, and the unknown control points $\mathbf{c}_j$ at the j^{th} iteration. We compute the decreasing direction $\boldsymbol{\delta}_j$ for $\mathbf{c}_j$ by solving the following linear system:

$$\begin{bmatrix} H_j & A^T \\ A & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}_j \\ \lambda \end{bmatrix} = \begin{bmatrix} -G_j \\ \mathbf{0} \end{bmatrix}. \tag{12}$$

Any indefinite Hessian (detected by the Cholesky decomposition) should be modified to be positive-definite. We add a positive number τ to the diagonal entries of H_j. We update τ iteratively by $\tau_{l+1} = \min(10^{16}, 10\tau_l)$, $\tau_0 = 10^{-5}$ until the modified hessian is positive-definite. Then, new control points are determined by backtracking line search:

$$\mathbf{c}_{j+1} = \mathbf{c}_j + \alpha_j \boldsymbol{\delta}_j, \tag{13}$$

where α_j is the step size to keep the locally bijective property and decrease the energy. The initial value for each iteration is set as half of the maximum available step size $\alpha_{\max}$ to maintain the locally bijective property, which is computed according to the method of [Smith and Schaefer 2015]. When all of the mapped sample points do not violate the exterior boundary constraints, we include these constraints to compute $\alpha_{\max}$ in order to restrict the mapped sample points inside S^R. Because $A\boldsymbol{\delta}_j = \mathbf{0}$, our new control points $\mathbf{c}_{j+1}$ satisfy $A\mathbf{c}_{j+1} = b$, meaning that the smoothness constraints are maintained during the optimization. We terminate the optimization when the relative change of the whole energy is smaller than 10^{-6}.

4.8 Discussions

Mapping formulation. Sun et al. [2016] represent the mapping as a linear combination of RBFs and impose hard inequality constraints on distortion and exterior boundaries. However, their optimization problem is complicated and the solver is hard to proceed. Instead we use the patch-based Bézier surfaces to represent the mappings, which can be easily conducted with C^2 smoothness constraints on the mapping. Furthermore, we convert the hard inequality constraints to unconstrained energies, which is easy to be optimized. For large input VEs, such as the VE in Fig. 1, [Sun et al. 2016] cannot work well while our SAM can generate low distortion result.

Non-uniform pathways in VE. Although we elaborate on how our method operates in special cases, our SAM method can easily handle general cases. Generally, the pathways in S^V have different widths. A straightforward scheme is to compute the maximum w_{max} for all widths and then cover a rectangular patch $w_{max} \times w_{max}$ at each intersection of a horizontal and vertical pathway. Instead, we compute adaptive rectangular patches at the pathway intersections according to their sizes. Then, we cover road sections using rectangular patches with sizes that are interpolated by those at the intersections.

For S^V with curve-shaped pathways, we compute the skeletons of all pathways and generate a quadrilateral mesh over S^V where each quad is centered on the skeleton. For quads that are not rectangles, we parameterize them into a unit square $[0, 1] \times [0, 1]$ and represent local maps accordingly.

Arbitrary RW. For non-rectangular S^R, we first compute its oriented bounding box S_b^R. Then, we use S_b^R as an input RW and regard the difference $S_b^R \setminus S^R$ as interior obstacles in the optimization.

Successive global optimization scheme. In our SAM method, the global optimization is performed after all super-patches are assembled together. For very large VEs, there too many variables in the final global optimization that may cause very expensive computational cost. An alternative scheme is to perform a successive global optimization (SGO), i.e, successively applying the global optimization on existing super-patches during the assembly process. In order to reduce the number of variables, the optimization can be performed on the current super-patch and its neighbor super-patches. The SGO scheme can reduce the computational cost and improve the results for large-scale VEs, but does not improve much for small VEs (see the experiments shown in Fig. 13).

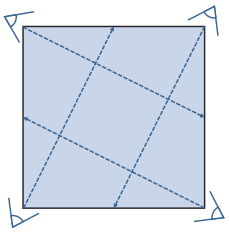
5 EXPERIMENTS AND APPLICATIONS

We have done a number of experiments to validate our SAM method and have compared it with state-of-the-art methods (also see the accompanying video).

5.1 Validation system

Setup. To validate our method, we built a real walking VR system in a $6m \times 5.5m$ seminar room in the university. The Oculus DK2 HMD, which is connected to and driven by a laptop with Intel i7-6700HQ CPU, NVIDIA GTX1070 GPU and 16 GB of RAM, is adopted to supply a pair of identical binocular images of the mapped VE. Users carry the laptop and wear the HMD when they navigate the VE by walking. The hardware is powered by an uninterruptible power supply in a backpack. We use the same approach as [Sun et al. 2016] to compute the inverse map of $\mathbf{f}$ and render the warped VEs.

Tracking system. The user's orientation and position are needed to compute the inverse map of $\mathbf{f}$. The orientation sensor in the Oculus DK2 HMD provides the user's orientation. Generally, an extra capture system, such as the OptiTrack motion capture system [Sun et al. 2016], is adopted to track the user's position. Instead of using such an expensive capture system (about $52,000), we use four Microsoft Kinect V2 (about $150 for one Kinect) to build a low-cost tracking system as follows. At each corner of the room, we place a Kinect on a camera tripod to make a $3.6m \times 3.6m$ real walking workspace, S^R, as shown in Fig. 1 (c). Each Kinect, whichis set toward the mid-point of its opposite edge (see the right inset), is connected to a laptop. The signals of the four laptops are synchronized through network sockets.



The four Kinects are kept in the same heights and are calibrated in the same world coordinates, so that the room floor is the XOY plane and the gravity upright is the Z axis. As the user walks around, each Kinect returns its skeleton, and we use the neck joint of the skeleton to represent the user's position, which is more stable than the head joint. Then, we uniformly average the four captured positions and use this as the user's position in S^R. If one Kinect fails to capture the position at a certain time, we ignore it in the average.

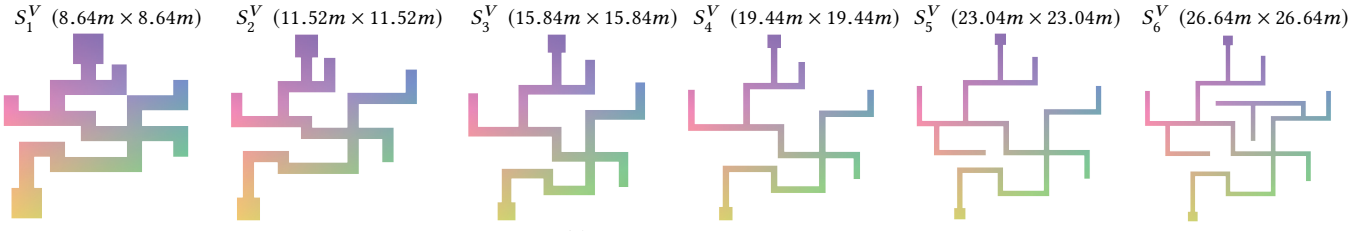
We further smooth the user's walking path by amending the current position through using their previous position as a prediction. Specifically, we denote the user's position at time t as $\mathbf{X}_t$ (in the XOY plane). The averaged position captured by the Kinects at time $t + 1$ is denoted as $\widehat{\mathbf{X}}_{t+1}$. Then, the the user's position at time $t + 1$ is computed as:

$$\mathbf{X}_{t+1} = \mathbf{X}_t + \|\widehat{\mathbf{X}}_{t+1} - \mathbf{X}_t\|_2 \mathbf{D}_t, \tag{14}$$

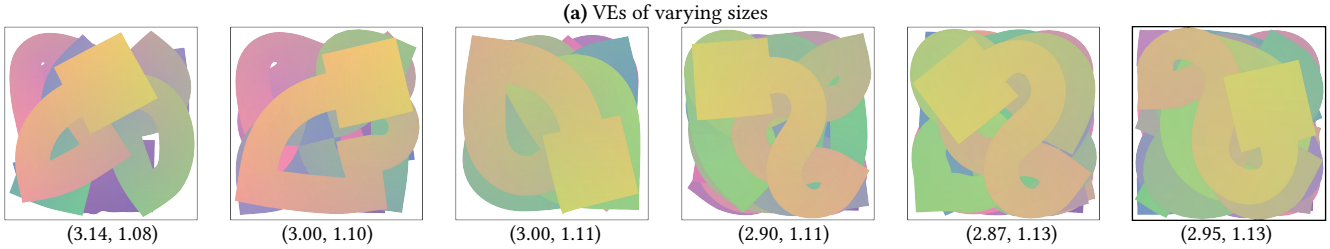
where $\mathbf{D}_t$ is the normalized direction of the blended direction:

$$\widehat{\mathbf{D}}_t = \omega_X \frac{\widehat{\mathbf{X}}_{t+1} - \mathbf{X}_t}{\|\widehat{\mathbf{X}}_{t+1} - \mathbf{X}_t\|_2} + (1 - \omega_X) \frac{\mathbf{X}_t - \mathbf{X}_{t-1}}{\|\mathbf{X}_t - \mathbf{X}_{t-1}\|_2}, t > 0, \tag{15}$$

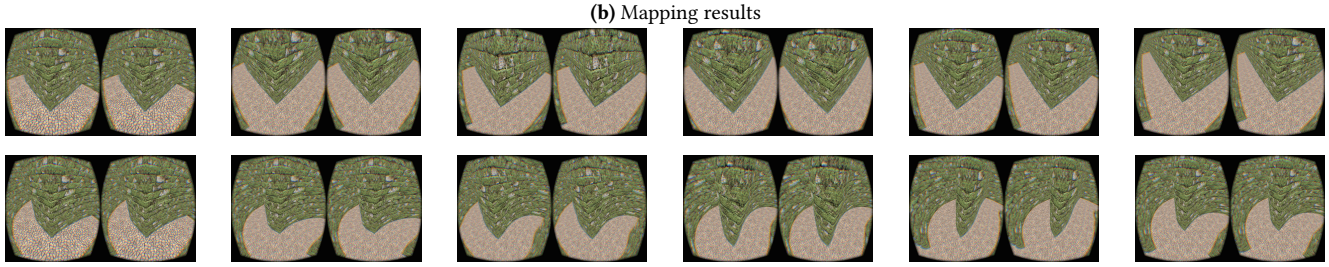
where $\mathbf{X}_0 = \widehat{\mathbf{X}}_0$, and ω_X is a weight ($\omega_X = 0.8$ by default).



(a) VEs of varying sizes

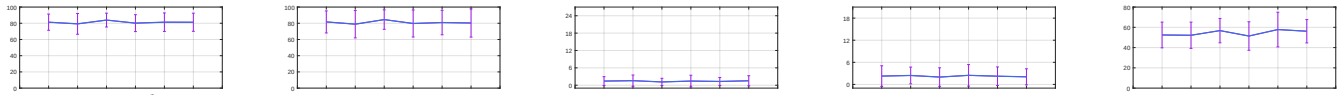


(b) Mapping results



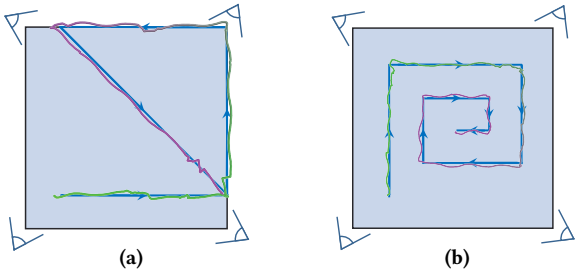
(c) Rendering of the virtual world (upper row) and the user's HDM view (lower row)

Fig. 3. Testing our SAM method on various VEs. First row: six VEs of varying sizes. Second row: the mapping results. Numbers in brackets denote maximal distortion and average distortion. Third row: the rendering of the virtual world (upper row) and the user's HDM view (lower row) at similar viewpoints. From the distortion, we can see that SAM obtains low distortion mappings for these VEs, even for the maximal one S_6^V which is about 54 times S^R in area.



(a) ρ^{loc} **(b)** ρ^{vir} **(c)** ρ^{nau} **(d)** ρ^{ocu} **(e)** Distance

Fig. 4. Plots of averaged locomotion fidelity, visual fidelity, nausea score, oculomotor score, and walking distance, respectively, for the user study that 30 participants are asked to walk freely in six scenes (one by one in a random order).



(a) **(b)**

Fig. 5. Validation of our low-cost tracking system (shown in Fig. 1 (c)). Two paths (shown in blue) are designed in S^R and ten users are asked to walk along them. The average walking paths of the users are shown as curves in gradient color which are quite close to the designed paths, which illustrates that our tracking system is suitable for real walking applications.

To verify our tracking system, we design two paths in S^R (Fig. 5 (a)) and ask ten users to walk along them. It can be seen that the tracked and designed curves are very similar, and the average distance error between them is less than $5cm$. Our tracking system is cheap and easy to setup and is used for all our experiments.

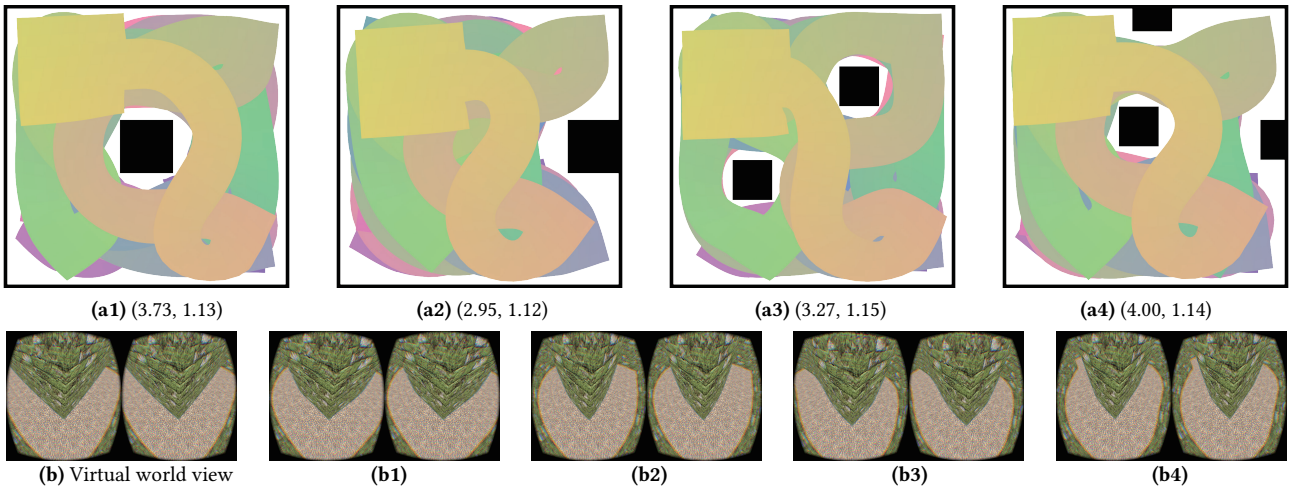
5.2 User study metrics

We conduct numerous user studies to evaluate our SAM method.

Participants. For each user study, we enroll 30 participants between the ages of 18 and 35. Some of the participants had used the HMD before, while others never have the experience. Table 1 shows details of the participants, such as their average age and deviation, and the number of males and females, etc., in each user study.

Objective metric. We adopt the isometric distortion metric used in [Fu et al. 2015] as the objective distortion metric. For each sample point $\mathbf{z}$, its distortion is defined as $\delta_{\mathbf{z}}^{iso} = \max\{\sigma_{\mathbf{z}}^1, 1/\sigma_{\mathbf{z}}^1\, \sigma_{\mathbf{z}}^2, 1/\sigma_{\mathbf{z}}^2\}$, where $\sigma_{\mathbf{z}}^1$ and $\sigma_{\mathbf{z}}^2$ are the two singular values of the Jocabian of the map $\mathbf{f}$ at $\mathbf{z}$. $\delta_{\mathbf{z}}^{iso}$ reaches the optimal value 1 when $\sigma_{\mathbf{z}}^1 = \sigma_{\mathbf{z}}^2 = 1$. We uniformly sample 1000^2 points in each patch and denote $\delta_{\max}^{iso}$ and $\delta_{\text{avg}}^{iso}$ as the maximal and average distortion values on the sample points respectively.

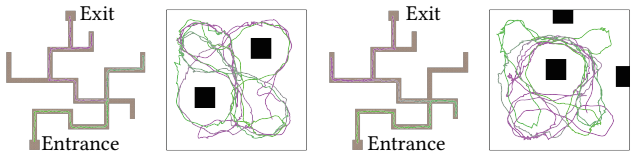
Subjective metric. As in [Bowman et al. 2002], we use the locomotion fidelity ρ^{loc} (out of 100) and the visual fidelity ρ^{vis} (out of 100) to measure the VR usability for the user when navigating a VE. We



(a1) (3.73, 1.13) **(a2)** (2.95, 1.12) **(a3)** (3.27, 1.15) **(a4)** (4.00, 1.14)

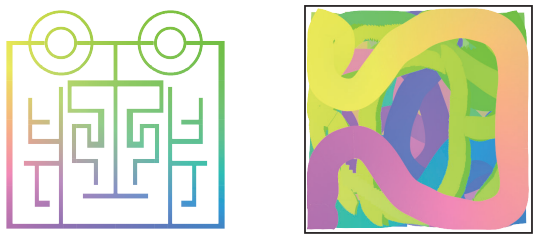
(b) Virtual world view **(b1)** **(b2)** **(b3)** **(b4)**

Fig. 6. (a1-a4) The S_4^V in Fig. 3 is mapped into S^R with four different interior obstacles (represented by black blocks). The virtual world appearance is rendered in (b) and the corresponding user's HMD views are shown (b1-b4).



(a1) Virtual path **(a2)** Real path **(b1)** Virtual path **(b2)** Real path

Fig. 7. Two real paths from the entrance to the exit of scenes (a1) and (b1), which are scenes in Fig. 6 (a3) and (a4) respectively, are shown in (a2) and (b2), respectively. See the accompanying video for users' real walking without bumping into any obstacles.



(a) **(b)**

Fig. 8. (a) The input VE as the reachable region of Fig. 1 (a). (b) Mapping result whose $\delta_{\max}^{iso} = 3.10$ and $\delta_{\text{avg}}^{iso} = 1.21$, respectively.

define ρ_{avg}^{loc} and ρ_{dev}^{loc} as the average and standard deviation of ρ^{loc} for all participants in one user study. ρ_{avg}^{vis} and ρ_{dev}^{vis} are similarly defined.

Furthermore, the standard post-interview-based method is used to evaluate the motion sickness by the simulator sickness questionnaire (SSQ) [Kennedy et al. 1993] and we record the scores of nausea ρ^{nau} (out of 27) and of oculomotor ρ^{ocu} (out of 21). Their average and standard deviation are denoted as ρ_{avg}^{nau}, ρ_{dev}^{nau}, ρ_{avg}^{ocu}, and ρ_{dev}^{ocu} respectively.

Task failure. Because we have not implemented collision detection, the participants are told not to cross the wall in the VE before the experiments. For each experiment, the participants are asked to fulfil a task in the VE via real walking, and we terminate it after they either feel sick or walk across the wall. We denote N_{fail} as the number of participants who fail to fulfil this task.

5.3 Experiments

VEs of varying sizes. We test our SAM method on six different VEs of varying sizes; the maximal one (S_6^V) is about 54 times of S^R in area, as shown in Fig. 3. All of the $\delta_{\max}^{iso}$ and $\delta_{\text{avg}}^{iso}$ for six scenes are close to 3.0 and 1.1, respectively (see Table 1), which means that our SAM method obtains low distortion mappings for these VEs.

Each participant is asked to walk freely in six scenes (one by one in a random order) within 2 minutes, and we record the walking distance (in meters) for each task. We ask the participants to take a break for more than half an hour after completing one task; they carry on to the next task after the break.

All participants have successfully fulfilled all tasks, i.e., $N_{fail} = 0$. From their feedback we see that they had similar experiences in these scenes. Some of them even told us that there are no differences within these scenes. This is proved by the plots of ρ_{avg}^{nau}, ρ_{dev}^{nau}, ρ_{avg}^{ocu}, and ρ_{dev}^{ocu}, as shown in Fig. 4 (a)-(d), as these values are similar with small deviations. Both the nausea and oculomotor scores are in the comfortable range. Fig. 4 (e) shows the average and deviation of the walking distances for different scenes. The averages are similar, which means that the mapping results generated by our SAM method are similar for all scenes.

Obstacles in S^R. We map S_4^V in Fig. 3 into S^R with four different interior obstacles, as shown in Fig. 6. We ask five participants (four males and one female) to carry out the task of walking from the entrance and to find their way to the exit. All of the participants succeeded in completing this task without bumping into any obstacles. Fig. 7 shows the walking paths of two participants in both S^V and S^R, respectively.

Large-scale VE.. We map a large VE ($24m \times 24m$) into a small RW ($3.6m \times 3.6m$), as shown in Fig. 8. The virtual world floor plan of the VE is shown in Fig. 1 (a). In this VE, there are curve-shaped and non-uniform pathways, forming some loops.

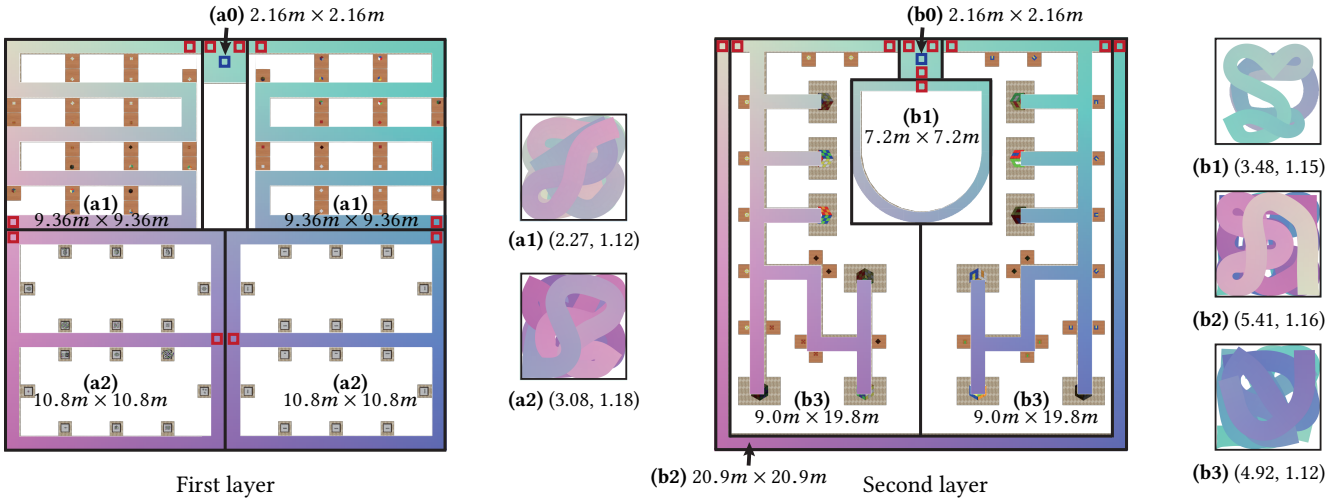


Fig. 9. Example for teleportation technique on a large-scale VE with two layers. There are five regions in the first layer and five regions in the second layer. The blue rectangles in (a0) and (b0) are used as the teleporters for connecting two layers. The adjacent red rectangles are the teleporters for connecting the regions. The mappings of various regions are shown with distortion.

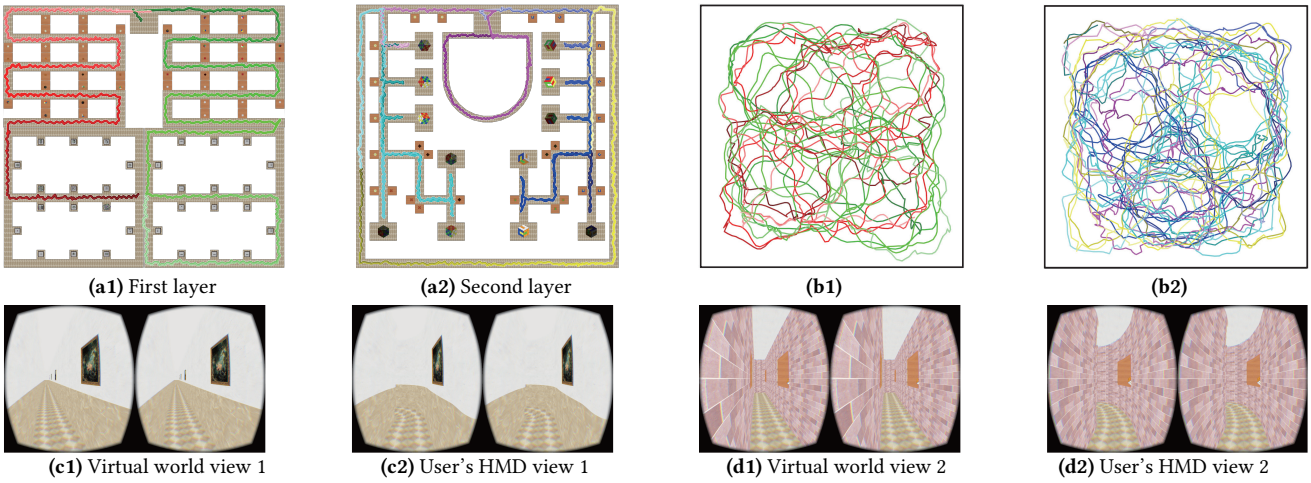


Fig. 10. (b1) and (b2) show the real paths of walking through the first and second layers in (a1) and (a2), respectively. The virtual world appearances at two viewpoints are rendered in (c1) and (d1), respectively, and the corresponding user's HMD views are shown in (c2) and (d2), respectively.

Large-scale VEs via teleportation. Fig. 9 shows a large-scale VE of an exhibition hall with two layers. There are five regions in the first layer (Fig. 9 (left)) and two regions, (a1) and (a2), are symmetric to the other two. There are five regions in the second layer (Fig. 9 (right)), and one region (b3) is symmetric to the other one. (b1) is a curved pathway, and (b2) is a long corridor. The sizes of these regions are shown in the Fig. 9. The blue rectangle in (a0) from the first layer and the blue rectangle from (b0) in the second layer are used as the teleporters to connect the two layers. The adjacent red rectangles are the teleporters connecting the other regions.

The mapping results for regions (a1) and (a2) in the first layer and (b1), (b2), and (b3) in the second layer, with the values of $\delta^{iso}_{\max}$ and $\delta^{iso}_{\text{avg}}$, are shown in Fig. 9. The values of $\delta^{iso}_{\max}$ and $\delta^{iso}_{\text{avg}}$ show that all mappings have low levels of distortion.

We invite five participants to freely walk through and look around the exhibition hall. All of the participants have navigated through some of the regions without feeling uncomfortable or sick. Fig. 10 shows the walking paths (b1) and (b2), which are composed of the walking paths of all participants in two layers, respectively. Two rendered views are shown in Fig. 10.

Performance. The mapping is computed offline. It takes about 7 minutes to compute the mapping for the scene Office, shown in Fig. 11 (a), and it takes 30 minutes for the S^V_3 in Fig. 3.

During users' navigation, the computation of the dynamic inverse mapping achieves more than 100 FPS for all scenes. The rendering of the images (with a resolution of 1182×1464) in the HMD is affected by the tracking rate. As Kinect provides only 30 FPS rates for tracking the users' skeletons, we extrapolate the users' location using their previous walking direction and then speed up the rendering rate to about 60 FPS, which is sufficient for a quality VR experience.

Size constraint between S^V and S^R. The quality of the computed map $\mathbf{f}_k$ highly depends on the sizes of the primitive patches P^V_k when compared to S^R. If a patch $P^V_{k^*}$ is larger than S^R, it is impossible

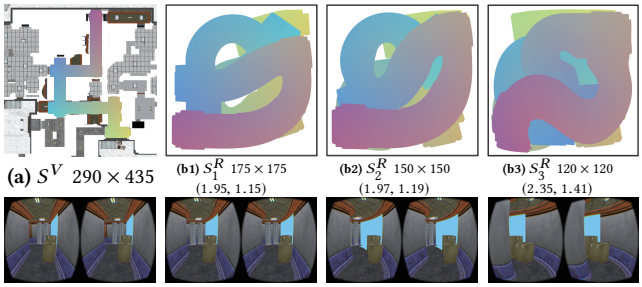


Fig. 11. (a) The input Office S^V with maximal patch of 60×50. (b1-b3) S^V is mapped into different RWs (S_1^R, S_2^R, and S_3^R). (c) The rendered virtual world appearance in S^V. (d1-d3) The corresponding warped VEs in user's HMD view of (b1-b3). Numbers in brackets denote $\delta_{\max}^{iso}$ and $\delta_{\text{avg}}^{iso}$. From the objective metric and rendered images, we see that mapping S^V to S_1^R and S_2^R are more acceptable.

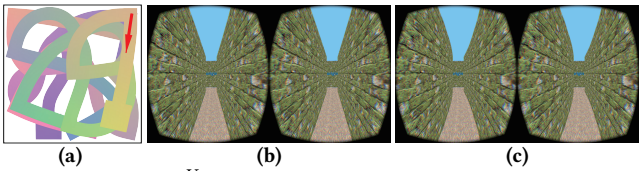


Fig. 12. (a) The VE S_5^V in Fig. 3 is mapped into a new RW ($7.2m \times 7.2m$). The rendered virtual world appearance in the original VE (b) and warped VE (c). The viewpoint for (c) is along the red arrow in (a).

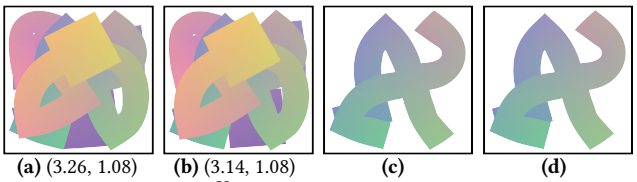


Fig. 13. We map the VE S_1^V in Fig. 3 using the successive global optimization scheme (a) and our SAM method (b) respectively, part of which are shown in (c) and (d) respectively. The results are very similar.

to map $P_{k^*}^V$ into S^R without small isometric distortion. Even if $P_{k^*}^V$ is smaller than S^R, the map $\mathbf{f}_{k^*}$ may have large distortions due to the smoothness constraints in the assembly. From numerous experiments and their analysis (see the varying sizes of RWs in Fig. 11), we have observed that if the largest width of the pathways in S^V is smaller than the size of S^R within a ratio of 0.35, then our SAM method can generate substantially acceptable mapping results for VR applications.

Note that our method provides good mappings with lower distortion when the size ratio is small. We show an example in Fig. 12. Compared to the mapping result of S_5^V in Fig. 3, we zoom-in the RW twice in length, achieving much smaller maximal distortion and average distortion (1.55, 1.03). From the rendered virtual world appearance difference between the input (Fig. 12(b)) and mapped VEs (Fig. 12(c)), our method achieves nearly straight roads.

Successive global optimization scheme. The successive global optimization scheme may use less time and generate less distortion result than our SAM method for very large-scale VEs. For other VEs, it generates similar results as our SAM method does and causes more time than ours because of its successive optimization manner. For example, we apply the successive global optimization scheme on mapping the VE S_1^V in Fig. 3, as shown in Fig. 13 (a,c). The generated result looks very similar to that generated by our SAM method as shown in Fig. 13 (b,d). However, it computational time is about 2 times our SAM method.

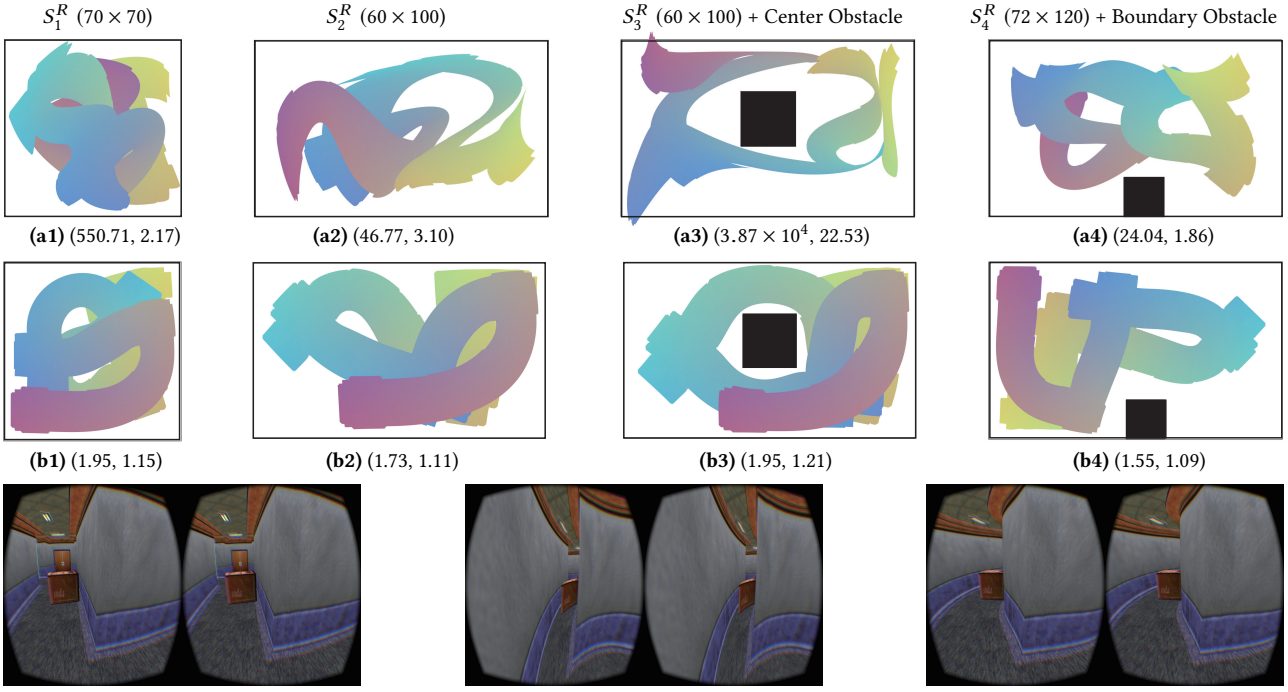
Fig. 14. The VE ($11.88m \times 23.04m$) (a) and its map by our method (b). We use it for comparing with S2C.

5.4 Comparisons

Comparison to the redirected walking method. Redirected walking allows users to navigate large-scale VEs by real walking via interactively and imperceptibly rotating the VE about the user. We choose the VE in Fig. 14 (a), which is a pathway of 52.56 meters with a few turns, to compare our SAM method with one of the redirected walking methods, i.e., the Steer-to-Center (S2C) technique proposed in [Hodgson and Bachmann 2013] and implemented by [Azmandian et al. 2016]. We conduct two user studies.

In the first user study, the task for each participant is to walk from the entrance to the exit using our SAM method and the S2C method, respectively. All of the participants succeeded in fulfilling the task with our SAM method. However, five participants failed to fulfil the task using the S2C method because they are not aware of the sign that says "Spin In Place" at the reset location; and thus, they went out of the RW. The average finish time with our SAM method is 119.40 seconds with a standard deviation of 36.17, and the average finish time with the S2C method is 221.41 seconds with a standard deviation of 46.36. The reason why the S2C method takes longer is that the participants are often interrupted by the 'reset', so they have to stop walking and turn themselves around according to the reset hints. On the other hand, our SAM method allows the participants to walk through the VE without any interruptions, making the navigation very smooth. From the subjective aspect, our SAM method provides comparable user experiences to the S2C method, which are verified by the subjective metrics (not much difference between two methods), as shown in Table 1.

In the second user study, the task is to ask the participants to play a game in the VE. The participants are chased by a scary wolf , and they must try not to be caught by the wolf. If they are caught during the navigation, they lose the game. Otherwise, if they succeed in reaching the exit without being caught, they win the game. The wolf starts from the entrance and the participant starts from a place $4m$ away from the entrance. The wolf moves at a constant speed of $0.6m/s$, which is much slower than the normal walking speed of people in the real world [Browning et al. 2006; Mohler et al. 2007]. 28 participants won the game using our SAM method while only one participant, who has extensive experience with VR, won the game using the S2C method. Since the participants waste time

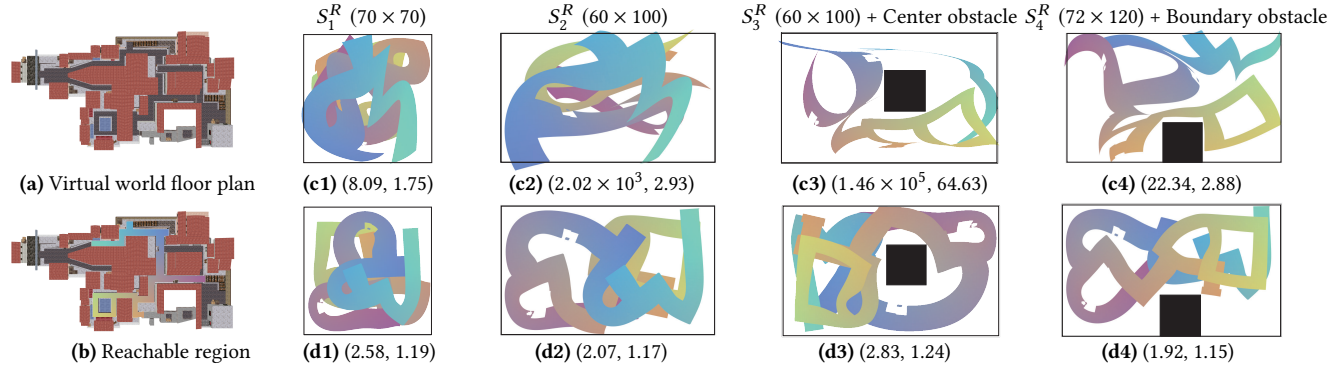


(a1) (550.71, 2.17) **(a2)** (46.77, 3.10) **(a3)** (3.87×10^4, 22.53) **(a4)** (24.04, 1.86)

(b1) (1.95, 1.15) **(b2)** (1.73, 1.11) **(b3)** (1.95, 1.21) **(b4)** (1.55, 1.09)

(c) Virtual world view **(d)** [Sun et al. 2016] **(e)** Our method

Fig. 15. Comparison to [Sun et al. 2016] on the VE Office in Fig. 11 (a). The first row shows the mapping results using [Sun et al. 2016] and the second row shows the results using our method. The third row shows the rendered images.



(a) Virtual world floor plan **(c1)** (8.09, 1.75) **(c2)** (2.02×10^3, 2.93) **(c3)** (1.46×10^5, 64.63) **(c4)** (22.34, 2.88)

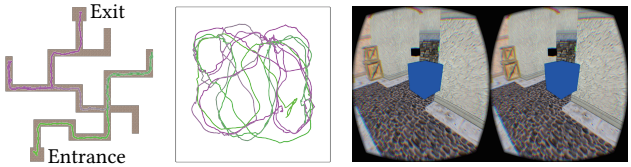
(b) Reachable region **(d1)** (2.58, 1.19) **(d2)** (2.07, 1.17) **(d3)** (2.83, 1.24) **(d4)** (1.92, 1.15)

(f) Virtual world view **(e)** [Sun et al. 2016] **(g)** Our method

Fig. 16. Comparison to [Sun et al. 2016] on the VE Italy (a) with size of 156 × 184. The first row shows the mapping results using [Sun et al. 2016] and the second row shows the results using our method. The third row shows the rendered images.

by stopping and making turns at the 'reset' locations, they have a greater possibility of being caught by the wolf, thus losing the game. On the other hand, many participants walk faster than usual in the games using our SAM method, because it provides a continuous and smooth map for the VE. Some participants even stop to wait for the wolf a bit when the wolf is left far behind.

Comparison to [Sun et al. 2016]. We compare our SAM method and the global mapping method [Sun et al. 2016] (GLM method for short) on two VEs (the scene Office in Fig. 11 (a) and the scene Italy in Fig. 16 (a)). From the values of $\delta_{\max}^{iso}$ and $\delta_{\text{avg}}^{iso}$ shown in Fig. 11 and 16, our SAM produces much lower isometric distortion than the GLM method. As the GLM method introduces large distance



(a) Virtual path **(b)** Real path **(c)**

Fig. 17. Two VR games using our method. Maze game: a typical user path for the maze (a) is shown in (b). Grasping game: the user is asked to grasp the floating colored boxes shown in (c).

distortions, the warped scenes may be unacceptable due to the degenerated pathways, such as the example shown in Fig. 16 (c3). As these narrow paths always lead to real movement failure, we compare with [Sun et al. 2016] on Fig. 16 (c3) and (d3) by simulation in the accompanying video.

We conduct a user study with 30 participants. We choose the scene Office (Fig. 11 (a)) as S^V and map it in S^R using our SAM method and the GLM method, respectively. Each participant is asked to walk through the pathway from one end to the other, and the finish time is recorded. The average finish time of our SAM method is 55.3 seconds with a standard deviation of 20.94, and the average time of 90.6 seconds with a standard deviation of 34.22 for the GLM method. The reason why it takes longer to walk through the GLM map is that its generated pathways are unrecognizable in many places, making the participants feel uncomfortable when walking through.

5.5 Applications

We design two games as the applications for our SAM method, including the maze game and the grasping game, shown in Fig. 17.

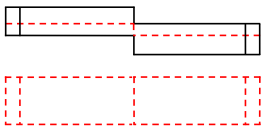
Maze game. In this game, the participants have to find their way out of a maze scene (S_3^V from Fig. 3). As there is only one feasible path from the entrance to the exit in the maze, the participants may go along wrong ways and turn around when they reach dead ends. Thirty participants are enrolled in the maze game. 28 participants succeeded in reaching the exit of the maze, and two failed in the game by walking across a wall. No one felt uncomfortable or sick while playing the game. From both objective and subjective metrics, shown in Table 1, our SAM method provides realistic and comfortable user experiences in this game.

Grasping game. In this game, the participants are asked to grasp as many boxes as possible in the VE (Fig. 16 (a)) within 2 minutes. Some floating boxes are generated and randomly placed in the VE below the height of 1.6m. The participants must move around, find the boxes, and grasp them. Thirty participants are enrolled to play this game. All of the participants succeeded in playing the game without walking across a wall, and no one felt uncomfortable or sick while playing the game. The average number of grasped boxes was 23.17 with a standard deviation of 3.40. All of the participants reported that they enjoyed playing this game, which means that our SAM method provides realistic and immersive user experiences (also see Table 1). Some participants even asked to play the game for more times.

6 CONCLUSION

Our proposed SAM method adopts a divide-and-conquer strategy to generate maps from large-scale virtual scenes to small real workspace. The success of SAM relies on two aspects: (1) new representations for mappings and new formulations for distortion and boundary constraints, which results in an effective solver generating better results; (2) the divide-and-conquer strategy can adapt for very large-scale VEs. A large number of experiments on various virtual scenes, as well as intensive user studies, have been done to validate our SAM methods from both subjective and objective aspects. Experimental results have shown the feasibility and practicability of our SAM method. Substantial comparisons to the redirected walking method [Hodgson and Bachmann 2013] and the global mapping method [Sun et al. 2016] have shown that our SAM method obtains better user experiences when navigating VEs. We believe our SAM method provides a general method to produce low-distortion maps between large-scale virtual scenes and small real workspaces, providing great potential in applying VR applications for home users.

Limitation. Our method needs to decompose the input pathways into rectangular patches. First, for the reachable region like the region bounded by the black polygon in the right inset, we need to decompose it in small subpatches using the red dotted lines (upper) or to enclose it with enlarged rectangular patches (bottom). Our method may generate high distortion mapping as there are many smoothness constraints on these small patches. Maybe the global method, such as [Sun et al. 2016], can achieve lower distortion mapping when the whole VE is small. Second, if there are pathways in the VE with large widths (compared to the size of the RW), the map generated by our method may have large isometric distortions, which may cause uncomfortable experience for the users. Actually, other methods also suffer this problem.



Future work. Our research opens many directions for future studies. First, it is interesting to consider of using T-spline surfaces to represent the mapping, which can handle more complicated input reachable regions. Second, we would like to combine different methods, such as redirected walking methods and space mapping methods, to achieve better real walking experiences. A straightforward idea is to apply the redirected walking method in some regions, which necessitate an absence of distortion, and apply the space mapping method in other regions, where users are not that sensitive to the distortion. Some perceptual and psychological studies on the virtual scenes might be helpful for this problem. Third, the input VE has to be reachable regions consisting of a set of narrow pathways and users are required to walk following the pathways. Studying real walking in open scenes, which allow users to walk freely in VE, is a promising research direction for future work. Last but not least, it is worthwhile studying on applying the mapping method in AR applications [Sousa et al. 2016]. It is feasible but challenging.

ACKNOWLEDGEMENTS

We would like to thank Qi Sun for providing their data in [Sun et al. 2016] and his assistance on the rendering part, authors of [Azmandian et al. 2016] for providing the implementation of S2C, user

Table 1. Statistics for user studies. We report the number of participants ("#part"), their age range, the average age a_{avg}, the standard deviation of their ages a_{dev}, the number of males and females, and the number of participants who have experienced the HMD before ("#used"). We also report the statistics of the user study results: the isometric distortion, the locomotion fidelity score, the visual fidelity , the nausea score, and the oculomotor score. A number in boldface emphasizes the best result for comparisons.

Scene	#part	Age Range	a_{avg}	a_{dev}	Male, Female	#used	$\delta^{iso}_{\max}/\delta^{iso}_{\text{avg}}$	N_{fail}	$\rho^{loc}_{\text{avg}}/\rho^{loc}_{\text{dev}}$	$\rho^{vis}_{\text{avg}}/\rho^{vis}_{\text{dev}}$	$\rho^{nau}_{\text{avg}}/\rho^{nau}_{\text{dev}}$	$\rho^{ocu}_{\text{avg}}/\rho^{ocu}_{\text{dev}}$
Fig. 3-S1	30	[20, 28]	23.53	2.19	21, 9	21	3.14/1.08	0	81.33/9.99	81.67/13.59	1.43/1.52	2.27/2.80
Fig. 3-S2	30	[20, 28]	23.53	2.19	21, 9	21	3.00/1.10	0	79.27/12.81	78.90/16.93	1.57/1.96	2.43/2.27
Fig. 3-S3	30	[20, 28]	23.53	2.19	21, 9	21	3.00/1.11	0	83.93/8.53	84.53/12.01	1.13/1.25	2.00/2.52
Fig. 3-S4	30	[20, 28]	23.53	2.19	21, 9	21	2.90/1.11	0	80.17/10.42	79.73/16.66	1.43/2.01	2.47/2.92
Fig. 3-S5	30	[20, 28]	23.53	2.19	21, 9	21	2.87/1.13	0	81.33/11.42	80.9/15.09	1.30/1.37	2.23/2.51
Fig. 3-S6	30	[20, 28]	23.53	2.19	21, 9	21	2.95/1.13	0	81.27/11.18	80.33/17.40	1.53/1.74	2.07/2.18
Fig. 14-S2C	30	[18, 26]	21.63	2.30	19, 11	18	**1.00**/**1.00**	5	75.10/19.71	**100.00**/**0.00**	2.53/2.42	3.20/**2.55**
Fig. 14-ours	30	[18, 26]	21.63	2.30	19, 11	18	2.35/1.11	**0**	**83.57**/**10.23**	75.9/19.13	**2.00**/**2.24**	**2.80**/2.98
Fig. 15-[Sun et al. 2016]	30	[19, 30]	23.77	2.50	21, 9	18	550.71/2.17	**0**	71.77/18.57	58.83/20.50	2.77/2.98	3.40/3.28
Fig. 15-ours	30	[19, 30]	23.77	2.50	21, 9	18	**2.19**/**1.19**	**0**	**81.37**/**14.10**	**83.5**/**10.92**	**1.87**/**1.89**	**2.00**/**2.39**
Fig. 17-Maze	30	[20, 35]	24.7	3.21	22, 8	10	3.00/1.11	2	79.80/13.63	76.20/16.40	1.80/1.71	2.40/2.09
Fig. 17-Grasp	30	[19, 28]	22.63	2.40	23, 7	12	2.58/1.19	0	81.83/12.44	82.00/9.34	2.73/3.22	3.30/3.32

study participants for evaluating our system, and the anonymous reviewers for their constructive suggestions and comments. This work is supported by the National Natural Science Foundation of China (61672482, 11626253) and the One Hundred Talent Project of the Chinese Academy of Sciences.

REFERENCES

Mahdi Azmandian, Timofey Grechkin, Mark Bolas, and Evan Suma. 2016. The redirected walking toolkit: a unified development platform for exploring large virtual environments. In *Everyday Virtual Reality (WEVR), 2016 IEEE 2nd Workshop on*. 9–14.

Mahdi Azmandian, Rhys Yahata, Mark Bolas, and Evan Suma. 2014. An enhanced steering algorithm for redirected walking in virtual environments. In *Virtual Reality (VR), 2014 IEEE*. 65–66.

Doug A Bowman, Joseph L Gabbard, and Deborah Hix. 2002. A survey of usability evaluation in virtual environments: classification and comparison of methods. *Presence* 11, 4 (2002), 404–424.

Raymond C Browning, Emily A Baker, Jessica A Herron, and Rodger Kram. 2006. Effects of obesity and sex on the energetic cost and preferred speed of walking. *Journal of Applied Physiology* 100, 2 (2006), 390–398.

Renjie Chen and Ofir Weber. 2015. Bounded Distortion Harmonic Mappings in the Plane. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015), 73:1–73:12.

Lung-Pan Cheng, Thijs Roumen, Hannes Rantzsch, Sven Köhler, Patrick Schmidt, Robert Kovacs, Johannes Jasper, Jonas Kemper, and Patrick Baudisch. 2015. TurkDeck: Physical Virtual Reality Based on People. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 417–426.

J.M. Escobar, E. Rodríguez, R. Montenegro, G. Montero, and J.M. González-Yuste. 2003. Simultaneous untangling and smoothing of tetrahedral meshes. *Comput. Methods Appl. Mech. Engrg.* 192 (2003), 2775–2787.

Tom Field and Peter Vamplew. 2004. Generalised algorithms for redirected walking in virtual environments. In *Artificial Intelligence in Science and Technology*. 21–25.

Xiao-Ming Fu, Yang Liu, and Baining Guo. 2015. Computing locally injective mappings by advanced MIPS. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015), 71:1–71:12.

Eric Hodgson and Eric Bachmann. 2013. Comparing four approaches to generalized redirected walking: Simulation and live user data. *IEEE. T. Vis. Comput. Gr.* 19, 4 (2013), 634–643.

Eric Hodgson, Eric Bachmann, and David Waller. 2011. Redirected walking to explore virtual environments: Assessing the potential for spatial interference. *ACM Transactions on Applied Perception (TAP)* 8, 4 (2011), 22.

Hiroo Iwata, Hiroaki Yano, and Hiroshi Tomioka. 2006. Powered Shoes. In *ACM SIGGRAPH 2006 Emerging Technologies*.

Robert S Kennedy, Norman E Lane, Kevin S Berbaum, and Michael G Lilienthal. 1993. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The international journal of aviation psychology* 3, 3 (1993), 203–220.

Zohar Levi and Ofir Weber. 2016. On the Convexity and Feasibility of the Bounded Distortion Harmonic Mapping Problem. *ACM Trans. Graph.* 35, 4 (2016), 106:1–106:15.

Betty J Mohler, William B Thompson, Sarah H Creem-Regehr, Herbert L Pick, and William H Warren. 2007. Visual flow influences gait transition speed and preferred walking speed. *Experimental brain research* 181, 2 (2007), 221–228.

Thomas Nescher, Ying-Yin Huang, and Andreas Kunz. 2014. Planning redirection techniques for optimal free walking experience using model predictive control. In *3D User Interfaces (3DUI), 2014 IEEE Symposium on*. 111–118.

Tabitha C Peck, Henry Fuchs, and Mary C Whitton. 2012. The design and evaluation of a large-scale real-walking locomotion interface. *IEEE. T. Vis. Comput. Gr.* 18, 7 (2012), 1053–1067.

Roi Poranne and Yaron Lipman. 2014. Provably good planar mappings. *ACM Trans. Graph. (SIGGRAPH)* 33, 4 (2014), 76:1–76:11.

Sharif Razzaque. 2005. *Redirected Walking*. Ph.D. Dissertation. Chapel Hill, NC, USA.

Sharif Razzaque, Zachariah Kohn, and Mary C Whitton. 2001. Redirected walking. In *Proceedings of EUROGRAPHICS*, Vol. 9. 105–106.

Sharif Razzaque, David Swapp, Mel Slater, Mary C Whitton, and Anthony Steed. 2002. Redirected walking in place. In *ACM International Conference Proceeding Series*, Vol. 23. 123–130.

Martin Schwaiger, Thomas Thümmel, and Heinz Ulbrich. 2007. Cyberwalk: Implementation of a Ball Bearing Platform for Humans. In *Proceedings of the 12th International Conference on Human-computer Interaction: Interaction Platforms and Techniques*. 926–935.

Jason Smith and Scott Schaefer. 2015. Bijective Parameterization with Free Boundaries. *ACM Trans. Graph. (SIGGRAPH)* 34, 4 (2015), 70:1–70:9.

J. L. Souman, P. Robuffo Giordano, M. Schwaiger, I. Frissen, T. Thümmel, H. Ulbrich, A. De Luca, H. H. Bülthoff, and M. O. Ernst. 2008. CyberWalk: Enabling Unconstrained Omnidirectional Walking Through Virtual Environments. *ACM Trans. Appl. Percept.* 8, 4 (2008), 25:1–25:22.

L. Sousa, R. Alves, and J. Rodrigues. 2016. Augmented reality system to assist inexperienced pool players. *Computational Visual Media* 2, 2 (2016), 183ÍC193.

Frank Steinicke, Gerd Bruder, Jason Jerald, Harald Frenz, and Markus Lappe. 2008. Analyses of human sensitivity to redirected walking. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*. 149–156.

Evan A Suma, Seth Clark, David Krum, Samantha Finkelstein, Mark Bolas, and Zachary Warte. 2011. Leveraging change blindness for redirection in virtual environments. In *Virtual Reality Conference (VR), 2011 IEEE*. 159–166.

Evan A Suma, Zachary Lipps, Samantha Finkelstein, David M Krum, and Mark Bolas. 2012. Impossible spaces: Maximizing natural walking in virtual environments with self-overlapping architecture. *IEEE. T. Vis. Comput. Gr.* 18, 4 (2012), 555–564.

Qi Sun, Li-Yi Wei, and Arie Kaufman. 2016. Mapping Virtual and Physical Reality. *ACM Trans. Graph. (SIGGRAPH)* 35, 4 (2016), 64:1–64:12.

Martin Usoh, Kevin Arthur, Mary C. Whitton, Rui Bastos, Anthony Steed, Mel Slater, and Frederick P. Brooks, Jr. 1999. Walking > Walking-in-place > Flying, in Virtual Environments. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. 359–364.

Khrystyna Vasylevska and Hannes Kaufmann. 2017. Towards efficient spatial compression in self-overlapping virtual environments. In *3D User Interfaces (3DUI), 2017 IEEE Symposium on*. 12–21.

Khrystyna Vasylevska, Hannes Kaufmann, Mark Bolas, and Evan A Suma. 2013. Flexible spaces: Dynamic layout generation for infinite walking in virtual environments. In *3D User Interfaces (3DUI), 2013 IEEE Symposium on*. 39–42.

Betsy Williams, Gayathri Narasimham, Bjoern Rump, Timothy P. McNamara, Thomas H. Carr, John Rieser, and Bobby Bodenheimer. 2007. Exploring Large Virtual Environments with an HMD when Physical Space is Limited. In *Proceedings of the 4th Symposium on Applied Perception in Graphics and Visualization*. 41–48.

Michael A Zmuda, Joshua L Wonser, Eric R Bachmann, and Eric Hodgson. 2013. Optimizing constrained-environment redirected walking instructions using search techniques. *IEEE. T. Vis. Comput. Gr.* 19, 11 (2013), 1872–1884.