Feature Guided Path Redirection for VR Navigation

Antong Cao[1], Lili Wang *[1], Yi Liu[1], and Voicu Popescu[2]

[1]State Key Laboratory of Virtual Reality Technology and Systems,
Beijing Advanced Innovation Center for Biomedical Engineering, Beihang University, Beijing, China
Peng Cheng Laboratory
[2]Department of Computer Science, Purdue University, United States

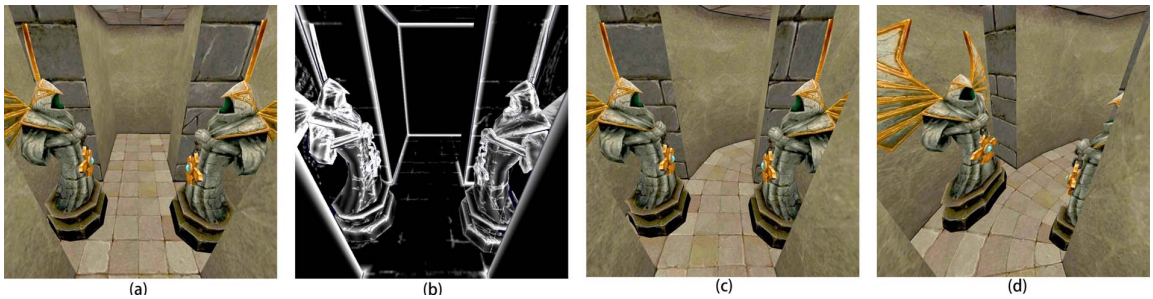

Figure 1: Feature guided path redirection. Our method analyses a conventional rendering (a) of the virtual environment (VE) to quantify visual detail (b), and then computes a deformation of the VE to make it fit in the available physical space, redirecting the user's path. This feature-guided path redirection minimizes distortions at regions with high visual detail (c). Path redirection without feature guidance (d) results in higher distortion, e.g. δ = 1.41 for (d) vs. δ = 1.16 for (c).

ABSTRACT

Path redirection for virtual reality (VR) navigation allows the user to explore a large virtual environment (VE) while the VR application is hosted in a limited physical space. Static mapping redirection methods deform the virtual scene to fit the physical space. The challenge is to deform the virtual scene in a reasonable way, making the distortions friendly to the user's visual perception. In this paper we propose a feature-guided path redirection method that finds and takes into account the visual features of 3D virtual scenes. In a first offline step, a collection of view-independent and view-dependent visual features of the VE are extracted and stored in a visual feature map. Then, in a second offline step, the navigation path is deformed to fit in the confines of the available physical space through a mass-spring system optimization, according to distortion sensitive factors derived from the visual feature map. Finally, a novel detail preserving rendering algorithm is employed to preserve the original visual detail as the user navigates the VE on the redirected path. We tested our method on several scenes, where our method showed a reduced VE 3D mesh distortion, when compared to the path redirection methods without feature guidance.

Index Terms: Virtual reality—Navigation—Path redirection—

1 INTRODUCTION

Exploring a Virtual Environment(VE) by walking gives users a sense of immersion and presence in the VE, as confirmed by many perceptual studies [17, 20, 28]. However, natural walking in large VEs is challenging due to real world obstacles such as walls or pieces of furniture, which do not have a counterpart in the VE. A possible solution is teleportation, which temporarily suspends the one-to-one mapping between the physical and virtual spaces, transferring the user from one location to another without commensurate actual user locomotion. Teleportation has the disadvantage of breaking visualization continuity, which leads to a loss of situational awareness, to a loss of the sense of immersion and presence in the VE, and even to simulator sickness. Another possible solution is path redirection, which modifies the user's path in the VE to abide by the physical constraints of the space hosting the VR application.

One approach of path redirection is dynamically redirected walking [10, 16, 19], which adds gains to the user's position and orientation in real time to steer the user clear of the boundaries and obstacles of the physical space. Experiments have shown that vision dominates people's perception of space. When vision conflicts with the input from other senses, the brain usually accepts the visual information and overrules the other senses. Therefore, in the process of dynamic redirected walking, changing the visual input provided to the user with slight scene rotations and translations gives the user the illusion of walking in a large space, avoiding interference from physical obstacles. The challenge of dynamically redirected walking is to devise manipulations of the user position and orientation that are simultaneously large enough to achieve obstacle avoidance, and small enough to remain imperceptible.

Another approach is static path redirection [5, 27], which takes descriptions of the available physical space and of the VE as input, and map the walkable sub-space of the VE onto the physical space as a pre-process. When the user navigates the VE, the scene in line of sight is rendered with distortions, in order to guide the user to change their path to conform to the physical space. The goal is to design the distortions to be as inconspicuous as possible, for the experience to be as close as possible to actually explore the original VE. Current static path redirection methods struggle with VEs that have rich geometry and texture detail, and with VEs that have regular features such as right angle corners or straight line segments, where the tampering with the VE is readily noticeable by the user, resulting in highly objectionable artifacts.

In this paper, we present a feature guided path redirection method that aims to reduce distortions at regions with abundant visual detail, in order to provide a high-quality visual exploration of the VE. Our

*e-mail: wanglily@buaa.edu.cn

2642-5254/20/$31.00 ©2020 IEEE
DOI 10.1109/VR46266.2020.00032

method relies on an offline analysis of the visual features of VE. Our method computes the distribution of view-independent, geometry features and of view-dependent, visual features, and uses the feature distribution to guide the mapping of the virtual space to the physical space. First, the view-independent and view-dependent features are found and stored in a visual feature map. Then, a mass-spring system is run offline based on distortion costs according to the visual feature map, and the equilibrium state computed by the optimization is used to define the virtual to physical space mapping. Finally, at runtime, a detail-preserving rendering algorithm is used to render the VE and to guide the user on the fly. Figure 1 shows that our method produces output frames that are comparable to the conventional rendering of the original VE, with smaller and less noticeable distortions than path redirection that does not take VE features into account.

In summary, our paper makes the following contributions: 1) A pipeline for path redirection based on the visual features of the VE. To the best of our knowledge, this is the first path redirection framework that takes into account visual features; 2) A visual feature map to store the distribution of geometry and appearance features of the VE in the context of VR navigation; 3) A mass-spring-based path optimization method to map the virtual path onto the physical space, which takes into account distortion-sensitive factors computed by the visual feature map; 4) A detail-preserving method for rendering 3D scenes to redirect users in VR navigation. 5) A user study that validates the benefits of our method objectively, based on distortion metrics, and subjectively, based on user perception.

2 Prior Work

In recent years, a variety of path redirection methods have been proposed, and they achieve the primary goal of fitting the VE into the smaller physical space. However, problems remain, as summarized, for example, by Nilsson et al. [13]. These problems include: (1) subtlety, (2) safety, (3) generalizability, and (4) unwanted side effects such as user disorientation and simulator sickness. Different methods focus on a subset of these problems. In addition to the overview of prior work given below, we also refer the reader to the recent survey of techniques for path redirection [11, 14, 21].

Dynamic redirection One natural approach to path redirection is to change the mapping from virtual to real dynamically. Dynamic redirection keeps users in the confines of the physical space through translation and rotation gains to the tracking data. Razzaque et al. [16] proposed a dynamic mapping approach that steers the user towards an intermediate guidance point (i.e. a *waypoint*), and then to the next waypoint after that. A large direction change occurs when a waypoint is reached. Razzaque subsequently proposed three general navigation strategies: Steer-to-Center, Steer-to-Orbit, and Steer-to-Multiple Targets [15]. More recently, Langbehn et al. [10] introduced a type of redirection method based on bending gains, which introduce discrepancies between physical and virtual paths that are concentrated in regions where both paths are bent. In addition, Ivleva [9] and Langbehn et al. [12] took advantage of the lesser attention level associated with user's eye blinks. When a blink was detected, the visual input was suppressed and a large direction manipulation was performed.

Sun et al. [26] suggested that the redirection operation can be performed during rapid eye movement, i.e. saccades, when the user simply changes view direction abruptly without focusing on the scene in the process. He also proposed a method to trigger a larger additional saccades for the system to exploit. Azmandian et al. [1] investigated redirection in the context of multiple VR users, with the primary concern of avoiding collisions between the user who share the same physical space.

The advantages of the dynamic redirection method are its simplicity and effectiveness at keeping the user within the confines of the physical space. However, due to the real-time change in gain, the redirection does not always go completely undetected by the user, who occasionally feels manipulated. Dynamic redirection is less conspicuous in large physical environments, where there is more physical distance over which to dilute the rotation and translation gains. The recent development of inside-looking-out trackers do remove the constraint of the limited tracked space, however, most applications will continue to be run in tight physical spaces, many times smaller than the VEs they host. Furthermore, triggering extra blinks and saccades does influence the user VR experience, and research remains to be conducted to estimate what is acceptable, when, and in what contexts.

Manipulation of virtual scenes. Suma et al. [22–24] proposed a different approach to redirection, based on altering the VE geometry when the user "is not looking". The method changes the geometry outside the current user view frustum, e.g. it changes the position of the doorways and corridors to modify the walking paths. Also relying on the manipulation of geometry, Suma et al. proposed the impossible spaces redirection technique [25]. Impossible spaces compresses a large interior scene into a smaller physical area by folding the floor plan manually. Vasylevska et al. [29] described flexible spaces, which overcomes the over-constrained problem of geometry folding by employing procedural layout generation, which "grows" the VE geometry into the available space. The method succeeds in some building interior environments, but it does not generalize to handle all VE geometry.

Static virtual to real path mapping. Another approach is to precompute a mapping of the virtual space to the real space by Sun et al. [27]. The advantage is that the offline processing affords the time needed to solve complex global optimizations. However, whereas the mapping has low overall distortion, the approach could cause large distortions in a specific region, that result in sizeable artifacts noticeable to the user when the user explores that specific region. Dong et al. [5] extended this research direction further, adopting a divide-and-conquer strategy that focused on mapping individual sub-parts of the walkable path, to avoid that any of these parts are distorted excessively. Our method falls into this category of static virtual to real path mapping. Unlike previous methods, our method considers the visual features of the virtual scene when the static mapping is generated, avoiding distortions where they matter most, i.e. in areas on which the user is likely to focus, and concentrating distortions in areas where they matter less, i.e. in areas on which the user is unlikely to focus. Our path redirection method makes use of the known visual features of the VE in two stages, first at a high level, in an offline process, for the entire VE, and then at a low level, at run time, for the part of the VE visible in each frame.

3 Feature Guided Path Redirection Method

We first give an overview of our pipeline.

3.1 Method overview

Given a 3D VE modeled with textured triangle meshes and a rectangular physical space, we provide a path redirection method that allows the user to explore the VE within the confines of the physical space. Our method takes into account VE features to minimize the negative impact of path redirection on the visual quality of the VE exploration. Our method proceeds with the following main steps.

- Step 1 (offline). Visual feature map construction (Sect. 3.2).
  a) Extract view-independent visual features of VE;
  b) Extract view dependent visual features of VE;
  c) Generate visual feature map.
- Step 2 (offline). Path redirection optimization (Sect. 3.3).
  a) Load mass-spring system based on feature map;

b) Compute mass-spring system equilibrium state that defines virtual to physical mapping.

- Step 3 (online). Detail-preserving rendering (Sect. 3.4).
  a) Fold the VE based on a Bezier transformation that relies on the virtual to physical mapping computed offline.
  b) Refine the VE folding to avoid large disortions at distortion sensitive regions close to the current user viewpoint.
  c) Render folded VE.

In a first offline step, we extract both view-dependent and view-independent visual features from VE geometry and textures, we combine them, and we bake them into a visual feature map. The feature map is a texture of VE surfaces that contains the distribution of the visual features extracted (Sect. 3.2). In a second offline step, a mapping between the VE walkable subspace and the physical space is computed with a mass-spring optimization. The masses are placed at the vertices of the walkable virtual space, and the springs are loaded with forces based on the feature map. The equilibrium state defines the virtual to physical mapping (Sect. 3.3). Then, at run time, the VE is rendered with the mapping, which is optimized for the current view, to abide by the physical space constraints while minimizing visual distortion (Sect. 3.4).

3.2 Visual feature map construction

The goal of our path redirection method is to make the distortion of the 3D scene more acceptable, which means: 1) local distortion should be correlated to VE visual detail, i.e. if a region has many details to which the user is likely to pay attention, the distortion for this region should be small, while if the region has no details, the local distortion can be larger; 2) the overall distortion of the VE should be as small as possible. The first step towards satisfying these requirements is to find the VE features on which users are likely to focus, such as edges and complex objects and textures.

We introduce the *visual feature map*, which is a texture atlas that stores the concentration of visual features in the VE. The higher the value of a visual feature map texel, the richer in features the corresponding VE surface patch. Similar to a lightmap, the visual feature map is a surface texture that stores pre-computed information to be conveniently reused at run time. The visual feature map is constructed by extracting the visual features, and then by baking them into the texture atlas, with the steps outlined in Alg.1.

The algorithm takes as input the VE and the possible user paths W, and it produces the visual feature map A. The algorithm also takes as input a Gaussian kernel G by which the atlas is denoised. At line 1, the visual feature map is defined as a texture atlas A that covers the entire VE. Every texel a of A stores the visual feature intensity f, the geometric detail g, the indices $\{v_0, v_1, v_2\}$ of the vertices of the VE triangle sampled by a, and the barycentric coordinates $\{\alpha, \beta, \gamma\}$ that define a within its VE triangle. This initialization computes $\{v_0, v_1, v_2\}$ and $\{\alpha, \beta, \gamma\}$, and sets f and g to 0.

At lines 2-3, the algorithm computes the geometric detail at each vertex of the VE. The geometric detail is a view independent visual feature of the VE. The geometric detail $v.g$ for a vertex v is computed as the average normal gradient over the vertices v_i neighboring v:

$$v.g = \frac{1}{n}\sum_{i=1}^{n}\frac{arccos(v_i.n \cdot v.n)}{\|v_i.xyz - v.xyz\|} \quad (1)$$

n denotes the normal and xyz denotes the position of the vertex. A vertex neighbors another vertex if there is a triangle to which they both belong. Fig. 2b illustrates the geometric detail computed by our algorithm.

Algorithm 1: Feature map computation

Input: VE geometry + texture, walkable subspace W of VE, Gaussian kernel G
Output: visual feature map atlas A

1 $A(f, g, v_0, v_1, v_2, \alpha, \beta, \gamma)$ = TextureAtlas(VE)
2 **for** *each vertex v in VE* **do**
3 $v.g$ = GeometricDetail(VE, v)
4 **for** *each texel a in A* **do**
5 $a.g = a.v_0.g * a.\alpha + a.v_1.g * a.\beta + a.v_2.g * a.\gamma$
6 Sample W with set of viewpoints V
7 **for** *each viewpoint v in V* **do**
8 (CB, ZB) = Render(VE, v)
9 S = Salience(CB, ZB)
10 E = Edges(CB, ZB)
11 L = Lines(E)
12 C = Corners(L, ZB)
13 **for** *each texel a in A* **do**
14 $P = a$.Unproject(); p = Project(P, v)
15 **if** *NotVisible(p, v, ZB)* **then**
16 **continue**
17 f_v=Combine($S_p, E_p, L_p, C_p, a.g$)
18 $a.f = (f_v > a.f)?f_v : a.f$
19 **return** $A \otimes G$

At lines 4-5, the algorithm sets the geometric detail for each texel a of the atlas by barycentric interpolation of the geometric detail values at the vertices of the triangle of a. At line 6, the algorithm defines a set of viewpoints V over the set of possible user viewpoints W, which will be used to estimate the view-dependent visual features.

For each viewpoint v in V, the view-dependent visual features are computed first (lines 8-12), and then they are baked into the atlas (lines 13-18). The view-dependent visual features are computed based on the color and depth buffers CB and ZB obtained with a conventional rendering of the VE from v (line 8). A salience map S that encodes how much a region stands out from its neighbors is computed with a prior art method [7], see Fig. 2c. An edge map E is computed with a Canny edge detector [3], which is run on both the color and depth channels (Fig. 2d). The edge map is used to compute a line map L with a Hough transform [8]. The line map is further processed to detect right-angle corners by finding line intersections and computing the angle between the intersecting lines in 3D. The line and corners map is visualized in Fig. 2e.

The view-dependent visual feature maps S, E, L, and C are computed they are transferred to the atlas A. For each texel a of A, the corresponding VE surface 3D point P is computed by unprojection (line 14), using the triangle vertices $a.\{v_0, v_1, v_2\}$ and the barycentric coordinates $a.\{\alpha, \beta, \gamma\}$; then P is projected with viewpoint v to visual feature map pixel p (line 14). If the texel is visible from v (lines 15-16), the visual feature intensity f_v of a is computed (line 17) by combining the visual feature map values at p and the geometric detail $a.g$ at a, as $f_v = (w_1 S_p + w_2)(w_3 E_p + w_4 L_p + w_5 C_p) + w_6 a.g$, where w_i are constant weights with values between 0 and 1. We use $w_1 = 0.6, w_2 = 0.4, w_3 = 0.7, w_4 = 0.4, w_5 = 0.4, w_6 = 0.4$ for all of our scenes. The edge, line, and corner visual features are only taken into account for salient texels, and their combined value is aggregated with geometric complexity. If the resulting feature intensity f_v is greater than what the atlas already stores at a, the atlas is updated (line 18). (The atlas feature intensity channel is initialized to 0, not shown in Alg.1 for conciseness.) Once all viewpoints are processed, the atlas is convolved with a 5×5 Gaussian kernel G to filter noise (line 19), and returned.

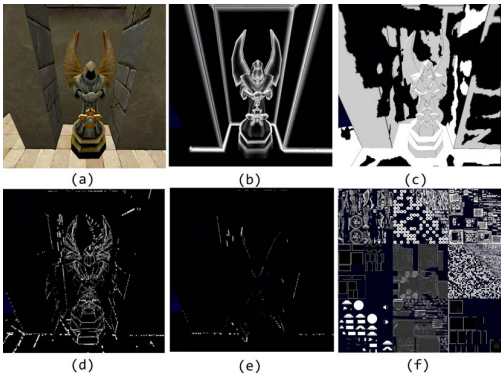

Figure 2: Conventional image (a), geometric detail (b), salience (c), edges (d), lines and corners (e), and resulting visual feature map (f).

3.3 Path redirection optimization

The visual feature map is used to compute a path redirection that folds the VE into the available physical space, with a mass-spring system optimization, according to Alg.2. The input is the visual feature map A, a 2D navigation mesh M modeling the walkable space of the VE, and a 2D rectangular bounding box of the available physical space P. M is constructed manually as a quad mesh over the floor plan of the walkable space. The output is the folded mesh M', which gives the mapping from virtual to physical.

Algorithm 2: Path redirection

Input: Visual feature map A, navigation mesh M of the VE, and physical space P
Output: Folded mesh M' that fits in P

1 S = MassSpringSystem(M, A)
2 **repeat for** *each mass* m_i *in* S **do**
3 **for** *each neighboring mass* m_j *of* m_i *in* S **do**
4 T_i += Tension(m_i, m_j)
5 **for** *each neighboring mass* m_k *of* m_i *in* S, $k > j$ **do**
6 (F_j, F_i, F_k) = AntiDeformation(m_j, m_i, m_k)
7 (D_j, D_i, D_k) += (F_j, F_i, F_k)
8 B_i = BoundaryConformance(m_i, P)
9 **for** *each mass* m_i *in* S **do**
10 m_i = UpdatePosition($T_i + D_i + B_i$, m_i)
11 **until** Equilibrium(S) and $S \subset P$
12 **return** M' = *Mesh(S)*

A mass-spring system is initialized using the navigation mesh M and the visual feature map A (line 1). One mass is placed at the middle of each shared edge of M, and a spring is defined for each pair of masses that belong to the same quadrilateral of M. A mass is connected with at most four springs, one for each neighboring mass. All mass particles have the same mass value, and all springs have the same stiffness. Initially, all springs are undeformed.

In addition to its position and mass, a particle m_i also has a property that defines the local sensitivity to distortion, which is constant, and it is computed using A. We first render the VE to a cubemap centered at m_i. Then an initial distortion sensitivity S_i is computed for m_i by averaging the $intensity \times depth$ over all texels in the cubemap. Once S_i is computed for all masses, it is discretized to one of four levels L_i, based on the normal distribution, as follows,

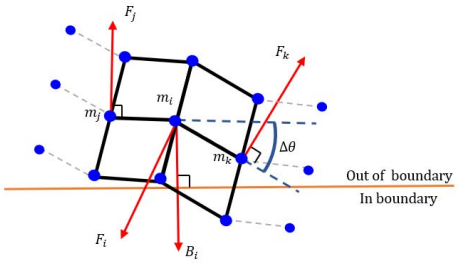

Figure 3: The blue dots represent masses and the black lines represent springs. The forces F_i, F_j, F_k oppose the change of the angle $\Delta\theta$. Mass m_i, which is outside the boundary of the physical space, is pushed by force B_i towards the boundary.

$$L_i = \begin{cases} 0, S_i < \mu - \sigma \\ 1, \mu - \sigma < S_i < \mu \\ 2, \mu < S_i < \mu + \sigma \\ 3, S_i > \mu + \sigma \end{cases} \quad (2)$$

where μ is the mean and σ is the std. dev. of S_i over all masses.

The algorithm simulates the mass-spring system dynamics iteratively until equilibrium is reached *and* all masses are inside the physical space P (lines 2-11). Each iteration first computes the forces acting upon each mass (lines 3-8). Three types of forces act upon each mass m_i. One type is spring tension force T_i, which is computed by summing up to four tensions, one for each of the springs (m_i, m_j) connected to m_i (line 4). A second type is a deformation resisting force D_i, which, for masses with high distortion sensitivity, opposes deformation. D_i is computed by adding up to six deformation resisting forces F_i, one for each pair (m_j, m_k) of neighboring masses of m_i (lines 5-7); F_i opposes the change of angle (m_j, m_i, m_k); this process also computes deformation resisting forces F_j and F_k which act on m_j and m_k, and are accumulated. The third type of force is a boundary conforming force B_i that acts on m_i only if it is outside the boundary of P, and pushes it towards P (line 8).

Spring tension T_{ij} is given by Hooke's law

$$T_{ij} = k\Delta x, \quad (3)$$

where k is the spring coefficient and Δx is the spring deformation.

The deformation resistant forces (F_i, F_j, F_k) (line 6), act upon m_i, m_j and m_k to resist the change of the angle between springs (m_j, m_i) and (m_i, m_k). In Fig. 3, the springs (m_j, m_i) and (m_i, m_k) were initially aligned, and they now form an angle $\Delta\theta$. Forces (F_i, F_j, F_k) aim to reduce $\Delta\theta$. F_j and F_k are perpendicular to springs (m_j, m_i) and (m_i, m_k), respectively. The magnitudes of F_j and F_k are

$$F_j = F_k = \Delta\theta * k * 2^{L_i * a}, \quad (4)$$

where L_i is the discrete distortion sensitivity level given by Equation 2. Constant a helps modulate the range of sensitivities. F_i is a reaction force equal and opposite to the resultant of F_j and F_k:

$$F_i = -(F_j + F_k) \quad (5)$$

The tension and distortion resistant forces prevent large and abrupt deformations of the path. A third type of force is needed to advocate for the folding of the VE to fit inside the boundary P of the physical

space. This force B_i is proportional to the distance between m_i and its closest point on the boundary P. The direction of B_i is perpendicular to and towards P (Fig. 3).

Once the forces are computed, the position of each mass is updated (lines 9-10), using the classic Newton equations. The folded mesh M' is recovered from the distorted mass-spring system at equilibrium (line 11). We run Alg.2 multiple times, initializing the mass-spring system with different stiffness coefficients at line 1, and we select the equilibrium state for which the angles between connected springs have changed the least from the initial state.

3.4 Detail-preserving rendering for path redirection

The computation of the deformation M' of M is a first, high level, step towards achieving a path redirection that meets the physical space limitations while avoiding distortions in feature-rich areas of the VE. This first step is performed off-line and it acts on the path graph, achieving a global deformation of the VE. A second, low level, step for avoiding these unwanted distortions is taken at run time, for each frame. This second step acts locally, on the geometry currently seen by the user. This second step removes some of the distortion introduced by the first step, based on the now known user view. The second step ensures that the geometry of a feature-rich region currently seen by the user is rendered closer to its original version, using a rigid body transformation that preserves the original geometry. The combination of these two global and local steps achieve greater output image fidelity in feature-rich regions (Fig. 4). The second step proceeds according to Alg.3.

Figure 4: Conventional rendering for ground truth (left), rendering using our method with both the global and local detail-preserving steps (middle), and rendering using our method with only the global step (right). The local step alleviates the stretching of the statue.

Algorithm 3: Detail-preserving rendering

Input: VE geometric model, feature atlas A, current user viewpoint v, navigation mesh M, deformed to M'
Output: output frame FB

1 $TB.r$ = RenderTopView(VE, A, v)
2 **for** *each pixel p in* TB **do**
3 $\quad p.gb$ = LocalMaximum($p, TB.r$)
4 $TB.r$ = NormalizeAndClamp($TB.r, R_{min}, R_{max}$)
5 FB = Render(VE, v, TB, **VertexProgram**)
6 **return** FB
7 Vertex **VertexProgram**(Vertex P)
8 $\quad p$ = TopViewProject(P, v)
9 $\quad p_{max} = TB[p].gb$
10 $\quad q = p_{max} + \|(p - p_{max})/\|p - p_{max}\| * \varepsilon$
11 $\quad (p', p'_{max}, q')$ = BezierTransform($(p, p_{max}, q), M, M'$)
12 $\quad W$ = ExtractRigidBodyTransformation(p_{max}, p'_{max}, q, q')
13 $\quad p_r = W \times p$; p_r = Clamp(p_r, p', p'_{max})
14 $\quad p' = p' + TB[p].r * (p_r - p')$
15 $\quad P'$ = TopViewUnproject($p', P'.y$)
16 **return** $MVP \times P'$

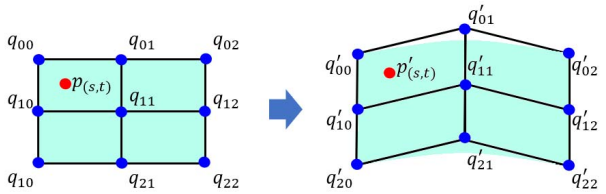

Figure 5: Bezier mapping of point p to p'. We construct in the neighborhood of p a quadratic Bezier surface with nine control points P_{ij}, derived from the undistorted navigation mesh; then we compute parameters (s,t); finally, we use (s,t) to sample the corresponding Bezier surface of the folded navigation mesh to obtain p'.

The region of the VE in the proximity of the current user viewpoint v is rendered orthographically from above, textured with the atlas A, into the red channel of an auxiliary color buffer TB (line 1). This orthographic projection switches from 3D to the 2D domain where the path redirection M to M' was defined, setting the stage for using it to compute the deformation of each vertex.

Then, for each pixel p of TB we find the pixel that has the highest red channel intensity in a square neighborhood of p (lines 2-3). The coordinates of this local maximum pixel are stored in the green and blue channels of p. Then the red channel of TB is normalized and clamped to a subinterval of feature intensities $[R_{min}, R_{max}]$, i.e. 0 for R_{min} and below and 1 for R_{max} and above (line 4). R_{min} and R_{max} are constants selected once per VE, which allow increasing the sensitivity with mid-range feature density. Then the output frame FB is rendered with the vertex program given in lines 7-16.

The vertex program starts by projecting the input vertex P with the top view to p (line 8). The local maximum of p is looked up in the green and blue channels of the TB pixel at p (line 9). A 2D point q is defined inside segment (p_{max}, p), ε close to p_{max}. The three 2D points p, p_{max}, and q are transformed using a Bezier mapping computed using the navigation mesh M and its deformation M' (line 11). Bezier mapping is a classical geometric deformation tool [4], except that we do it in 3D and not in 2D, as illustrated in Fig. 5. Then, the original and Bezier transformed points p_{max}, q and p'_{max}, q' are used to define a 2D rigid body transformation W (line 12), which is applied to p to obtain p_r (line 13). p_r is clamped with

$$p_r = \begin{cases} (p' - p'_{max}) * S_{max} + p'_{max}, & \frac{\|p'_{max} - p_r\|}{\|p'_{max} - p'\|} > S_{max} \\ p_r, & \frac{\|p'_{max} - p_r\|}{\|p'_{max} - p'\|} \leq S_{max} \end{cases} \quad (6)$$

where the maximum scaling factor S_{max} is selected once per VE.

Finally, the vertex is adjusted one more time by pushing p' towards p_r a fractional amount given by the feature density at the vertex $TB[p]$ (line 14). The final position of the vertex is computed by restoring the y coordinate (line 15). The vertex program returns the distorted vertex P' transformed and projected by multiplication with the usual modelview and projection matrix MVP (line 16).

4 Results and Discussion

We have tested our method on five scenes: Maze (11m × 15m, S1), Apartment (13m × 32m, S2), Office (14m × 16m, S3), Passage (9.6m × 9.6m, S4), and Corridor (17m × 18m, S5). We have conducted a user study using an HTC Vive system which has a tracked head-mounted display (HMD), an external tracker, and a wireless hand-held controller. The HMD is tethered to a desktop PC (Intel i7 processor, 16GB RAM, and NVIDIA 1080-ti graphics card). The size of the tracked physical space is 4m × 4m. The VE's

Table 1: Path distortion with and w/o feature guidance, for the entire path, and for the part of the path with high detail.

Scene	$\Delta\theta^\circ$ (Avg/Max) w/ FG	$\Delta\theta^\circ$ (Avg/Max) w/o FG	Red. of avg	$\Delta\theta_{high\_detail}{}^\circ$ (Avg/Max) w/ FG	$\Delta\theta_{high\_detail}{}^\circ$ (Avg/Max) w/o FG	Red. of avg
S1	13° / 79°	15° / 56°	15%	13° / 28°	17° / 32°	23%
S2	22° / 76°	26° / 85°	18%	23° / 49°	30° / 82°	28%
S3	13° / 57°	18° / 55°	28%	18° / 38°	21° / 49°	14%

are folded with our approach to fit in the physical space and the user navigates the environment naturally, by walking.

4.1 Path distortion

We investigated the amount of distortion introduced by our method with an objective path distortion metric computed as the average distortion angle $\Delta\theta$ (Equation 4) over all springs in the spring-mass system. Table 1 shows the average and maximum path distortions for several scenes, with and without feature guidance. Average distortion is quantified both over the entire path, as well as just over the parts of the VE with high feature density, i.e. the top 50% densities in the feature map. Feature guidance is disabled by setting all feature density values (L_i in Equation 2) to 0. Our method yields smaller average distortions at the cost of occasionally larger distortions when the entire VE is considered. In the high feature areas, both maximum and average path distortion values are reduced with our feature guided method.

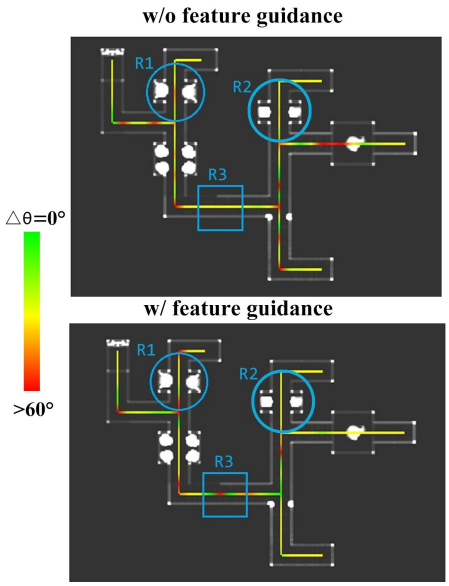

Figure 6: Path distortion visualization. Regions $R1$ and $R2$ have high feature density, and region $R3$ has low feature density. Without feature guidance, path distortion is high at $R1$ and $R2$ and low at $R3$; our feature guidance method moves the distortion away from $R1$ and $R2$, and concentrates it at $R3$.

4.2 View-independent geometry distortion

We measure the distortion of the VE geometry using Sorkine's distortion metric δ [18]. For a triangle T and its distortion T', δ is defined as follows.

$$\delta_T = max\{\gamma_{max}, 1/\gamma_{min}\} \quad (7)$$

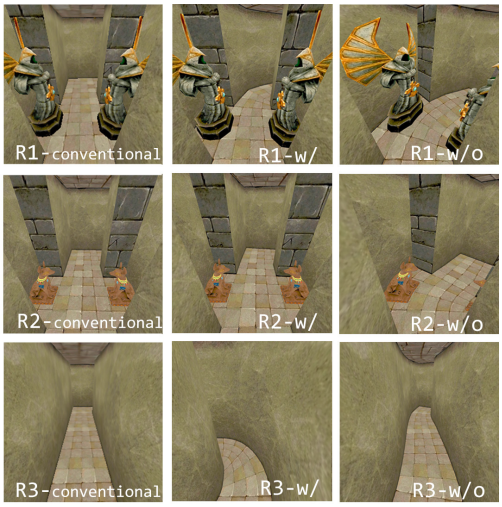

Figure 7: The three rows show frames at $R1$, $R2$, and $R3$, rendered with our method with feature guidance (middle), without feature guidance (right), and conventionally (left). For the feature dense regions $R1$ and $R2$ our method produces frames that are similar to those obtained by conventional rendering, whereas not using feature guidance results in distortions and occlusions at $R1$ and $R2$. For $R3$, our method with feature guidance distorts the path, whereas not using feature guidance doesn't distort the path, having inappropriately distorted the virtual environment at $R1$ and $R2$.

Table 2: VE geometry distortion, with and without feature guidance, for the entire VE, and for the parts of the VE with high detail.

Scene	$\delta_g^{avg}/\delta_g^{max}$ w/ FG	$\delta_g^{avg}/\delta_g^{max}$ w/o FG	Red.	$\delta_g^{high\_detail}$ w/ FG	$\delta_g^{high\_detail}$ w/o FG	Red.
S1	1.47 / 1.59	1.57 / 1.60	10%	1.10/1.59	1.40/1.60	26%
S2	1.53 / 1.93	1.71 / 3.09	18%	1.82/1.93	2.78/3.09	53%
S3	1.25 / 1.62	1.28 / 1.70	3%	1.23/1.62	1.25/1.70	2%

In Equation 7, γ_{max} and γ_{min} are the largest and smallest scaling factors over T as it is distorted to T'. A δ of 1 indicates no distortion. Table 2 gives the average δ_g^{avg} and maximum δ_g^{max} distortion over the entire VE geometry, and over the VE geometry with feature density values in the top 50% of the feature map, with and without feature guidance. For feature guidance, the VE distortion is computed after the view-independent path redirection optimization (Step 2 in Sect. 3.1), so it does not include the view-dependent refinement brought by Step 3. Our feature guidance method reduces VE geometry distortion in all cases, as it strives to reduce distortion over regions with high geometric complexity, therefore exempting a large number of triangles from significant distortion. The distortion reduction is more noticeable in the high detail areas for S1 and S2. The small distortion reduction of S3 is because the visual detail features in S3 are more evenly distributed, making it difficult to distinguish between high detail and low detail areas.

4.3 View-dependent geometry distortion

We have also investigated the distortion introduced by our method in screen space, for individual output frames. Given a frame where a triangle T is visible, we define the view-dependent distortion δ_v of T as the distortion of the projected triangle, computed according to Equation 7. δ_v is computed after the final per-frame distortion refinement (Step 3). Fig. 8 illustrates the benefits of the view-dependent

Table 3: Time cost

Scene	num of tris	area [m^2]	Step 1	Step 2	fps
S1	563k	165	549s	100s	57
S2	478k	416	815s	195s	49
S3	167k	224	603s	90s	60
S4	209k	92	494s	62s	89
S5	318k	306	754s	79s	88

refinement in terms of reducing view-dependent geometry distortion. The refinement corrects the appearance of the objects of interest, e.g. by removing the stretching of the statue.

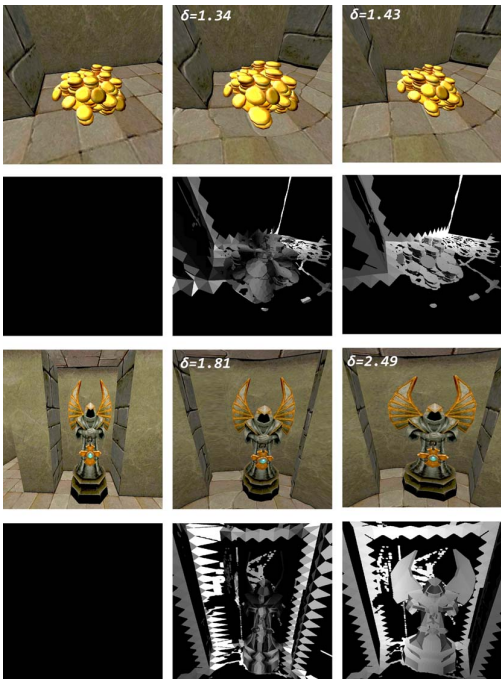

Figure 8: Frames rendered conventionally (left), with our method and with refinement (middle), and with our method but without refinement (right). Rows 2 and 4 show distortion in white. The view dependent distortion, shown for each frame, is reduced by the refinement.

4.4 Rendering performance

Table 3 shows the time cost of our method, broken down into the time for the offline Step 1 of visual feature map construction and the time for the offline Step 2 of path redirection optimization, as well as the frame rate provided to the user exploring the VE. Step 1 dominates the preprocessing time. Preprocessing time for step 1 does not correlate with the number of scene triangles, but rather with the scene area, as the scene is sampled uniformly to collect visual features. For these five scenes, the total preprocessing time is below 20 minutes. The frame rate is comfortably interactive for all scenes.

4.5 Comparison to prior state-of-the-art method

We compared our method with the state-of-the-art Smooth Assembled Mappings method (SAM) [5], using the authors' code, for which we are grateful. SAM achieves redirection with low distortion by subdividing the path into segments, optimizing the mapping on individual segments, and then combining the individual mappings. Table 4 shows that our method reduces the average view-independent distortion δ_g. The maximum view-independent distortion of our method is slightly larger for S5. We also calculate $\delta_g^{high\_detail}$ of the high feature density areas of the 3D mesh. Both average and maximum view-independent distortion values are reduced. The SAM paper introduces an additional distortion metric δ_n, defined on the geometry of the navigation path. On average, our method has a lower δ_n (Table 4), at the cost of an occasional slightly larger maximum δ_n value, as our method distributes distortion non-uniformly, protecting regions with high density of visual features.

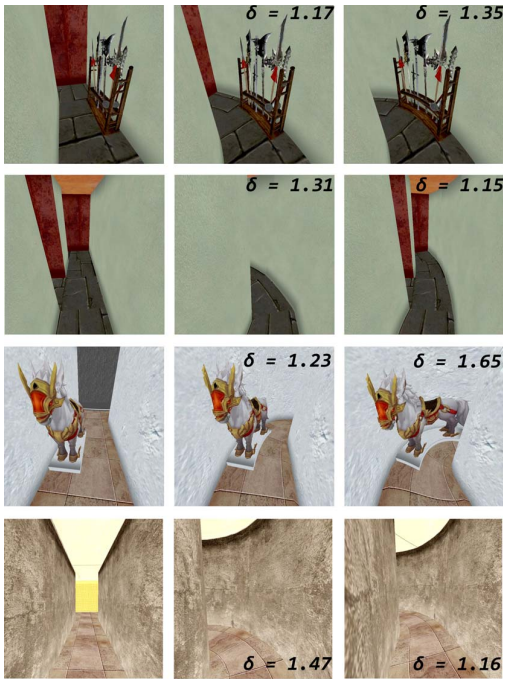

Figure 9: Frames rendered conventionally (left), with our method (middle), and with SAM (right). Our method distorts qualitatively and quantitatively less compared to SAM for frames with high feature density (rows 1 and 3), and more than SAM for frames with low feature density (rows 2 and 4).

Table 4: Comparison to prior art method SAM, over the entire VE, and over the VE regions with high feature density.

Group	$\delta_g^{avg}/\delta_g^{max}$	$\delta_n^{avg}/\delta_n^{max}$	$\delta_g^{high\_detail}$
S4-our	1.21/2.00	1.21/1.86	1.21/1.80
S4-SAM	1.32/2.47	1.21/1.90	1.32/2.16
S5-our	1.23/3.19	1.22/2.04	1.24/ 1.85
S5-SAM	1.30/2.91	1.23/1.82	1.37/2.04

Fig. 9 gives a visual comparison between our method and SAM. Our method concentrates the distortion at regions with low feature density (rows 2 and 4), to show regions of interest with little distortion.

4.6 User study

We have evaluated our method in a VE exploration user study.

Participants, tasks, and experimental design. We recruited (n=16) participants from the graduate student population of our institution. The age range was 22 to 26, and 4 participants were female. All participants had experience with HMD VR applications. The participants were asked to perform two tasks with each of three methods: the Steer to Center (S2C) method [15], SAM, and our

Table 5: User study results: view-dependent distortion δ_v, SSQ Total Severity score TS, locomotion fidelity ρ_l, and visual fidelity ρ_v

.

Condition	$\delta_v^{avg} \pm \delta_v^{dev}$	Pre TS	Post TS	ρ_l [%]	ρ_v [%]
S4 S2C	1.00 ± 0.00	3.8 ± 2.9	19.2 ± 3.4	50.0	100
S4 our	1.25 ± 0.08	1.5 ± 2.1	1.8 ± 1.9	80.0	80.0
S4 SAM	1.29 ± 0.04	7.3 ± 7.4	1.5 ± 2.1	76.7	70.0
S5 S2C	1.00 ± 0.00	2.1 ± 1.7	22.9 ± 10.2	40.0	100
S5 our	1.26 ± 0.02	4.4 ± 2.3	5.2 ± 3.6	75.0	70.0
S5 SAM	1.39 ± 0.03	4.1 ± 2.0	5.2 ± 3.6	72.5	57.5

method. S2C and SAM served as control. S2C is a real time method, i.e. with no preprocessing required. SAM requires preprocessing. For example, for S4, SAM preprocessing takes 24min, whereas the preprocessing for our method takes 9min on S4. The tasks were to find exits in S4 and S5, respectively. We used a within-subject design, with each participant performing both tasks in each of the three conditions, in random order. Each task took approximately five minutes. The subjects took a two minute break between tasks and between conditions.

Objective metric. For each participant, we calculate the average view-dependent 3D mesh distortion δ_v, over all frames. The calculation is performed off-line, based on the participant HMD trace, to avoid lowering the frame rate.

Subjective metrics. Participants completed a simulator sickness questionnaire (SSQ) [8] before the experiment, and then after the experiment. Participants also completed a fidelity questionnaire [2] that asks participants to rate locomotion and visual fidelity. Locomotion fidelity was defined for participants as the degree of similarity to conventional VR locomotion, on a scale from 0 to 100, with 0 corresponding to "completely different" and 100 corresponding to "exactly the same". Visual fidelity was defined for participants as the degree to which the images seen are similar to the images seen in conventional VR, on the same 0 to 100 scale.

Results and analysis Table 5 gives the results of our study. As S2C does not distort the scene, the view dependent distortion is always 1 and the visual fidelity is always 100, which comes at the cost of substantially lower locomotion fidelity and higher post TS. Our method achieves a lower distortion than SAM. A repeated measures ANOVA test run on the three conditions reveals that there is a significant difference between at least one pair of conditions for the view dependent distortion on S4 ($p < 0.0001, F = 58.02$) and on S5 ($p < 0.0001, F = 107.7$). The subsequent Bonferroni correction shows an advantage of our method over SAM for S4 ($p = 0.0949$), and a significant advantage of our method over SAM for S5 ($p < 0.0001$). We explain the greater advantage of our method over SAM for S5 than for S4 due to the larger size of S5, which gives more room to relocate the distortion of feature-rich regions.

The TS does not increase from pre to post beyond 70, which is the threshold above which the increase would indicate onset of motion sickness [6]. Our method and SAM have similar TS scores, while the S2C scores are substantially higher. Locomotion fidelity is low for S2C, as the method redirects the participants with viewpoint translation and rotations dynamically, and the participants can feel the inconsistency between real movements and the movements in the virtual scenes. The locomotion fidelity for our method is slightly higher than those of SAM, but the advantage is not significant. The same ANOVA test shows that there is a significant difference between the three conditions in terms of visual fidelity for S4 ($p < 0.0001, F = 88.08$) and for S5 ($p < 0.0001, F = 140$). The Bonferroni correction shows that our method has a significant advantage over SAM on both S4 ($p = 0.0051$) and S5 ($p < 0.0001$). Several participants noted for SAM that regions with high complexity are occasionally distorted, which leads to incorrect depth and scale perception, which in turn affects navigation.

5 Conclusions, Limitations, and Future work

We have presented an approach for path redirection that takes into account the visual features of the virtual environment. The central idea of our approach is to use visual features to optimize the virtual to real mapping and make the distortions of the VE less noticeable and less distracting to the user as they navigate the VE. The visual features are extracted from the VE with view-dependent and view-independent methods, and the distribution of features is stored in a *visual feature map* texture atlas of the VE. The virtual to real path mapping is generated by optimizing a mass-spring system, which takes into consideration the visual feature distribution. At run-time, the regions of the VE that are visible to the user and that are sensitive to distortion, are rendered with as little distortion as possible based on a detail-preserving rendering algorithm. Our method achieves path redirection with less 2D and 3D distortions for regions with abundant visual detail, and, as a result, provides a more comfortable VR user experience.

One limitation of our current implementation is that it cannot handle planar vertical surfaces with high (texture) detail, such as a painting on a wall, as these have no footprint in the horizontal plane where the deformation is computed. This could be remedied by encasing any such region in a thin box used for the purpose of the orthographic projection. A second limitation of our current implementation is that the transition from the low-deformation feature-rich region to the adjacent high-deformation feature-poor regions can be too fast. This fast transition distorts, for example, the texture on the floor surrounding the statue in Fig. 8, row 3, middle, and a more gradual transition could alleviate the problem.

Our approach falls into the category of static mapping methods for path redirection. In the case of a VE with narrow corridors, this is a reasonable choice as the user can only see a small part of the VE at any given moment due to the occlusion culling provided by the corridor walls. On the other hand, for VE's with large empty spaces, static path redirection will result in objectionable distortions, and dynamic mapping methods are to be preferred. One possible direction of future work is to bring the idea of visual feature aware optimization to the realm of dynamic mapping methods.

Another limitation of the current work is that we derive visual features from standard features of the VE geometry and texture, such as salience, presence of long lines, corners, and complex 3D geometry detail. Whereas often these are features that indeed attract the attention of the user, future work could explore making use of application specific knowledge of what is truly important to the user in each application context. The same VE could have its visual features weighted differently based on each task. Another approach is to learn user interests from user traces.

Longer term, research should continue to address the practicality of the VR interface, such that the effectiveness of experiencing a 3D dataset while immersed in it is leveraged beyond niche applications in entertainment.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China through Projects 61932003 and 61772051, by National Key R&D plan 2019YFC1521102, by the Beijing Natural Science Foundation L182016, by the Beijing Program for International ST Cooperation Project Z191100001619003.

REFERENCES

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

[2] D. A. Bowman, J. L. Gabbard, and D. Hix. A survey of usability evaluation in virtual environments: classification and comparison of methods. *Presence: Teleoperators —& Virtual Environments*, 11(4):404–424, 2002.

[3] J. Canny. A computational approach to edge detection. In *Pattern Analysis and Machine Intelligence*, p. 679–698. IEEE Transactions on, 1986.

[4] S. Coquillart. Extended free-form deformation: A sculpturing tool for 3d geometric modeling. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, p. 187–196. Association for Computing Machinery, New York, NY, USA, 1990. doi: 10.1145/97879.97900

[5] Z.-C. Dong, X.-M. Fu, C. Zhang, K. Wu, and L. Liu. Smooth assembled mappings for large-scale real walking. *ACM Transactions on Graphics (TOG)*, 36(6):211, 2017.

[6] J. A. Ehrlich and E. M. Kolasinski. A comparison of sickness symptoms between dropout and finishing participants in virtual environment studies. *Proceedings of the Human Factors —& Ergonomics Society Annual Meeting*, 42(21):1466–1470, 1998.

[7] X. Hou and L. Zhang. Saliency detection: A spectral residual approach. In *IEEE Conference on Computer Vision Pattern Recognition*, p. 1–8, 2007.

[8] P. Hough. Method and means for recognizing complex patterns. *US Patent 3 069 654*, 1962.

[9] V. Ivleva. *Redirected Walking in Virtual Reality during eye blinking*. PhD thesis, Bachelor's thesis, University of Bremen, 2016.

[10] E. Langbehn, P. Lubos, G. Bruder, and F. Steinicke. Bending the curve: Sensitivity to bending of curved paths and application in room-scale vr. *IEEE transactions on visualization and computer graphics*, 23(4):1389–1398, 2017.

[11] E. Langbehn and F. Steinicke. Redirected walking in virtual reality, 2018.

[12] E. Langbehn, F. Steinicke, M. Lappe, G. F. Welch, and G. Bruder. In the blink of an eye: leveraging blink-induced suppression for imperceptible position and orientation redirection in virtual reality. *ACM Transactions on Graphics (TOG)*, 37(4):66, 2018.

[13] N. C. Nilsson, T. Peck, G. Bruder, E. Hodgson, S. Serafin, M. Whitton, F. Steinicke, and E. S. Rosenberg. 15 years of research on redirected walking in immersive virtual environments. *IEEE computer graphics and applications*, 38(2):44–56, 2018.

[14] N. C. Nilsson, S. Serafin, and R. Nordahl. Walking in place through virtual worlds. In *International Conference on Human-Computer Interaction*, pp. 37–48. Springer, 2016.

[15] S. Razzaque. *Redirected walking*. University of North Carolina at Chapel Hill, 2005.

[16] S. Razzaque, D. Swapp, M. Slater, M. C. Whitton, and A. Steed. Redirected walking in place. In *EGVE*, pp. 123–130. Eurographics Association, 2002.

[17] R. A. Ruddle and S. Lessels. The benefits of using a walking interface to navigate virtual environments. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 16(1):5, 2009.

[18] O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski. Bounded-distortion piecewise mesh parameterization. In *Proceedings of the conference on Visualization'02*, pp. 355–362. IEEE Computer Society, 2002.

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

[20] E. Suma, S. Finkelstein, M. Reid, S. Babu, A. Ulinski, and L. F. Hodges. Evaluation of the cognitive effects of travel technique in complex real and virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 16(4):690–702, 2010.

[21] E. A. Suma, G. Bruder, F. Steinicke, D. M. Krum, and M. Bolas. A taxonomy for deploying redirection techniques in immersive virtual environments. In *Virtual Reality Short Papers and Posters (VRW), 2012 IEEE*, pp. 43–46. IEEE, 2012.

[22] E. A. Suma, S. Clark, S. L. Finkelstein, and Z. Wartell. Exploiting change blindness to expand walkable space in a virtual environment. In *Virtual Reality Conference (VR), 2010 IEEE*, pp. 305–306. IEEE, 2010.

[23] E. A. Suma, S. Clark, D. Krum, S. Finkelstein, M. Bolas, and Z. Warte. Leveraging change blindness for redirection in virtual environments. In *Virtual Reality Conference (VR), 2011 IEEE*, pp. 159–166. IEEE, 2011.

[24] E. A. Suma, D. M. Krum, and M. Bolas. Redirected walking in mixed reality training applications. In *Human Walking in Virtual Environments*, pp. 319–331. Springer, 2013.

[25] E. A. Suma, Z. Lipps, S. Finkelstein, D. M. Krum, and M. Bolas. Impossible spaces: Maximizing natural walking in virtual environments with self-overlapping architecture. *IEEE Transactions on Visualization and Computer Graphics*, 18(4):555–564, 2012.

[26] Q. Sun, A. Patney, L.-Y. Wei, O. Shapira, J. Lu, P. Asente, S. Zhu, M. McGuire, D. Luebke, and A. Kaufman. Towards virtual reality infinite walking: dynamic saccadic redirection. *ACM Transactions on Graphics (TOG)*, 37(4):67, 2018.

[27] Q. Sun, L.-Y. Wei, and A. Kaufman. Mapping virtual and physical reality. *ACM Transactions on Graphics (TOG)*, 35(4):64, 2016.

[28] M. Usoh, K. Arthur, M. C. Whitton, R. Bastos, A. Steed, M. Slater, and F. P. Brooks Jr. Walking¿ walking-in-place¿ flying, in virtual environments. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pp. 359–364. ACM Press/Addison-Wesley Publishing Co., 1999.

[29] K. Vasylevska, H. Kaufmann, M. Bolas, and E. A. Suma. Flexible spaces: Dynamic layout generation for infinite walking in virtual environments. In *3D user interfaces (3DUI), 2013 IEEE Symposium on*, pp. 39–42. IEEE, 2013.