





**SPECIAL ISSUE PAPER**

# DreamWalk: Dynamic Remapping and Multiperspectivity for Large-Scale Redirected Walking

Yuan Xiong  | Tong Chen  | Tianjing Li  | Zhong Zhou\* 

<sup>1</sup>State Key Laboratory of Virtual Reality  
Technology and Systems, Beihang  
University, Beijing, China

**Correspondence**

Zhong Zhou, State Key Laboratory of  
Virtual Reality Technology and Systems,  
Beihang University, Beijing 100191, China.  
Email: zz@buaa.edu.cn

**Summary**

Redirected walking (RDW) provides an immersive user experience in virtual reality applications. In RDW, the size of the physical play area is limited, which makes it challenging to design the virtual path in a larger virtual space. Mainstream RDW approaches rigidly manipulate gains to guide the user to follow predetermined rules. However, these methods may cause simulator sickness, boundary collision, and reset. Static mapping approaches warp the virtual path through expensive vertex replacement in the stage of model pre-processing. They are restricted to narrow spaces with non-looping pathways, partition walls, and planar surfaces. These methods fail to provide a smooth walking experience for large-scale open scenes. To tackle these problems, we propose a novel approach that dynamically redirects the user to walk in a non-linear virtual space. More specifically, we propose a Bezier-curve-based mapping algorithm to warp the virtual space dynamically and apply multiperspective fusion for visualization augmentation. We conduct comparable experiments to show its superiority over state-of-the-art large-scale redirected walking approaches on our self-collected photogrammetry dataset.

**KEYWORDS:**

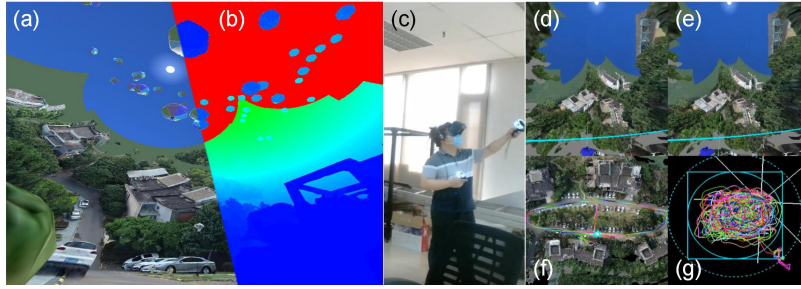
redirected walking, dynamic mapping, multiperspectivity, locomotion, virtual reality

## 1 | INTRODUCTION

In redirected walking (RDW) applications, the size of the virtual space is much larger than the physical space. It is helpful to limit the user's walking within physical boundaries. Conventional RDW approaches apply manipulation gains in a linear space<sup>1,2,3,4,5,6</sup>. These approaches have problems, including simulator sickness, collision, and reset. While static mapping approaches fold virtual pathways to fit the physical play area in pre-processing, but can hardly handle open field and looping circulation<sup>7,8,9,10</sup>. On the other hand, with the development of large-scale 3D reconstruction, the dimension of the virtual space increases rapidly. Existing VR locomotion approaches using the pinhole camera model can hardly handle large-scale open scenes. Thus, a novel method is needed to solve this problem. Recently, multiperspective rendering techniques have been proposed for occlusion exploration and visualization augmentation<sup>11,12,13</sup>. In this paper, we are inspired to propose a **Dynamic Remapping and Multiperspectivity (DREAM)** method for large-scale RDW with the following contributions:

- We illustrate a real-time remapping algorithm that dynamically warps the virtual space using 3D Bezier curves.
- We propose an adaptive multiperspective algorithm for visualization augmentation and occlusion exploration.
- We propose a pitching-up strategy to reduce the user's awareness of camera manipulation gains.

Our experiments reveal the difficulty of mapping between small physical space and large virtual scenes in RDW. To improve the performance of RDW in small play area, a large-scale multiperspective warping is required.



**Figure 1** We propose DREAM to dynamically remap virtual and real scenes. (a) Street view of a 240m  $\times$  200m virtual scene. (b) Depth map. (c) 3.5m  $\times$  3.5m play area. (d) Left HMD view. (e) Right HMD view. (f) Virtual path. (g) Physical path.

## 2 | RELATED WORK

**Redirected walking**<sup>14</sup> manipulates gains so that the virtual path behaves differently from the physical path. Nillson<sup>15</sup> et al. give an overview of recent RDW approaches. Ana et al.<sup>16</sup> summarizes conventional detection thresholds and lateral motion. Typical gains are introduced in<sup>17,18,19,1</sup>, including translation, rotation, curvature, bending, and their derivative. Detection thresholds of lateral and backward steps are measured<sup>20</sup>. Nariaki et al.<sup>21</sup> illustrate the concept of inclination manipulation in pitch RDW applications. Cheng et al.<sup>22</sup> designed a tower climbing game to provide an infinite walking experience. Zhang and Xu et al. manipulate the reset to optimize the interruption experience<sup>23,24,25,26</sup>.

There are few large-scale RDW approaches. The Giant<sup>2</sup> manipulates the avatar size and speed gains. VRoamer<sup>27</sup> and DreamWalker<sup>28</sup> dynamically localize the user with SLAM techniques but require large physical space. Infinite walking<sup>3</sup> includes eye detectors to detect saccadic suppression and leverage temporary blindness for dynamic path planning. The Steer-to-Orbit(S2O) and Steer-to-Center(S2C) algorithms are proposed<sup>4</sup> for calculating desired steering targets. Learning-based steering and path prediction are widely studied<sup>29,30</sup>. Dong<sup>5</sup> illustrates the density-based pushing strategy to keep users away from the center. Telewalk<sup>6</sup> includes the pre-defined real circle for endless walking routes. Simulator sickness is widely studied<sup>31,32</sup>. RDW applications include Simulator Sickness Questionnaire(SSQ)<sup>33</sup> to analyze the severity of simulator sickness symptoms.

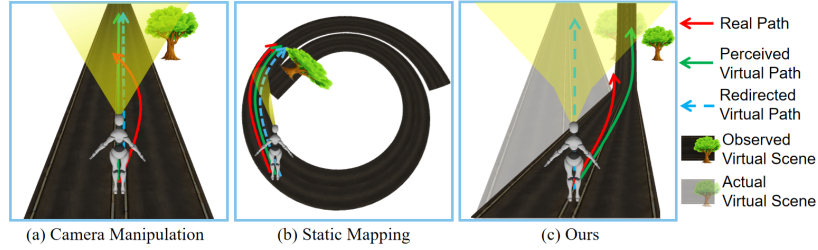
**Static mapping** approaches fold the virtual space into a smaller play area using curvature mapping. Sun<sup>7</sup> proposes a globally surjective and locally injective mapping strategy for space warping. It also solves the self-overlapping problem, including nearest non-occluded pixel searching and environment mapping. Dong<sup>8</sup> proposes a smoothly assembled mapping (SAM) method in which the virtual pathway is decomposed into small patches for distortion optimization. The distortion in SAM is defined by patch-based Bezier surfaces. We are motivated by this idea to replace static 2D mapping with dynamic 3D remapping. Other approaches are proposed for multi-user real walking<sup>9</sup>. Bending gains in planar mapping<sup>9</sup> is similar to those in RDW, which use a derivation of curvature gains. Cao<sup>10</sup> points out that feature-guided mapping is better than globally optimized mapping in terms of visual quality. In real walking, researchers statically map the virtual scene to the physical space using vertex replacement. The performance of static mapping is low, and its limitations for narrow planar scenes make it difficult to generalize to complex VR applications.

**Multiperspectivity** is widely used in VR occlusion exploration. Lin<sup>11</sup> includes top and side view cameras for multiperspective disocclusion. Wang<sup>12,13</sup> manages occlusion exploration by detecting depth discontinuity. UrbanRama<sup>34</sup> provides a navigation interface with vertical perspective warping in one direction and uses cylindrical projection to mix local and global views. These approaches use vertex replacement, partial updates, and multiple rendering strategies, which can lead to low performance.

### 3 | DYNAMIC REMAPPING AND MULTIPERSPECTIVITY

#### 3.1 | Overview

As shown in Figure 1, our goal is to design an RDW method that dynamically and smoothly warps the virtual space to hide the severe optic flow change caused by manipulation gains. We implement two proposed algorithms, the dynamic 3D Bezier warping and multiperspective fusion. The major difference between the proposed and conventional locomotion methods is illustrated in Figure 2. Unlike conventional RDW approaches, our method uses a non-pinhole multiperspective camera model for rendering. In contrast to static mapping approaches, we consider a more general scenario with complex virtual elements other than the pathway and dynamically warp the virtual space using shaders. In computer vision, a virtual scene is modeled as a projective 3D space  $\mathbb{R}_3$ ,



**Figure 2** The major difference between (a) manipulation-based RDW, (b) static mapping, and (c) our method.

which can be represented as a linear combination of three eigenvectors,  $\mathbf{x}$ ,  $\mathbf{y}$ , and  $\mathbf{z}$ . In this paper, for convenience, we always use the right-handed coordinate system with X-right-Y-up-Z-back orientation for the camera coordinates, and the earth coordinates with X-East-Y-North-Z-Up(ENU) orientation for scene modeling. Besides, we are using the homogeneous coordinates for matrix and vector multiplication. The perspective imaging of the Helmet-Mounted Display(HMD) can be represented as:

$$X_{img} = M_{persp} M_{c2d} M_{r2c} M_{v2r} X_v, \quad (1)$$

where  $M_{v2r}$ ,  $M_{r2c}$ , and  $M_{c2d}$  are  $4 \times 4$  homogeneous matrixes that transform the homogeneous vector of a model vertex  $X_v = [x \ y \ z \ 1]^T$  from the virtual world to the room-scale coordinates, the HMD camera coordinates, and the final device coordinates. Each homogeneous matrix  $M = [R|t]$  consists of an orthogonal rotation matrix  $R$  and a translation vector  $t$ .  $M_{c2d}$  can be acquired from the room-scale API using laser trackers or visual SLAM. Display devices simulating binocular vision require  $M_{persp}$  for perspective imaging. Its result is a point  $X_{img} = [u \ v \ w \ 1]^T$ , in a view volume determined by the field of view, aspect ratio, near clipping, and far clipping. In our scenario, given a large-scale model, we dynamically map its vertexes to the camera coordinates, then remap their eigenvectors using Bezier curves and multiperspective fusion.

#### 3.2 | Dynamic remapping

To avoid inclination gains, we dynamically distort the scene along with the user movement. While other approaches use the pinhole camera model in Equation 1 for rendering, we begin to rethink the distorted camera model by involving non-linear transformations. Unlike Wang's<sup>12,13</sup> methods which use partial multiperspective rendering and vertex replacement, we use 3D Bezier curves to remap space eigenvectors according to the current user position. Such a warping can be programmed in a vertex shader in the parallel GPU rendering pipeline, improving the performance remarkably.

**Formulation.** Since the remapping is dynamic and user-oriented, we need to transform the model vertices in the ENU coordinates to the camera coordinates for further distortion. Therefore, we divide the pinhole transformation into two parts:

$$X_{img} = M_{persp} M_{c2d} B(M_{r2c} M_{v2r} X_v), \quad (2)$$

The inner part  $M_{persp}M_{c2d}$  applies a camera-to-device transformation and perspective imaging. The outer part  $M_{r2c}M_{v2r}X_v$  transforms a vertex to the room and camera coordinates. We employ the Bezier warping  $B$  using the function  $b$ :

$$b(t) = \begin{cases} \sum_{i=0}^{n-1} \binom{n-1}{i} t^i (1-t)^{n-1-i} P_i, & 0 \leq t \leq 1 \\ b(1) - (t-1)L \left. \frac{\partial}{\partial t} \right|_{t=1} b(t), & t > 1 \\ -b(-t), & t < 0 \end{cases} \quad (3)$$

where variable  $t$  represents the location on the Bezier curve,  $L$  is the integral length of the Bezier curve in range  $0 \leq t \leq 1$ ,  $\binom{n-1}{i} = \frac{(n-1)!}{i!(n-1-i)!}$  are binomial coefficients, and  $P_i$  is the  $i$ th anchor ( $0 \leq i \leq n-1$ ) of  $n$  anchor points used to determine the Bezier curve. Unlike the explicit definition of the Bezier curve mentioned in<sup>35</sup>, our modification restrains its behavior beyond the interval  $0 \leq t \leq 1$ . For negative  $t$ , we use the symmetric value  $-b(-t)$  as its returns. This is important because it is designed to remove the accumulative shift caused by jitter errors of laser trackers, which return non-zero translation. When  $t > 1$ , we use the extended line of the Bezier curve to avoid the bend-back issue in which the model may collide with itself. The extended line can be determined by the derivative vector of the last anchor at  $t = 1$ . Notice that the sign is always negative because we are using the X-right-Y-up-Z-back coordinate system in which the camera is looking at the  $-z$  direction from the origin.

A vertex  $X = [x_0 \ y_0 \ z_0 \ 1]^T$  in the camera coordinates can also be represented as a linear combination of eigenvectors:  $X = x_0\mathbf{x} + y_0\mathbf{y} + z_0\mathbf{z} + \mathbf{w}$ . Instead of changing  $x_0$ ,  $y_0$  and  $z_0$ , we dynamically remap space eigenvectors  $\mathbf{x} = [1 \ 0 \ 0 \ 0]^T$ ,  $\mathbf{y} = [0 \ 1 \ 0 \ 0]^T$ ,  $\mathbf{z} = [0 \ 0 \ 1 \ 0]^T$  to normalized vectors  $\mathbf{x}_b$ ,  $\mathbf{y}_b$ ,  $\mathbf{z}_b$  using,

$$\begin{cases} \mathbf{z}_b = - \left. \frac{\partial}{\partial t} \right|_{t=t_0} b(t), \quad t_0 = \frac{|X|}{L} \\ \mathbf{x}_b = \mathbf{z}_b \times \mathbf{y} \\ \mathbf{y}_b = \mathbf{z}_b \times \mathbf{x}_b \end{cases} \quad (4)$$

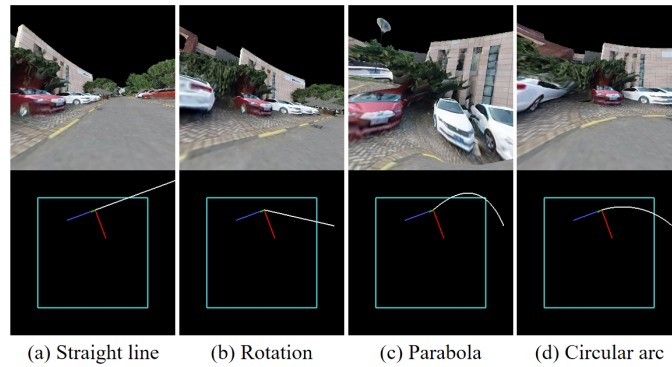
Similarly,  $\mathbf{z}_b$  can be calculated using the derivative vector at  $t = t_0$ .  $\mathbf{x}_b$  equals the cross-product of  $\mathbf{z}_b$  and up direction  $\mathbf{y}$ .  $\mathbf{y}_b$  equals the cross-product of  $\mathbf{z}_b$  and  $\mathbf{x}_b$ . Finally, we compose the new eigenvectors to a Bezier warping:

$$B(X) = J_b X, \quad (5)$$

where  $J_b = [\mathbf{x}_b \ \mathbf{y}_b \ \mathbf{0} \ \mathbf{w}] \in \mathbb{R}_4$  denotes the Jacobian of Bezier function  $b$  at  $X$ . In shader programming, the derivative vector  $\mathbf{z}$  in  $J_b$  can be calculated using an arbitrarily chosen shift  $\Delta t = 1.0e^{-5}$ ,

$$\mathbf{z}_b = - \left. \frac{\partial}{\partial t} \right|_{t=t_0} b(t) = - \frac{b(t_0 + \Delta t) - b(t_0)}{\Delta t}, \quad (6)$$

**Implementation details.** Once we have determined the Bezier function, we can simulate space warping through eigenvector remapping. However, choosing the design pattern for the Bezier anchor points is a hard decision, which is related to factors including implementation difficulty, run-time performance, visual appearance, and corner cases.



**Figure 3** Comparison between different Bezier patterns. Top: HMD view. Bottom: the top view of the play area with physical boundaries in cyan, x-axis in red, y-axis in green, z-axis in blue, and bent view direction in white.

In order to generate smooth Bezier curves, we design several patterns for creating anchor points, as shown in Figure 3. Figure 3a shows the straight line pattern in which anchor points are equally distributed along the  $-z$  view direction. The only parameter for this pattern is  $L$  used by Equation 4. When  $L$  equals the total distance between anchors, the Bezier warping model degenerates into a pinhole model. Otherwise, it is equivalent to a zooming model with a focal length enlarged by  $L$ . The straight line pattern can be used as a backup pattern for other patterns which failed to return anchors in corner cases. Figure 3b shows the rotation pattern. The rotation pattern can be used to simulate a camera rotation. Figure 3c shows the parabola pattern. In 3D geometry, a parabola function can be well-determined by two points and an initial direction of the starting points. In most cases, it provides a smooth U-shaped curve. However, the integral length of a rotated parabola is difficult to compute. In practice, we use the arc pattern shown in Figure 3d. An arc of a circle is defined as a segment of the circle, which can also be well-determined by two anchors and an initial direction. The integral length of an arced curve can be calculated explicitly:

$$L = \int_{\theta_0}^{\theta_{n-1}} r d\theta = r\theta \Big|_{\theta_0}^{\theta_{n-1}} = (\theta_{n-1} - \theta_0)r, \quad (7)$$

where  $\theta_{n-1} - \theta_0$  defines the opening angle from the first anchor to the last anchor, and  $r$  is the radius of the arc. To avoid self-overlapping, we cap the opening angle to  $\frac{\pi}{2}$ . We use Bezier curve approximation instead of the explicit equation of the circular arc, because Bezier curve is more convenient and universal. In the implementation of the circular arc, an additional branch is included in determining whether we need a major arc or a minor arc. Besides, it requires additional rotation and trigonometric operations, which may slow down the rendering. In our implementation, we flexibly integrate different patterns into our system depending on requirements.

As we are using control anchors to determine Bezier curves, the more anchors used, the higher precision we can achieve. However, in consideration of the shader performance, the number of anchors should be small enough. Let  $n$  be the number of anchors. When  $n = 2$ , the Bezier curve model degenerates into a rotation model, like in Figure 3b. So we only consider cases where  $n \geq 3$ . As mentioned in Equation 3, the computational cost of the dynamic remapping mainly comes from the calculation of Bezier function  $b(t)$ . We can optimize it by enumerating binomial coefficients  $\binom{n}{i}$  as constants, replacing the expensive power function with multiplication operations, and reusing previous terms. For example, by reusing  $t^n(1-t)^i$  and  $\frac{1-t}{t}$  in each iteration, we reduce its computational cost. In our implementation, we set  $n = 5$  for best practice.

### 3.3 | Multiperspective rendering

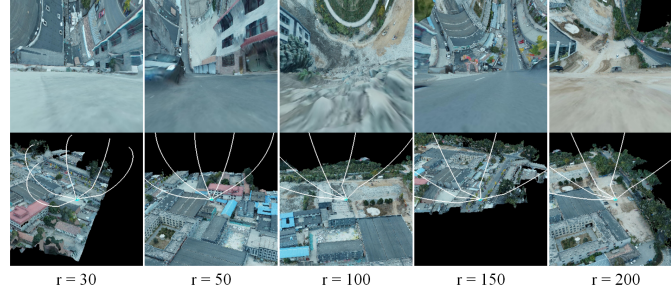
As described in the previous section, the Bezier curve can be used for remapping space eigenvectors. As shown in Figure 3c and Figure 3d, lateral pixels are stretched and become blurry. We are motivated to improve the rendering of lateral pixels. On the other hand, we are illuminated by the horizontal multiperspective rendering<sup>12</sup> for occlusion exploration. Besides the sky, filling the upper half of the view with scene elements is a good choice because it provides visual cues to the moving direction.

**Multiperspective fusion.** In our scenario, we expect to provide a natural walking experience with free viewports and moving directions. Unlike other approaches which place explicit obstacles to block the user view, we use the original large-scale reconstruction of real-world scenes. We improve the fidelity of dynamic warping in all directions and include the multiperspectivity fusion to assemble renderings from different perspectives smoothly. In our design, the general multiperspective warping is defined as a linear combination of several Bezier curves and is deployed as a replacement of the single warping:

$$B_{multi}(X) = \frac{\sum_{i=1}^m w_i R_i^{-1} B_i(R_i X)}{\sum_{i=1}^m w_i}, w_i = (1 + \mathbf{v}_i \cdot (-\mathbf{z}))^2, \quad (8)$$

where  $X$  is a vertex in camera coordinates, rotated by  $R_i$  before the Bezier warping, and  $R_i^{-1}$  for inverse rotation after the warping. The weight factor  $w_i$  is defined by the square of the cosine similarity, which is 1 plus the dot product of two eigenvectors,  $\mathbf{v}_i$  and  $-\mathbf{z}$ , where  $\mathbf{v}_i$  is the normalized direction of the  $i$ th perspective ( $1 \leq i \leq m$ ).  $R_i$  is constrained by  $R_i \mathbf{v}_i = -\mathbf{z}$ . It rotates the axis  $\mathbf{v}_i$  to  $-\mathbf{z}$ , around the up direction  $\mathbf{y}$ . Note that the single warping  $B$  in Equation 2 is a special case of  $B_{multi}$  with  $m = 1$ ,  $\mathbf{v} = -\mathbf{z}$ , and  $R = I$ , where  $I$  is the identity matrix. The multiperspective fusion increases the computational cost of shaders. To balance performance and rendering quality, we recommend using 8 perspectives for best practice.

**Vertical multiperspectivity.** Unlike conventional approaches<sup>12,13</sup> which use multiple pinhole renderings for occlusion exploration, we solve the occlusion problem in real-time. In our system, we include the vertical warping using the modified



**Figure 4** Vertical multiperspective rendering with different bending radii in meters. The top row shows the user’s HMD view. The bottom row shows the bird view of virtual scenes with Bezier curves in white.

eigenvectors remapping. The mixed Bezier warping of a vertex  $X$  is

$$B_{mix}(X) = \begin{cases} B(X), & t_0 \leq T \\ B(X) + b_v(t_0 - T), & t_0 > T \end{cases} \quad (9)$$

where  $B$  is the horizontal Bezier warping mentioned in Equation 5, and  $b_v$  is the vertical horizontal Bezier function.  $T$  is the distance threshold to activate the vertical multiperspective rendering. In our implementation, we arbitrarily use  $T = -30/L$ . When  $X$  is 30m away from the camera, it starts bending up, controlled by vertical Bezier anchors. When choosing the pattern for vertical anchors, we follow the same strategy in Figure 3d, by using the circular arc pattern. In Figure 4, we show different bending radii of the circular arc. Whether to use a small or large bending radius depends on the map size of the virtual world. In our implementation, we set  $r = 100$  for large-scale RDW. Another parameter that can be artificially changed is the scale factor  $L$ , defined in Equation 4. For maps that are not big enough to cover the entire front view after vertical bending, it is suggested that using a scaled  $L$  for vertical zooming. The mixed Bezier warping  $B_{mix}$  can be used to replace the original warping  $B$  in multiperspective rendering Equation 8. As shown in Figure 4, the vertical multiperspective rendering successfully solves the occlusion problem by rendering faraway objects into the user’s HMD view on the upper half.

**Geometry remapping.** The major difference between our methods and other multiperspective approaches<sup>34</sup> is that we not only distort the view but also remap the geometry representation to support geometry-based interaction, including illumination, collision detection, and model manipulation. Equation 9 is friendly to linear interpolation and smoothness. However, the real divisor decomposition of such a polynomial may not exist. For approximate inverse mapping of a virtual point  $Y$ , we calculate

$$B^{-1}(Y) = \arg \min_X ||B_{mix}(X) - Y||^2, \quad (10)$$

where we try to solve the least-square problem using gradient descent to find the inverse point  $X \in \mathbb{R}_3$ . In the stage of initialization, we can either use  $X = Y$ , or the closest Bezier anchor. In our experience, both of them work as expected. With the momentum optimization, their results converge after 300 iterations. The computation of Equation 10 is CPU-intensive, working for tasks with discrete point searching, such as collision detection. For tasks with massive requests, it is recommended to implement Equation 9 in the GPU pipeline and store original positions in a render buffer for fast looking up.

The normal vector  $n$  of a given vertex  $X$  can be calculated using the original normal  $n_0$  and a small shift  $\Delta x$  by,

$$n = \text{normalize}(B_{mix}(X + \Delta x * n_0) - B_{mix}(X)), \quad (11)$$

### 3.4 | Camera manipulation gains

The calculation of manipulation gains is defined in the objective metrics. The main purpose of our approach is to visually bend the virtual space that the user can see and redirect the user to a virtual path. In general, when a user walks from position  $X_t$  to  $X_{t+1}$ , conventional RDW approaches to convert the real translation  $T_r$  to virtual translation  $T_v$  so that  $X_{t+1}$  is shifted to a virtual place  $X'_{t+1}$ , as shown in Figure 2a. A similar manipulation is applied to the rotation components. Rotation matrices are not addable and should be decomposed to the addable form of yaw-pitch-roll angles, written as  $Y_{t+1}$ . In our approach, to relieve the cybersickness caused by locomotion, we define the pose transformation of a new frame,

$$\begin{cases} Y'_{t+1} = Y_{t+1} = Y_t + \mathbf{R}_r \\ X'_{t+1} = X_t + |\mathbf{T}_r| \cdot \mathbf{v}_t \\ \mathbf{v}_t = \frac{(X_{target} - X_t)}{|X_{target} - X_t|} \end{cases} \quad (12)$$






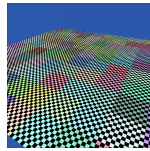



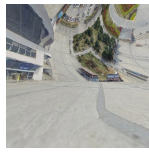
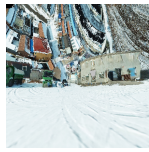
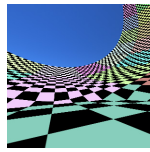
where  $\mathbf{v}_t$  denotes the direction to the steering target  $X_{target}$ , which is manually updated by the user controller in our system.

The design purpose of Equation 12 is to keep the user moving in a virtual path  $X_t \rightarrow X'_{t+1}$ , while physically walking along the path  $X_t \rightarrow X_{t+1}$  without too much awareness of the difference between them. We are motivated to reduce the awareness of manipulation gains<sup>16,17,18,19,1</sup> to reduce simulator sickness by multiperspective augmentation and pitching-up strategy.

## 4 | EXPERIMENTS

This section expounds on detailed experiments of our framework. We present experimental evaluations on RDW tests, including basic motion analysis and long-distance walking. Because no dynamic warping approaches in RDW have been found recently<sup>15</sup>, we can only compare our result with conventional RDW and static mapping approaches. Both numerical analysis and visual comparison are given in detail. All experiments are deployed on a personal workstation with 6 cores 3.2 GHz processors, 32 GB RAM, and a NVIDIA GeForce RTX 2080Ti GPU. In the real walking test, we use the basic HTC VIVE room-scale set with 2 laser trackers to support legacy devices. In our latest implementation, we also use the HTC Cosmos with the wireless adapter. The rendering application is inherited from the official SteamVR example with C++, OpenGL shaders, and OpenVR toolkits. All HMD screenshots are taken from the companion window, which copies the HMD buffer directly.

**Table 1** Dataset for walking experiments

Metrics	Town	Apartment	Campus	Station	Snowy	Chessboard
Size (m) Triangles	360 × 420 13.2M	240 × 200 4.7M	686 × 499 1.0M	1080 × 959 4.3M	280 × 285 3.4M	100 × 100 20k
Bird view						
HMD view						

**Participants.** We conducted user studies to evaluate our DREAM method. We enrolled 43 participants, 29 males and 14 females, whose ages were between 21 and 59 ( $\rho_{age} = 30.58 \pm 9.53$ ). Most of them were college students and staff. They communicated well without trouble understanding the game. Few of them were familiar with VR applications, while others had no such experience. Before testing, participants were given tutorial materials about comparable methods, including introductions, algorithms, rendering samples, demo videos, and implementation details. Participants were invited to play VR games for at least 20 minutes. Most of them were immersed for more than 30 minutes and walked more than 100 meters.

**Dataset.** In Table 1 we show our self-collected large-scale urban scenes. Detailed size, meshes, scene bird views, and HMD screenshots are presented. These real scenes are reconstructed using oblique photography and structure from motion techniques. Although the texture resolution and point cloud accuracy of photogrammetric data are limited by the flight height (120m on average) of the unmanned aerial vehicle(UAV), it can provide more detail and diversity than manually created simple 3D geometries. We propose our photogrammetric dataset to test performance bottlenecks of existing large-scale RDW algorithms. However, we also provide a lightweight chessboard virtual scene with fewer triangles for low-performance algorithms.

**Subjective metrics.** We use the subjective metrics mentioned in<sup>36</sup> to evaluate user involvement in RDW applications. For subjective features, we ask participants to grade them according to the acceptance of a certain category from 1 to 10. We compare

**Table 2** Scoring standard for the user study

Metrics	Score 1~3	Score 4~6	Score 7~9	Score 10
Rendering	Completely unacceptable	Acceptable	Less than perfect, visible defects	Exactly the same as real scenes
Locomotion	Fails to provide a smooth walking experience	Hard to finish walking	Moderately different from the expectation	Fluid walking experience
Performance	Severe lagging or long waiting	Lagging (fps $\geq 30$ )	Perceptible frame drop (fps $\geq 60$ )	Fluid (fps $\geq 90$ )
Immersible Scene Size	Small	Medium	Large	Extremely Large
Learnability	Hard to learn	Learnable after training	Easy to learn from a tutorial	Pick-up-and-play
Interesting	Boring	Playable	Very Interesting	Can't stop playing

the average scoring and standard deviation of conventional locomotion approaches, including the Giant<sup>2</sup>, Telewalk<sup>6</sup>, and SAM<sup>8</sup> to the proposed DREAM method, denoted as  $\rho_G, \rho_T, \rho_S, \rho_{ours}$  accordingly. The detailed scoring standard is shown in Table 2. After each test, users were asked to fill out the SSQ<sup>33</sup>, which contains 16 symptoms with a 4-point scale(0,1,2,3) for all items.

**Objective metrics.** In comparison with conventional large-scale RDW methods, we evaluate four types of manipulation gains: the translation gain  $g_T = \frac{|T_{virtual}|}{|T_{real}|}$ , the rotation gain  $g_R = \frac{|R_{virtual}|}{|R_{real}|}$ , the curvature gain  $g_C = \frac{1}{r_{real}}$  and the bending gain  $g_B = \frac{r_{virtual}}{r_{real}}$ , where  $T_{virtual}$  and  $T_{real}$  are the virtual and real translation;  $R_{virtual}$  and  $R_{real}$  are the virtual and real rotation;  $r_{real}$  and  $r_{virtual}$  are the radii of curvature of the bent real and virtual path. When the user walks freely, curvature gains and bending gains are measured separately, depending on whether the walking route is straight enough.

To determine the quality and performance of the mappings, we adopt two objective metrics, the mapping speed, and fidelity rate. For the mapping speed, because the map size varies from scene to scene, we evaluate the average number of processed triangles per second. In terms of fidelity, we quantitatively measure the distortion, denoted as  $\delta = \frac{r}{r_d}$ , where  $r$  is the circumcircle radius of the original triangle and  $r_d$  is the circumcircle radius of the distorted triangle. The average and standard deviation of the distortion rate are denoted as  $\delta_{avg}$  and  $\delta_{std}$ .

## 4.1 | Subjective experiments

### 4.1.1 | User Experience Study.

We conducted the user experience study to evaluate the presence score defined in Table 2, by providing a survey with single-item questions scoring different locomotion approaches. Before and after each test, users were asked to fill out the SSQ.

**Rendering quality.** The difference between rendering quality varies slightly ( $\rho_G = 7.0 \pm 1.98, \rho_T = 7.33 \pm 1.2, \rho_S = 6.67 \pm 1.61, \rho_{ours} = 7.54 \pm 1.5$ ). Participants blame the SAM for the wall blocking and prefer open field renderings. Our approach benefits from the multiperspective rendering and impresses participants with extraordinary user experience.

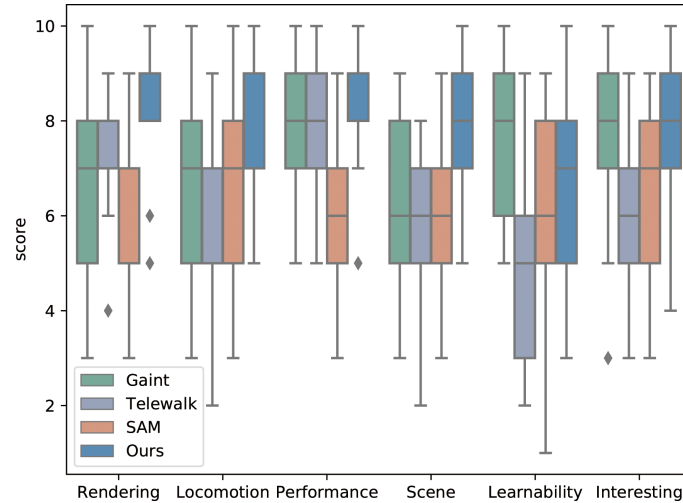
**Locomotion.** In terms of locomotion experience ( $\rho_G = 5.71 \pm 1.97, \rho_T = 5.29 \pm 1.9, \rho_S = 6.83 \pm 1.74, \rho_{ours} = 7.79 \pm 1.28$ ), participants consider the flexibility more than walking distance. On this point, the mono-directional strategy in Telewalk and pathway restriction in SAM does not perform well. While our movement control is flexible.

**Performance.** It turns out that performance is not a critical issue for users unfamiliar to the VR applications ( $\rho_G = 7.67 \pm 1.37, \rho_T = 7.38 \pm 1.53, \rho_S = 5.88 \pm 1.65, \rho_{ours} = 8.21 \pm 1.14$ ). The SAM is exceptional, with a pre-processing delay and overlapping removal cost. Our space warping is GPU-intensive, in which shader programs execute heavy computational tasks parallelly. The rendering cost of our system is 3ms/frame, and the capped frame rate is 90fps.

**Virtual scene size.** Users prefer large maps with rich diversity ( $\rho_G = 5.92 \pm 2.0, \rho_T = 5.58 \pm 1.41, \rho_S = 6.17 \pm 1.76, \rho_{ours} = 8.38 \pm 1.21$ ). As an open-world implementation, our approach performs the best. Virtual scenes in SAM and Telewalk are not fully walkable because they have restrictions in the walking route or suffer from self-overlapping.

**Learnability** Users can hardly learn the perfect circular walking restricted by Telewalk but can quickly learn the Giant walking. They show different confidence in learning lateral, backward steps and our free walking with steering targets ( $\rho_G = 8.13 \pm 1.51, \rho_T = 5.17 \pm 2.01, \rho_S = 5.33 \pm 2.41, \rho_{ours} = 6.67 \pm 1.79$ ).





**Figure 5** Scoring boxplots of user experience study results.

**Interesting.** The result of this category varies differently ( $\rho_G = 7.0 \pm 1.97$ ,  $\rho_T = 5.5 \pm 2.06$ ,  $\rho_S = 6.33 \pm 2.06$ ,  $\rho_{ours} = 7.75 \pm 1.62$ ). It means that users have different preferences related to the content instead of implementation details.

**SSQ analysis.** We followed<sup>33</sup> and measured the mean and standard deviation of SSQ scores ( $M_G = 9.04 \pm 4.49$ ,  $M_T = 30.29 \pm 9.10$ ,  $M_S = 17.06 \pm 8.76$ ,  $M_{ours} = 4.17 \pm 2.39$ ). The results indicate different simulator sickness levels for the Giant (low), Telewalk (severe), SAM (moderate), and our method (low). It turned out that translation gains mainly caused the simulator sickness of the Giant. For Telewalk, rotation gains caused severe disorientation. SAM works for narrow scenes with block walls. However, in large-scale open scenes, it brings oculomotor symptoms because of the over-distortion and high overlapping. Our method performed stably in terms of low-level simulator sickness.

**Data analysis.** According to the overall scoring concerning all categories in Figure 5, our method has the highest average score and lowest standard deviation ( $\rho_G = 6.90 \pm 1.98$ ,  $\rho_T = 6.04 \pm 1.93$ ,  $\rho_S = 6.2 \pm 1.92$ ,  $\rho_{ours} = 7.72 \pm 1.52$ ). We further analyzed the data by including the Friedman test, which showed the difference between our approach and others. The average scoring data shown in Figure 5 was converted into a matrix with 6 columns (scoring criteria) and 4 rows (methods). From average ranks, we calculated the statistic value  $Q = 9.936$ . Since the degree of freedom was small, and its chi-square distribution approximation was poor, we obtained the p-value and upper critical threshold from the Friedman test table. The upper critical value  $F_{0.05}[4, 6] = 7.6$ , and the p-value ( $p < 0.05$ ) was significant. We followed the rules to include the Nemenyi test for post-hoc comparison. The critical distance for average ranking is  $CD_{0.05} = 1.915$  of condition ( $q_{\alpha=0.05}[4] = 2.569$ ). It showed that the performance of compared methods was significantly different. Our method surpasses others in terms of all scoring factors except for learnability. Our method generally increased the rendering performance and brought a fluid walking experience.

We found that women experienced the same level of simulator sickness compared to men while immersed in the large-scale open scene. However, the attention to gaming factors varied according to gender. Female users were more concerned about the visualization, including the texture resolution, artifacts, and color schema. While male users pay more attention to game tasks, including the interactivity of the user interface, the smoothness of walking, and the fun of shooting.

#### 4.1.2 | Ablation Study.

In order to study the effectiveness of vertical multiperspectivity and pitching-up strategy in reducing simulation sickness and visual augmentation, we design an ablation study to determine the threshold of the ideal pitching-up angle. The result is shown in Table 3. In this test, we invited 20 hardcore users to play our shooting game with 100 targets randomly deployed all over the virtual world. Before and after each test, users were asked to fill out the SSQ to observe whether sickness scoring rises. In each test, we bend the visual scene up with a certain angle to trick the user into looking up.

A Friedman test on the user representation results for each pitching angle showed a significant effect of the pitching-up strategy. The representation data in Table 3 shows a matrix with 5 columns (representation factors) and 6 rows (pitching angles). By calculating the ranks, we found the statistic value  $Q = 17.686$ . Since its degree of freedom is small and the approximation to chi-square distribution is poor, we obtain the p-value from the Friedman test table. It showed that  $Q$  was much larger than the

**Table 3** Ablation study on the pitching-up test with different pitching angles  $\theta$ . We provide a shooting game where users are asked to finish a task with 100 aerial targets to destroy. We measure changes in representation factors for each test.

$\theta$	Awareness of gains ↓	SSQ Score ↓	Task completion ↑	Walking distance ↑	Game duration ↑
0°	100%	15.2	26.5%	23.1m	5.5min
15°	60%	11.4	45.1%	45.8m	12.1min
30°	65%	5.23	55.1%	127.1m	41.5min
45°	35%	4.13	82.2%	156.9m	48.2min
60°	15%	4.02	78.3%	69.2m	35.3min
75°	10%	4.67	79.9%	34.1m	37.1min

upper critical value  $F_{0.05}[6, 5] = 10.49$ , and the p-value ( $p < 0.05$ ) was significant. We detected the different representations for different pitching angle test. We further employed the Nemenyi test for post-hoc comparison, with critical distance  $CD_{0.05} = 3.37$  ( $q_{\alpha=0.05}[6] = 2.85$ ) and  $CD_{0.1} = 3.0$  ( $q_{\alpha=0.1}[6] = 2.589$ ). The post-hot test showed performance differences between two pitching angle tests if their average performance rank is greater than the  $CD$ . In this case, according to the average ranks, users in groups with pitching-up angles ( $r_{15^\circ} = 4.6, r_{30^\circ} = 3.4, r_{45^\circ} = 1.6, r_{60^\circ} = 2.6, r_{75^\circ} = 2.8$ ) performed differently than those in groups without pitching angles ( $r_{0^\circ} = 6$ ) with better ranks. Users in groups with pitching-up angles greater than  $30^\circ$  performed similarly without significant statistical difference.

The ablation study showed that the pitching-up strategy successfully relieved the simulator sickness caused by camera manipulations by hiding close objects and their optic flow beyond the user’s view window. On the other hand, the vertical multiperspective rendering kept most game factors in the user view, including structured patterns, shooting targets, and direction cues. Users quickly got used to our game design of changing the terrain appearance for occlusion exploration and visual augmentation without additional training. They have no difficulty in shooting discovered targets and walking around. However, the pitching-up angle should be smaller than  $75^\circ$  for best practice. While walking with a pitching angle smaller than  $15^\circ$ , most users can obviously notice the translation and curvature manipulations reflected in closer objects. According to the research in<sup>37</sup>, optical patterns can provide a sense of self-motion, and the sensory mismatch between the visual and bodily motion is the main cause of simulator sickness. Our test results showed that users in the large pitching angle group reported low sensitivity to the manipulation gains because they did not see much optic flow in the first place. Besides, users in those groups immersed better in terms of walking distance, game duration, and task completion. As a result, we suggest that the pitching up angle for the vertical multiperspective system be set to  $45^\circ$  for best practice.




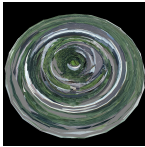
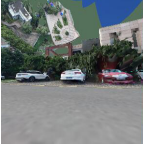

In general, the test results show that the pitching-up strategy can successfully reduce simulator sickness caused by the sensitivity of optic flow change of manipulation gains, and visual cues and scene appearance can still be observed well.

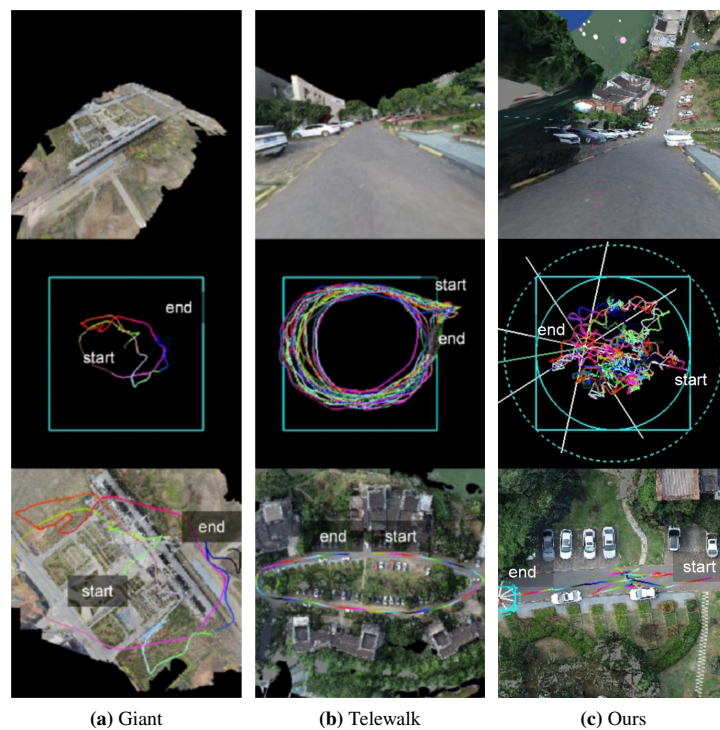
## 4.2 | Quantitative experiments

### 4.2.1 | Mapping evaluations

We compare our method with mainstream static mapping approaches, Sun’s method<sup>7</sup> and the SAM<sup>8</sup>. They statically map the large-scale virtual space to the narrow physical space in the pre-processing. We do not compare our work to<sup>9</sup> because it is a multi-user application with the same warping as SAM. The detailed result is shown in Table 4. Although optimized by multithreading, Sun’s method spent 14 hours to warp the large-scale open scene, and SAM spent 76 minutes. Regarding visual fidelity, the distortion rate of them varies from triangle to triangle, depending on the viewpoint. Static mapping approaches have a higher self-overlapping rate, bringing in additional costs for depth detection and occlusion removal. As shown in Table 4, the visual fidelity of static mapping on large-scale uneven open surfaces is poor. On the other hand, our method can leverage the GPU pipeline and finish the scene warping within 3 milliseconds. The multiperspective fusion guaranteed the rendering smoothness in all directions. In our system, the dynamic warping was user-oriented, and no overlapping occurred.

**Table 4** Comparison between static mapping and ours on warping large-scale uneven open scenes.

Methods	Acceleration	Speed	Overlapping	$\delta_{avg}$	$\delta_{std}$	Street View	Bird View
Sun's <sup>7</sup>	CPU	$2.68 \times 10^2/s$	99.97%	1.714	7.781		
SAM <sup>8</sup>	CPU	$2.40 \times 10^3/s$	69.86%	0.708	0.429		
Ours	GPU	$1.57 \times 10^9/s$	0.00%	0.636	0.310		

**Figure 6** Comparison between different large-scale RDW approaches. (a) the Giant<sup>2</sup>; (b)Telewalk<sup>6</sup>; (c) our method. Top row: HMD view. Middle row: top view of the play area. Bottom row: top view of the virtual scene.

#### 4.2.2 | Redirected Walking

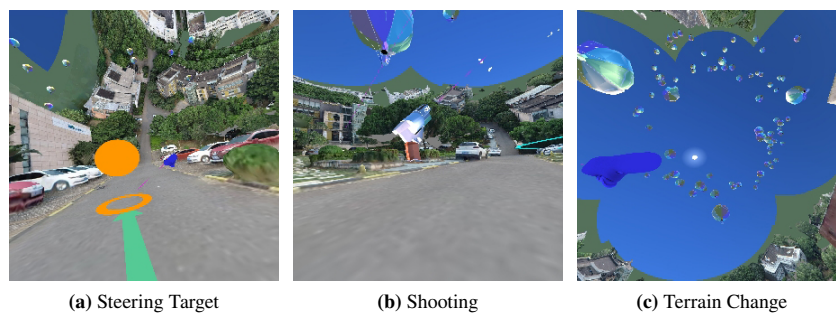
We compare our method with large-scale RDW approaches, as shown in Figure 6 and Table 5. The Giant<sup>2</sup> increases the avatar size for a bird-view VR experience. For large-scale scenes, the proposed ground level scaling of Giant increased to 30 and eye level scaling becomes 10. As shown in Table 5, the translation gains of Giant is larger than expected. Telewalk<sup>6</sup> uses the S2O<sup>4</sup> algorithm that enables infinite walking by mapping the virtual path to a physical orbit. However, Telewalk only allows one-way walking on the pre-defined virtual path without turning and lateral walking. It is hard for common users to maintain their walking path in a perfect circle. Manipulation gains of Telewalk are larger than the detection thresholds mentioned in<sup>16</sup>. In contrast,

**Table 5** Comparison of manipulation gains in large-scale RDW.

Methods	$g_T$	$g_R$	$g_C$	$g_B$
Giant <sup>2</sup>	$300 \pm 0.00$	$1.00 \pm 0.00$	$0.00 \pm 0.00$	$1.00 \pm 0.00$
Telewalk <sup>6</sup>	$1.17 \pm 4.34$	$1.73 \pm 27.67$	$3.97 \pm 17.20$	$0.13 \pm 2.99$
Ours	$1.00 \pm 0.03$	$1.00 \pm 0.01$	$0.21 \pm 0.63$	$1.01 \pm 0.68$

our method allows walking on all directions and applies distorted multiperspectivity for visual deception. With target focusing, users can change their paths. Our average translation and rotation gains are close to 1, in the range of detection thresholds. Our translation and rotation gains are not exactly 1 because of the pose error brought by sensors. As a result, the user's sense of discomfort is relieved in our system. Infinite walking approaches using special eye detectors<sup>3</sup> are beyond our consideration. We do not conduct long-distance walking experiments for static mapping approaches because they have no manipulation gains.

For the RDW task, all users successfully walked for at least 50 meters and 10 minutes. The longest walking was finished in 20.5 minutes, in which the user walked 642.2 meters and shoot down all targets. As for the shooting task, the average and standard deviation of orbs used to hit a balloon target was  $11.05 \pm 10.6$ . The average and deviation of balloons shot in a game is  $64.53 \pm 27.6$ . For all tests, no simulator sickness were reported.



**Figure 7** The DREAM world game design. (a) Selecting the steering target by the left controller. (b) Shooting orbs isomg the right controller rendered as a minigun. (c) Changing the terrain appearance by bending Bezier curves.

### 4.2.3 | Application

We design the DREAM world game with two tasks to verify our system.

The first task is long-distance walking. As shown in the teaser Figure 1 and Figure 7b, users are asked to walk in a virtual open scene. They can walk freely with their own targets. To avoid unexpected collision with the physical boundary in the play area, we render a teal circle in the air for notification. At the beginning of the game, all Bezier curves for multiperspectivity are set straight forward. As shown in Figure 7c, users can adjust them according to their preferences using the left controller. The steering target is initially positioned a few meters away on the ground and is rendered as an orange orb. A green arrow pointing to the steering target is rendered for direction indication. The user can use the left controller to adjust the steering target. This design aims to verify that our system can relieve cumulative simulator sickness in large-scale RDW.

The second task is the balloons shooting, as shown in Figure 7b. In the game, the right controller is rendered as a minigun, continuously shooting bouncing orbs. Since we have dynamically remapped the virtual coordinates, we can extract the position and normal vectors of every triangle in real-time. The user can play with the gun by watching the orbs bounce in the virtual scene. We randomly place 100 balloons in the air for shooting. The bullets in the game are designed as small orbs which can be elastically collided. This design aims to verify the correctness of coordinates transformation and inverse transformation.

## 5 | GENERAL DISCUSSION

Our subjective experiments revealed that the smoothness of rendering and playable scene size is important. The locomotion experience and learnability vary from person to person, depending on the tolerance to motion sickness and educational background. Our method performs better than compared algorithms according to the statistical report. From the objective experiments, we learn that the vertical multiperspective fusion successfully reduces the user's awareness of manipulation gains and makes them relaxed by focusing on main tasks. Moreover, our results show that the translation and curvature gains are less imported compared with the rotations when looking up. In our scenario, the multiperspective augmentation of terrain appearance plays an essential role in providing visual cues so as to relieve symptoms of disorientation. The ablation study in Table 3 shows that the ideal pitching-up angle is between  $45^\circ$  and  $60^\circ$ .

The application experiments have proved that users can be immersed in our game and learn how to play the shooting game while walking. This shows that our method can effectively solve the simulator sickness and view occlusion problems of RDW. It makes the RDW in large-scale open scenes possible, which is a well-known difficult problem in the RDW field.

The major difference between our design and other approaches is that we modify the user view and meanwhile change the coordinates system. Without changing the geometric topology of the large-scale virtual scene, we can calculate the multiperspective transformation and inverse transformation in real-time. This allows game factors such as elastic collision and illumination to be implemented in the warped scene. The innovation of our approach is replacing the traditional pinhole camera model with the Bezier-curve-based dynamic coordinates remapping.

However, our warping method can also cause textural artifacts and visual defects. This is mainly due to the over-distortion of large triangles. Our experiments show that increasing the density and resolution of the model can alleviate this problem.

## 6 | CONCLUSIONS

In this paper, we have presented a dynamic remapping and multiperspectivity method for RDW applications. We show that our method can enhance the performance of space warping and effectively relieve the simulator sickness. Comprehensive experiments demonstrate the usefulness of our method in RDW in large-scale open scenes.

The proposed system has limitations. It works for large-scale virtual scenes but can hardly handle indoor situations. In our datasets, due to the flight height of oblique photography, the rendering quality of the ground-level objects needs improvement. This issue can be resolved by using more high-quality aerial images for 3D model reconstruction.

We use dynamic remapping to solve simulator sickness and occlusion problems in RDW. However, we will improve our manipulation algorithm in our future work by using advanced control algorithms to achieve better walking performance.

## ACKNOWLEDGMENTS

The authors thank Ligang Liu for his insightful feedback and comments. This work is supported by the Natural Science Foundation of China under Grant No. 62272018.

## References

1. Langbehn E, Lubos P, Bruder G, Steinicke F. Bending the curve: Sensitivity to bending of curved paths and application in room-scale vr. *IEEE transactions on visualization and computer graphics* 2017; 23(4): 1389–1398.
2. Abtahi P, Gonzalez-Franco M, Ofek E, Steed A. I'm a giant: Walking in large virtual environments at high speed gains. In: ACM. ; 2019: 1–13.
3. Sun Q, Patney A, Wei LY, et al. Towards virtual reality infinite walking: dynamic saccadic redirection. *ACM Transactions on Graphics (TOG)* 2018; 37(4): 1–13.
4. Hodgson E, Bachmann E. Comparing four approaches to generalized redirected walking: Simulation and live user data. *IEEE transactions on visualization and computer graphics* 2013; 19(4): 634–643.

5. Dong T, Shen Y, Gao T, Fan J. Dynamic Density-based Redirected Walking Towards Multi-user Virtual Environments. In: IEEE. ; 2021: 626–634.
6. Rietzler M, Deubzer M, Dreja T, Rukzio E. Telewalk: Towards Free and Endless Walking in Room-Scale Virtual Reality. In: ACM. ; 2020: 1–9.
7. Sun Q, Wei LY, Kaufman A. Mapping virtual and physical reality. *ACM Transactions on Graphics (TOG)* 2016; 35(4): 1–12.
8. Dong ZC, Fu XM, Zhang C, Wu K, Liu L. Smooth assembled mappings for large-scale real walking. *ACM Transactions on Graphics (TOG)* 2017; 36(6): 1–13.
9. Dong ZC, Fu XM, Yang Z, Liu L. Redirected smooth mappings for multiuser real walking in virtual reality. *ACM Transactions on Graphics (TOG)* 2019; 38(5): 1–17.
10. Cao A, Wang L, Liu Y, Popescu V. Feature guided path redirection for VR navigation. In: IEEE. ; 2020: 137–145.
11. Meng-Lin W, Popescu V. Efficient VR and AR Navigation through Multiperspective Occlusion Management. *IEEE transactions on visualization and computer graphics* 2017; 99: 1–12.
12. Wang L, Wu J, Yang X, Popescu V. Vr exploration assistance through automatic occlusion removal. *IEEE transactions on visualization and computer graphics* 2019; 25(5): 2083–2092.
13. Wang L, Chen J, Ma Q, Popescu V. Disocclusion Headlight for Selection Assistance in VR. In: IEEE. ; 2021: 216–225.
14. Razzaque S, Swapp D, Slater M, Whitton MC, Steed A. Redirected walking in place. In: . 2. Citeseer. ; 2002: 123–130.
15. Nilsson NC, Peck T, Bruder G, et al. 15 Years of Research on Redirected Walking in Immersive Virtual Environments. *IEEE Computer Graphics and Applications* 2018; 38(2): 44–56.
16. Serrano A, Martin D, Gutierrez D, Myszkowski K, Masia B. Imperceptible manipulation of lateral camera motion for improved virtual reality applications. *ACM Transactions on Graphics (TOG)* 2020; 39(6): 1–14.
17. Steinicke F, Bruder G, Jerald J, Frenz H, Lappe M. Estimation of detection thresholds for redirected walking techniques. *IEEE transactions on visualization and computer graphics* 2009; 16(1): 17–27.
18. Williams N, Peck TC. Estimation of rotation gain thresholds for redirected walking considering FOV and gender. In: IEEE. ; 2019: 1229–1230.
19. Kim D, Shin Je, Lee J, Woo W. Adjusting Relative Translation Gains According to Space Size in Redirected Walking for Mixed Reality Mutual Space Generation. In: IEEE. ; 2021: 653–660.
20. Cho YH, Min DH, Huh JS, Lee SH, Yoon JS, Lee IK. Walking Outside the Box: Estimation of Detection Thresholds for Non-Forward Steps. In: IEEE. ; 2021: 448–454.
21. Sugamoto N, Ueta K, Ujitoko Y, Sakurai S, Nojima T, Hirota K. Inclination Manipulator. In: 2019 (pp. 23–24).
22. Cheng JH, Chen Y, Chang TY, Lin HE, Wang PYC, Cheng LP. Impossible staircase: Vertically real walking in an infinite virtual tower. In: IEEE Computer Society. ; 2021: 50–56.
23. Xu SZ, Lv T, He G, Chen CH, Zhang FL, Zhang SH. Optimal Pose Guided Redirected Walking with Pose Score Precomputation. In: ; 2022.
24. Zhang SH, Chen CH, Zollmann S. One-step out-of-place resetting for redirected walking in VR. *IEEE Transactions on Visualization and Computer Graphics* 2022.
25. Xu SZ, Liu TQ, Liu JH, Zollmann S, Zhang SH. Making Resets away from Targets: POI aware Redirected Walking. *IEEE Transactions on Visualization and Computer Graphics* 2022; 28(11): 3778–3787.
26. Zhang SH, Chen CH, Zheng F, Yang YL, Hu SM. Adaptive Optimization Algorithm for Resetting Techniques in Obstacle-Ridden Environments. *IEEE Transactions on Visualization and Computer Graphics* 2023; 29(4): 2080–2092. doi: 10.1109/TVCG.2021.3139990

27. Cheng LP, Ofek E, Holz C, Wilson AD. Vroamer: generating on-the-fly VR experiences while walking inside large, unknown real-world building environments. In: IEEE. ; 2019: 359–366.
28. Yang JJ, Holz C, Ofek E, Wilson AD. DreamWalker: Substituting Real-World Walking Experiences with a Virtual Reality. In: ACM. ; 2019: 1093–1107.
29. Strauss RR, Ramanujan R, Becker A, Peck TC. A steering algorithm for redirected walking using reinforcement learning. *IEEE transactions on visualization and computer graphics* 2020; 26(5): 1955–1963.
30. Azmandian M, Grechkin T, Bolas M, Suma E. Automated path prediction for redirected walking using navigation meshes. In: IEEE. ; 2016: 63–66.
31. Hu P, Sun Q, Didyk P, Wei LY, Kaufman AE. Reducing simulator sickness with perceptual camera control. *ACM Transactions on Graphics (TOG)* 2019; 38(6): 1–12.
32. Feigl T, Roth D, Gradl S, et al. Sick moves! motion parameters as indicators of simulator sickness. *IEEE transactions on visualization and computer graphics* 2019; 25(11): 3146–3157.
33. Kennedy RS, Lane NE, Berbaum KS, Lilienthal MG. Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness. *The International Journal of Aviation Psychology* 1993; 3(3): 203-220.
34. Chen S, Miranda F, Ferreira N, et al. Urbanrama: Navigating cities in virtual reality. *IEEE Transactions on Visualization and Computer Graphics* 2021; 28(12): 4685–4699.
35. Mortenson ME. *Mathematics for computer graphics applications*. Industrial Press Inc. . 1999.
36. Bowman DA, Gabbard JL, Hix D. A survey of usability evaluation in virtual environments: classification and comparison of methods. *Presence: Teleoperators & Virtual Environments* 2002; 11(4): 404–424.
37. LaViola JJ. A Discussion of Cybersickness in Virtual Environments. *SIGCHI Bull.* 2000; 32(1): 47–56.

## 7 | AUTHOR BIOGRAPHY



**Yuan Xiong** received the B.S. degree from Beihang University in 2010 and the M.S. degree in computer science from Clemson University in 2014. He is currently pursuing the Ph.D. degree with the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, China. His research interest includes camera calibration, multiple view geometry in computer vision and virtual reality.



**Tong Chen** received the M.S. degree from North China Electric Power University, Beijing, China, in 2022. He is currently pursuing the Ph.D. degree with the School of Computer Science, Beihang University, Beijing, China. His research interests concern on computer graphics, virtual reality, 3D Reconstruction and localization.



**Tianjing Li** received the B.S. degree from the Soochow University in 2021. He is currently pursuing the master's degree with the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University, China. His research interest includes computer graphics and 3D reconstruction.



**Zhong Zhou** (Member, IEEE) received the B.S. degree from Nanjing University in 1999 and the Ph.D. degree from Beihang University, Beijing, China, in 2005. He is currently a Professor and the Ph.D. Adviser with the State Key Laboratory of Virtual Reality Technology and Systems, Beihang University. His main research interests include virtual reality/augmented reality/mixed reality, computer vision, and artificial intelligence.

**How to cite this article:** Yuan Xiong., Tong Chen, Tianjing Li, and Zhong Zhou (2023), DreamWalk: Dynamic Remapping and Multiperspectivity for Large-Scale Redirected Walking, *Comput Anim Virtual Worlds*, 2023;00:1–6.