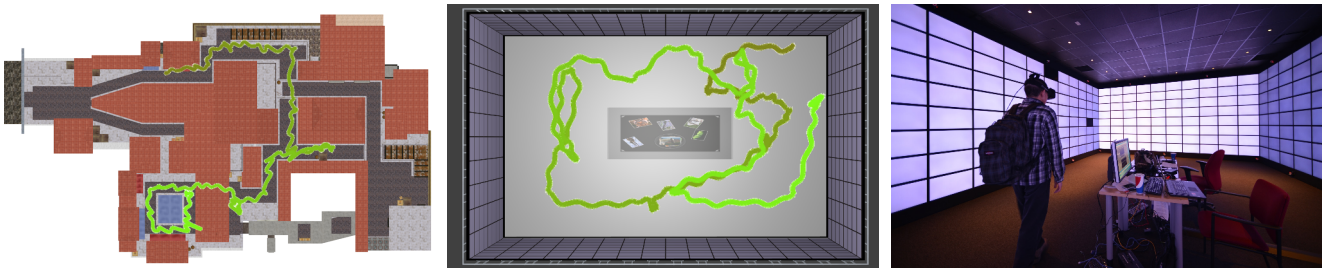
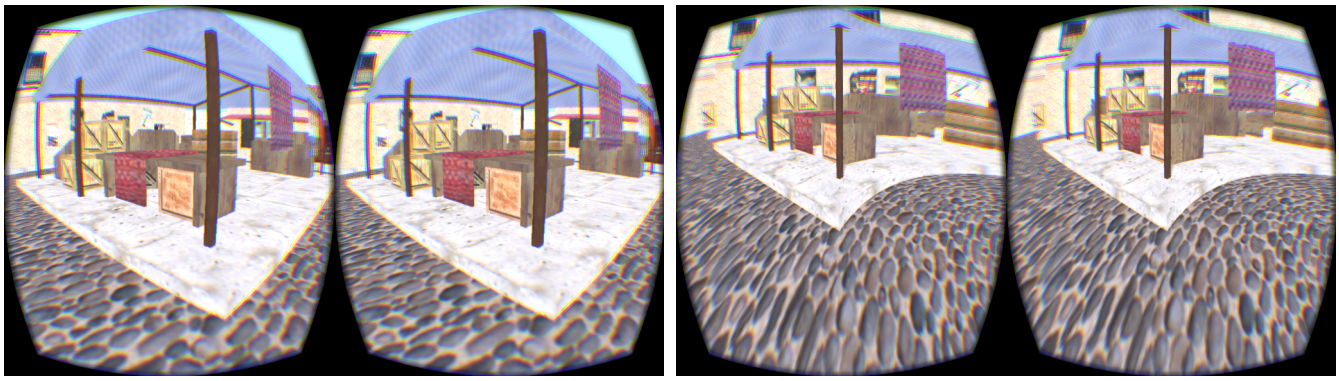
Mapping Virtual and Physical Reality

Qi Sun† Li-Yi Wei‡ Arie Kaufman†

Stony Brook University† University of Hong Kong‡



(a) *virtual world floor plan, size* $33m \times 21m$ **(b)** *real world floor plan, size* $10m \times 5.8m$ **(c)** *lab setup, for the real world in (b)*



(d) *virtual world view* **(e)** *user's HMD view*

Figure 1: *Overview of our system. (a) and (b) are aerial views of the virtual and physical worlds, with their dimensions in meters. Typically, (a) is much larger than (b). The goal is to enable the user to walk freely in (b) while experiencing (a) using a head-mounted display (HMD). Our system first computes a planar map between (a) and (b), with the mapped user walking paths overlaid in green gradients. (c) is a photograph of the lab setup, including the equipment and surroundings. Our system then renders the virtual world appearance (d) for the user's HMD view (e) in a way that is compatible with the real world geometry, so that the user can faithfully see the former and comfortably navigate the latter. Note that even though the user's view is entirely blocked by the HMD, our system guides the user away from boundaries and obstacles such as the walls and furniture. (Scene in (a) is courtesy of Counter Strike, Italy.)*

Abstract

Real walking offers higher immersive presence for virtual reality (VR) applications than alternative locomotive means such as walking-in-place and external control gadgets, but needs to take into consideration different room sizes, wall shapes, and surrounding objects in the virtual and real worlds. Despite perceptual study of impossible spaces and redirected walking, there are no general methods to match a given pair of virtual and real scenes.

We propose a system to match a given pair of virtual and physical worlds for immersive VR navigation. We first compute a planar map between the virtual and physical floor plans that minimizes angular and distal distortions while conforming to the virtual environment goals and physical environment constraints. Our key idea is to design maps that are globally surjective to allow proper folding of large virtual scenes into smaller real scenes but locally injective to avoid locomotion ambiguity and intersecting virtual objects. From these maps we derive altered rendering to guide user navigation within the physical environment while retaining visual fidelity to the virtual environment. Our key idea is to properly warp the virtual world appearance into real world geometry with sufficient quality and performance. We evaluate our method through a formative user study, and demonstrate applications in gaming, architecture walkthrough, and medical imaging.

Keywords: virtual reality, head-mounted display, redirected walking, warped space, planar map, geometry morphing, camera projection, real-time rendering, human perception

Concepts: •**Computing methodologies** → **Virtual reality;**

1 Introduction

With the confluence of virtual reality (VR) hardware and software developments, a variety of devices and setups are now offering different costs, features, and capabilities. Ideally, a VR environment should facilitate full immersion and natural movement. Current devices such as projected rooms (CAVEs) and head mounted displays

SIGGRAPH '16 Technical Paper, July 24 - 28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925883

ACM Reference Format
Sun, Q., Wei, L., Kaufman, A. 2016. Mapping Virtual and Physical Reality. ACM Trans. Graph. 35, 4, Article 64 (July 2016), 12 pages. DOI = 10.1145/2897824.2925883 http://doi.acm.org/10.1145/2897824.2925883.

(HMDs) can provide realistic rendering, but often require users to remain stationary or walk within a limited area due to hardware limitations (e.g., CAVE space or HMD cables) or navigation concerns (e.g., real environments not visible through HMD). Users thus need to employ less natural means such as gamepads and walk-in-place devices to control their movements, which can negatively impact their sense of presence compared with natural interaction [Witmer and Singer 1998] and real walking [Usoh et al. 1999].

Combining realistic VR displays and real walking has the potential for immersive presence and natural interaction. Free walking is already viable on the hardware side, as upcoming HMDs are equipped with low cost motion trackers (e.g., $799 for HTC Vive including motion trackers compared to $52000 for OptiTrack motion capture). However, virtual and real worlds often differ significantly in sizes, shapes, and objects. Thus, proper mapping between the two is needed to offer a believable presence in the virtual world and feasible navigation in the real world (e.g., users remain perceptually comfortable and without bumping into objects). Devising such mapping remains an important open problem in VR display and navigation.

Techniques such as redirected locomotion [Razzaque et al. 2002; Hodgson et al. 2008], distorted space [Suma et al. 2012; Vasylevska et al. 2013], and physical props [Cheng et al. 2015] have shown promise for bridging the gap between virtual and real scenes. Behavioral studies, such as [Steinicke et al. 2008; Nilsson et al. 2014; Bruder et al. 2015], have indicated the possibility of navigating a large-scale virtual environment while physically remaining in a reasonably small real space. However, those existing methods use procedurally generated content for feasibility studies, but do not provide general methods to map between a given pair of virtual and real environments. Since the virtual environment (e.g., a game or an architectural design) is usually orthogonal to the physical environment (which varies depending on the end users), a general method to bridge the two is crucial for real VR walkthrough.

We propose a VR method to maximize presence in a given virtual world and facilitate real walking in a given physical world. Within the scope of this paper we represent both worlds as planar floor plans, and use an HMD (Oculus DK2) attached to a notebook computer as the VR device to allow free navigation. Our method faithfully renders the virtual world inside the HMD but alters the camera projection for navigating the real environment, so that users can retain perceptual comfort while being guided to stay within the boundary walls and away from obstacles such as furniture. Figure 1 is an overview of our system and the user experience.

Our method consists of two key components: a planar map between virtual and real world floor plans, and a camera projection derived from the planar map and scene content. The planar map aims to preserve both angle and distance between the virtual and real worlds for visual and locomotive consistence. The camera rendering aims to preserve the virtual world appearance and the real world geometry, while guiding user navigation to avoid physical obstacles and vestibular discomfort.

Both planar maps [Poranne and Lipman 2014; Fu et al. 2015; Chen and Weber 2015] and projective rendering [McMillan 1997; Yang et al. 2011; Popescu et al. 2009] have been extensively studied in computer graphics. However, our VR method has different requirements from these prior art. Prior planar maps often require bijectivity to avoid folds, but our method does not require bijectivity as it looks for proper folding of the virtual world into the real one. We instead optimize for maps that can preserve angle and distance and can be efficiently computed for VR navigation. Prior projective rendering methods focus on speed and realism. In addition to that, our method also relies on the projection to properly guide user locomotion and hide perceptual discrepancies between the virtual and real worlds. We thus derive our camera projection according to the planar map and scene content to balance between visual realism, geometric consistency, and perceptual comfort.

Our method allows users to wear wireless HMDs to navigate virtual scenes via free locomotion in real worlds. We evaluate our system through a formative user study and applications in gaming, architecture walkthrough, and medical imaging.

In summary, the main contributions of this paper include:

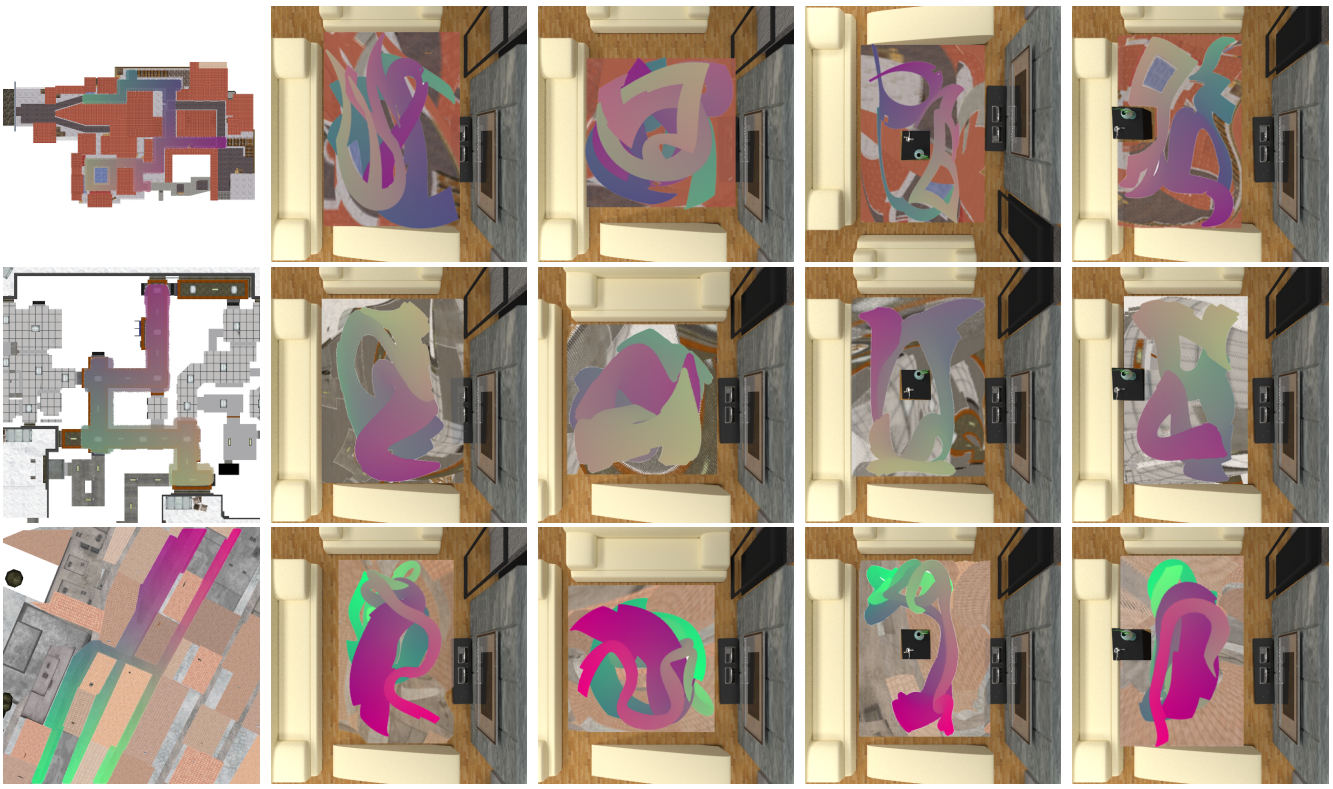
- An HMD-VR system that allows real walking in a given physical environment while perceiving a given virtual environment;
- A custom planar map that is globally surjective but locally injective between the virtual and physical floorplans to minimize angular and distal distortions for walkthroughs;
- Optimization methods to compute the aforementioned planar maps as two parts: a static forward map that minimizes angular and distal distortions while avoiding obstacles and boundaries, and a dynamic inverse map that guides natural locomotion and resolves local ambiguities;
- A rendering method that preserves the virtual world appearance while observes the physical world geometry to balance between visual fidelity and navigation comfort.

2 Previous Work

Immersive virtual environments There are various forms of immersive virtual environments. Some, such as rooms or cabins (CAVEs), offer semi-immersive experiences in which users can see virtual worlds projected on physical displays. Others, such as head mounted displays (HMDs), have more compact setup and fuller immersion than CAVEs, and have gained recent popularity due to improvement in hardware and software (see e.g., [Huang et al. 2015; Li et al. 2015]). However, users' immersive experiences depend on not only rendering/display performance but also interaction and navigation capabilities. HMDs can block perception of the surrounding real world with negative impacts on user interaction and navigation, such as hand motion [Jang et al. 2015], obstacle avoidance, and walking direction. Walking-in-place (WIP) techniques, such as omni-directional treadmills [Souman et al. 2008], robot tiles [Iwata et al. 2006], and motion carpets [Schwaiger et al. 2007], can reduce some of these issues, but have yet to gain popularity due to barriers in hardware and usability (see e.g., [Wong 2015]).

Real walking Studies have shown that real walking outperforms walking-in-place and other indirect means of VR navigation [Usoh et al. 1999]. However, real walking requires sufficiently large physical spaces, which are almost always in different (usually smaller) sizes and shapes from the corresponding virtual spaces (unless the latter are designed from the former, as in [Simeone et al. 2015]). Techniques such as physical props [Cheng et al. 2015], warped spaces [Suma et al. 2012; Vasylevska et al. 2013], and redirected walking [Razzaque et al. 2001; Maesen et al. 2013; Zmuda et al. 2013; Nescher et al. 2014] have been proposed to reconcile the virtual and physical worlds, and behavior studies have indicated that limited amounts of space distortion can be acceptable for VR navigation [Steinicke et al. 2008; Zhang and Kuhl 2013; Nilsson et al. 2014; Bruder et al. 2015]. However, existing methods are not general enough to map between a given pair of virtual and physical environments. Our paper aims to address this important problem.

Planar mapping Various planar mapping methods have been proposed to achieve application-specific goals such as minimiz-



(a) *virtual scene* 200 × 200 **(b)** *real scene* 60 × 100 **(c)** *real scene* 70 × 70 **(d)** 60 × 100 + *obstacle* **(e)** 60 × 100 *non-rectangular*

Figure 2: *Static mapping examples. (a): original input virtual scene overlaid with the user paths. (b) and (c): map to real spaces with different sizes and shapes. (d) and (e): map to real spaces with interior obstacles within and adjacent to the boundaries. (Scenes courtesy from top to bottom: Counter Strike Italy, Counter Strike office, and Venice.) Additional detailed information are in Section 6 and Figure 11.*

ing distortion and avoiding folding (see e.g., [Poranne and Lipman 2014; Fu et al. 2015; Chen and Weber 2015] and the references therein). Our system also relies on planar mapping, but has needs beyond existing methods. For example, sparse constrained deformation [Schüller et al. 2013] and convex SOCP optimization [Poranne and Lipman 2014] are not suitable for our problem, and we need local isometry for geodesics rather than global isometry for distance-preserving. Moreover, most traditional planar mapping and deformation applications are based on user manipulation. In our application, the output domain is pre-defined by the real space. We thus propose a custom planar mapping with application-specific objectives, constraints, and solvers.

Re-projective rendering Re-projective rendering has a long history in computer graphics, including image-based rendering [McMillan 1997], general camera models [Popescu et al. 2009], and shading reuse for acceleration [Yang et al. 2011]. Our system also uses re-projective rendering for HMDs, but faces a unique challenge: combining the appearance of the virtual world and the geometry of the physical world to strike the right balance between visual fidelity and navigation comfort. We thus propose a custom re-projective rendering method to address this challenge.

3 Method

Given the 2D floor plans for the virtual S_v and real S_r scenes, we first compute a static forward map $\mathbf{f}$ from S_v to S_r (Section 3.1). This map is surjective but not bijective in general when $S_v > S_r$, but should minimize both distance and angle distortion for VR walkthroughs. It should reach every point in both S_v and S_r, while keeping inside S_r and away from interior obstacles. Folding is introduced without tearing or breaking apart the virtual world.

At run time during user navigation, we compute a dynamic reverse map of $\mathbf{f}$ to determine the virtual location in S_v from the tracked user position in S_r (Section 3.2). This reverse map should be consistent with the forward map $\mathbf{f}$ while maintaining motion and perception consistency for the users.

Finally, we render the virtual scene into the HMD (Section 3.3). The rendering should have enough quality and speed, and fit the appearance of the virtual scene into the geometry of the real scene to balance between visual and motion fidelity.

3.1 Static Forward Mapping

In this step, we surjectivly map each virtual scene pixel $\mathbf{x} = (x, y) \in S_v$ to a real scene point $\mathbf{u} = (u, v) \in S_r$, where S_v and S_r represent 2D planar regions. Unlike most prior 2D planar mapping methods, our application does not require global bijectivity to allow proper folding of large virtual scenes into small real scenes. Instead, our map relies more on conformality and isometry to minimize angular and distal distortion during VR navigation. Figure 2 shows mapping examples with different inputs and outputs.

Inputs and outputs Both the virtual S_v and real S_r scenes are represented by planar 2D regions bounded by their external and internal boundaries (for domain shapes and obstacle regions, respectively). For computational purposes, we represent both spaces as polygonal shapes. In practice, those polygons can be extracted as convex/non-convex hulls from scanned data or design figures.

Representation Similar to prior meshless warping and planar mapping methods [Poranne and Lipman 2014; Chen and Weber 2015], our method also adopts a basis-function form to facilitate analytical computation of Jacobians and Hessians:

$$(u(x,y),\ v(x,y)) = \mathbf{u} = \mathbf{f}(\mathbf{x}) = \sum_{i=1}^{p} \mathbf{c}_i \mathbf{b}_i(\mathbf{x}) + \mathbf{T}\mathbf{x}, \quad (1)$$

where $\{\mathbf{b}_i\}$ are basis functions with weights $\{\mathbf{c}_i\}$, and $\mathbf{T}$ is an affine transformation matrix. We use Gaussians for $\mathbf{b}$, i.e.,

$$\mathbf{b}_i(\mathbf{x}) = e^{\frac{-|\mathbf{x}-\mathbf{x}_i|^2}{2s^2}}, \quad (2)$$

where $\mathbf{x}_i$ is the i-th basis center (blue points in Figure 3) and $\mathbf{x}$ is a sample point in S_v (green points in Figure 3). In our experiments, we perform stratified sampling with each stratum containing 0.025% pixels of S_v and set s as $5\times$ the average sample distance.

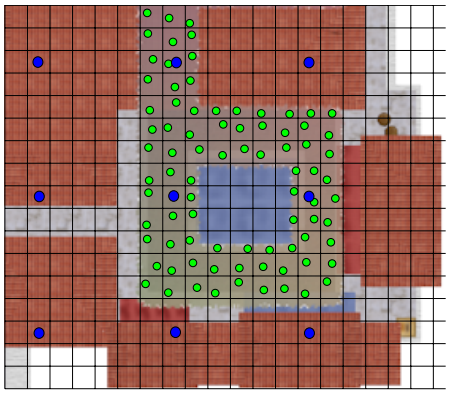


Figure 3: *Stratified sampling example for part of the Italy scene floor plan.*

Goal The general goal is to find proper $\mathbf{c} = \{\mathbf{c}_i\}$ and $\mathbf{T}$ so that the mapping $\mathbf{f}$ is as globally conformal and locally isometric as possible. In general, S_v is larger than S_r. To allow folding S_v into S_r, $\mathbf{f}$ should be surjective but not necessarily bijective which is the goal of most of other planar mapping methods. Such folding will also prevent $\mathbf{f}$ from being globally-isometric. Thus, we target our ideal mapping as globally conformal but *locally* isometric, via a collection of objectives and constraints described below.

Conformal objective As 2D mappings satisfy the Cauchy-Riemann function when it preserves angles [Lévy et al. 2002; Chen and Weber 2015], we define the conformal objective as:

$$\mathbf{E}_{conf}(\mathbf{c}) = \max_{\mathbf{x}} \left(\left(\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} \right)^2 + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)^2 \right). \quad (3)$$

We then minimize this energy (i.e., a minimax formulation) to maintain smooth energy distribution without extra regularization.

Distance constraint Unlike a global isometric mapping which requires $\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} = 1$, our mapping only needs to be locally isometric, which requires its Jacobians J to satisfy $J^T J = \mathbf{1}$, i.e.,

$$\begin{aligned} \left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2 &= 1 \\ \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 &= 1 \\ \frac{\partial u}{\partial x}\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\frac{\partial v}{\partial y} &= 0. \end{aligned} \quad (4)$$

Since local isometry maps geodesics to geodesics [Gray 1996], it suffices for VR locomotion. Note that minimizing $\mathbf{E}_{conf}$ in Equation (3) also addresses the last term in Equation (4), so we only need to focus on the first two terms.

Analogous to feature-aware texturing [Gal et al. 2006], different virtual regions may need different amounts of distance preservation in VR applications. For example, distances near region boundaries should be more strictly preserved as the users can examine the virtual walls close by, than when the users are in the middle of a large empty space. Due to this practical consideration, instead of putting the first two terms in Equation (4) as objective functions, we treat them as bounded constraints for more flexible control:

$$\begin{aligned} \alpha(\mathbf{x}) < \left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial x} \right)^2 &< \beta(\mathbf{x}) \\ \alpha(\mathbf{x}) < \left(\frac{\partial u}{\partial y} \right)^2 + \left(\frac{\partial v}{\partial y} \right)^2 &< \beta(\mathbf{x}), \end{aligned} \quad (5)$$

where $\alpha \in [0,1]$ and $\beta \in [1,+\infty)$ are stretching ranges for each virtual scene point $\mathbf{x}$. When both α and β equal to 1, the mapping is strictly locally isometric. However, for better conformality, we can relax the isometry into a range: the lower/higher the α/β value is, the more shrinking/stretching is allowed. There are three ways to set those parameters: constant values, user specification, or automatically computed via RANSAC line detection over S_v.

Exterior boundary constraint To keep all $\mathbf{u}$ inside the real space S_r, we construct the polygonal convex hull of S_r as a set of straight line functions $\{\mathbf{B}_i\}$, and add a series of linear constraints:

$$(\mathbf{B}_i \mathbf{u})^T (\mathbf{B}_i \mathbf{C}_r) > 0, \quad (6)$$

where $\mathbf{C}_r$ is the center of the physical space. The idea is to keep $\mathbf{u}$ and $\mathbf{C}_r$ on the same side of each $\mathbf{B}_i$ for testing point inclusion.

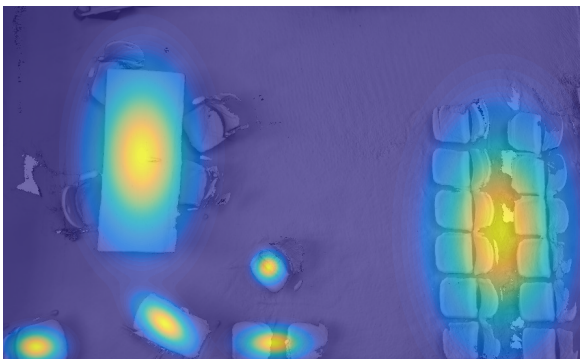


Figure 4: *Energy distribution of the obstacle barrier in Equation* (7) *over a reconstructed real indoor scene from [Choi et al. 2015]. Notice the higher energies in obstacle areas as indicated by brighter colors.*

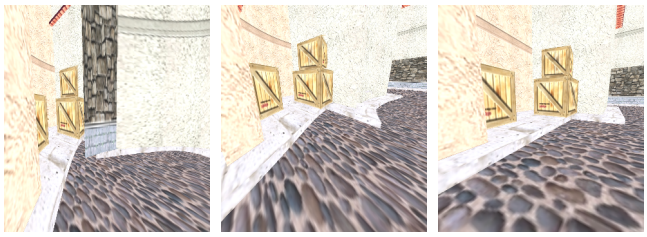
Interior obstacle barrier Preventing users from hitting interior obstacles can be formulated as the opposite of the point inclusion test in Equation (6). However, such formulation will require the calculation of directed areas or angles and solving a large quadratic instead of linear constraint system. For faster computation, we instead use a 2D Gaussian based barrier function for each interior object. For each object, we fit a minimal-covering 2D ellipse area $E(\mathbf{u}_c, E_w, E_h, \theta_c)$, where $\mathbf{u}_c$ is the center, E_w and E_h are width and height, θ_c is the rotation angle. Based on the scale of the ellipse, we define a Gaussian-based barrier:

$$E_b\left(E(\mathbf{u}_c, E_w, E_h, \theta_c), \mathbf{u}\right) = \exp\left(\frac{-1}{2\sigma^2} \left(\frac{u'^2}{E_w^2} + \frac{v'^2}{E_h^2} \right) \right), \quad (7)$$

where

$$\mathbf{u}' = \left(\mathbf{u} \begin{bmatrix} \cos\theta_c & \sin\theta_c \\ -\sin\theta_c & \cos\theta_c \end{bmatrix} - \mathbf{u}_c \right). \quad (8)$$

In our experiment, we set $\sigma^2 = 0.2$. Figure 4 depicts an example.



(a) *no local bijection* **(b)** *with local bijection* **(c)** *virtual view*

Figure 5: *Local bijection. Without local bijection, local fold-over in the static mapping may block the whole path with a wall (a). Adding the local bijection constraint can help prevent this artifact (b). (c) shows the original virtual scene for comparison.*

Local bijectivity Our mapping $\mathbf{f}$ allows global surjectivity to fold large S_v into small S_r. However, a local fold-over may produce visible artifacts, as exemplified in Figure 5. To prevent such fold-overs, we add a *local* bijectivity control, as described below.

From [Schüller et al. 2013], a mapping at a given point $\mathbf{x} \in S_v$ is locally bijective (i.e., no fold-overs) when it satisfies:

$$\det(\mathbf{J}_\mathbf{u}(\mathbf{x})) > 0. \quad (9)$$

Directly applying this constraint to all points in S_v can be computationally expensive. More efficient barrier functions and optimizers [Schüller et al. 2013] require sparse objective functions, whereas our objective function is dense. The method in [Poranne and Lipman 2014] can express Equation (9) as eigenvalues over all points, but such constraints cannot improve performance in our non-convex quadratic constraint problem.

To address this performance issue, we add local bijective constraints in a coarse-to-fine process. At the beginning, we partition S_v into a collection of cells (Figure 3). Then, during optimization, we add the following constraints to each sample point $\mathbf{x} \in S_v$:

$$\det\left(\mathbf{J}_\mathbf{u}(\mathbf{x})\right) = \frac{\partial x}{\partial u}\frac{\partial y}{\partial v} - \frac{\partial x}{\partial v}\frac{\partial y}{\partial u} > 0. \quad (10)$$

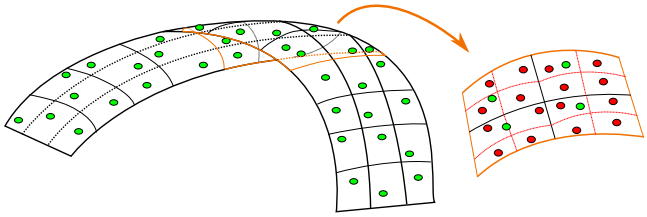


Figure 6: *Local bijection sampling. When fold-over is detected in the orange area, we restart the optimization, split all sample grids and add more samples. Distance constraints are then relaxed in the folding area.*

After convergence, if we find fold-over inside any cell, we iteratively split all cells into four smaller ones and add one more sample for each. We split all cells instead of only those in the fold-overs for faster convergence. Specifically, in our experiments we have found that up-sampling only the fold-over areas tends to push fold-overs to other areas with original sampling, which will require even more optimization rounds to fix than up-sampling all cells. Compared with active-set methods used in [Poranne and Lipman 2014], the coarse-to-fine process is more stable for a non-convex problem.

Relaxed distance constraint To facilitate local bijectivity, we relax the distance constraints in Equation (5) to encourage stretching over folding. Here, imagine the virtual domain is a plastic floor plan sheet that can be bent or fold, but never cut. Intuitively, bending will cause point-wise stretching but folding will not. Thus, to encourage bending over folding, we maintain an extra point set L from those samples in a folding area, i.e., the red points in Figure 6. We increase the upper limit of Equation (5) for all points in L to encourage stretching:

$$\begin{aligned} \left(\frac{\partial x}{\partial u}\right)^2 + \left(\frac{\partial y}{\partial u}\right)^2 < \beta(\mathbf{x}) \rightarrow \left(\frac{\partial x}{\partial u}\right)^2 + \left(\frac{\partial y}{\partial u}\right)^2 &< \lambda\beta(\mathbf{x}) \\ \left(\frac{\partial x}{\partial v}\right)^2 + \left(\frac{\partial y}{\partial v}\right)^2 < \beta(\mathbf{x}) \rightarrow \left(\frac{\partial x}{\partial v}\right)^2 + \left(\frac{\partial y}{\partial v}\right)^2 &< \lambda\beta(\mathbf{x}). \end{aligned} \quad (11)$$

We set $\lambda = 1.2$ in our experiments.

Solver The local isometric requirement in Equation (5) makes the terms quadratic and thus cannot be directly solved via the SOCP methods as in [Poranne and Lipman 2014; Chen and Weber 2015]. With the combined conformal objective and various constraints and requirements, the problem becomes a quadratically constrained quadratic programming (QCQP) in a minmax format. However, due to the dual-bounded constraints Equations (4) and (5), the constraints are not convex and thus not suitable for QCQP solvers.

To address this large, dense, and non-linear optimization problem, we adopt an interior-point method [Bonnans et al. 2006]. In order to match the solver format, we rewrite the conformal objective Equation (3) as follows:

$$\min z, s.t. \left(\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2 < z. \quad (12)$$

Combining all those constraints and barriers we discussed above, the final static mapping problem to be solved by interior-point method becomes:

$$\min z + w_o E_b(E, \mathbf{u}), \quad (13)$$

where w_o is the weight for obstacle barrier function, which we set as 600. We initialize $\{\mathbf{c}_i\}$ and $\mathbf{T}$ in Equation (1) as zeros and an identity matrix to satisfy Equations (5), (6) and (10) to (12).

3.2 Dynamic Inverse Mapping

The static map in Section 3.1 forwards positions from the virtual world S_v to the real world S_r. However, for VR walkthroughs we need the reverse map, from the current user position in S_r to S_v. This reverse map needs to deal with the fact that the forward map might not be bijective and thus there can be multiple solutions. In addition, it should also minimizes perceptual angle and distance distortion during navigation. Below, we describe how we compute this inverse map dynamically during user navigation.

Start Given the user positions $\mathbf{u}(t)$ and $\mathbf{u}(t+1)$ as well as orientations $\mathbf{U}(t)$ and $\mathbf{U}(t+1)$ tracked in the real world S_r at time steps t and $t+1$, and the corresponding virtual position $\mathbf{x}(t)$ and orientation $\mathbf{X}(t)$ at time t, our goal is to compute the corresponding virtual position $\mathbf{x}(t+1)$ and orientation $\mathbf{X}(t+1)$. Note that this is a path dependent process as $\mathbf{x}(t+1)$ and $\mathbf{X}(t+1)$ are computed from $\mathbf{x}(t)$, $\mathbf{X}(t)$, $\mathbf{u}(t+1)$, and $\mathbf{U}(t+1)$. We manually assign $\mathbf{x}(0)$ and $\mathbf{X}(0)$ for the initial virtual world position and orientation.

Direction update To compute $\mathbf{x}(t+1)$, we first compute the moving direction:

$$\hat{\delta}\mathbf{x}(t) = \frac{\mathbf{x}(t+1) - \mathbf{x}(t)}{\|\mathbf{x}(t+1) - \mathbf{x}(t)\|} \triangleq \begin{pmatrix} \hat{\delta}x \\ \hat{\delta}y \end{pmatrix}. \tag{14}$$

The virtual and real world directions are related by the Jacobians of their mapping:

$$\begin{pmatrix} \hat{\delta}x \\ \hat{\delta}y \end{pmatrix} = \begin{pmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{pmatrix} \begin{pmatrix} \hat{\delta}u \\ \hat{\delta}v \end{pmatrix}, \tag{15}$$

where

$$\begin{pmatrix} \hat{\delta}u \\ \hat{\delta}v \end{pmatrix} = \hat{\delta}\mathbf{u}(t) = \frac{\mathbf{u}(t+1) - \mathbf{u}(t)}{\|\mathbf{u}(t+1) - \mathbf{u}(t)\|} \tag{16}$$

is the real world direction. Thus, the goal is to find the Jacoabian of the reverse function of $\mathbf{f}$ in Equation (1):

$$\mathbf{J}_{\mathbf{u}}(\mathbf{x}) = \begin{bmatrix} \frac{\partial x}{\partial u} & \frac{\partial x}{\partial v} \\ \frac{\partial y}{\partial u} & \frac{\partial y}{\partial v} \end{bmatrix}. \tag{17}$$

Even though $\mathbf{f}$ might not be globally bijective, the local bijectivity (Section 3.1) satisfies the inverse function theorem [Rudin 1976] and allows us to compute the inverse Jacobian via:

$$\mathbf{J}_{\mathbf{u}}(\mathbf{x}) = \mathbf{J}_{\mathbf{x}}^{-1}(\mathbf{u}), \tag{18}$$

where $\mathbf{J}_{\mathbf{x}}(\mathbf{u})$ can be computed from the analytic function $\mathbf{f}$ at position $\mathbf{x}(t)$.

Position update We next compute the new virtual position $\mathbf{x}(t+1)$ based on the estimated direction $\hat{\delta}\mathbf{x}(t)$. We focus on the 2D x-y position, as the z/height value of $\mathbf{x}$ can be directly assigned from $\mathbf{u}$ after an initial correspondence. For computation purposes, we define $\mathbf{\Delta x}(t) = \mathbf{x}(t+1) - \mathbf{x}(t)$, and represent it in a polar coordinate system, i.e., $\mathbf{\Delta x}(t) = \mathbf{\Delta x}_t(d, \theta) = (d\cos(\theta), d\sin(\theta))$. The goal is to find optimized (d, θ) to minimize an energy function as follows.

The first energy term measures how close the actual direction is to the estimated direction $\hat{\delta}\mathbf{x}(t)$:

$$E_{dir}(\theta) = \left\| \theta - \arctan\left(\frac{\hat{\delta}y}{\hat{\delta}x}\right) \right\|^2. \tag{19}$$

The second term is to keep the virtual distance close to the real distance:

$$E_{dis}(d) = \|d - \mathbf{\Delta u}(t)\|^2. \tag{20}$$

The last term is to match the mapping function $\mathbf{f}$ in Equation (1):

$$E_{map}(d, \theta) = \|\mathbf{f}(\mathbf{x}(t) + \mathbf{\Delta x}(t)) - \mathbf{u}(t+1)\|^2. \tag{21}$$

We find $\mathbf{x}(t+1) = \mathbf{x}(t) + \mathbf{\Delta x}(t)$ to minimize

$$E_{rev} = E_{map} + \lambda_{dir} E_{dir} + \lambda_{dis} E_{dis}, \tag{22}$$

where λ_{dir} and λ_{dis} are relative weights. In our experiments, we set $\lambda_{dir} = 0.1$ and $\lambda_{dis} = 0.05$.

For fast convergence, we make the initial guess as:

$$\begin{aligned} \theta &= \arctan\left(\frac{\hat{\delta}y}{\hat{\delta}x}\right) \\ d &= \|\mathbf{\Delta u}(t)\|. \end{aligned} \tag{23}$$

Orientation update For rendering, we also need to compute virtual camera orientation $\mathbf{X}(t)$ from real camera orientation $\mathbf{U}(t)$, which is tracked by the HMD. We represent both orientations by their Euler angles:

$$\begin{aligned} \mathbf{U}(t) &= (yaw_{\mathbf{u}}(t), pitch_{\mathbf{u}}(t), roll_{\mathbf{u}}(t)) \\ \mathbf{X}(t) &= (yaw_{\mathbf{x}}(t), pitch_{\mathbf{x}}(t), roll_{\mathbf{x}}(t)). \end{aligned} \tag{24}$$

Since our planar map $\mathbf{f}$ has only x-y positions, we compute only $yaw_{\mathbf{x}}$ and simply copy $pitch_{\mathbf{x}}$ and $roll_{\mathbf{x}}$ from $pitch_{\mathbf{u}}$ and $roll_{\mathbf{u}}$:

$$\begin{aligned} pitch_{\mathbf{x}}(t) &= pitch_{\mathbf{u}}(t) \\ roll_{\mathbf{x}}(t) &= roll_{\mathbf{u}}(t). \end{aligned} \tag{25}$$

A straightforward way to compute $yaw_{\mathbf{x}}(t)$ is copying the optimized angle θ from Equation (22). However, empirically we found that this may cause nausea and dizziness. To explain this, the static mapping (and thus the estimated orientation correspondence) is non-linear. Consequently, when users rotate their heads with uniform speed, the corresponding virtual camera may rotate non-uniformly. We thus compute $yaw_{\mathbf{x}}$ as a combination of λ_a and $yaw_{\mathbf{u}}$ to balance between accuracy and consistency:

$$\begin{aligned} yaw_{\mathbf{x}}(t) &= \lambda_a \theta(t) + \lambda_c yaw_{\mathbf{u}}(t) \\ 1 &= \lambda_a + \lambda_c \end{aligned}, \tag{26}$$

where λ_a and λ_c are subjective parameters set via user evaluation, as discussed in Section 5.1.

3.3 Rendering

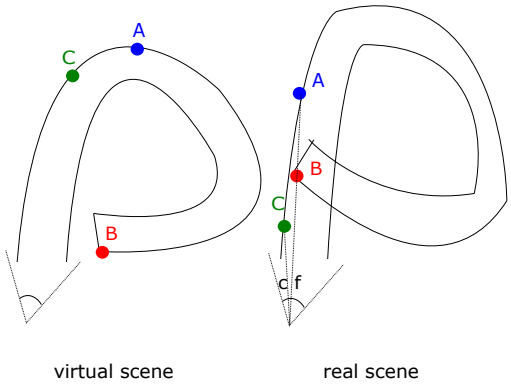
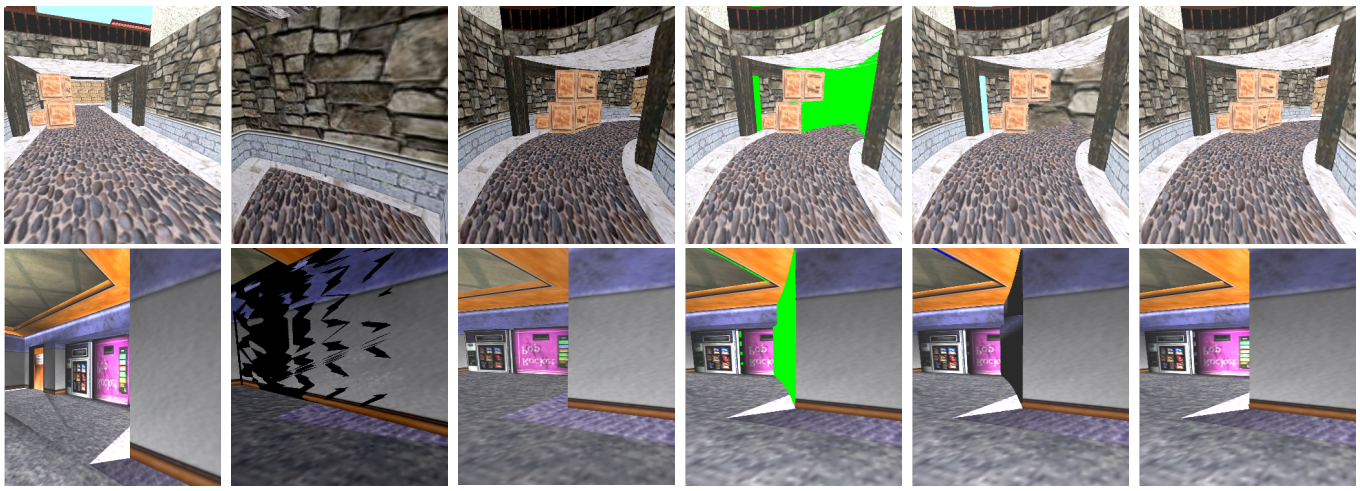


Figure 7: *Handling occlusion in rendering. The left and right are the virtual and real scenes. A camera is looking down a walled path, with three corresponding scene points shown in different colors. Point A is occluded in the virtual camera $\mathbb{C}_v$ but visible (i.e., dis-occluded) in the real camera $\mathbb{C}_r$. The goal is to decide how to render this dis-occluded pixel f in $\mathbb{C}_r$. Direct rendering the real scene G_r into $\mathbb{C}_r$ will pick B and cause highly occluded geometry in general (Figure 8b). The naive method improves this problem somehow (Figure 8e), but still picks point B for f because it is nearer the center of $\mathbb{C}_v$ than A. Thus, for each pixel in the dis-occluded area such as f, we search for its nearest non-occluded pixel in $\mathbb{C}_r$, which is c in this example. We then find the corresponding scene point C of pixel c, and assign its 3D distances to A and B in the virtual scene G_v as their depth values. Here, since C is closer to A than B, A will be picked for f. This strategy works because the local bijectivity of our mapping will prevent the incorrect solution B to be closer to C than the correct solution A.*

From the tracked/computed real/virtual user positions/orientations (Section 3.2), we have the real and virtual cameras $\mathbb{C}_r = \{\mathbf{u}(t), \mathbf{U}(t)\}$ and $\mathbb{C}_v = \{\mathbf{x}(t), \mathbf{X}(t)\}$ at each moment t. Our goal



(a) *virtual camera view* **(b)** *real camera view* **(c)** *local view of (b)* **(d)** *warp + dis-occlusion* **(e)** *naive solution* **(f)** *our method*

Figure 8: *Rendering examples. (a): virtual camera $\mathbb{C}_v$ rendering $\mathbf{I}_v$. (b): real camera $\mathbb{C}_r$ rendering $\mathbf{I}_r$; notice the ghosting objects blocking most of the views. (c): manually cropping only the relevant parts of the real scene to remove the ghosting objects in (b); this is for comparison only, not for practical use. (d): warping $\mathbf{I}_v$ towards $\mathbb{C}_r$ with dis-occlusion areas visualized in green color. (e): naive solution for the dis-occlusion in (d) using fragment depth values in the virtual camera $\mathbb{C}_v$; however, this is also not enough. (f): our method fitting the virtual image $\mathbf{I}_v$ into the real camera $\mathbb{C}_r$; notice the combination of rendering in (a) and geometry in (c).*

is to render the appearance of the virtual world into the environment of the real world, so that users can perceive the former while navigating in the latter. As visualized in Figures 7 and 8, direct mapping the virtual scene geometry into the real scene via $\mathbf{f}$ in Equation (1) can have overlaps and intersections and thus not suitable for rendering. The original virtual scene rendering, however, cannot be used for direct navigation as it would cause motion sickness due to incompatibility with the real scene. We thus fit the rendering of the virtual world into the geometry of the real world, as discussed below.

Algorithm We first render the virtual image $\mathbf{I}_v$ with virtual scene geometry G_v and virtual camera $\mathbb{C}_v$. We then initialize the real image $\mathbf{I}_r$ by mapping/warping [McMillan 1997] $\mathbf{I}_v$ into $\mathbb{C}_r$ via $\mathbf{f}$ to maintain visibility consistency with $\mathbf{I}_v$. Parts of $\mathbf{I}_r$ might remain uncovered due to dis-occlusion, for which we perform another rendering pass via the real scene geometry G_r and camera $\mathbb{C}_r$.

Note that it is important to warp the original virtual image $\mathbf{I}_v$ into $\mathbf{I}_r$ first, followed by rendering G_r only into the uncovered portions of $\mathbf{I}_r$. Otherwise, parts of G_r may erroneously occlude points in $\mathbf{I}_v$ that should remain visible in $\mathbf{I}_r$, as demonstrated in Figure 8b. For the dis-occluded parts in $\mathbf{I}_r$, simply rendering G_r via $\mathbb{C}_r$ will show artifacts in Figure 8b in these dis-occluded regions. A naive method that partially improves the outcome is to use depth values in the virtual camera $\mathbb{C}_v$ instead of the real camera $\mathbb{C}_r$. However, this is still not enough, as demonstrated in Figure 8e. Thus, we propose a more accurate method for dis-occlusion areas as follows. For each fragment (xy pixel + z depth) f_r visible in $\mathbf{I}_r$ but not in $\mathbf{I}_v$ (i.e., in dis-occlusion areas), we find the nearest (in terms of 2D xy coordinates) fragment f_v visible (i.e., non-occluded) in $\mathbf{I}_r$. We then assign the 3D Euclidean distance between f_v and f_r (in $\mathbb{C}_v$ space) as the depth value to f_r for rendering in $\mathbb{C}_r$. Figure 7 illustrates an example. Intuitively, this strategy works because the local bijection in Section 3.1 prevents overlapping among nearby virtual scene points. Notice the geometric similarity between the local real scene in Figure 8c and our result in Figure 8f.

Implementation Our rendering algorithm is amenable for GPU implementation. We first render the virtual image $\mathbf{I}_v$ (via polygon rasterization) of the virtual geometry G_v into virtual camera $\mathbb{C}_v$. For each pixel/fragment, we record the usual color and depth, as well as the optional G-buffer parameters [Crassin et al. 2015] for re-shading non-Lambertian materials. We then forward warp [McMillan 1997; Yang et al. 2011] $\mathbf{I}_v$ into $\mathbb{C}_r$ to initialize $\mathbf{I}_r$ (and optionally re-render non-Lambertian fragments), and record the coverage via a stencil buffer. If $\mathbf{I}_r$ is not entirely covered, we render G_r via $\mathbb{C}_r$ culled by the stencil buffer.

Because it is computationally complex to do reverse projection of $\mathbf{I}_v$ as $\mathbf{f}$ is a transcendental function, we store the 3D position of a real rendered pixel as a texture coordinate.

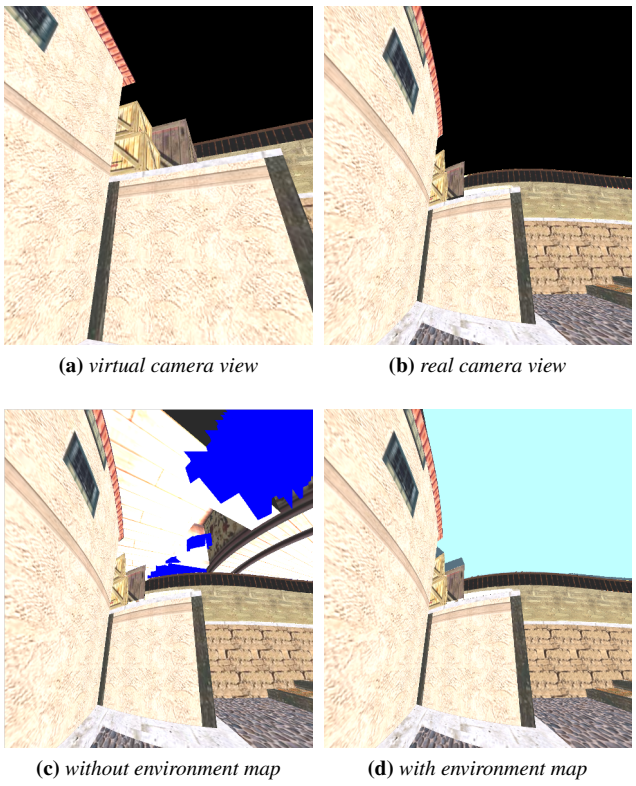
Similar to standard game level design, we surround the scene with an environment-map box to ensure all pixels in $\mathbf{I}_v$ are initially covered. Thus, all uncovered pixels in forward-warped $\mathbf{I}_r$ are caused by dis-occlusion. The environment map is important to ensure robust dis-occlusion to prevent far-away objects being mistakenly rendered into the background, as exemplified in Figure 9.

For more robust handling of larger/more-complex dis-occlusions, for each dis-occluded fragment f_r we find its four instead of just one nearest visible fragment, as described above. Specifically, we find the nearest non-occluded pixel $c_i, i \in [1, 4]$ visible (i.e., non-occluded) along each image domain direction ($\pm x$ and $\pm y$) in $\mathbf{I}_r$. Each of them has 2D distance d_i to f_r and corresponding scene position C_i. We then estimate the hypothetical matching point C (as in Figure 7) by:

$$C = \frac{\sum_{i=1}^{4} d_i^{-1} C_i}{\sum_{j=1}^{4} d_j^{-1}}. \tag{27}$$

To improve quality, we follow the standard IBR tricks [Debevec et al. 1996] of rendering multiple images with different camera parameters for G_r, and blend all fragments with the same screen position and depth (within numerical precision range) with weights proportional to their quality. For example, fragments with normal directions closer to the camera ray will have higher weights.

The camera $\mathbb{C}_r$ might go beyond the field of view of the original camera $\mathbb{C}_v$. We thus render $\mathbf{I}_v$ into a cylinder or 6 sides of a cube to ensure sufficient coverage. For efficiency, we estimate and render



(a) *virtual camera view* **(b)** *real camera view*

(c) *without environment map* **(d)** *with environment map*

Figure 9: *Importance of the environment map, a sky blue background in this example. (a): rendered from the virtual camera view. (b): rendered from the real camera view. (c): our method without using environment map; notice the ugly artifacts in the sky. (d): our method with environment map; the sky becomes clear, similar to (b).*

only the needed subset of $\mathbf{I}_v$ from $\mathbb{C}_v$ and $\mathbb{C}_r$, which includes the relevant cube faces stenciled with the necessary portions.

Discussion Conceptually, the mapping $\mathbf{f}$ in Equation (1) can have a meta parameter w that morphs the scene [Turk and O'Brien 2005]: $\mathbf{f}_0$ maps to the virtual scene $\mathbf{G}(0) = G_v$, $\mathbf{f}_1$ maps to the real scene $\mathbf{G}(1) = G_r$, and $\mathbf{f}_w$ with $w \in (0\ 1)$ maps to a scene in-between $\mathbf{G}(w)$. The w parameter trades off between visual fidelity to the virtual image $\mathbf{I}(0) = \mathbf{I}_v$ and motion fidelity to the real scene $\mathbf{G}(1) = G_r$. Figure 10 compares renderings with different mixing weights.

4 Implementation

Hardware As shown in Figure 1c, our experiments run in a $5m \times 10m$ room with 24 OptiTrack motion capture cameras installed along the floor and ceiling edges, and a rectangular desk in the middle of the room serving as an obstacle. We attach a rigid body OptiTrack tracker on top of an Oculus DK2 HMD, which is connected to and driven by a laptop with Intel i7-5700HQ CPU, NVIDIA GTX980m GPU, and solid-state drive. A light-weighted uninterruptible power supply in a backpack provides power to the HMD. During user movement, the position $\mathbf{u}$ is streamed from the motion capture system, and the orientation $\mathbf{U}$ is returned from Oculus HMD orientation sensor. We use a hybrid VR framework [Febretti et al. 2014] to drive all hardware and synchronize signals.

The Oculus HMD supports two types of rendering: a pair of identical binocular images or a pair of different monocular images with disparity. We choose the former for all images shown in the paper and the main video, and the latter for additional video segments.

Reachable area extraction To optimize space usage and computation speed, we only consider a subset of the virtual space S_v during offline optimization (Section 3.1) and online walkthrough (Section 3.2). This set should be reachable with a navigation starting from the initial virtual position. Thus, we perform a flood fill method over the virtual scene with root as $\mathbf{x}(0)$. All covered reachable regions serve as the input domain of the static mapping function $\mathbf{f}$.

5 Evaluation

We have evaluated subjective and objective aspects of our system pipeline via various experiments. We have recruited 7 participants with ages between 25 and 31. Among these participants, one participant has no prior experience with HMDs, six others have at least some basic knowledge or experiences with HMDs. One of the participants suffered from a light degree of vertigo.

5.1 Subjective Parameters

Design In addition to objective parameters which are empirically set based on scene properties, we also have to evaluate two subjective parameters, w in Section 3.3 and $\frac{\lambda_a}{\lambda_c}$ in Section 3.2. Due to limited human sensitivity to small parameter differences, we evaluate ranges instead of individual values. For w, we uniformly split its valid range $[0, 1]$ into 3 subranges. For $\frac{\lambda_a}{\lambda_c}$, values lower than 0.6 may cause significant mismatch between virtual and real camera orientations, triggering large dis-occlusions and rendering artifacts (Figure 14). We thus chose the $[0.6, 1]$ range and uniformly split it into 4 sub-ranges. For all experiments, each participant was tested with a random value sampled from each sub-range.

For w, we asked the participants to follow paths inside the virtual office scene (Figure 10) which are mapped to the physical lab scene with both boundaries and obstacles. Intuitively, larger w values will favor locomotion over visual fidelity, and w that is too small can cause motion sickness. We asked the participants to choose the most favorite w value to balance between visual and locomotion fidelity. Because people are more sensitive to locomotion discomfort, the participants were evaluated with high to low w values until feeling uncomfortable. Since $\frac{\lambda_a}{\lambda_c}$ is for orientation, we asked the participants to remain stationary, rotate their heads, and choose which values provide the most natural experience.

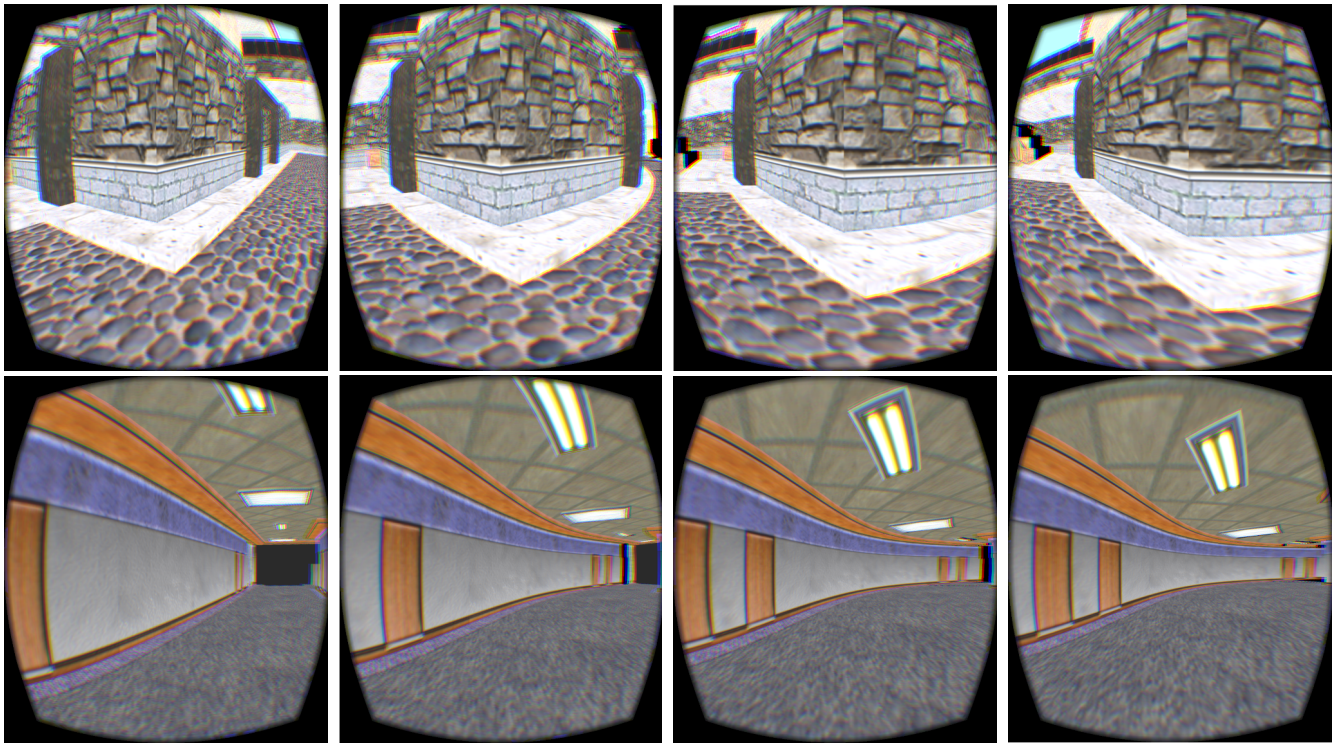
Result For w, 6 participants chose values within the 33% to 66% range as their preferences. One of those 6 users reported unbearable locomotion experience when w lies in 0 to 33%. The user with 3D vertigo reported light but bearable dizziness during this range and prefers the 66% to 100% range instead.

For $\frac{\lambda_a}{\lambda_c}$, all participants reported visual discomfort when the value is lower than 0.7. Among all participants, 1 chose $[0.7, 0.8]$ while the others chose $[0.8, 0.9]$. They reported that the proper values should be position-sensitive, i.e., the level of mismatch between real and virtual scenes caused by $\mathbf{f}$.

We observed that participants did not rotate their head much during the w experiments. As a result, they did not detect strong inconsistency when λ_c is low until they were asked to do the rotation experiment.

5.2 VR Usability

Design Similar to other VR applications [Bowman et al. 2002], we have conducted a formative user study to evaluate the usability



(a) $w = 0\%$ **(b)** $w = 33\%$ **(c)** $w = 66\%$ **(d)** $w = 100\%$

Figure 10: *Comparison of different virtual-real mixing weights w in rendering. (Scene courtesy of Counter Strike, Italy (top) and office (bottom).)*

of our VR system. We chose the task-based method to evaluate locomotion and the post-interview-based method to evaluate visual fidelity and motion sickness. All subjective parameters in this study session are taken from users' choices described in Section 5.1.

For this experiment, we chose the Counter Strike office scene (Figure 11b) as it has simple and uniform paths for task-based studies. We picked the purple vending machine at one end of the scene as the goal object. This particular object is unique in the scene and far from the entrance door, and thus suitable for a search task. Without prior knowledge of the scene, the participants started free locomotion to find the target. During the experiment, if a participant hits any real world objects or feels sick, we stopped the experiment. To evaluate locomotion fidelity, the participants can click a counter whenever they feel lost. To evaluate visual fidelity, the participants were given the original virtual scene to explore and grade for similarity at the end of the experiments. At the closing interview session, the participants shared their comments with us and graded their sickness and fatigue levels.

Result For the search task, 2 participants had a failure experience. One was caused by wall crossing; since our current pipeline does not implement collision detection, when users cross a wall our algorithm will get stuck by regarding all pixels as dis-occlusion. Another failure case was caused by the particular user passing-by the vending machine without noticing it until being explicitly prompted. No participant hit any real world objects.

For locomotion fidelity, no direction loss was reported.

For motion sickness, we have conducted the simulator sickness questionnaire (SSQ) [Kennedy et al. 1993] at the end of the experiments. Compared to prior literature, such as [Schild et al. 2012] for gaming environments, our overall numbers as shown in Table 1 are in a comfortable range. Specifically, one user reported bearable fatigue after the w experiment. Two users (one with VR experience and another without) reported dizziness right after the first experiment, but they recovered and felt comfortable for the remaining two experiments.

measure \ participants	1	2	3	4	5	6	7
nausea (out of 27)	5	8	6	1	0	0	0
oculomotor (out of 21)	5	6	4	1	1	1	0

Table 1: *SSQ results using the questionnaire from [Bouchard et al. 2007].*

For visual fidelity, one participant graded 90 (out of 100), five graded 80, and one graded 40. The reason for this latter low grade is that the participant is more sensitive to corner angles. Since our algorithm warps the virtual scene, global angle (not local conformality) might change significantly.

For additional interview comments, the participant without VR experience was excited about the walkthrough experience. Another participant expressed concerns about highly bended angles, which may cause users fatigue and discomfort after sustained usage.

5.3 Performance

Static forward mapping Since the static mapping is a preprocessing step, there is no need for real-time or interactive performance. However, as shown in Table 2, without our Gaussian-based obstacle barrier function, the optimization will be impractical for real spaces with obstacles.

Dynamic inverse mapping The inverse mapping part is usually very fast with speed mainly depends on the complexity of $\mathbf{f}$. For example, the office scene with the largest number of basis func-

tions (Table 3) has FPS ranging from 160 to 240 based on users' movement speed.

Rendering Table 3 provides performance measurements and scene statistics. The frame rate is between interactive (30 FPS) and real-time (60 FPS), and the rendering cost depends much more on the image resolution (1182 × 1461 per eye) than geometry details due to our use of image warping.

6 Applications

Our method can be applied for various VR applications, such as gaming/entertainment, architecture/design walkthrough, and medical imaging/visualization.

First person shooting games, such as Counter Strike, have been a main target for VR designers. We have applied our method to two different scenes in Counter Strike: the Italy scene for outdoor views and the office scene for indoor views, as shown in Figures 11a and 11b. The corresponding results in Figures 1 and 10 and the accompanying video show the promise of our method for navigating first person shooter games while moving in a real room using HMD.

Virtual architectural walkthroughs provide immersive experience for design and education. Figure 11c displays a portion of the Venice city 3D model, with our rendering result shown in Figure 12a. Using our system, users can virtually walk to every corner of the city while staying in their homes.

Modern virtual reality techniques have been introduced in medical imaging with clinical use. As a representative application, virtual colonoscopy is a non-invasive computer-aided 3D medical imaging technology to replace traditional optical colonoscopy [Hong et al. 1997]. To make radiologists feel being inside a human colon while having realistic navigation experience, we straighten a folded colon for 2D navigation Figure 11d. A sample rendering by our system can be seen in Figure 12b.

7 Limitations and Future Work

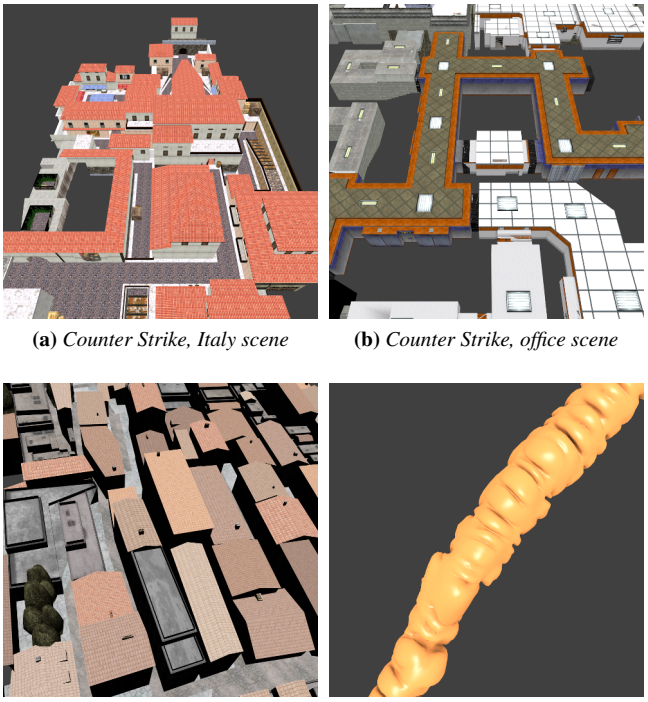
We have presented a method to support real walking within a given physical environment, while perceiving a given virtual world inside an HMD for VR applications. Our system focuses on the graphics aspects: 2D mapping and 3D rendering to balance between visual fidelity to the virtual world and locomotion comfort for the physical world. These graphics algorithms depend on but are largely orthogonal and complementary to a collection of perceptual and physiological parameters, for which we have performed a preliminary study but are definitely worth more thorough investigation.

real / virtual	no obstacle		60×100 with obstacle	
	60×100	70×70	barrier	constraint
Italy	406.8s	487.9s	928.7s	>3h
Office	282.6s	214.3s	444.7s	>3h
Venice	75.6s	61.6s	340.7s	>3h
Colonoscopy	22.4s	35.0s	203.0s	35m

Table 2: *Performance measurement for static mapping. Note the faster performance of barriers than of constraints for interior obstacles.*

info / scene	# vertices	# triangles	# basis	FPS
Italy	11018	21038	143	49.7
Office	11752	22249	165	50.4
Venice	6492	8431	154	44.2
Colonoscopy	5094	10108	48	40.5

Table 3: *Performance measurement for rendering.*



(a) *Counter Strike, Italy scene* **(b)** *Counter Strike, office scene*

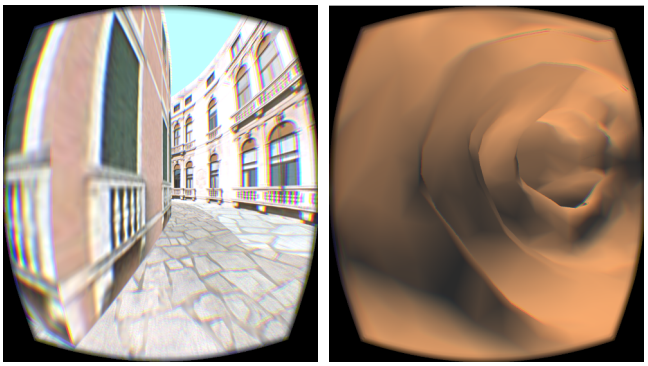
(c) *Venice scene* **(d)** *virtual colonoscopy*

Figure 11: *External views of all scenes used. (a) and (b) are from Counter Strike game level data, (c) is freely available from http://tf3dm.com/, and (d) is from our in-house anonymous patient database.*

Some specific evaluations and extensions including a larger scale user study as well as the incorporation of physics laws and tactile interaction [Cheng et al. 2015] are planned.

Our current method is designed for mapping 2D floor plans. It can be used to navigate scenes with height variations by tracking user head elevation and add it over the virtual scene height value for rendering. This can work if the users actively change their body postures such as jumping or crouching, but they might not sense the proper motion with plain walking up a ramp.

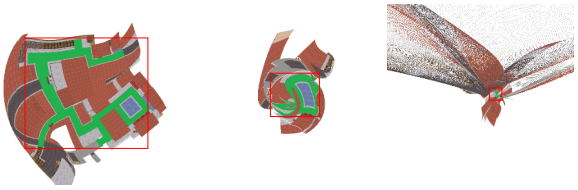
Although our static mapping solver can estimate a constraint-satisfying solution for any given pair of virtual and real spaces, the quality depends on their shape and size differences. When their sizes differ substantially, the mapped virtual space may be strongly



(a) *Venice scene* **(b)** *virtual colonoscopy*

Figure 12: *Sample HMD rendering for the Venice scene and virtual colonoscopy. The corresponding results for the Italy and office scenes can be found in Figures 1 and 10.*

distorted, reducing both visual and locomotion fidelity to the users. On the other hand, similar virtual and physical worlds can certainly improve the output quality of our system. Please refer to Figure 13 for comparisons.



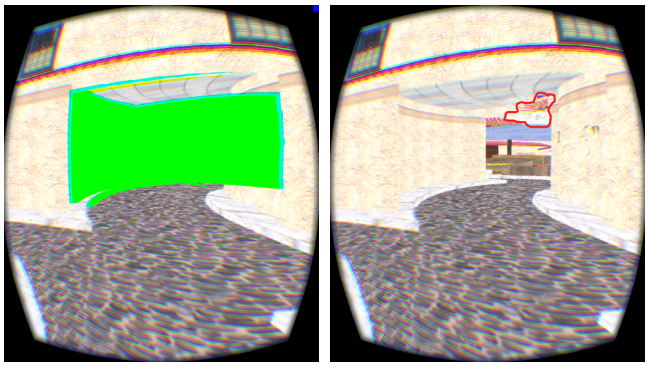
(a) *small* **(b)** *medium* **(c)** *large*

Figure 13: *Differences in size/shape between the virtual and real spaces. Our method can compute static mapping between a given pair of virtual and real scenes, but the quality depends on their size/shape differences. The red rectangles indicate the extents of the real spaces.*

Large open virtual spaces cannot be folded into a small real space without being noticed by the users. Space manipulation mechanisms such as teleportation might help and are worth further study. Fortunately, many virtual scenes in popular VR applications such as gaming, architecture, and medical imaging are highly occluded, and thus can benefit from our method.

In some cases, narrow virtual pathways can become even narrower than regular human footsteps after the static mapping. This, combined with motion capture accuracy limit, can cause our estimated $\mathbf{\Delta x}$ to cross walls and obstacles.

Our rendering algorithm cannot handle a large area that is entirely dis-occluded and also has very different depth from the surrounding regions, as shown in Figure 14. In particular, our dis-occlusion handling method, as depicted in Figure 7, requires parts of the nearby geometry to be visible for a dis-occluded fragment. Our experiments and analysis indicated that these dis-occlusion artifacts can be resolved by using more virtual images. Exactly how many additional virtual cameras are necessary, and where to place them, remain future research problems.



(a) *dis-occlusion area* **(b)** *artifacts*

Figure 14: *Limitation in our rendering algorithm. (a) shows an entire dis-occluded area in green color, which has vastly different depth from the surrounding rendered areas as shown in (b), which circles the rendering artifact in red.*

During the initial phase of the project, we have explored the use of non-pinhole cameras [Cui et al. 2010; Popescu et al. 2009] for rendering, but settled for traditional pinhole projection due to perceptual and performance reasons. In particular, pinhole projection is faster to render, and causes less motion disorientation for VR walkthroughs. However, a limited amount of non-linear projection is still worth further investigation to better balance between rendering and perception.

Acknowledgements

We would like to thank Ping Hu for helping us setup the motion capture system and straightening the virtual colonoscopy data, our user study participants for evaluating our prototype system, and the anonymous reviewers for their valuable suggestions. This work has been partially supported by National Science Foundation grants CNS-0959979, IIP1069147, CNS-1302246, Marcus Foundation grant, and Hong Kong RGC general research fund *Workflow Textures* (project number 17202415).

References

BONNANS, J. F., GILBERT, J. C., LEMARÉCHAL, C., AND SAGASTIZÁBAL, C. A. 2006. *Numerical Optimization: Theoretical and Practical Aspects (Universitext)*. Springer-Verlag.

BOUCHARD, S., ROBILLARD, G., AND RENAUD, P. 2007. Revising the factor structure of the simulator sickness questionnaire. *Annual Review of CyberTherapy and Telemedicine 5*, 128–137.

BOWMAN, D. A., GABBARD, J. L., AND HIX, D. 2002. A survey of usability evaluation in virtual environments: Classification and comparison of methods. *Presence: Teleoper. Virtual Environ. 11*, 4 (Aug.), 404–424.

BRUDER, G., LUBAS, P., AND STEINICKE, F. 2015. Cognitive resource demands of redirected walking. *IEEE Transactions on Visualization and Computer Graphics 21*, 4 (April), 539–544.

CHEN, R., AND WEBER, O. 2015. Bounded distortion harmonic mappings in the plane. *ACM Trans. Graph. 34*, 4 (July), 73:1–73:12.

CHENG, L.-P., ROUMEN, T., RANTZSCH, H., KÖHLER, S., SCHMIDT, P., KOVACS, R., JASPER, J., KEMPER, J., AND BAUDISCH, P. 2015. Turkdeck: Physical virtual reality based on people. In *UIST '15*, 417–426.

CHOI, S., ZHOU, Q.-Y., AND KOLTUN, V. 2015. Robust reconstruction of indoor scenes. In *CVPR '15*, 5556–5565.

CRASSIN, C., MCGUIRE, M., FATAHALIAN, K., AND LEFOHN, A. 2015. Aggregate G-buffer anti-aliasing. In *I3D '15*, 109–119.

CUI, J., ROSEN, P., POPESCU, V., AND HOFFMANN, C. 2010. A curved ray camera for handling occlusions through continuous multiperspective visualization. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (Nov.), 1235–1242.

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH '96*, 11–20.

FEBRETTI, A., NISHIMOTO, A., MATEEVITSI, V., RENAMBOT, L., JOHNSON, A., AND LEIGH, J. 2014. Omegalib: A multi-view application framework for hybrid reality display environments. In *Virtual Reality (VR), 2014 IEEE*, 9–14.

FU, X.-M., LIU, Y., AND GUO, B. 2015. Computing locally injective mappings by advanced mips. *ACM Trans. Graph. 34*, 4 (July), 71:1–71:12.

GAL, R., SORKINE, O., AND COHEN-OR, D. 2006. Feature-aware texturing. In *EGSR '06*, 297–303.

GRAY, A. 1996. *Modern Differential Geometry of Curves and Surfaces with Mathematica*, 1st ed. CRC Press, Inc.

HODGSON, E., BACHMANN, E., AND WALLER, D. 2008. Redirected walking to explore virtual environments: Assessing the

potential for spatial interference. *ACM Trans. Appl. Percept. 8*, 4 (Dec.), 22:1–22:22.

HONG, L., MURAKI, S., KAUFMAN, A., BARTZ, D., AND HE, T. 1997. Virtual voyage: Interactive navigation in the human colon. In *SIGGRAPH '97*, 27–34.

HUANG, F.-C., CHEN, K., AND WETZSTEIN, G. 2015. The light field stereoscope: Immersive computer graphics via factored near-eye light field displays with focus cues. *ACM Trans. Graph. 34*, 4 (July), 60:1–60:12.

IWATA, H., YANO, H., AND TOMIOKA, H. 2006. Powered shoes. In *SIGGRAPH '06 Emerging Technologies*.

JANG, Y., NOH, S.-T., CHANG, H. J., KIM, T.-K., AND WOO, W. 2015. 3d finger cape: Clicking action and position estimation under self-occlusions in egocentric viewpoint. *IEEE Transactions on Visualization and Computer Graphics 21*, 4 (April), 501–510.

KENNEDY, R. S., LANE, N. E., BERBAUM, K. S., AND LILIENTHAL, M. G. 1993. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The International Journal of Aviation Psychology 3*, 3, 203–220.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21*, 3 (July), 362–371.

LI, H., TRUTOIU, L., OLSZEWSKI, K., WEI, L., TRUTNA, T., HSIEH, P.-L., NICHOLLS, A., AND MA, C. 2015. Facial performance sensing head-mounted display. *ACM Trans. Graph. 34*, 4 (July), 47:1–47:9.

MAESEN, S., GOORTS, P., AND BEKAERT, P. 2013. Scalable optical tracking for navigating large virtual environments using spatially encoded markers. In *VRST '13*, 101–110.

MCMILLAN, JR., L. 1997. *An Image-based Approach to Three-dimensional Computer Graphics*. PhD thesis. UMI Order No. GAX97-30561.

NESCHER, T., HUANG, Y.-Y., AND KUNZ, A. 2014. Planning redirection techniques for optimal free walking experience using model predictive control. In *3DUI '14*, 111–118.

NILSSON, N., SERAFIN, S., AND NORDAHL, R. 2014. Establishing the range of perceptually natural visual walking speeds for virtual walking-in-place locomotion. *IEEE Transactions on Visualization and Computer Graphics 20*, 4 (April), 569–578.

POPESCU, V., ROSEN, P., AND ADAMO-VILLANI, N. 2009. The graph camera. *ACM Trans. Graph. 28*, 5 (Dec.), 158:1–158:8.

PORANNE, R., AND LIPMAN, Y. 2014. Provably good planar mappings. *ACM Trans. Graph. 33*, 4, 76:1–76:11.

RAZZAQUE, S., KOHN, Z., AND WHITTON, M. C. 2001. Redirected Walking. In *Eurographics 2001 - Short Presentations*, Eurographics Association.

RAZZAQUE, S., SWAPP, D., SLATER, M., WHITTON, M. C., AND STEED, A. 2002. Redirected walking in place. In *EGVE '02*, 123–130.

RUDIN, W. 1976. *Principles of mathematical analysis*, third ed. McGraw-Hill Book Co., New York. International Series in Pure and Applied Mathematics.

SCHILD, J., LAVIOLA, J., AND MASUCH, M. 2012. Understanding user experience in stereoscopic 3d games. In *CHI '12*, 89–98.

SCHÜLLER, C., KAVAN, L., PANOZZO, D., AND SORKINE-HORNUNG, O. 2013. Locally injective mappings. In *SGP '13*, 125–135.

SCHWAIGER, M., THÜMMEL, T., AND ULBRICH, H. 2007. Cyberwalk: Implementation of a ball bearing platform for humans. In *Human-Computer Interaction. Interaction Platforms and Techniques*. Springer, 926–935.

SIMEONE, A. L., VELLOSO, E., AND GELLERSEN, H. 2015. Substitutional reality: Using the physical environment to design virtual reality experiences. In *CHI '15*, 3307–3316.

SOUMAN, J. L., GIORDANO, P. R., SCHWAIGER, M., FRISSEN, I., THÜMMEL, T., ULBRICH, H., LUCA, A. D., BÜLTHOFF, H. H., AND ERNST, M. O. 2008. Cyberwalk: Enabling unconstrained omnidirectional walking through virtual environments. *ACM Trans. Appl. Percept. 8*, 4 (Dec.), 25:1–25:22.

STEINICKE, F., BRUDER, G., JERALD, J., FRENZ, H., AND LAPPE, M. 2008. Analyses of human sensitivity to redirected walking. In *VRST '08*, 149–156.

SUMA, E., LIPPS, Z., FINKELSTEIN, S., KRUM, D., AND BOLAS, M. 2012. Impossible spaces: Maximizing natural walking in virtual environments with self-overlapping architecture. *IEEE Transactions on Visualization and Computer Graphics 18*, 4 (April), 555–564.

TURK, G., AND O'BRIEN, J. F. 2005. Shape transformation using variational implicit functions. In *SIGGRAPH '05 Courses*.

USOH, M., ARTHUR, K., WHITTON, M. C., BASTOS, R., STEED, A., SLATER, M., AND BROOKS, JR., F. P. 1999. Walking $>$ walking-in-place $>$ flying, in virtual environments. In *SIGGRAPH '99*, 359–364.

VASYLEVSKA, K., KAUFMANN, H., BOLAS, M., AND SUMA, E. 2013. Flexible spaces: Dynamic layout generation for infinite walking in virtual environments. In *3DUI '13*, 39–42.

WITMER, B. G., AND SINGER, M. J. 1998. Measuring presence in virtual environments: A presence questionnaire. *Presence: Teleoper. Virtual Environ. 7*, 3 (June), 225–240.

WONG, R., 2015. Vr startups: Stop trying to make virtual reality treadmills a thing. http://mashable.com/2015/06/20/virtual-reality-treadmills/.

YANG, L., TSE, Y.-C., SANDER, P. V., LAWRENCE, J., NEHAB, D., HOPPE, H., AND WILKINS, C. L. 2011. Image-based bidirectional scene reprojection. *ACM Trans. Graph. 30*, 6 (Dec.), 150:1–150:10.

ZHANG, R., AND KUHL, S. A. 2013. Human sensitivity to dynamic rotation gains in head-mounted displays. In *SAP '13*, 71–74.

ZMUDA, M., WONSER, J., BACHMANN, E., AND HODGSON, E. 2013. Optimizing constrained-environment redirected walking instructions using search techniques. *IEEE Transactions on Visualization and Computer Graphics 19*, 11 (Nov), 1872–1884.