Integrating Both Parallax and Latency Compensation into Video See-through Head-mounted Display

Atsushi Ishihara, Hiroyuki Aga, Yasuko Ishihara, Hirotake Ichikawa, Hidetaka Kaji, Koichi Kawasaki, Daita Kobayashi, Toshimi Kobayashi, Ken Nishida, Takumi Hamasaki, Hideto Mori, and Yuki Morikubo

Fig. 1: (a) Our head-mounted display prototype. (b) Results of view synthesis with disocclusion. (c) Results without disocclusion (ours). (d) Results for composition without occlusion. (e) Results of occlusion (our method).

Abstract—This work introduces a perspective-corrected video see-through mixed-reality head-mounted display with edge-preserving occlusion and low-latency capabilities. To realize the consistent spatial and temporal composition of a captured real world containing virtual objects, we perform three essential tasks: 1) to reconstruct captured images so as to match the user's view; 2) to occlude virtual objects with nearer real objects, to provide users with correct depth cues; and 3) to reproject the virtual and captured scenes to be matched and to keep up with users' head motions. Captured image reconstruction and occlusion-mask generation require dense and accurate depth maps. However, estimating these maps is computationally difficult, which results in longer latencies. To obtain an acceptable balance between spatial consistency and low latency, we rapidly generated depth maps by focusing on edge smoothness and disocclusion (instead of fully accurate maps), to shorten the processing time. Our algorithm refines edges via a hybrid method involving infrared masks and color-guided filters, and it fills disocclusions using temporally cached depth maps. Our system combines these algorithms in a two-phase temporal warping architecture based upon synchronized camera pairs and displays. The first phase of warping is to reduce registration errors between the virtual and captured scenes. The second is to present virtual and captured scenes that correspond with the user's head motion. We implemented these methods on our wearable prototype and performed end-to-end measurements of its accuracy and latency. We achieved an acceptable latency due to head motion (less than 4 ms) and spatial accuracy (less than 0.1° in size and less than 0.3° in position) in our test environment. We anticipate that this work will help improve the realism of mixed reality systems.

Index Terms—Video see-through, mixed reality, occlusion, latency compensation

1 Introduction

The mixed reality (MR) industry is rapidly expanding in fields such as entertainment and education, and is expected to play an important role in freeing users from spatial constraints by presenting virtual worlds in front of them. There are two types of MR: optical see-through (OST) and video see-through (VST). In OST, users see the real world directly through transparent displays; in contrast, VST captures the world using cameras mounted in front of the head-mounted display (HMD) and projects it onto a display. Existing commercial OSTs suffer from a narrow field of view and poor color reproducibility. VST can overcome these limitations with wider-field-of-view cameras and displays.

As Xiao et al. [72] pointed out, because the position of the cameras and users' eyes are not identical, differences in size and position arise between the objects captured and displayed on the HMD and those seen with the users' eyes. Moreover, because captured images are typically processed independently of virtual scene rendering, it is important to compose them as temporally matching [25] and to reproject the composed images such that they correspond with the user's head motion, to reduce latency [12]. When composing both captured and rendered images, the depth information must be properly managed for correct occlusion. Failure to do so results in a lack of depth cues and can cause cybersickness.

Numerous image warping and view synthesis approaches have been proposed to generate novel views from the originally captured ones. Numerous occlusion methods have also been proposed, including depth- and model-based approaches. However, these approaches are typically computationally heavy (even with high-end processors), and dense and accurate depth maps are often required, which increases the processing time. Longer processing times tend to result in larger latencies.

In this study, we integrate both parallax and latency compensation methods into a VST-HMD. To provide a spatially and temporally consistent presentation, we propose 1) the perspective-corrected reconstruction of captured images, 2) the composition of virtual and reconstructed scenes with edge-preserving occlusion, and 3) a latency compensation scheme that composes virtual and reconstructed scenes with the same timestamp and then deforms composed scenes to correspond to the user's head motions.

To reconstruct captured images, we reproject images captured by the color camera to the user's eye position, using depth maps. Reprojected images exhibit disocclusion areas, attributable to positional differences between the cameras and eyes. We fill these disocclusions using cached

• *Atsushi Ishihara, Hiroyuki Aga, Yasuko Ishihara, Hirotake Ichikawa, Hidetaka Kaji, Koichi Kawasaki, Daita Kobayashi, Toshimi Kobayashi, Ken Nishida, Takumi Hamasaki, Hideto Mori and Yuki Morikubo are with Sony Group Corporation*
E-mail: Atsushi.A.Ishihara@sony.com

Manuscript received 14 October 2022; revised 13 January 2023; accepted 30 January 2023. Date of publication 27 February 2023; date of current version 29 March 2023. Digital Object Identifier no. 10.1109/TVCG.2023.3247460

depth maps estimated in previous frames, instead of relying upon computationally heavy inpainting procedures. Our method is not only fast but also spatially correct because it is based upon observations (at least in static scenes).

For occlusion, we refine the edges of the depth maps using both infrared (IR) masks and color-guided filters. Our algorithms adopt the layered depth image (LDI) [60] approach. We separate the layers into foreground (more important for occlusion) and background (essential for disocclusion filling).

For latency compensation, we propose a two-phase temporal warping method. In the first warp, we three-dimensionally reproject the reconstructed scenes to temporally match the virtual ones. In the second warp, we use two-dimensional warping to match the composed scenes with the user's head motions. Our system synchronizes the frame starts of the stereo cameras and displays. A single timestamp generator is used to provide matching timestamps.

We tested our method in a prototype experiment, in which users see virtual objects in front of them and verify the size of those objects using their hands. We performed end-to-end measurements of the accuracy and latency.

Our technical contributions are as follows:

- LDI-based algorithms that offer perspective-corrected reconstructed scenes and virtual and reconstructed scene composition with edge-preserving occlusions
- A two-phase temporal warping architecture to ensure consistency between the virtual and reconstructed scenes and between the composed scenes and user head motions
- A wearable VST MR prototype implementation and end-to-end assessment

In this paper, we first review the related approaches (Section 2). After describing our system (Section 3), the algorithms (Section 4), and the implementation (Section 5), we explain and discuss our measurement procedure (Section 6). Finally (Section 7), we summarize our contributions and discuss future work.

2 Related Work

2.1 View Synthesis

The technique of creating a novel view from another is referred to as "image-based rendering" [61] or "novel view synthesis" [56]. As a representative example, the "view morphing" method has been proposed to generate a new image from two input images, without using depth information [59]. No black hole pixels remained after conversion; however, the distances to objects in the converted image were incorrect. Methods of creating new perspective images using depth information are sometimes called "3D warping" or "depth-based" image rendering.

Warping is classified according to the direction of reference [9]. Forward warping is relatively lightweight but prone to artifacts. It converts the source-perspective images into a destination viewpoint. Inverse warping can reference the source images to generate a destination viewpoint. Many forward warping methods have been proposed, including the merging of multi-point inputs as point clouds [37], the generation of interpolated perspectives using video and depth inputs [63], hole-filling using multi-perspective inputs [39], foreground silhouette guides [9], geometric defect compensation [14], remote rendering implementation [58], graphics processing unit (GPU) optimization [57], and real-time processing in stand-alone HMDs [15]. Inverse warping [47, 49, 54] tends to take longer than forward warping because it must search for the epipolar line, although it can eliminate holes in the converted images.

LDIs divide an image into multiple layers [38,42]. These approaches use inpainting to fill holes; however, they tend to take a long time to process. "Texture-mapped models" [33, 34, 53] have been proposed, in which multiple images are point-clouded to build a three-dimensional model and paste textures. These models are reusable but often take time to build.

In recent years, machine-learning methods have been proposed to generate multiplane images from multiple input images [24, 71, 72]. In addition, many neural radiance field-based methods have been proposed to create free-viewpoint images, using multiple images as inputs [26, 27]. These approaches are computationally demanding to run in real time.

2.2 Occlusion

Occlusion has been a subject of research since the 1990s; it can be managed using either depth- or model-based methods [35].

In depth-based methods, occlusion is handled using an estimated depth map. Methods using multi-view stereo inputs [26] and structured light [18, 32, 44, 69] have been proposed. Structured light approaches focus on closing the gaps generated by the different layouts of IR projectors and cameras. The results of these methods strongly depend on the quality of the depth maps. Methods for estimating the depth from a monocular camera image (including machine learning methods) have been proposed [19, 20, 43]. Methods that use multiple image sensor inputs have also been proposed [28, 40], as well as depth estimation methods based upon the continuity of a time series [13, 29]. Depth estimation methods that use image sensor inputs and sparse depths are known as depth completion. Sparse depths are generally generated from image sensor inputs or the outputs of other sensors [17, 23, 36, 45]. Model-based methods can be further sub-classified. Virtual phantom methods use a hand model derived from hand pose estimation and tracking results. One representative method used the estimation results of an off-the-shelf optical hand tracking module [22, 70]. Hand pose estimation methods can fail when an object is being held in the hand. Other techniques include foreground–background separation [67] , foreground–background separation using color as a cue [1,64], and methods combining segmentation and tracking [65]. All these approaches either rely upon depth sensors or are computationally heavy.

2.3 Warping

Latency compensation via post-render warping resembles view synthesis in that it produces a novel view from other view images. Many methods can deform a rendered result into an image, to reduce the delay and/or rendering load. Several classical approaches have been proposed [48, 50]. Since the mid-2000s, many methods have been proposed [11, 52, 55, 66, 74]. All these compensation methods neglect the consistency between the virtual scene and captured image. Efforts have been made to develop low-latency VST [5]; however, consistency with virtual scenes has been neglected. These approaches are too computationally demanding to run in real time.

Techniques for matching captured images with virtual scene delays have been proposed [7, 25, 41] . Freiwald et al. [25] proposed a method of warping (instead of buffering) captured images to ensure temporal consistency with virtual scenes. In addition, asynchronous timewarp [51] installed in the HMD has been applied to the captured image and virtual scene to reduce delay. However, this approach warps only with orientation and does not use a translation component.

2.4 Acceptable Latency

In VR, latency due to head motion can lead to cybersickness and poor realism [62]; ideally, latency should be less than 17–20 ms [3,12,21,46].

In contrast to VR, where the field of view is completely covered, optical see-through MR allows the real world (and deviations therefrom) to be directly observed. Blate et al. [10] used a tracking system with a reduced latency of 28 μs and found that the perceived limit of deviation was ~1 ms.

However, certain delays cannot be compensated for by warping according to head motion (e.g., the movements of others or users' hands captured in VST images). This is sometimes referred to as "capture-to-display" latency [16] or latency due to scene motion [41], and its acceptable value is undetermined. For example, Attig et al. [6] argued that latencies of less than 100 ms for shape and texture changes in the object do not affect interactions. Gruen et al. [30] measured the capture-to-display latency of several commercial VST-HMDs and reported it to be between 50 and 60 ms.

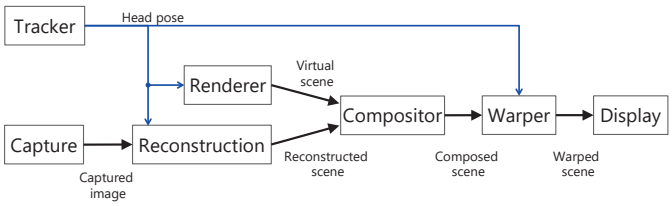

Fig. 2: Block diagram of our system.

3 System Overview

3.1 Aims and Requirements

We developed a VST-HMD prototype to explore new MR values (besides the features). We focus on presenting a correct depth cue within users' personal space via a VST-HMD, because intuitive hand-eye coordination is necessary in an MR space. Therefore, we designed our system to achieve this under the following three requirements:

- Wearable form factor: The VST-HMD should be small and lightweight enough to be worn by users as they walk around within a room (at least 3 × 3 m). The HMD can be tethered like PC-based VR HMDs, however, no other equipment fixed to the room should be needed as the entire system must be moved from room to room.
- Low-latency: The latency of the VST-HMD should be sufficiently short to prevent cybersickness.
- Made of COTS: The VST-HMD should consist of a commercial off-the-shelf module (except for the near-eye display module, which consists of a display panel and optics). We accepted the display's dimension and refresh rate as given conditions for the system design due to availability.

3.2 Major Blocks

Before describing our approach, we explain the terminology and functions of the seven major blocks in our VST-HMD system, as shown in Figure 2. The tracker block estimates the head pose by applying the visual–inertial odometry method to data obtained from cameras and an inertial measurement unit (IMU). The capture block captures a real scene in front of the HMD using color and depth image sensors. The reconstruction block estimates the scene depth from the captured images and integrates it into the reconstructed scene (i.e., an internal 3D representation). The renderer block renders virtual objects (as viewed from the tracked head pose) as virtual scene images. The composite block composes the virtual and reconstructed scenes in the composed image. The warper block applies image warping to the composed image, to match the head motions and correct the distortion of the display optics. The display block presents a warped image of the user's eyes, using near-eye display panels and optics.

3.3 Our Approach

In this section, we describe our primary approach. As discussed in Section 2, most view synthesis and occlusion approaches are computationally heavy; meanwhile, low latency is an indispensable requirement in MR.

We establish acceptable latency as the main condition to be maintained when considering our algorithm. As described in Section 2.4, two types of latencies are possible: latency due to head motion and latency due to scene motion. VST includes two types of latency due to head motion: one arises between virtual and reconstructed scenes, for which we set the acceptable latency to 2 ms (because it is visible and easy to notice); the other describes the scene's latency to (actually invisible) real scenes, which should be less than 17 ms, according to previous studies. We set the acceptable latency due to scene motion as 50–60 ms (a relatively poor sensitivity), following studies by Attig et al. [6] and Gruen et al. [30] .

Our approach is composed of two key ideas:

1. to reconstruct captured images within acceptable latencies due to scene motion
2. to compensate for the latency due to head motion

Reconstruction of Captured Images To maintain capture-to-display latency due to scene motion, lightweight reconstruction algorithms of an acceptable quality are required. Rather than improving the quality of the entire depth map, we focus on edge refinement and disocclusion filling. Our reconstruction method divides the captured images into foreground and background images. We define the foreground region as personal space. Users' hands and the objects held by them move around as occluders. The edges of occluders affect the occlusion results. We employed a hybrid method using IR masks and color-guided filters to refine the edges. We assumed that the background region was a static environment and rapidly filled the disocclusion with cached depth maps accumulated from past frames.

Latency Compensation For latency due to head motion, we used two different types of warping algorithms: (1) three-dimensional warping in the composition block, to reduce registration errors between captured and virtual scenes (because users are sensitive to this error) and (2) two-dimensional warping, to reduce errors between the composed scenes when the captured and virtual scenes are combined and corresponded to head motions (this error is minor compared to that between captured and virtual scenes). Moreover, the processing time of the warping block should fall within a certain time limit (the threshold for judder-free frame updates). This block compensates for latency and interpolates frames using the last composed block.

3.4 Timing Budget

Here, we detail the timing budget of our system, which includes the frame rate and processing time. As mentioned in Section 3.1, we implemented a given display panel with an 80 Hz refresh rate.

We established the timing budget as below, based upon our display panel's specifications and an acceptable latency due to scene motion. A frame interval of 12.5 ms was applied for the capture component (i.e., from the center of exposure to until the completed transfer of the captured image), the rendering component (i.e., to render virtual scenes), and the combined reconstruction and compositor component (i.e., to reconstruct captured images and composite the reconstructed and virtual scenes, including edge refinement and disocclusion filling). The capture component was run at 80 Hz. Rendering, reconstruction, and composite components were processed sequentially, and this series ran at 40 Hz. We created a warper component that ran at 80 Hz to interpolate frames and compensate for the latency. Because our system involves line-scanning displays and rolling shutter cameras for capture, pixels are processed from top to bottom. Even though the processed timing differed between the top and bottom pixels, we considered the timing of pixels in a certain line in both the captured and displayed images to be roughly the same within a frame. As a result, the latency due to scene motion was at least 50 ms and at most 62.5 (in cases of interpolated frames) in the absence of frame drops.

4 Proposed Pipeline

4.1 Definitions

Here, we define the common terms and notations used in following sections.

Device X on our HMD prototype (shown in Figure 3) can refer to the left and right sides of the tracker, color, and depth stereo cameras, respectively, denoted as *sl* and *sr*, *cl* and *cr*, or *dl* and *dr*. X can also refer to the eye-box center of the left or right side of the binocular display, which can be treated as a quasi-device and is denoted as *el* or *er*. The left or right side of each stereo camera or the eye-box of the display defines its own frame of reference. When we do not need to distinguish between the left and right sides of these devices, *s*, *c*, *d*, or *e* can be used in isolation.

The transformation matrix ${}_B\boldsymbol{T}_A$ represents the transformation of a point from frame of reference A to frame of reference B. It has the form $\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix}$ where $\boldsymbol{R}$ is a 3×3 matrix that represents the rotation, and $\boldsymbol{t}$ is

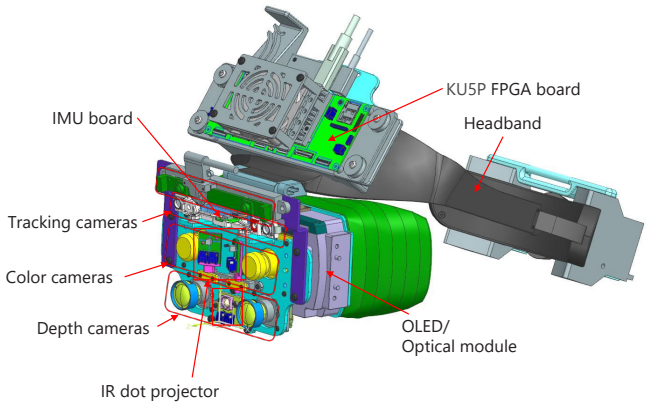

Fig. 3: Components of our prototype.

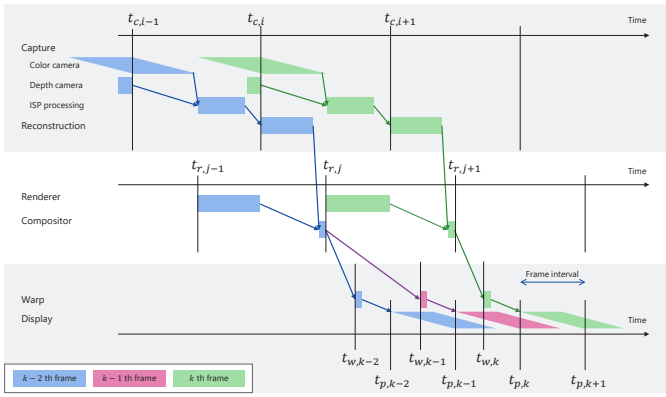

Fig. 4: Processing timeline.

a 3×1 vector that represents the translation. In addition to the frame of reference formed by device X, we can also use a stationary frame of reference, often referred to as the world frame, denoted as w.

The pose of device X, denoted as $\boldsymbol{V}_X$, refers to its orientation and position relative to a reference frame, which can be represented using a transformation matrix with rotation component $\boldsymbol{R}$ and translation component $\boldsymbol{t}$. The rotation component describes the orientation of device X, while the translation component describes its position.

The projection matrix of device X, which can be either a camera or the eye-box of the display, is denoted as $\boldsymbol{P}_X$, which has the form $\begin{bmatrix} \boldsymbol{K} & 0 \\ 0 & 1 \end{bmatrix}$ where $\boldsymbol{K}$ is a 3×3 matrix that represents the intrinsic matrix defined in the pinhole camera model.

The time $t_{Y,Z}$ denotes the time to start the processing of Y in frame index Z. The processing Y can refer to capturing, rendering, warping, or presenting, and is denoted as g,r,w, or p, respectively, as shown in Figure 4. The frame index Z can be i,j, or k for capturing, rendering, or presenting, respectively, as the frame intervals or phases for these processes are different.

The color, IR intensity, or depth image can be represented by functions C, L, or D, respectively, which map normalized pixel coordinates (x,y) to pixel values that represent color $\boldsymbol{c}$, IR intensity l, or depth d. In the depth image, a pixel with a depth value of zero signifies a lack of corresponding points, even though the depth image represents a point cloud.

Note that the image referred to here is typically a stereo pair of images. Unless stated otherwise, the left and right images are processed independently by image operators.

The time series of images generated periodically is denoted with its frame index. (e.g., the series of color images captured at $t_{g,i}$ is denoted as $C_{g,i}$)

The image is associated with the view, which is a combination of the projection matrix $\boldsymbol{P}_X$ and the pose $\boldsymbol{V}_X[t_{Y,Z}]$ of the device used to generate it. For example, the color image captured by the color camera is associated with the view $(\boldsymbol{P}_c, \boldsymbol{V}_c[t_{g,i}])$, which is denoted concisely as $\mathbb{C}_i = (\boldsymbol{P}_c, \boldsymbol{V}_c[t_{g,i}])$.

When the color image C and the depth image D are associated with the same view, the tuple $(C_{\mathbb{X}}, D_{\mathbb{X}})$ can be denoted as $\mathfrak{B}_{\mathbb{X}}$.

The reprojection operator converts the depth image D_A in view at the time $\mathbb{A} = (\boldsymbol{P}_A, \boldsymbol{V}_A)$ to another depth image D_B in view at the time $\mathbb{B} = (\boldsymbol{P}_B, \boldsymbol{V}_B)$. It is denoted as $D_B = {}_B\mathcal{Q}_A(D_A)$ where D_A and D_B are depth images, and ${}_B\boldsymbol{H}_A$ is the reprojection matrix that maps points in D_A to corresponding points in D_B.

This mapping can be represented as $\boldsymbol{p}_b = {}_B\boldsymbol{H}_A \cdot \boldsymbol{p}_a$ where $\boldsymbol{p}_a = (x,y,z,1)$ and $\boldsymbol{p}_b = (x',y',z',w')$ are vectors, and ${}_B\boldsymbol{H}_A = \boldsymbol{P}_B \cdot \boldsymbol{V}_B \cdot \boldsymbol{V}_A^{-1} \cdot \boldsymbol{P}_A^{-1}$. When (x,y) are the coordinates of a point in the depth image D_A, z is a depth value at that point $D_A(x,y)$. When $(x'/w', y'/w')$ are the coordinates of a point in the depth image D_B, z'/w' is a depth value at that point $D_B(x'/w', y'/w')$.

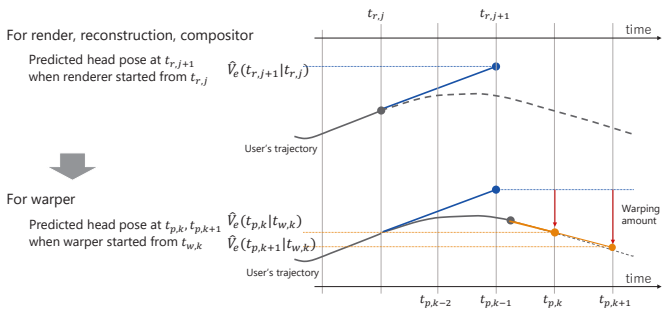

Fig. 5: Predicted head pose on rendering and on warping.

4.2 Tracker

The tracker block estimates the pose of device X at the current time t_0 and also predicts the pose at a future time t by extrapolating the estimated results until now. These poses are denoted as $\hat{\boldsymbol{V}}_X[t_0]$ and $\hat{\boldsymbol{V}}_X(t|t_0)$, respectively.

To do this, the tracker block first estimates or predicts the camera pose $\hat{\boldsymbol{V}}_{sl}$ using inputs from the stereo cameras sl and sr and the IMU, using a visual-inertial odometry (VIO) method similar to that described in [4]. Then, the estimated or predicted camera poses are converted to the pose of device X using the pre-calibrated transform matrix ${}_X\boldsymbol{T}_{sl}$.

4.3 Renderer

The renderer block generates a color and depth image, denoted as $(C_{v,j}, D_{v,j})$, by rendering a virtual scene from the eye space in the predicted eye pose at the time to start rendering next frame.

This view is denoted as $\mathbb{R}_j = (\boldsymbol{P}_e, \hat{\boldsymbol{V}}_e(t_{r,j+1}|t_{r,j}))$, and the generated color and depth images of the j-th frame can be represented as $\mathfrak{B}_{v,j} = (C_{v,j}, D_{v,j})$.

4.4 Capture

This block captures a real scene in front of the HMD using two types of cameras. A pair of color cameras is used to capture the visible image and a pair of IR cameras with an IR dot projector are applied for depth sensing (referred to as a depth camera) . All cameras start their exposures at the same time, although the exposure duration differs between the color and IR cameras.

Raw images taken by the stereo color camera are developed and stereo-rectified. The processed result is denoted as the grabbed color image $C_{g,i}$ viewed from $\mathbb{C}_i = (\boldsymbol{P}_c, \boldsymbol{V}_c[t_{g,i}])$.

When the stereo IR camera starts its exposure, the IR dot projector is triggered to flash . The emitted IR dot patterns are reflected on a real-world surface and grabbed via a stereo IR camera. These raw images are developed and stereo-rectified. The processed result of the i-th frame is the IR intensity image, $L_{g,i}$ viewed from $\mathbb{D}_i = (\boldsymbol{P}_d, \boldsymbol{V}_d[t_{g,i}])$.

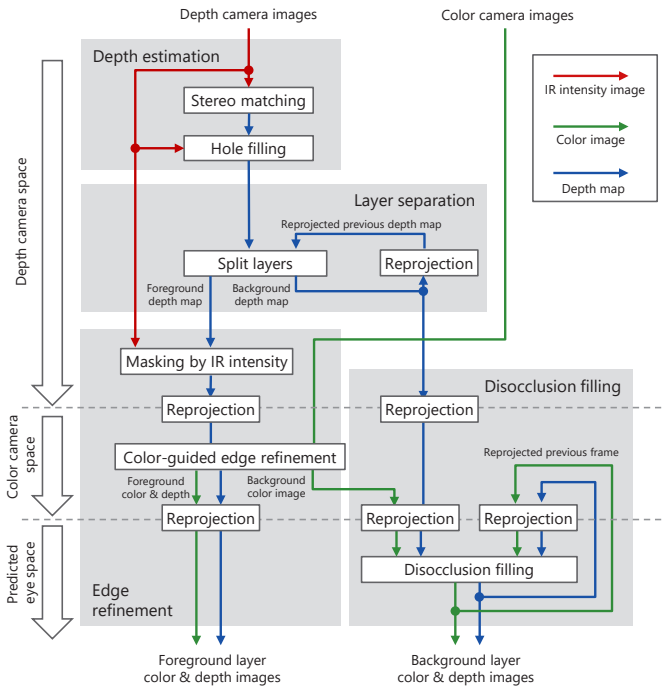

Fig. 6: Reconstruction pipeline.

4.5 Reconstruction

The reconstruction block consists of four sub-blocks. Figure 6 presents an overview of the pipeline from capture to reconstruction. The details of each sub-block are described as follows:

4.5.1 Depth Estimation

The depth estimation sub-block estimates the depth maps from the captured stereo IR image $L_{g,i}$; it involves two steps: stereo matching and hole filling.

Stereo Matching Our system uses libSGM [2] (a GPU-accelerated implementation of a semi-global matching algorithm [35]), because its processing time is sufficiently short (3 ms per frame in our system), and its recall is sufficiently high if a random IR dot pattern is projected on the real surface. This step creates a base depth map $D_{s,i}$ from the captured IR image $L_{g,i}$.

Hole Filling We configured the emission intensity of the IR dot projector to cover a measurement range of up to 6 m; this constraint is a result of our testing environments.

However, the side effects of IR emission make it too strong for closer surfaces, and the reflected dot patterns are saturated on the IR intensity image. The stereo-matching algorithm cannot function for such saturated regions.

A hole-filling operator f_h is introduced to improve these regions, by replacing the hole with the closest successful stereo-matched pixel in the neighborhood, $\mathcal{N}(\boldsymbol{p})$ is a set of pixel coordinates neighbor to the coordinate $\boldsymbol{p}$ that satisfy the conditions $D_{s,i}(\boldsymbol{p}) > 0$ and $L_{g,i}(\boldsymbol{p}) < k_{sat}$ (as shown in Figure 7). This process can be represented as:

$$f_h(\boldsymbol{p}) = \begin{cases} D_{s,i}(\boldsymbol{q'}) & \text{if } D_{s,i}(\boldsymbol{p}) = 0 \text{ or } L_{g,i}(\boldsymbol{p}) < k_{sat} \\ D_{s,i}(\boldsymbol{p}) & \text{otherwise} \end{cases}$$

where $D_{s,i}$ and $L_{g,i}$ are short hands of $D_{s,i}[\mathbb{D}_i]$ and $L_{g,i}[\mathbb{D}_i]$ respectively. $\boldsymbol{p}, \boldsymbol{q'}$ are 2D pixel coordinates on the base depth map $D_{s,i}$ and the IR intensity image L_g^i. $\boldsymbol{q'}$ is the closest successful stereo-matched neighbor to $\boldsymbol{p}$, which is $\underset{\boldsymbol{q}\in\mathcal{N}(\boldsymbol{p})}{\operatorname{argmin}} |\boldsymbol{p}-\boldsymbol{q}|$.

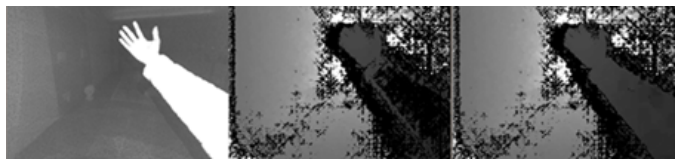

Fig. 7: Hole filling result: (left) IR intensity image, (center) depth map before hole filling, and (right) depth map after hole filling.

4.5.2 Layer Separation

This sub-block layer separation splits the hole-filled depth image $D_{h,i}$ into a foreground layer depth image $D_{f,i}$ and background layer depth image $D_{b,i}$.

The basic concept is to compare the depths of each pixel in the depth map between the current and previous frames. If the current pixel is closer (further away) than the previous one, it should be classified as the foreground (background) layer, because it indicates the arrival (removal) of an occluder.

However, the depth map for the previous frame cannot be directly compared because it is viewed from the previous camera pose. Therefore, we reproject the depth map from the previous camera pose $\mathbb{D}_{i-1} = (\boldsymbol{P}_d, \boldsymbol{V}_d[t_{g,i-1}])$ to the current one $\mathbb{D}_i = (\boldsymbol{P}_d, \boldsymbol{V}_d[t_{g,i}])$ before comparison, as follows:

$$D_{h,i-1}[\mathbb{D}_i] = {}_{\mathbb{D}_i}\mathcal{Q}_{\mathbb{D}_{i-1}}(D_{h,i-1}[\mathbb{D}_{i-1}])$$

Then, we apply the current and reprojected previous depth images, $D_{h,i}[\mathbb{D}_i]$ and $D_{h,i-1}[\mathbb{D}_i]$ respectively, to the split layer (SL) operator, and it returns a tuple of the foreground and background layers of the depth map $(D_{f,i}[\mathbb{D}_i], D_{b,i}[\mathbb{D}_i])$. The operator f_s can be defined as follows:

$$f_s(\boldsymbol{p}) = \begin{cases} (d_i, d_{i-1}) & (d_i - d_{i-1} < -k_{th}) \\ (0, \alpha d_i + (1-\alpha) d_{i-1}) & (|d_i - d_{i-1}| < k_{th}) \\ (0, d_i) & \text{otherwise} \end{cases}$$

where d_i and d_{i-1} are depth values at point $\boldsymbol{p} = (x, y)$ in depth maps $D_{h,i}[\mathbb{D}_i]$ and $D_{h,i-1}[\mathbb{D}_i]$, respectively. k_{th} is a threshold value to split, and α is a weight to blend current and previous depth values.

It is important to note that the depth of the region occluded by the foreground layer in the background layer is preserved, as we assume that the background layer is mostly stable and its depth remains unchanged during occlusion.

4.5.3 Edge Refinement

This edge refinement sub-block creates color and depth images of the foreground layer with plausible edges, using the depth image of the foreground layer $D_{f,i}[\mathbb{D}_i]$, the captured color image $C_{g,i}[\mathbb{C}_i]$, and the IR image from the depth camera $L_{g,i}[\mathbb{D}_i]$.

The estimated and hole-filled depth map tends to dilate the occluder's edge regions because regions around the edge of the curved surface exhibit blind spots that cannot be observed from one of the stereo depth cameras; this causes stereo-matching failures. As a result, the foreground depth map appears dilated.

Therefore, we introduced two methods to refine the edges: IR-intensity based masking and color-guided filtering.

Masking by IR intensity The IR dot projector is a type of spotlight, and its radiance decreases with respect to the distance to the real-world surface. Therefore, some of this distance information is reflected in each pixel value of the IR image captured by the depth camera.

This is not sufficient to use as a depth map, but we can exploit it to improve the falsely dilated edge regions by intersecting it with the mask, which is generated from the IR depth camera image using the pre-configured threshold.

Therefore, we introduce masking operator f_m that takes the foreground depth image $D_{f,i}[\mathbb{D}_i]$ and IR intensity image $L_{g,i}[\mathbb{D}_i]$ as inputs

and generates the masked foreground depth map $D_{fm,i}[\mathbb{D}_i]$, which can be represented as follows:

$$f_m(\boldsymbol{p}) = \begin{cases} D_{f,i}(\boldsymbol{p}) & (L_{g,i}(\boldsymbol{p}) < k_{th}) \\ 0 & \text{otherwise} \end{cases}$$

where $\boldsymbol{p} = (x, y)$ is a point in the image, $D_{f,i}(\boldsymbol{p})$ is the depth value at point $\boldsymbol{p}$, and $L_{g,i}(\boldsymbol{p})$ is the IR intensity value at point $\boldsymbol{p}$. The threshold value k_{th} is a constant that determines when the operator returns the depth value or zero.

We tuned the masking threshold to distinguish objects inside the personal distance [31] (mostly the users' hands or arms) from static backgrounds.

Depth contour refinement with color guide The depth contour refinement filter is used to improve misalignments or jagged edges between the depth and color images that result from reprojection errors and differences in image resolution. The process can be described as follows:

Firstly, the masked foreground depth image $D_{fm,i}[\mathbb{D}_i]$ is reprojected to the same view of the color camera, as follows:

$$D_{fm,i}[\mathbb{C}_i] = {}_{\mathbb{C}_i}\mathcal{Q}_{\mathbb{D}_i}(D_{fm,i}[\mathbb{D}_i])$$

Now, we have the color and depth images $(C_{g,i}[\mathbb{C}_i], D_{fm,i}[\mathbb{C}_i])$ as viewed from the same color camera pose, and we denote this tuple as $\mathfrak{B}_{fm,i}[\mathbb{C}_i]$.

Secondly, the depth contour refinement filter, denoted as f_r, is applied to this tuple, producing a refined tuple of the foreground depth and color images.

$$\mathfrak{B}_{fr,i}[\mathbb{C}_i] = f_r(\mathfrak{B}_{fm,i}[\mathbb{C}_i])$$

Thirdly, the color image is masked with the refined foreground depth image to extract the background color image (this image operation is denoted as f_e):

$$C_{be,i}[\mathbb{C}_i] = f_e(C_{g,i}[\mathbb{C}_i], D_{fr,i}[\mathbb{C}_i])$$

Fourthly, the refined tuple of the foreground depth and color images is reprojected to the view matched with the renderer. This process can be represented as:

$$\mathfrak{B}_{fr,i}[\mathbb{R}_j] = {}_{\mathbb{R}_j}\mathcal{Q}_{\mathbb{C}_i}(\mathfrak{B}_{fr,i}[\mathbb{C}_i])$$

where $D_{fm,i}[\mathbb{C}_i]$ is the masked foreground depth map reprojected to the color camera view, $C_{fr,i}[\mathbb{C}_i]$ is the refined foreground color image, $C_{br,i}[\mathbb{C}_i]$ is the background color image, $\mathfrak{B}_{fr,i}[\mathbb{R}_j]$ is the refined tuple of the foreground depth and color images.

The depth contour refinement filter f_r takes a tuple of the color and depth images as input, denoted as (C_{fm}, D_{fm}), and the resulting tuple of color and depth images. It can be represented as follows:

$$f_r(\boldsymbol{p}) = \begin{cases} (\bar{C}_{fm}(\boldsymbol{p}), \bar{D}_{fm}(\boldsymbol{p})) & \text{if} |C_{fm}(\boldsymbol{p}) - \bar{C}_{fm}(\boldsymbol{p})| < k_{th} \\ (C_{fm}(\boldsymbol{p}), D_{fm}(\boldsymbol{p})) & \text{otherwise} \end{cases}$$

where $\boldsymbol{p} = (x, y)$ is a point in these images.

For a set of pixels $\mathcal{N}(\boldsymbol{p})$ that are neighbors of pixel $\boldsymbol{p}$, the average color and the average depth of the neighbors are denoted as $\bar{C}_{fm}(\boldsymbol{p})$ and $\bar{D}_{fm}(\boldsymbol{p})$ respectively, and they can be represented as: $\frac{1}{|\mathcal{N}(\boldsymbol{p})|}\sum_{\boldsymbol{q}\in\mathcal{N}(\boldsymbol{p})} C_{fm}(\boldsymbol{q})$, $\frac{1}{|\mathcal{N}(\boldsymbol{p})|}\sum_{\boldsymbol{q}\in\mathcal{N}(\boldsymbol{p})} D_{fm}(\boldsymbol{q})$ The threshold value k_{th} is a constant that determines when the operator replaces the depth value with the average depth of the neighbors.

The filter f_e can be defined as follows:

$$f_e(\boldsymbol{p}) = \begin{cases} 0 & (D_{fr,i}(\boldsymbol{p}) > 0) \\ C_{g,i}(\boldsymbol{p}) & \text{otherwise} \end{cases}$$

where $C_{g,i}$ is a color image to be masked, $D_{fr,i}$ is a depth image used as a mask, and $\boldsymbol{p} = (x, y)$ is a point in these images.

Some of the results with and without the filter are shown in Figure 8.

Fig. 8: Edge refinement results (left: before refinement; right: after refinement).

4.5.4 Disocclusion Filling

This sub-block takes the background depth map $D_{b,i}[\mathbb{D}_i]$ and color image $C_{be,i}[\mathbb{C}_i]$, as inputs, and fills the occluded region with the foreground layer, using the previous background color and depth images, and generates the color and depth images at the same view with the renderer, which is denoted as $\mathfrak{B}_{bd,i}[\mathbb{R}_j]$. This process can be represented as follows:

Firstly, the background layer depth image $D_{b,i}[\mathbb{D}_i]$ is reprojected to the view matched with the color camera via

$$D_{b,i}[\mathbb{C}_i] = {}_{\mathbb{C}_i}\mathcal{Q}_{\mathbb{D}_i}(D_{b,i}[\mathbb{D}_i])$$

Now, we have the color and depth images $(C_{be,i}[\mathbb{C}_i], D_{b,i}[\mathbb{C}_i])$ as viewed from the same color camera pose, and we denote this tuple as $\mathfrak{B}_{be,i}[\mathbb{C}_i]$.

Secondly, this color and depth images are reprojected to the view matched with the renderer, as follows:

$$\mathfrak{B}_{be,i}[\mathbb{R}_j] = {}_{\mathbb{R}_j}\mathcal{Q}_{\mathbb{D}_i}(\mathfrak{B}_{be,i}[\mathbb{C}_i])$$

Thirdly, the color and depth images filled by the disocclusion filter in the previous frame, denoted as $\mathfrak{B}_{be,i-1}[\mathbb{C}_{i-1}]$, are reprojected to the current eye space for disocclusion, as follows:

$$\mathfrak{B}_{bd,i-1}[\mathbb{R}_j] = {}_{\mathbb{C}_i}\mathcal{Q}_{\mathbb{C}_{i-1}}(\mathfrak{B}_{bd,i-1}[\mathbb{R}_{j-1}])$$

Finally, the occluded region at the background layer color depth image in the current frame $\mathfrak{B}_{be,i}[\mathbb{R}_j]$ is filled with the corresponding pixels in the previous frame $\mathfrak{B}_{be,i-1}[\mathbb{R}_j]$ by the disocclusion filter, denoted as f_d, as follows:

$$\mathfrak{B}_{bd,i}[\mathbb{R}_j] = f_d(\mathfrak{B}_{be,i}[\mathbb{R}_j], \mathfrak{B}_{bd,i-1}[\mathbb{R}_j])$$

The disocclusion filter f_d takes the color and depth images in the current frame and the reprojected previous frame as inputs, denoted as C_c, D_c, C_p, and D_p, respectively, and returns a tuple of the color and depth images, denoted as C_o and D_o. It can be represented as follows:

$$D_o(\boldsymbol{p}) = \begin{cases} \alpha D_c(\boldsymbol{p}) + (1-\alpha) D_p(\boldsymbol{p}) & \text{if } D_c(\boldsymbol{p}), D_p(\boldsymbol{p}) > 0 \\ D_c(\boldsymbol{p}) & \text{else if } D_p(\boldsymbol{p}) = 0 \\ D_p(\boldsymbol{p}) & \text{otherwise} \end{cases}$$

$$C_o(\boldsymbol{p}) = \begin{cases} C_c(\boldsymbol{p}) & \text{if } D_c(\boldsymbol{p}) > 0 \\ C_p(\boldsymbol{p}) & \text{otherwise} \end{cases}$$

where $\boldsymbol{p} = (x, y)$ is a pixel coordinate, $D_c(\boldsymbol{p})$, $D_p(\boldsymbol{p})$, and $D_o(\boldsymbol{p})$ are depth values at point p in depth images D_c, D_p, and D_o, respectively. $C_c(\boldsymbol{p})$, $C_p(\boldsymbol{p})$, and $C_o(\boldsymbol{p})$ are color values at point p in color images C_c, C_p, and C_o, respectively.

It is important to note that the depth value for a given pixel in the current and previous frames is averaged, if both values are available, because we assume that the background layer is mostly stable, and its depth remains unchanged over the frame interval.

4.6 Composition

This block composes the color and depth images of the reconstructed foreground $\mathfrak{B}_{fr,i}[\mathbb{R}_j]$, the reconstructed background $\mathfrak{B}_{bd,i}[\mathbb{R}_j]$, and the rendered virtual scenes $\mathfrak{B}_{v,j}[\mathbb{R}_j]$. Occlusion between the virtual and reconstructed scenes is realized by depth testing between input images. The resulting composition is denoted as $C_{m,j}[\mathbb{R}_j]$.

4.7 Warper

This warper block applies image warping (referred to as "temporal warping") to the composed scene $C_{m,j}[\mathbb{R}_j]$, to correct the prediction error of the user's head pose used at rendering $\boldsymbol{V}_e(t_{r,j+1}|t_{r,j})$ with the latest one predicted just before display scanning $\boldsymbol{V}_e(t_{p,k}|t_{w,k})$. And this warper corrects the distortion produced by the display optics. The warped results are sent to the display panel.

Temporal warping is a type of inverse warping in which the source image $C_{m,j}[\mathbb{R}_j]$ is assumed to be located on a plane (a "stabilization plane") but its destination plane is twisted by user head movements as the display scans from top to bottom.

This twisted destination mesh is formed by independently reprojecting the upper and lower corners of the composed scene. The upper left and right corners of the composed scene are re-projected to the eye space as viewed from the predicted eye pose at the start of display scanning $\boldsymbol{V}_e(t_{p,k}|t_{w,k})$. The lower left and right corners are reprojected to the eye space as viewed from the predicted eye pose at the end of display scanning $\boldsymbol{V}_e(t_{p,k+1}|t_{w,k})$, as shown in Figure 8.

5 Implementation

In this section, we describe the implementation of our method. We implemented this approach in a wearable prototype HMD that allowed the wearer to walk around a space of ~3 × 3 m. Figure 9 shows a block diagram of the prototype system. To keep the HMD lightweight, heavy processing tasks (e.g., reconstruction and rendering) were handled by a desktop PC connected to the device. Data collected from sensors mounted on the front of the HMD prototype were sent to the desktop PC, where they were processed to generate an image and returned to the prototype for display.

Sensors and Projectors The prototype had three stereo camera pairs. The first was for VIO; it consisted of global shutter image sensors (each with a 640 × 480 resolution) and a lens with an IR cut filter. The second was for depth estimation; it consisted of global shutter image sensors (each with a 1440 × 1080 resolution) and a lens with an IR bandpass filter. We used this camera pair for depth measurement at a resolution of 720 × 540 in the binning mode. The third was for video see-through capture; it consisted of rolling shutter image sensors (each with a 4056 × 3044 resolution). We used this camera pair for video see-through at a 2028 × 1522 resolution in the binning mode. All camera sensors were operated at 80 Hz.

To add feature points in texture-less areas, the prototype featured two IR dot projectors; these were placed side-by-side at an angle to the main optical axis, to obtain a wider field of view (FoV); it emitted 15372 dots in a H93° × V70° FoV.

For VIO, our prototype also featured a gyro sensor running at 1600 Hz and an accelerometer sensor that ran at 1100 Hz but was up-sampled to 1600 Hz.

We used three stereo cameras to obtain the best results for each objective. For example, the baseline of the stereo camera pair for VIO was wider than that for the depth estimation. Before use, the position of each stereo camera and display was measured and calibrated via image processing.

Displays The prototype was equipped with a near-eye display system using a Fresnel lens at H99° × V105° FoV. It used an organic light emitting diode display panels with a resolution of 1920 × 1920 per eye; these were line scanning displays that ran at 80 Hz. All cameras and displays were synchronized with the vertical sync signal, to obtain the corresponding images from different cameras and display them.

HMD Controls The prototype was equipped with a field programmable gate array (FPGA) board Xilinx KU5P to control the sensors and displays. All cameras and displays were synchronized with the vertical sync signal, to obtain corresponding images from different cameras and display them. The cameras and IMUs signals were timestamped by the FPGA and transferred to the PC. This timestamp was generated using the same clock source.

This FPGA board also handled the data transfer, which proceeded via a multi-fibre push-on cable between the prototype and PC. The data transfer rate was 5.84 Gbps from the HMD to the PC, and 14.16 Gbps from the PC to the HMD.

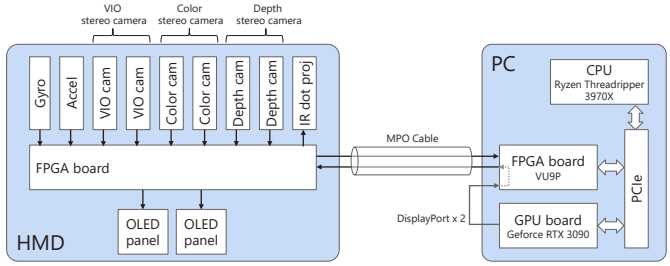

Fig. 9: Our prototype system block diagram.

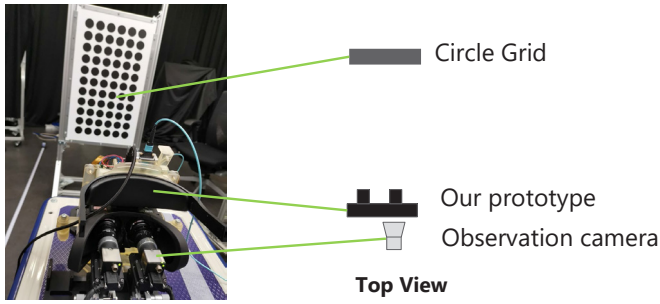

Fig. 10: Setup for measuring perspective correctness.

PC as processing box The PC connected to the prototype was equipped with an AMD Ryzen Threadripper 3970X central processing unit (CPU), NVIDIA Geforce RTX 3090 Graphics board, and Xilinx VU9P FPGA board. All image processing was executed on a CPU or GPU using DirectX12 or compute unified device architecture. The generated result was sent from the GPU and relayed via the FPGA board to the prototype through the MPO cable.

6 Experiments

6.1 Test Environments

We assume that our test environments are indoor office rooms, which have an area of ~3 × 3 m. Although our prototype works in typical office rooms with white walls, we used a black curtain or gray panels on some regions of the walls, to avoid strong IR reflection and subsequent inaccurate depth estimations.

6.2 Perspective-correctness

We measured the perspective correctness of the reconstructed scene. The experimental setup is illustrated in Figure 10. A circular grid was placed in front of our prototype as an object of known size. A circular grid is often used to calibrate cameras. We placed our observation camera (UI-3480CP-M-GL [86]) at the position of the user's eye. This position is where the circular grid can be most clearly seen through the display from the observation camera. We marked these positions to ensure that the cameras are placed in the same position each time. The position of the observation camera and those of the video see-though cameras were calibrated in advance.

The experiment comprised two steps: 1) The prototype captured the circular grid and displayed it. The observation camera then captured the display of the prototype. 2) We removed the prototype so that the observation camera could capture the circle grid directly, to obtain ground truth images. We compared the results of Steps 1 and 2. We binarized captured images and detected circles from them using the "findContour" function of OpenCV [85], and we calculated the center of each circle. We compared the absolute and relative positions of the centers of the captured images.

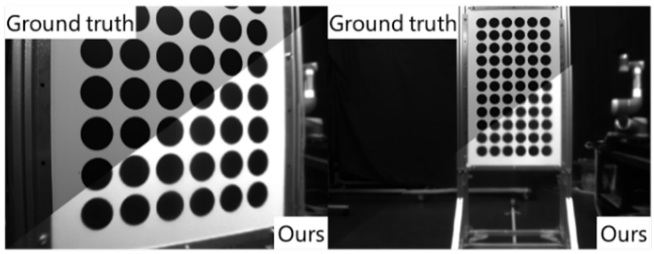

Fig. 11: Circle grid as seen by observation camera with and without prototype. The circular grid was placed (left) 60 cm and (right) 180 cm from the prototype.

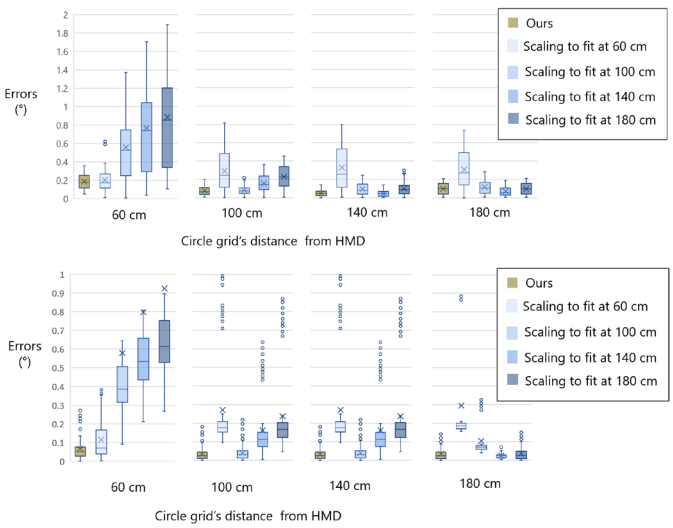

Fig. 12: Accuracy measurement results: (upper) position errors from circle centers to ground truth, which are errors in position; (lower) distance errors between circles and ground truth, which are errors in size.

These steps were performed ten times under different positions of the circular grid and at different distances (Figure 11). We also compared the results of our method with a two-dimensional scaling of the captured images at each distance. For example, images were two-dimensionally scaled to fit the sizes of objects at 60 cm, and the sizes of objects at other distances were incorrect. As shown in Figure 12, the two-dimensional scaling results were resembled the ground truth at only one certain distance, whereas ours resembled the ground truth at any distance (less than 0.3° in position and less than 0.1° in size).

6.3 Occlusion

We measured the accuracy of the proposed occlusion algorithm. We recorded three different occlusion scenarios: (a) the back of a hand, (b) the palm of a hand, and (c) a note and pen held by hands. We took five different frames for each scene, each with different poses for the foreground objects. In each frame, we manually created a reference mask. We made our prototype render a blue square and compared the occlusion results of our algorithm against those of the reference masks. We adopted the commonly used precision and recall metrics [8] [68]. The following values are all averaged over five frames: The precision of Scene (a) was 0.9645 and the recall was 0.9836. The precision of Scene (b) was 0.9427, and the recall was 0.9938. The precision of Scene (c) was 0.9185, and the recall was 0.95 (Figure 13).

We also observed scenes in which the users pretended to hold a virtual camera. Although our occlusion masks occluded fingers in more distant areas whilst retaining fingers in nearer ones, there were noticeable artifacts.

To understand how the color contrast between foreground objects

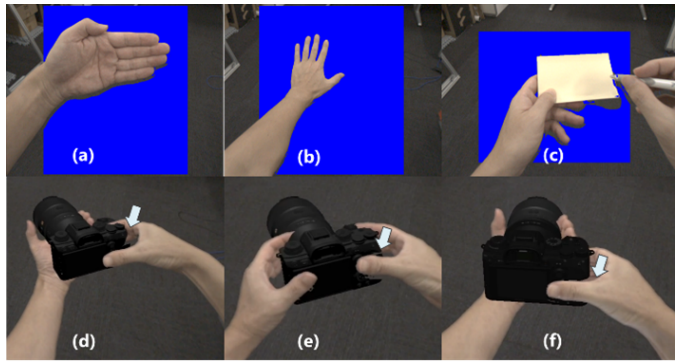

Fig. 13: Occlusion by (a) palm of hand, (b) back of hand, (c) hand-held note and pen. Scenes (d), (e), and (f) show a user holding a virtual camera. White arrows indicate artifacts.

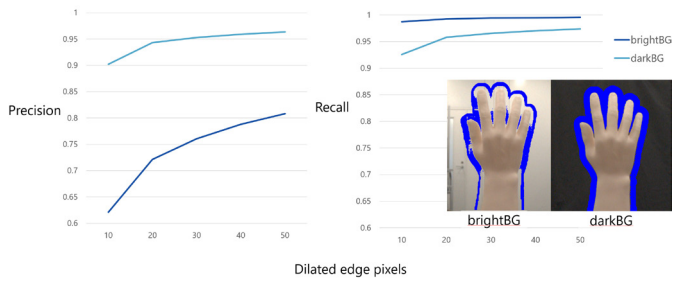

Fig. 14: Occlusions results for two different backgrounds.

and background scenes affected the result, we measured the accuracy of occlusion under two different backgrounds: bright and dark. We used a mannequin hand as the foreground occluder. The prototype rendered virtual objects as dilated shapes of foreground objects. We measured the precision and accuracy by increasing the pixels from the edges in a stepwise fashion. When the contrast was low, the precision remained low because our color-guided filters tended to fail. The results are shown in Figure 14.

6.4 Latency Due to Head Motion

In this section, we describe how we measure the latency due to head motion. Figure 15 shows our setup. Two cameras (Ximea MC031MG-SY@240fps [73]) were used: (a) captures the real-world image. (b) captures the display of the HMD prototype. (a) and (b) were synchronized and calibrated with each other in advance. We attached the prototype to a vibration generator that could vibrate from -5° to 5° at 1 Hz. In the figure, (c) denotes an indicator placed in front of the prototype in the real world, (d) shows an indicator captured and displayed by the prototype, and (e) shows an indicator rendered according to the prototype's self-pose estimation.

While (c) is set at the center of (a)'s field of view, either (d) or (e) is also set at the center of (b)'s field of view. (a) and (b) captures multiple

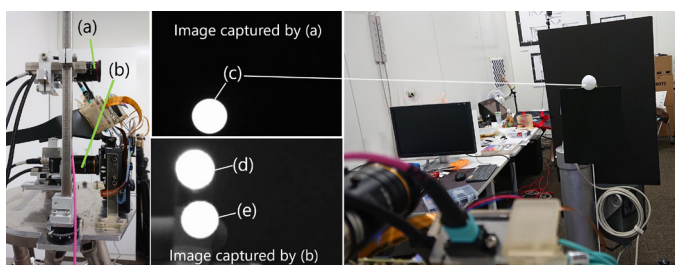

Fig. 15: Latency measurement setup.

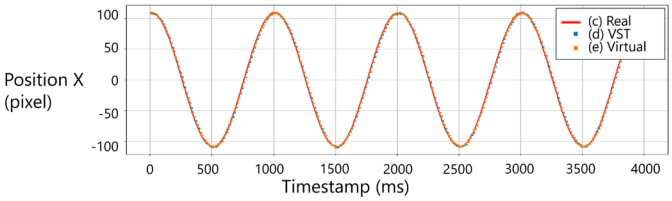

Fig. 16: Indicator trajectories.

Reconstruction and composition	Processing time (ms)
Depth estimation	3.10
Edge refinement filter	2.08
Disocclusion and composition	2.06
Other filters	4.11
Warper	Processing time (ms)
Two-dimensional warping	0.36

Table 1: Processing time for each block.

images as the vibrator moves. When comparing VST image and real world, (d) is set as a target for the camera (b), (e) is set as a target when comparing rendered scene and real world.

We detected the center of each circle and calculated RMSE of horizontal position of both (a) and the target. Latency is defined as follows:

$$latency = \underset{\Delta t}{\operatorname{argmin}} \sqrt{\frac{1}{n}\sum_{t=1}^{n}(\hat{P}(t+\Delta t) - P(t))^2}$$

Here, P and $\hat{P}$ represent functions return the horizontal position of both (a) and the target respectively, and t is a timestamp. By varying the offset, Δt, between -100 ms and 100 ms, we found that it can be used to minimize the value of RMSE. We therefore regard it as the latency. Since the resolution of the offset is limited by the frame rate of the observation camera, if there is a minimum RMSE between certain timestamps A and B, a more accurate offset is calculated by interpolating a virtual timestamp. The results are shown in Figure 16, where the x-axis denotes time and the y-axis denotes position. The delay of the reconstructed scenes compared to the real scenes was 3.3 ms and that of the virtual scenes compared to the real ones was 1.8 ms. Comparing these in each frame of the observation camera, the average delay of the captured image from the virtual scenes was 1.9 ms.

6.5 Latency Due to Scene Motion

In this section, we describe how we measured the latency due to scene motion. We used an LED array that counted every 1 ms. We placed this in front of our prototype and captured both the real LED and that displayed in our prototype, using two time-synchronized cameras (Ximea MC031MG-SY@240fps [73]). We compared the two methods by detecting a timestamp from the LED lighting pattern. The latency varied as 50–60 ms, with an average of 56 ms.

6.6 Performance

We measured the processing time per frame for the proposed algorithm. The results are shown in Table 1. The processing time for reconstruction and composition was ~12 ms, and the warper's processing time was 0.36 ms.

6.7 Discussion

In this section, we discuss the system configuration. For edge refinement, we combined IR masks and color-guided filters, as previously described. Without IR masks, it typically takes more than 50 ms to refine edges to the same quality as our setup. Without color-guided filters, the shapes of the depth maps featured jagged edges and appeared smaller than they should. The filter application depended on the use case. In our test environments, the combination of IR masks and color-guided filters produced reasonable proposals, even though the color-guided filters impoverished the result when the color contrast of the foreground and background were low.

For view synthesis, we stored only one frame to fill the disocclusion, instead of storing a series of past frames. This is because the surroundings continued to change, and reprojection errors accumulated. The pre-reconstruction method can be chosen if the environment is limited and static. For warping, we compensated for the latency of the head-motion related components in captured images (~50–60 ms). However, the latency because of scene motion remained the same. This may be too long for the applications requiring faster responses (e.g., when users physically play instruments). However, as defined in Section 3, 50–60 ms may be acceptable for even standard physical contact (e.g., grasping a mug or taking notes).

7 Conclusion and Future Work

We introduced an approach to show captured and virtual scenes in a spatially and temporally consistent manner whilst also reducing the delays caused by user head motions. We proposed novel LDI-based algorithms that realized lightweight reconstruction and edge-preserving occlusion management when combining virtual and reconstructed scenes. To reduce the latency due to head motion, we designed a two-phase warping architecture that included virtual and reconstructed scenes in the same timestamp and deformed them to keep up with the user's head poses. We implemented our approach on a wearable prototype and measured its accuracy and latency. We achieved an acceptable latency due to head motion of less than 4 ms. We also achieved a spatial accuracy of less than 0.1° (in size) and less than 0.3° (in position) in our test environments.

Although we achieved acceptable results in our test environments, the depth estimation results remain inaccurate in scenes where the color contrast between the foreground and background is low. Shiny walls also reflect IR lights and result in inaccurate occlusion. We believe that we can further improve our approach by utilizing sensors with high environmental robustness and/or combining sensors with different environmental characteristics. We need to address the latency of scene motion when warping, and also need other depth cues (e.g., focus cues), especially for interactions within a personal space. Moreover, for lighter and smaller form factors, it is necessary to dramatically reduce the processing load, which is a challenging task. We hope that our approach will help improve the quality and efficiency of mixed and augmented reality systems.

References

[1] A. F. Abate, F. Narducci, and S. Ricciardi. An Image Based Approach to Hand Occlusions in Mixed Reality Environments. In D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, A. Kobsa, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, D. Terzopoulos, D. Tygar, G. Weikum, R. Shumaker, and S. Lackey, eds., *Virtual, Augmented and Mixed Reality. Designing and Developing Virtual and Augmented Environments*, vol. 8525, pp. 319–328. Springer International Publishing, Cham, 2014. Series Title: Lecture Notes in Computer Science. doi: 10.1007/978-3-319-07458-0_30 2

[2] adaskit Team. libSGM. https://github.com/fixstars/libSGM, Feb. 2016. 5

[3] B. D. Adelstein, T. G. Lee, and S. R. Ellis. Head Tracking Latency in Virtual Environments: Psychophysics and a Model. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 47(20):2083–2087, Oct. 2003. doi: 10.1177/154193120304702001

[4] H. Aga, A. Ishihara, K. Kawasaki, M. Nishibe, S. Kohara, T. Ohara, and M. Fukuchi. Latency Compensation for Optical See-Through Head-Mounted with Scanned Display. *SID Symposium Digest of Technical Papers*, 50(1):330–333, June 2019. doi: 10.1002/sdtp.12923 4

[5] T. Ai. FPGA design & implementation of a very-low-latency video-see-through (VLLV) head-mount display (HMD) system for mixed reality (MR) applications. In *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry - Volume 1*, pp. 39–42. ACM, Zhuhai China, Dec. 2016. doi: 10.1145/3013971.3014020 2

[6] C. Attig, N. Rauh, T. Franke, and J. F. Krems. System Latency Guidelines Then and Now – Is Zero Latency Really Considered Necessary? In

D. Harris, ed., *Engineering Psychology and Cognitive Ergonomics: Cognition and Design*, vol. 10276, pp. 3–14. Springer International Publishing, Cham, 2017. Series Title: Lecture Notes in Computer Science. doi: 10.1007/978-3-319-58475-1_1 2, 3

[7] M. Bajura and U. Neumann. Dynamic registration correction in video-based augmented reality systems. *IEEE Computer Graphics and Applications*, 15(5):52–60, Sept. 1995. doi: 10.1109/38.403828 2

[8] C. Battisti, S. Messelodi, and F. Poiesi. Seamless Bare-Hand Interaction in Mixed Reality. In *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pp. 198–203. IEEE, Munich, Germany, Oct. 2018. doi: 10.1109/ISMAR-Adjunct.2018.00066 8

[9] M. Bauer. Image-based rendering for real-time applications. p. 7. 2

[10] A. Blate, M. Whitton, M. Singh, G. Welch, A. State, T. Whitted, and H. Fuchs. Implementation and Evaluation of a 50 kHz, $28\mu\mathrm{s}$ Motion-to-Pose Latency Head Tracking Instrument. *IEEE Transactions on Visualization and Computer Graphics*, 25(5):1970–1980, May 2019. doi: 10.1109/TVCG.2019.2899233 2

[11] H. Bowles, K. Mitchell, R. W. Sumner, J. Moore, and M. Gross. Iterative Image Warping. *Computer Graphics Forum*, 31(2pt1):237–246, May 2012. doi: 10.1111/j.1467-8659.2012.03002.x 2

[12] J. Carmack. Latency Mitigation Strategies. https://web.archive.org/web/20130225013015/http://www.altdevblogaday.com/2013/02/22/latency-mitigation-strategies/, Feb. 2013. 1, 2

[13] V. Casser, S. Pirk, R. Mahjourian, and A. Angelova. Depth Prediction without the Sensors: Leveraging Structure for Unsupervised Learning from Monocular Videos. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:8001–8008, July 2019. doi: 10.1609/aaai.v33i01.33018001 2

[14] G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics*, 32(3):1–12, June 2013. doi: 10.1145/2487228.2487238 2

[15] G. Chaurasia, A. Nieuwoudt, A.-E. Ichim, R. Szeliski, and A. Sorkine-Hornung. Passthrough+: Real-time Stereoscopic View Synthesis for Mobile Mixed Reality. 3(1):17. 2

[16] H. Chen, C. Wei, M. Song, M.-T. Sun, and K. Lau. Capture-to-display delay measurement for visual communication applications. *APSIPA Transactions on Signal and Information Processing*, 4(1), 2015. doi: 10.1017/ATSIP.2015.21 2

[17] Z. Chen, V. Badrinarayanan, G. Drozdov, and A. Rabinovich. Estimating Depth from RGB and Sparse Sensing. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, eds., *Computer Vision – ECCV 2018*, vol. 11208, pp. 176–192. Springer International Publishing, Cham, 2018. Series Title: Lecture Notes in Computer Science. doi: 10.1007/978-3-030-01225-0_11 2

[18] C. Du, Y.-L. Chen, M. Ye, and L. Ren. Edge Snapping-Based Depth Enhancement for Dynamic Occlusion Handling in Augmented Reality. In *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 54–62. IEEE, Merida, Yucatan, Mexico, Sept. 2016. doi: 10.1109/ISMAR.2016.17 2

[19] D. Eigen and R. Fergus. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-scale Convolutional Architecture. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 2650–2658. IEEE, Santiago, Chile, Dec. 2015. doi: 10.1109/ICCV.2015.304 2

[20] D. Eigen, C. Puhrsch, and R. Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network, June 2014. arXiv:1406.2283 [cs]. doi: 10.48550/arXiv.1406.2283 2

[21] S. R. Ellis, M. J. Young, B. D. Adelstein, and S. M. Ehrlich. Discrimination of Changes of Latency during Voluntary Hand Movement of Virtual Objects. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 43(22):1182–1186, Sept. 1999. doi: 10.1177/154193129904302203 2

[22] Q. Feng, H. P. H. Shum, and S. Morishima. Resolving occlusion for 3D object manipulation with hands in mixed reality. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, pp. 1–2. ACM, Tokyo Japan, Nov. 2018. doi: 10.1145/3281505.3283390

[23] D. Ferstl, C. Reinbacher, R. Ranftl, M. Ruether, and H. Bischof. Image Guided Depth Upsampling Using Anisotropic Total Generalized Variation. In *2013 IEEE International Conference on Computer Vision*, pp. 993–1000. IEEE, Sydney, Australia, Dec. 2013. doi: 10.1109/ICCV.2013.127 2

[24] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker. DeepView: View Synthesis With Learned Gradient Descent. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2362–2371. IEEE, Long Beach, CA, USA, June 2019. doi: 10.1109/CVPR.2019.00247

[25] J. P. Freiwald, N. Katzakis, and F. Steinicke. Camera time warp: compensating latency in video see-through head-mounted-displays for reduced cybersickness effects. In *Proceedings of the 24th ACM Symposium on Virtual Reality Software and Technology*, pp. 1–7. ACM, Tokyo Japan, Nov. 2018. doi: 10.1145/3281505.3281521 1, 2

[26] C. Gao, A. Saraf, J. Kopf, and J.-B. Huang. Dynamic View Synthesis from Dynamic Monocular Video. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5692–5701. IEEE, Montreal, QC, Canada, Oct. 2021. doi: 10.1109/ICCV48922.2021.00566 2

[27] S. J. Garbin, M. Kowalski, M. Johnson, J. Shotton, and J. Valentin. FastNeRF: High-Fidelity Neural Rendering at 200FPS. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14326–14335. IEEE, Montreal, QC, Canada, Oct. 2021. doi: 10.1109/ICCV48922.2021.01408 2

[28] C. Godard, O. M. Aodha, and G. J. Brostow. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6602–6611. IEEE, Honolulu, HI, July 2017. doi: 10.1109/CVPR.2017.699

[29] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova. Depth From Videos in the Wild: Unsupervised Monocular Depth Learning From Unknown Cameras. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8976–8985. IEEE, Seoul, Korea (South), Oct. 2019. doi: 10.1109/ICCV.2019.00907

[30] R. Gruen, E. Ofek, A. Steed, R. Gal, M. Sinclair, and M. Gonzalez-Franco. Measuring System Visual Latency through Cognitive Latency on Video See-Through AR devices. *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 791–799, 2020. doi: 10.1109/VR46266.2020.1580498468656 2, 3

[31] E. T. Hall. *The hidden dimension*. Anchor Books, New York, 1990. 6

[32] A. K. Hebborn, N. Hohner, and S. Muller. Occlusion Matting: Realistic Occlusion Handling for Augmented Reality Applications. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 62–71. IEEE, Nantes, Oct. 2017. doi: 10.1109/ISMAR.2017.23 2

[33] P. Hedman, S. Alsisan, R. Szeliski, and J. Kopf. Casual 3D photography. *ACM Transactions on Graphics*, 36(6):1–15, Nov. 2017. doi: 10.1145/3130800.3130828 2

[34] P. Hedman and J. Kopf. Instant 3D photography. *ACM Transactions on Graphics*, 37(4):1–12, Aug. 2018. doi: 10.1145/3197517.3201384

[35] H. Hirschmuller. Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, Feb. 2008. doi: 10.1109/TPAMI.2007.1166 5

[36] A. Holynski and J. Kopf. Fast depth densification for occlusion-aware augmented reality. *ACM Transactions on Graphics*, 37(6):1–11, Dec. 2018. doi: 10.1145/3272127.3275083

[37] A. Hornung and L. Kobbelt. Interactive Pixel-Accurate Free Viewpoint Rendering from Images with Silhouette Aware Sampling. *Computer Graphics Forum*, 28(8):2090–2103, Dec. 2009. doi: 10.1111/j.1467-8659.2009.01416.x 2

[38] V. Jantet, C. Guillemot, and L. Morin. Object-based Layered Depth Images for improved virtual view synthesis in rate-constrained context. In *2011 18th IEEE International Conference on Image Processing*, pp. 125–128. IEEE, Brussels, Belgium, Sept. 2011. doi: 10.1109/ICIP.2011.6115662 2

[39] Ji-Youn Choi, Sae-Woon Ryu, Hong-Chang Shin, and Jong-Il Park. Real-time view synthesis system with multi-texture structure of GPU. In *2010 Digest of Technical Papers International Conference on Consumer Electronics (ICCE)*, pp. 171–172. IEEE, Las Vegas, NV, Jan. 2010. doi: 10.1109/ICCE.2010.5418768 2

[40] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-End Learning of Geometry and Context for Deep Stereo Regression. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 66–75. IEEE, Venice, Oct. 2017. doi: 10.1109/ICCV.2017.17 2

[41] P. Kim, J. Orlosky, and K. Kiyokawa. AR Timewarping: A Temporal Synchronization Framework for Real-Time Sensor Fusion in Head-Mounted Displays. *Proceedings of the 9th Augmented Human International Conference*, 2018. doi: 10.1145/3174910.3174919 2

[42] J. Kopf, K. Matzen, S. Alsisan, O. Quigley, F. Ge, Y. Chong, J. Patterson, J.-M. Frahm, S. Wu, M. Yu, P. Zhang, Z. He, P. Vajda, A. Saraf, and M. Cohen. One shot 3D photography. *ACM Transactions on Graphics*, 39(4), Aug. 2020. doi: 10.1145/3386569.3392420

[43] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper Depth Prediction with Fully Convolutional Residual Networks. In *2016*

Fourth International Conference on 3D Vision (3DV), pp. 239–248. IEEE, Stanford, CA, Oct. 2016. doi: 10.1109/3DV.2016.32

[44] T. Luo, Z. Liu, Z. Pan, and M. Zhang. A Virtual-real Occlusion Method Based on GPU Acceleration for MR. *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 1068–1069, 2019. doi: 10.1109/VR.2019.8797811

[45] F. Ma and S. Karaman. Sparse-to-Dense: Depth Prediction from Sparse Depth Samples and a Single Image. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4796–4803. IEEE, Brisbane, QLD, May 2018. doi: 10.1109/ICRA.2018.8460184 2

[46] K. Mania, B. D. Adelstein, S. R. Ellis, and M. I. Hill. Perceptual sensitivity to head tracking latency in virtual environments with varying degrees of scene complexity. In *Proceedings of the 1st Symposium on Applied perception in graphics and visualization - APGV '04*, p. 39. ACM Press, Los Angeles, California, 2004. doi: 10.1145/1012551.1012559 2

[47] R. W. Marcato. Optimizing an Inverse Warper. p. 51. 2

[48] T. Mazuryk and M. Gervautz. Two-step Prediction and Image Deflection for Exact Head Tracking in Virtual Environments. *Computer Graphics Forum*, 14(3):29–41, Aug. 1995. doi: 10.1111/j.1467-8659.1995.cgf143_0029.x 2

[49] J. L. McMillan. An image-based approach to three-dimensional computer graphics. p. 206, 1997. 2

[50] L. McMillan and G. Bishop. Head-tracked stereoscopic display using image warping. pp. 21–30. San Jose, CA, Mar. 1995. doi: 10.1117/12.205865

[51] A. Michael. Asynchronous Timewarp Examined. https://developer.oculus.com/blog/asynchronous-timewarp-examined/, Mar. 2015. 2

[52] M. Misiak, A. Fuhrmann, and M. E. Latoschik. Impostor-based Rendering Acceleration for Virtual, Augmented, and Mixed Reality. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, pp. 1–10. ACM, Osaka Japan, Dec. 2021. doi: 10.1145/3489849.3489865

[53] Y. Nakashima, F. Okura, N. Kawai, r. Kimura, H. Kawasaki, K. Ikeuchi, and A. Blanco. Realtime Novel View Synthesis with Eigen-Texture Regression. In *Procedings of the British Machine Vision Conference 2017*, p. 83. British Machine Vision Association, London, UK, 2017. doi: 10.5244/C.31.83 2

[54] P. Ndjiki-Nya, M. Koppel, D. Doshkov, H. Lakshman, P. Merkle, K. Muller, and T. Wiegand. Depth Image-Based Rendering With Advanced Texture Synthesis for 3-D Video. *IEEE Transactions on Multimedia*, 13(3):453–465, June 2011. doi: 10.1109/TMM.2011.2128862

[55] D. Nehab, P. V. Sander, J. Lawrence, N. Tatarchuk, and J. R. Isidoro. Accelerating Real-Time Shading with Reverse Reprojection Caching. p. 11. doi: 10.2312/EGGH/EGGH07/025-036 2

[56] S. Niklaus. Novel View Synthesis in Time and Space. Technical report, Feb. 2020. doi: 10.15760/etd.7294 2

[57] J. Ogniewski. High-Quality Real-Time Depth-Image-Based-Rendering. p. 8, 2017. 2

[58] B. Reinert, J. Kopf, T. Ritschel, E. Cuervo, D. Chu, and H.-P. Seidel. Proxy-guided Image-based Rendering for Mobile Devices. *Computer Graphics Forum*, 35(7):353–362, Oct. 2016. doi: 10.1111/cgf.13032 2

[59] S. M. Seitz and C. R. Dyer. View morphing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques - SIGGRAPH '96*, pp. 21–30. ACM Press, Not Known, 1996. doi: 10.1145/237170.237196 2

[60] J. Shade, S. Gortler, L.-w. He, and R. Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*, pp. 231–242. ACM Press, Not Known, 1998. doi: 10.1145/280814.280882 2

[61] H. Shum and S. B. Kang. Review of image-based rendering techniques. p. 2. Perth, Australia, May 2000. doi: 10.1117/12.386541 2

[62] J.-P. Stauffert, F. Niebling, and M. E. Latoschik. Latency and Cybersickness: Impact, Causes, and Measures. A Review. *Frontiers in Virtual Reality*, 1:582204, Nov. 2020. doi: 10.3389/frvir.2020.582204 2

[63] W. Sun, L. Xu, O. C. Au, S. H. Chui, and C. W. Kwok. An overview of free viewpoint Depth-Image-Based Rendering (DIBR). p. 8. 2

[64] X. Tang, X. Hu, C.-W. Fu, and D. Cohen-Or. GrabAR: Occlusion-aware Grabbing Virtual Objects in AR. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, pp. 697–708. ACM, Virtual Event USA, Oct. 2020. doi: 10.1145/3379337.3415835

[65] Y. Tian, T. Guan, and C. Wang. Real-Time Occlusion Handling in Augmented Reality Based on an Object Tracking Approach. *Sensors*, 10(4):2885–2900, Mar. 2010. doi: 10.3390/s100402885 2

[66] J. M. P. van Waveren. The asynchronous time warp for virtual reality on consumer hardware. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*, pp. 37–46. ACM, Munich Germany, Nov. 2016. doi: 10.1145/2993369.2993375 2

[67] J. Ventura and T. Hollerer. Online environment model estimation for augmented reality. In *2009 8th IEEE International Symposium on Mixed and Augmented Reality*, pp. 103–106. IEEE, Orlando, FL, USA, Oct. 2009. doi: 10.1109/ISMAR.2009.5336493 2

[68] G. P. Visa and P. Salembier. Precision-Recall-Classification Evaluation Framework: Application to Depth Estimation on Single Images. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, eds., *Computer Vision – ECCV 2014*, vol. 8689, pp. 648–662. Springer International Publishing, Cham, 2014. Series Title: Lecture Notes in Computer Science. doi: 10.1007/978-3-319-10590-1_42 8

[69] D. R. Walton and A. Steed. Accurate real-time occlusion for mixed reality. In *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology*, pp. 1–10. ACM, Gothenburg Sweden, Nov. 2017. doi: 10.1145/3139131.3139153 2

[70] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler. Analysis of the Accuracy and Robustness of the Leap Motion Controller. *Sensors*, 13(5):6380–6393, May 2013. doi: 10.3390/s130506380 2

[71] S. Wizadwongsa, P. Phongthawee, J. Yenphraphai, and S. Suwajanakorn. NeX: Real-time View Synthesis with Neural Basis Expansion. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8530–8539. IEEE, Nashville, TN, USA, June 2021. doi: 10.1109/CVPR46437.2021.00843 2

[72] L. Xiao, S. Nouri, J. Hegland, A. G. Garcia, and D. Lanman. NeuralPassthrough: Learned Real-Time View Synthesis for VR. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings*, pp. 1–9. ACM, Vancouver BC Canada, Aug. 2022. doi: 10.1145/3528233.3530701 1, 2

[73] Ximea. MC031MG-SY Specifications. https://www.ximea.com/en/products/usb-31-gen-1-with-sony-cmos-xic/mc031mg-sy. 8, 9

[74] L. Yang, Y.-C. Tse, P. V. Sander, J. Lawrence, D. Nehab, H. Hoppe, and C. L. Wilkins. Image-based bidirectional scene reprojection. In *Proceedings of the 2011 SIGGRAPH Asia Conference on - SA '11*, p. 1. ACM Press, Hong Kong, China, 2011. doi: 10.1145/2024156.2024184 2