Scalable Image-based Indoor Scene Rendering with Reflections

ANONYMOUS AUTHOR(S)
SUBMISSION ID: 445

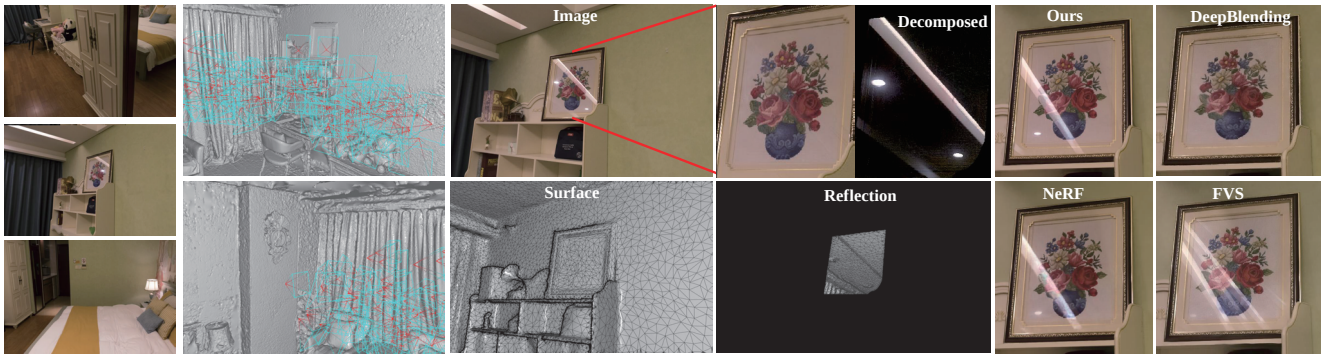


Fig. 1. Two-layer representation and its rendering result with reflections. Decomposed images: the input image inside the frame is decomposed into surface and reflection layer images. While state-of-the-art view synthesis methods, such as DeepBlending [Hedman et al. 2018], NeRF [Mildenhall et al. 2020], and FVS [Riegler and Koltun 2020a], render images with blurred reflections or without reflections at a novel viewpoint in this case, our image-based rendering pipeline can achieve high-quality rendering result using two-layer meshes and decomposed images. Best viewed with zoom-in.

This paper proposes a novel scalable image-based rendering (IBR) pipeline for indoor scenes with reflections. Observing that the reconstructed global mesh of an indoor scene can be used as a geometric prior to improve the robustness of the reflection decomposition algorithm but still inadequate for high-quality reflection rendering, we propose a global-mesh-guided alternating optimization algorithm that can construct front surface and back reflection layer RGB images and meshes, a two-layer representation, to support the accurate rendering of various reflections effectively. The algorithm alternatively optimizes two-layer RGB images and meshes to minimize the image composition error to reduce blurred artifacts in view warping. Moreover, to support densely sampled images required for two-layer mesh construction and high-frequency reflection rendering, we propose to integrate convolutional neural network (CNN) based super-resolution network and a motion refinement module to render high-resolution images with low-resolution textures. The motion refinement module can predict local offsets to correct errors of mesh-based warping further to improve rendering quality. Hence, our IBR pipeline can substantially save memory storage and exploit the multi-scale feature learned by CNN to reduce the artifacts caused by floating geometries. Experimental results show that our method can produce highly realistic rendering results with different kinds of reflections, and the rendering quality is superior to state-of-the-art IBR or neural rendering algorithms.

CCS Concepts: • **Computing methodologies** → **Image-based rendering, Neural network**; **Virtual reality**.

XXXX-XXXX/2021/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Additional Key Words and Phrases: Image-based rendering, Two-layer mesh, Reflection, Super-resolution, Neural network

ACM Reference Format:
Anonymous Author(s). 2021. Scalable Image-based Indoor Scene Rendering with Reflections. 1, 1 (January 2021), 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Image-based rendering (IBR) algorithms have been applied to synthesize photo-realistic images at novel viewpoints for indoor scenes, which is crucial to immersive virtual reality applications, such as free-viewpoint navigation of real-estate or museum. However, it is technically challenging due to two reasons. First, objects in indoor scenes are often observed near cameras, which results in severe occlusions and large motion parallax. Second, high-frequency view-dependent effects, such as sharp highlights and reflections from reflective or glossy surfaces, frequently occur due to the existence of mirrors, TV screens, and smooth surfaces of man-made objects.

Layered representations, such as layered depth images [Shade et al. 1998], multi-plane images (MPI) [Flynn et al. 2019; Zhou et al. 2018], and multi-spherical images (MSI) [Broxton et al. 2020], are developed to handle occlusions and view-dependent effects in IBR simultaneously. These representations can be used to store the RGB and compositing coefficient α of reflected scenes in separate layers and synthesize new images with reflections through layer blending. In [Sinha et al. 2012], two-layer RGBD images, the front surface and the rear reflection layer RGBD images, are constructed using the semi-global multi-view stereo algorithm for high-quality rendering of reflections. However, the construction algorithm of layered representations is either sensitive to hyper-parameters or time-consuming. Besides, it is also important to investigate how to construct the layer representation adaptively to reduce the required

memory storage. An alternative way for IBR of indoor scenes is to train neural networks, such as neural radiance fields [Mildenhall et al. 2020] and deep view synthesis network [Xu et al. 2019], to model the scene structure from sampled images implicitly. These networks can render reflective surfaces realistically, but it is computationally expensive to train them to render a large scale indoor scene in real-time. In addition, it is still challenging for deep neural networks to model sharp edges of reflections, as shown in Fig. 1.

In this paper, we address the problem of IBR of indoor scenes with reflections, which involves two issues: robust front surface and reflection layer decomposition and the balance between memory storage and the requirement of high-resolution images in virtual reality applications. The motivation of our IBR pipeline is based on two observations. First, with the fast development of structure from motion (SFM) and depth-camera-based 3D scene reconstruction techniques [Dong et al. 2019; Furukawa and Ponce 2010; Hartley and Zisserman 2004; Schönberger and Frahm 2016; Xu et al. 2017], the ability to capture high-quality geometry of an indoor scene has been significantly improved, even for mirrors and glasses [Whelan et al. 2018]. Therefore, we can obtain the prior geometry for reflective surfaces through a well-reconstructed global geometry [Hedman et al. 2018, 2016]. Such prior geometry can be used to improve the robustness of the reflection decomposition algorithm for reflection rendering. However, it is still challenging to handle the noises in the reconstructed prior geometry and camera poses to obtain high-quality reflection rendering results. Second, the realistic rendering of high-frequency reflections requires a much denser sampling rate than that of diffuse scenes. Empirically, we observe that it is necessary to maintain 30% overlap of reflections among neighboring images to obtain a high-quality reflection decomposition results, resulting in thousands of captured high-resolution (HR) images used as texture images in IBR. The memory cost will be unaffordable to store HR images in GPU memory as pointed out in [Hedman et al. 2016].

These two observations motivate us to design a IBR pipeline with two novel technical components:

- A global-mesh-guided alternating optimization algorithm for robust per-view two-layer mesh construction. We construct a two-layer mesh for each view with reflections, including a surface mesh representing the RGB and geometry information of front surfaces and a reflection mesh representing that of the front surfaces' reflected part of a scene. This representation can effectively support image-based rendering of an indoor scene with reflections. To obtain such a representation, we project the indoor scene's global mesh to each view to initialize the surface layer mesh and then initialize the reflection layer mesh by the intersection of reflected rays with the global mesh. However, due to noises of reconstructed camera poses and geometries, there are errors in the initialized meshes that will severely downgrade the rendering quality of reflections. Thus, combining with the linear image composition rule in [Sinha et al. 2012] and a multi-view consistency constraint for surface layer RGB images, we propose an alternating optimization algorithm that can achieve high-quality reflection-decomposition results at each view. The decomposed images are used as textures for the constructed two-layer meshes. Furthermore, we propose a detection-then-decomposition procedure to improve the two-layer mesh representation for reflected highlights.
- A convolutional neural network (CNN) based super-resolution (SR) method to render HR images with low-resolution (LR) input images, which can substantially save memory storage. The network is adapted from the SR network in [Wang et al. 2020], and we add a motion refinement module to the network to mitigate the artifacts, for instance, blurring at object boundaries, caused by inaccurate geometry. Since the network is also trained to de-artifact, we term the network as DSRNet hereafter. The network's input is the image generated by our view warping algorithm designed to mitigate the discontinuity artifacts of the tile-based view warping algorithm in [Hedman et al. 2018, 2016]. Our view warping algorithm is based on camera pose to avoid discontinuous view selection among neighboring tiles. Coupling with blending weight decay at image boundaries and occlusion edges, the rendering results' smoothness in the free-viewpoint navigation can be achieved. Furthermore, the surface and reflection layers in the selected views are warped according to their meshes separately and then blended at the target viewpoint to render the reflections correctly.

We have conducted experiments with our IBR pipeline for a variety of indoor scenes, ranging from apartments to offices. Experimental results show that our method can produce highly realistic rendering results with different kinds of reflections, and the rendering quality is superior to state-of-the-art IBR or neural rendering algorithms.

2 RELATED WORK

IBR can be conducted in a wide spectrum, from no geometry with a densely arranged camera array to explicit geometry reconstruction to assist the image-warping-based view synthesis [Gortler et al. 1996; Levoy and Hanrahan 1996; Penner and Zhang 2017]. The comprehensive survey of IBR techniques can be found in [Shum and Kang 2000; Zhang and Chen 2004], and the recent deep-learning-based IBR methods are reviewed in [Tewari et al. 2020]. We review the literature mostly related to our work in this section.

IBR with geometry: Geometry information is mainly used to map pixels between images captured at different viewpoints and determine their visibility. The representation of the geometry in IBR can be geometric proxies for depth correction, depth images for view interpolation, visual and opacity hulls for pixel visibility, and 3D meshes for view-dependent texturing and surface light fields [Buehler et al. 2001; Chen and Williams 1993; Debevec et al. 1996; Matusik et al. 2000, 2002; Wood et al. 2000]. The 3D geometry of a scene can be reconstructed from captured images by multi-view stereo(MVS) algorithms [Furukawa and Ponce 2010; Goesele et al. 2007; Rhemann et al. 2011], and used to guide the image warping and blending for novel view synthesis [Cayon et al. 2015; Chaurasia et al. 2011; Goesele et al. 2010]. Chaurasia et al. [2013] utilized super-pixels as constraints to obtain per-pixel depth and then warp images. It significantly reduces the image warping artifacts along occlusion edges produced by the method in [Chaurasia et al. 2011].

For indoor scenes, piece-wise planes and Manhattan-world assumption are exploited to reconstruct 3D planes from input images for image-based indoor scene rendering [Furukawa et al. 2009; Sinha et al. 2009].

In [Hedman et al. 2016], the reconstructed global geometry is refined at each view to align depth and image edges. The resulting per-view meshes are beneficial to handle large occlusions and motion parallax in IBR. Afterward, Hedman et al. [2018] proposed to train a deep neural network to blend images warped with per-view meshes to reduce ghosting artifacts caused by inaccurate geometry. These two approaches can reproduce view-dependent effects to some extent. However, they can not handle reflections because blending artifacts will be obvious if warping images using reflective surface geometry only. Our work is inspired by these two works, but exploit two-layer mesh representation to render indoor scenes with reflections.

A pioneering work on layered representation used in IBR is layered depth images (LDI) [Shade et al. 1998]. LDI can be viewed as a projective volume at a specific viewpoint which stores not only what is visible in the input image, but also what is behind the visible surface. It is proposed to handle large occlusions resulting from the object observed at close distance. Penner et al. [2017] constructed a projective volume representation from the captured scene images where each voxel encodes the uncertainty in the MVS and achieved high quality view synthesis results even at occlusion edges. In [Hedman et al. 2017], for the photos captured by a mobile phone or hand-held DSLR camera, two color-and-depth layer panoramas are constructed to produce perspective views near captured viewpoints with motion parallax effects. Broxton et al. [2020] designed a spherical dome to capture light field videos. Each frame is represented first with multi-sphere images computed by extending the deep neural network in [Xu et al. 2019] and then simplified to multi-layer meshes.

Our two-layer mesh representation is used to approximate the image warping behavior of the surface and its reflections separately, which is mostly related to reflection decomposition work in [Sinha et al. 2012]. Kopf et al. [2013] proposed to render the reflections in gradient domain. The reflection decomposition can also be achieved according to the motion cue computed with SIFT flow [Li and Brown 2013], homography [Guo et al. 2014], and dense optical flow [Xue et al. 2015]. In contrast, we leverage the global mesh as a prior to robustly compute the color and geometry of reflection layers for scalable indoor scene rendering. While the multi-layered mesh representation in [Broxton et al. 2020] can handle view dependent effects, their target is to allow as-large-as-possible viewpoint movement near a given viewpoint in VR videos.

Deep Learning-based IBR: Given captured images, deep learning-based IBR methods are capable of learning multi-scale features as the scene representation to facilitate IBR, such as end-to-end deep stereo for unstructured view interpolation [Flynn et al. 2016], deep view synthesis for spares images captured under controlled conditions [Xu et al. 2019], multi-plane images [Mildenhall et al. 2019; Srinivasan et al. 2019; Xu et al. 2019; Zhou et al. 2018], neural texture [Thies et al. 2019a], and neural volume [Lombardi et al. 2019]. Implicit function representation of a 3D scene can be learned through coordinate-based multilayer perceptron (MLP) by minimizing the similarity between the rendered image and the captured image at the same viewpoint [Sitzmann et al. 2019]. Mildenhall et al. [2020] trained a rendering network connected with a coordinate-based MLP using positional encoding to effectively encode the radiance fields, termed as neural radiance fields (NeRF). However, the training and testing of the NeRF network are time-consuming. Hence, Liu at al. [2020a] proposed neural sparse voxel fields to prune unnecessary samples inside the empty space of a 3D scene. The volume rendering step can also be accelerated by training a network to approximate the integration [Lindell et al. 2020].

The reconstructed coarse scene geometry can be used as a scaffold to fuse the image features for novel view synthesis. Riegler et al. [2020a] designed a recurrent encoder-decoder network to process reprojected features from neighboring views for view synthesis. They improved the view synthesis results further through view-dependent on-surface feature aggregation [Riegler and Koltun 2020b]. A factored representation of a scene, including point cloud, semantic structure and latent appearance code, are used in [Meshry et al. 2019] to render the scene at new viewpoints with different appearances.

Deep Learning for Image and Video Super-resolution: The deep learning methods for the task of image super-resolution (SR) range from the CNN-based method to approaches using Generative Adversarial Network (GAN) [Dong et al. 2014; Ledig et al. 2017; Rakotonirina and Rasoanaivo 2020]. For a comprehensive survey on deep learning based image super-resolution methods, we refer to the survey by Wang et al. [Wang et al. 2020]. Temporal coherence of video super-resolution (VSR) methods can be realized through integrating motion compensation modules to the SR neural network, such as multi-resolution spatial transformer module in VESPCN [Caballero et al. 2017], sub-pixel motion compensation layer in SPMCVSR [Tao et al. 2017], pyramid, cascading and deformable (PCD) alignment module in EDVR Wang2019, and recurrent networks to accelerate the frame warping in video SR [Fuoli et al. 2019; Haris et al. 2019; Sajjadi et al. 2018].

In game industry, temporal supersampling methods are developed for the SR of rendered videos [Chaitanya et al. 2017; Edelsten et al. 2019; Tatarchuk et al. 2014]. Based on the motion vectors between frames computed using the camera and depth information provided by the game engine, Xiao et al. [2020] proposed a network to learn how to blend multiple-frames in the feature space for high-quality supersampling results. To handle possible warping errors induced by inaccurate geometry at occlusion edges, we integrate a motion correction module into this network to align images locally to improve SR results.

3 OVERVIEW

The developed IBR pipeline consists of three main stages: 1. After obtaining the global mesh of an indoor scene, the pipeline starts with constructing per-view two-layer meshes to represent the geometry of an indoor scene with reflections. 2. A view-warping algorithm is developed to synthesize the image with low-resolution textures ($\frac{1}{4}$ resolution of rendered HR images) at the user-specified viewpoint. 3. Such images are fed into the DSRNet to produce the HR rendering

result. As illustrated in Fig. 1, the constructed two-layer meshes along with the decomposed surface and reflection layer images can be used to render highly realistic reflections. Since we perform the refection decomposition on images before gamma-correction (see Sec. 4.2 for details), the decomposed images are gamma-corrected in all our figures for better visualization.

The quality of global mesh **G** is critical to the success of IBR of an indoor scene. In our system, we construct the global mesh **G** from captured multi-view images using structure from motion (SFM) software `RealityCapture` [CapturingReality 2016]. The images are captured with a `Canon EOS 60D` digital single-lens reflex camera. We convert the raw image (at the resolution of 6, 000 by 4, 000) into the 16-bit `tiff` format. Empirically, we found that exploiting raw images in SFM helps the software to calculate camera poses more accurately (slightly down-sampling the raw image could improve the convergence speed of the pose estimation). While this software provides a smooth global mesh estimate even in textureless regions [Jancosek and Pajdla 2011], there are still missing or incorrect geometries due to the severe occlusions and misalignment of feature points. Thus, we also consolidate the reconstructed global mesh using RGB-D data (captured by a `Microsoft Kinect4` RGB-D camera) for objects with strong reflections or under severe occlusions. The geometry of mirrors and their mask in an image are obtained by the method in [Whelan et al. 2018]. Additionally, for those light sources that can not be reconstructed automatically, we allow users to create their proxy meshes using the sweeping-based modeling method [Chen et al. 2013]. The consolidated global mesh serves as a prior in the follow-up processing of layered mesh construction.

Constructing a two-layer mesh for each reflective object in the scene could be time-consuming. For small-size reflective objects (around 30x30x30cm^3) or reflective surfaces far from the camera (larger than 3m), the reflection captured in the image is out of focus and thus blurred. We found it is usually acceptable to render their view-dependent reflections through image blending. Therefore, we choose to construct layered meshes for salient and large reflection planes. To detect reflective surfaces, we first detect 3D planar surfaces in the global mesh using the RANSAC method [Schnabel et al. 2007]. For those planes larger than $0.09m^2$ or its projection in at least one image is more than 1000 pixels, the set of images to which these planes are visible will be the target of the reflection detection. Similar to [Sinha et al. 2012], we calculate multi-view cost volumes for the images to check whether second local matching-cost minimums, close to the reflected scene geometry of the plane but different to the depth of the projection of global geometry, exists in the volume. If found, the plane should contain reflections. We also run state-of-the-art image-based mirror detection method in [Lin et al. 2020] to detect mirrors in images not captured with the device in [Whelan et al. 2018]. Such images are used as additional textures to improve rendering quality. With such a reflection detection procedure, more than 90% reflective planes can be detected. Our system also allows users to specify reflective surfaces to correct mislabeling manually. Afterward, using global mesh as a prior, we perform reflection decomposition (§ 4.2) for each image that has reflections to provide surface and reflection layer meshes and images as the two-layer representation.

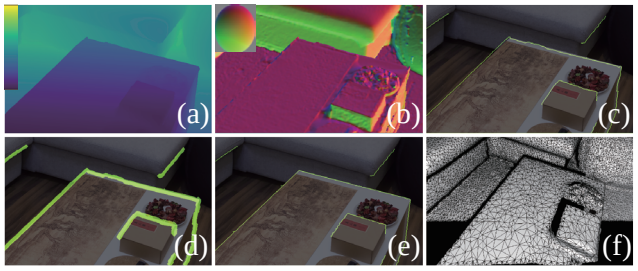


Fig. 2. Depth refinement to align depth and color edges. (a) Initial depth map. (b) Visualization of normal map. (c) Misalignment between depth and color edges. (d) Regions between depth and color edges. (e) Refined depth edges. (f) Constructed surface mesh. Lines in yellowgreen indicates the depth edges. Please zoom-in to view the details.

The rest of this paper is organized as follows. Sec. 4 describes the details of per-view depth refinement and the reflection decomposition in per-view two-layer mesh construction. Sec. 5 describes the details of two-layer view warping algorithm, and the details of DSRNet are described in Sec. 6. The implementation details and experimental results are described in Sec. 7 and Sec. 8 respectively. Finally, we conclude and discuss limitations and future work in Sec. 9.

4 PER-VIEW TWO-LAYER MESH CONSTRUCTION

In this section, we will describe the details of surface layer mesh construction and our reflection decomposition algorithm to construct per-view two-layer meshes for the subsequent rendering.

4.1 Surface Layer Mesh Construction

It starts with rendering the global mesh **G** at the viewpoint of image **I** to obtain a depth image **D**, and then refine **D** to align depth and color images to reduce tearing-apart, ghosting artifacts in IBR. The surface layer mesh is finally constructed according to the refined **D**. In contrast to align depth and color edges using guided median filter as in [Hedman et al. 2018, 2016], we integrate surface normal information in both depth edge detection and refinement to assist the edge alignment.

Depth edge detection: For each pixel in the depth image, we first calculate its vertex position $\mathbf{v}_i$ and normal $\mathbf{n}_i$. Second, we detect whether there exists a depth edge between two neighboring pixels p_i and p_j by checking their mutual planar distance dt_{ij}, which is :

$$dt_{ij} = \max\left(\left|(\mathbf{v}_i - \mathbf{v}_j)\cdot\mathbf{n}_i\right|, \left|(\mathbf{v}_i - \mathbf{v}_j)\cdot\mathbf{n}_j\right|\right) \quad (1)$$

If the calculated dt_{ij} exceeds threshold λ, it indicates a depth edge as shown in Fig. 2(c). As the remote pixels are less sensitive to depth change, we set λ as $0.01 * \max(1, \min(d_i, d_j))$ where d_i means the depth for pixel i that is in unit meter in this step. After obtaining depth edges, we generate a depth refinement mask to restrict the depth refinement area by rendering frontal parallel square patch of side length 4cm at each depth edge pixel to current view as shown in Fig. 2(d).

Depth refinement: Similar to [Hedman et al. 2018], we exploit the COLMAP software [Schönberger et al. 2016] to recover the detailed, pixel-wise depth inside the refinement mask. To this end, we import the camera poses and 3D points output in the stage

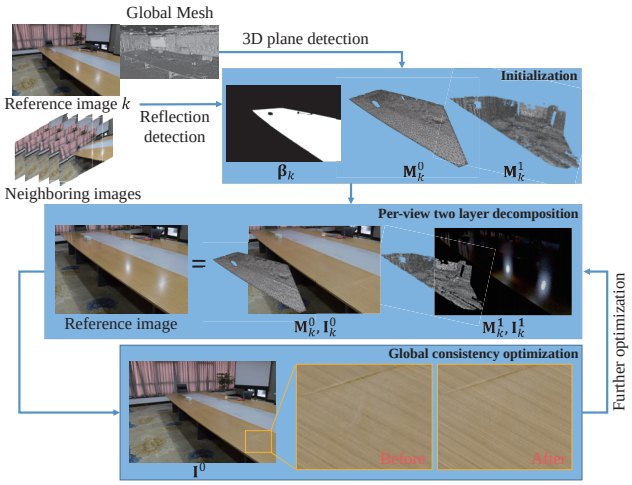


Fig. 3. The pipeline of reflection decomposition algorithm.

of sparse reconstruction from the RealityCapture software into the COLMAP, and run the pixel-wise view selection based multi-view stereo algorithm to obtain the depths inside the refinement mask. Afterward, if, for a pixel, its photometric depth and geometric depth consistency differ by more than 5%, the pixel is deemed to be uncertain, and we discard its depth. Usually, the re-detected depth edge after refinement is closer to the color edge. Afterward, we adapt edge-ware interpolation in Epicflow [Revaud et al. 2015] to compute the depth value d_i for a pixel i between the depth edges and their closest color edges as follows (Fig 2.e):

$$d_i = \sum_{j \in \mathcal{N}_i} \frac{w_g(i,j)}{\sum_k w_g(i,k)} \hat{d}_i^j, \tag{2}$$

where the pixels in $\mathcal{N}_i$ is the 4-closest neighbors outside the refinement mask of i computed using the geodesic distance $d_g(p_i, p_j)$, and $w_g(i,j) = \exp(-d_g(i,j))$, and $\hat{d}_i^j$ is the depth computed by the intersection between the line of sight through pixel i and the plane defined according to the 3D position and normal at pixel j. We also allow users to specify edges on color images if no color edge is close to a occlusion edge in almost constant color regions.

Converting refined depth to a surface mesh: We initialize a triangle mesh by creating edges between neighboring depth map pixels and a diagonal edge to convert each pixel square into two triangles. Then, the mesh is simplified to be the final per-view surface mesh [Garland and Heckbert 1997]. After simplification, the vertex number of per-view mesh is less than 40,000 on average. Thus, the vertex indices can be stored using unsigned short type. Moreover, we try to preserve mesh edges along depth occlusion edges by increasing their quadric error by a factor of 4 in mesh simplification to improve the mesh resolution around depth edges for the purpose of edge anti-aliasing in the rendering. Additionally, the meshes for image areas covered by reflections are constructed as described in the next section.

4.2 Reflection Decomposition

We follow the linear image composition rule in [Sinha et al. 2012] to decompose an image into surface and reflection layer images and optimize for two-layer meshes. Specifically, the RGB values of a reference image k with reflections can be defined as follows:

$$\mathbf{I}_k = \mathbf{I}_k^0 + \beta_k \mathbf{I}_k^1 \tag{3}$$

After obtaining the surface layer image $\mathbf{I}_k^0$ and the reflection layer image $\mathbf{I}_k^1$, they can be warped to a neighboring view k' to form a warped image $\tilde{\mathbf{I}}_{k'}$. Hence, we minimize the difference between $\tilde{\mathbf{I}}_{k'}$ and $\mathbf{I}_{k'}$ in the reflection decomposition. Given a reference image $\mathbf{I}_k$ and its neighboring views, the task of reflection decomposition is to figure out five quantities, a surface layer image $\mathbf{I}_k^0$, a reflection layer image $\mathbf{I}_k^1$, a mask β_k, a mesh of the surface layer $\mathbf{M}_k^0$, and a mesh of the reflection layer $\mathbf{M}_k^1$. The meshes $\mathbf{M}_k^{0,1}$ are used to warp $\mathbf{I}_k^0$ and $\mathbf{I}_k^1$ to a neighboring view k' to form a warped image $\tilde{\mathbf{I}}_{k'}$ using Eq. 3. The mask β is a binary mask on the surface layer to indicate whether reflections cover a pixel. The mask value is set to 1 for a covered pixel, otherwise 0.

Our reflection decomposition algorithm exploits the reconstructed global mesh and multi-view consistency constraints across decomposed layers to improve the algorithm's robustness against noise. It consists of three stages: two-layer mesh $\mathbf{M}_k^{0,1}$ and β mask initialization, per-view two-layer decomposition, and global consistency optimization to correct the possible errors in the per-view decomposition stage. The corrected layer images are fed to per-view decomposition as priors to refine the decomposition result further, as illustrated in Fig. 3. The reflection decomposition uses images before gamma correction due to the linear composition rule in Eq. 3. Usually, two-layer decomposition converges to a good-quality within two rounds. For reflected highlights with saturated intensities that break the composition equation in Eq. 3, we propose first to detect highlights and then rely on global consistency to obtain $\mathbf{I}_k^0$. In the following, we describe the details of each stage in our algorithm.

4.2.1 Initialization. Per-image reflection mask β_k is initialized by projecting the global plane with detected reflections onto the image k. The mask values of pixels covered by the projection are set to 1, otherwise 0. Afterward, we check whether a pixel belongs to the plane according to each pixel's distance to the plane and the difference between their normals. If the distance is below $0.03m$ and the angle between the two normal vectors is below $60°$, we set its mask value to 1 and 0 otherwise. Since we have refined the depth map to align depth and color edges, the β_k after this step will align to color edges. Note that we use the obtained β_k to indicate the reflection plane in the image, but do not optimize for β_k as in [Sinha et al. 2012].

As to two-layer mesh initialization for the plane with reflections, our system first computes a plane p_k using the refined depth in β_k and then utilize this plane to set the depth values for pixels inside β_k. Afterward, we initialize the depth of a reflection layer through reflections. Specifically, we create a virtual camera by reflecting the camera of image k with p_k and render a depth map of the indoor scene using the virtual camera. The rendered depth map inside β_k is used to initialize the mesh $\mathbf{M}_k^1$ as illustrated in Fig.3. A surface layer

mesh $\mathbf{M}_k^0$ is directly initialized using the refined depth inside β_k. Both meshes are constructed as described in Sec. 4.1 and simplified to reduce the degree of freedoms in optimization.

We initialize $\mathbf{I}_k^{0,1}$ using Eq. 8, but the median operation is replaced with minimum to initialize $\mathbf{I}_k^0$ and the initial $\mathbf{I}_k^1$ is set to $\mathbf{I}_k - \mathbf{I}_k^0$. Specifically, we warp the reference view to neighboring views through initialized two-layer meshes to compute Eq. 8. However, due to the error in the reconstructed depth and camera poses, there exists the misalignment between the warped reference image and images at neighboring views, leading to the degradation of the initialized $\mathbf{I}_k^0$ and the subsequent decomposition results. To this end, we perform as-rigid-as-possible local warping with 40×30 grids [Zhou and Koltun 2014] to reduce the misalignment in the initialization (Please refer to Sec.1 in "other supplementary materials" for details).

4.2.2 Alternating optimization for two-layer decomposition. The objective function developed to estimate two-layer meshes $\mathbf{M}_k^{0,1}$ and color images $\mathbf{I}_k^{0,1}$ is as follows:

$$\underset{(\mathbf{R},\mathbf{T})_k^1,\mathbf{M}_k^{0,1},\mathbf{I}_k^{0,1}}{\operatorname{argmin}} \quad E_d + \lambda_s E_s + \lambda_p E_p, \quad \text{s.t. } \mathbf{I}_k^{0,1}(\mathbf{u}) \in [0..1], \tag{4}$$

where $\mathbf{u}$ indicates a pixel in the reference image $\mathbf{I}_k$, and $(\mathbf{R},\mathbf{T})_k^1$ are the rotation and translation transformations for $\mathbf{M}_k^1$ respectively, two sets of augmented variables to estimate the global transformation error of reflection layer in the initialization. The weights λ_s, λ_p are used to balance between the data term E_d, the smoothness term E_s, and the prior term E_p, which are set to 0.04 and 0.01 respectively. We detail each term in the following:

Data Term: The data term E_d measures the difference between the image $\widehat{\mathbf{I}}_{k'}$ composed by rendering two-layer images and depth maps at a neighboring viewpoint k' and the captured photographs $\mathbf{I}_{k'}$, which is:

$$\mathrm{E}_{\text{data}} = \sum_{k' \in \mathcal{N}_k} \sum_{\mathbf{u}} \|\widehat{\mathbf{I}}_{k'}(\mathbf{u}) - \mathbf{I}_{k'}(\mathbf{u})\|^2 \tag{5}$$

$$\widehat{\mathbf{I}}_{k'}(\mathbf{u}) = \mathbf{I}_{k'}^0\left(\omega^{-1}\left(\mathbf{u}, \hat{\mathbf{D}}_k^0\right)\right) + \beta_{k'}\left(\omega^{-1}\left(\mathbf{u}, \hat{\mathbf{D}}_k^0\right)\right)\mathbf{I}_{k'}^1\left(\omega^{-1}\left(\mathbf{u}, \hat{\mathbf{D}}_k^1\right)\right),$$

where $\mathcal{N}_k$ consists of the image k and its 6 nearest neighboring images determined by their camera poses, $\mathbf{u}' = \omega^{-1}(\mathbf{u})$ is the warping function from k' to k based on depth maps $\hat{\mathbf{D}}_k^{0,1}$ that are obtained by rendering $\mathbf{M}_k^0$ and $\mathbf{R}_k^1\mathbf{M}_k^1 + \mathbf{T}_k^1$ at image k'. We determine $\mathcal{N}_k$ by rendering the reflection layer mesh at reference view to the neighboring views and then checking the depth overlap (depth difference $< 0.05 * \min(\text{depth})$) in β_k. The images with more that 30% depth overlap are sorted according to the camera pose difference (Eq.11), and top six images are kept to form $\mathcal{N}_k$.

Smoothness term: The smoothness term aims to minimize the gradient of separated color images $\mathbf{I}_k^{0,1}$ and the mean curvature normal of $\mathbf{M}_k^{0,1}$. We downscale the smoothness weights according to the color edges.

$$E_s = \sum_{\mathbf{u}} \left(e^{-\nabla \mathbf{I}_k^{0,1}(\mathbf{u})} \|\nabla \mathbf{I}_k^{0,1}(\mathbf{u})\|^2\right) + \sum_{\mathbf{v}} \|\mathbf{H}\mathbf{M}_k^{0,1}(\mathbf{v})\|^2 \tag{6}$$

where $\mathbf{v}$ is the vertex of $\mathbf{M}_k^{0,1}$, and $\mathbf{H}$ indicates the Laplacian matrix computed using cotangent weights [Desbrun et al. 1999].

Prior Term: We add a prior term on $\mathbf{I}_k^{0,1}$ To make the optimization stable:

$$E_p\left(\mathbf{I}_k^{0,1}\right) = \sum_{\mathbf{u}} \left(\|\mathbf{I}_k^0(\mathbf{u})\|^2\right) + \sum_{\mathbf{u}} \left(\|\mathbf{I}_k^1(\mathbf{u})\|^2\right) \tag{7}$$

Optimization: Since the optimization problem is complicated, we develop an alternating optimization algorithm to minimize the objective function of Eq. 4 as in [Sinha et al. 2012]. Specifically, after initializing the mask β and depth of surface and reflection layers at an image k, the algorithm alternatively solves for two sets of variables, namely $\mathbf{I}_k^{0,1}$ and $\left\{(\mathbf{R},\mathbf{T})_k^1, \mathbf{M}_k^{0,1}\right\}$ until convergence. In each iteration, each set of variables is solved by fixing another set of variables. Note that the 2D warping field is also used in the first iteration optimization for $\mathbf{I}_k^{0,1}$, since two-layer meshes are not optimized in this stage, and it is necessary to reduce the misalignment between 2D images caused by warping with initial meshes. In the second iteration optimization of $\mathbf{I}_k^{0,1}$, we discard the 2D warping field since the two-layer meshes have been optimized.

4.2.3 Global consistency optimization. It is used to enforce that the decomposed two-layer color images should be consistent among neighboring views. Precisely, we first warp the neighboring surface layer images $\mathbf{I}_{k'}^0$, in mask $\beta_{k'}^0$, to reference view according to $\mathbf{D}_{k'}^0$, and calculate the median value at each pixel to filter out erroneous RGB values output by the per-view decomposition. The updated surface layer images $\tilde{\mathbf{I}}_k^0$ at a pixel u is computed as follows:

$$\tilde{\mathbf{I}}_k^0(\mathbf{u}) = \text{median}\left(\left\{\mathbf{I}_{k'}^0\left(\omega^{-1}\left(\mathbf{u}, \mathbf{D}_k^0\right)\right) | k' \in \mathcal{N}_k\right\}\right) \tag{8}$$

The RGB values of reflection layer are updated by enforcing the linear composition rule in Eq.3:

$$\tilde{\mathbf{I}}_k^1(\mathbf{u}) = \begin{cases} \mathbf{I}_k(\mathbf{u}) - \tilde{\mathbf{I}}_k^0(\mathbf{u}) & \text{if } \mathbf{I}_k(\mathbf{u}) - \tilde{\mathbf{I}}_k^0(\mathbf{u}) - \mathbf{I}_k^1(\mathbf{u}) < 0 \\ \mathbf{I}_k^1(\mathbf{u}) & \text{otherwise} \end{cases} \tag{9}$$

After both layer images are updated for each view, all the updated images are fed into per-view reflection decomposition algorithm for another round of refinement. In this round, the updated surface and reflection layer images are used as additional priors such that the global consistency constraint is implicitly enforced. It can be simply implemented by adding the following two energy terms in the per-view decomposition:

$$\lambda_g \left(\sum_{\mathbf{u}} \|\mathbf{I}_k^0(\mathbf{u}) - \tilde{\mathbf{I}}_k^0(\mathbf{u})\|^2 + \sum_{\mathbf{u}} \|\mathbf{I}_k^1(\mathbf{u}) - \tilde{\mathbf{I}}_k^1(\mathbf{u})\|^2\right), \tag{10}$$

where the weight λ_g is set to 0.05.

4.2.4 Highlights. In indoor scenes, highlights often appear on glossy surfaces due to their reflection of light sources. However, pixels inside highlights have over-saturated intensities, which can not be modeled by Eq. 3. Therefore, we detect the pixels with highlights as shown in Fig. 4 to avoid the computation of E_{data} for these pixels. We rely on multi-view consistency to improve the robustness of highlight detection. First, we leverage mean-shift clustering (spatial radius equals 8, color radius equals 14) to extract the candidate highlight clusters where most pixels (larger than 90%) have high intensity (larger than 0.8) for all the captured images. Second, we eliminate

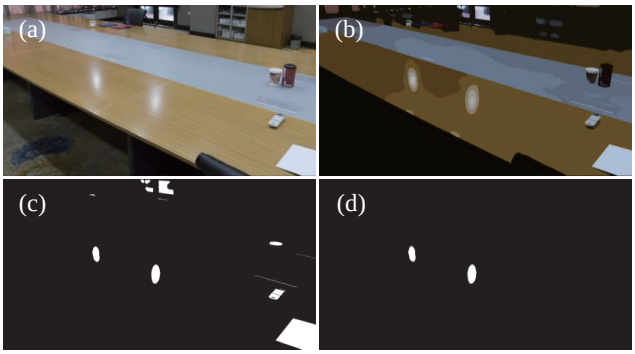


Fig. 4. The procedure of highlights detection. (a) Reference color image. (b) Mean-shift clusters. (c) Candidate highlight clusters. (d) Final highlight clusters.

false positives, such as white papers, by warping candidate highlight clusters to images neighboring to the reference image according to surface layer depth $\mathbf{D}^0_{k'}$. If the overlap between a cluster and other clusters in the neighboring images is less than 80% on average, this cluster is considered as true highlight clusters. This because that the reflected highlights should not be warped according to surface layer depth. Fig. 4 shows the highlight detection results.

In reflection decomposition, we first initialize the highlight pixels in $\mathbf{I}^0_k(\mathbf{u})$ with 0 and $\mathbf{I}^1_k(\mathbf{u})$ with $\mathbf{I}_k(\mathbf{u})$, and then optimize for $\mathbf{I}^{0,1}_k$ without detected pixels. However, after the global consistency optimization, these 0 values in $\mathbf{I}^0_k$ will be filled by the median RGB values of corresponding pixels in its neighboring views. The filled values can be used in the smoothness term and Eq. 10 in the second round optimization of $\mathbf{I}^{0,1}_k$. In our implementation, the RGB values of highlight pixels in $\mathbf{I}^1_k$ will not be modified during global consistency optimization. The mesh optimization of surface and reflection layer remains the same, and we directly use the RGB values of highlight pixels inside $\mathbf{I}^1_k$ to compute the E_{data}.

4.2.5 Mirrors. As mirrors are perfectly reflective surfaces without texture, we choose to set $\mathbf{I}^1_k = \mathbf{I}_k$ and $\mathbf{I}^0_k = \mathbf{0}$ for the pixels inside a mirror. Next, instead of optimize $\mathbf{D}^{0,1}_k$ and β_k, we fixed the $\mathbf{D}^0_k$ for the mirror using the reconstructed 3D mirror plane during scanning and only optimize for the $\mathbf{D}^1_k$ in reflection decomposition. Besides, if there are opaque materials attached to the mirror, its reconstructed geometry will be on the 3D mirror plane. We leverage this condition to set the pixel covered by opaque materials to 0 in β_k, which means that there are no reflections inside these pixels. These pixels are not considered in the reflection decomposition. Besides, to obtain clear textures of mirrors, we develop an algorithm to remove the camera reflected in a mirror. Please refer to Sec.2 in "other supplementary materials" for details.

5 VIEW WARPING USING TWO-LAYER MESH REPRESENTATION

The view warping algorithm first warps surface and reflection layer images with a two-layer mesh to a novel view $\mathbf{V}_n$ respectively and then obtain the synthesized image by compositing the warped surface and reflection images using Eq. 3. While tile-based inside-out IBR algorithm can apply to the rendering of each layer [Hedman et al. 2016], we observe that this pipeline often leads to discontinuity artifacts as illustrated in Fig. 5. We hypothesize that the discontinuity artifact is related to two factors. First, the view selection of tile-based rendering is independent in each visible grid cell, which may result in inconsistent view selection in neighboring cells. Second, IBR-cost based weighting scheme is independently computed at each pixel. This scheme aims to select the closest ray for each pixel, a principle often used in classical IBR algorithms [Gortler et al. 1996; Levoy and Hanrahan 1996]. However, it may also result in non-smooth weight distribution. Therefore, we design a view-based IBR algorithm to improve the smoothness of the rendering results. The term view-based indicates that the view selection and blending weights are based on the camera poses of views relative to the novel view $\mathbf{V}_n$. The composited images at $\mathbf{V}_n$ are fed into the DSRNet to obtain the final high-resolution rendered images. The overall two-layer rendering algorithm consists of two steps: front-most depth map (FMDP) generation and camera-pose-based view warping. Front-most depth is used in the fuzzy depth test in the rendering, similar to [Hedman et al. 2016]. We will describe the details of each step below.

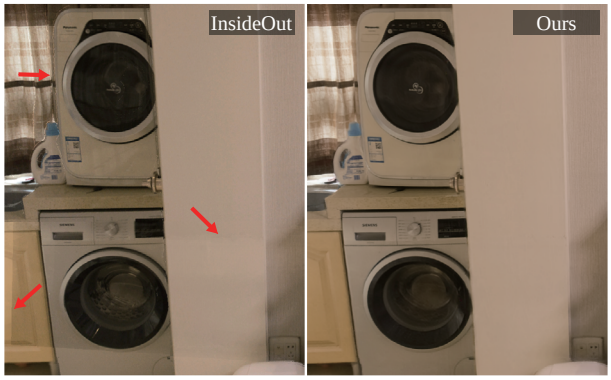


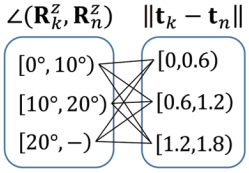
Fig. 5. Compared with InsideOut[Hedman et al. 2016], our view warping can avoid discontinuity artifacts and produce smoother image blending result.

View selection:We define a distance d_k between a view $\mathbf{V}_k$ and the novel view $\mathbf{V}_n$ to facilitate the view selection as follows:

$$d_k = \angle\left(\mathbf{R}^z_k, \mathbf{R}^z_n\right) * \pi/180 + \lambda\|\mathbf{t}_k - \mathbf{t}_n\|/\|\mathbf{t}_n\| \tag{11}$$

where the camera pose information for a view $\mathbf{V}_k$ includes the optical center $\mathbf{t}_k$ and optical axis $\mathbf{R}^z_k$, i.e. the z axis of the camera rotation matrix. This distance focuses on the consistency of the look-at direction of each camera by only using $\mathbf{R}^z_n$ as a simplified representation of a camera orientation. Empirically, we found it worked well when there seldom exist the camera rotations around its optical axis. The weight λ equals 0.1, and $\angle\left(\mathbf{R}^z_k, \mathbf{R}^z_n\right)$ is the angle between $\mathbf{R}^z_k$ and $\mathbf{R}^z_n$.

after view warping, we choose to independently select closest views at eight quadrants of the local coordinate system of $\mathbf{V}_n$ and continue to split each quadrant into nine sub-regions based on the pairwise combination of three angle and three distance intervals (see the right inset). This design



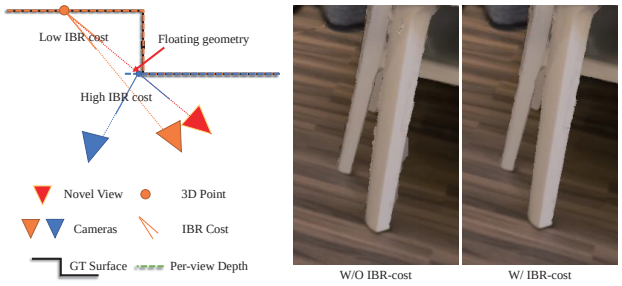


Fig. 6. The floating geometry at one pixel will be removed if its IBR-cost is high with respect to the novel view.

can select views surrounding $\mathbf{V}_n$ with a more uniform distribution and is empirically beneficial to the view-blending. With this setting, there will be 72 views in maximum. However, due to the distance conditions, it is usually around 20 views selected.

It is also important to maintain the overlap between selected views for $\mathbf{V}_n$ and that for previous view $\mathbf{V}_{n-1}$ to reduce the temporal flickers. We achieve this by delaying the removal of selected views at $\mathbf{V}_{n-1}$. Precisely, we first calculate $\delta\angle(\mathbf{R}_k^z, \mathbf{R}_n^z)$ and $\delta\|\mathbf{t}_k - \mathbf{t}_n\|$ between $\mathbf{V}_n$ and $\mathbf{V}_{n-1}$, then add them to corresponding sub-region conditions. The selected views at $\mathbf{V}_{n-1}$ that satisfies the updated conditions are also selected for $\mathbf{V}_n$.

Front-most Depth Map (FMDP) Generation: The FMDP is used in fuzzy depth test: If the absolute distance between the depth of a pixel and the corresponding depth in the FMDP is less than 3cm, this pixel is deemed to be visible at $\mathbf{V}_n$ and will be used in view-blending.

Given the view selection result, we can blend the depth maps generated by rendering the meshes of these views at $\mathbf{V}_n$ to obtain FMDP. However, floating geometry, i.e., the geometry errors, of per-view meshes will downgrade the blended depth map's quality and lead to incorrect depth test results. Thus, for pixels with large depth value variances, we utilize per-pixel IBR-cost to reduce the influence of floating geometry [Hedman et al. 2018]. That is, we first determine the least IBR-cost from the 3D point projected to the same pixel of $\mathbf{V}_n$ and then only keep those points whose difference of its IBR-cost to the least IBR-cost is less than a threshold (default 0.17). The minimal depth of these remained 3D points is then selected to be the front-most depth. The IBR-cost is defined in the same way as in [Hedman et al. 2016]:

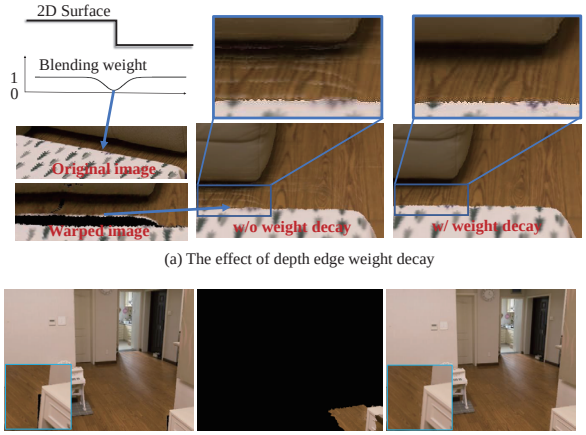
$$c(\mathbf{t}_k, \mathbf{t}_n, \mathbf{x}) = \angle(\mathbf{t}_k - \mathbf{x}, \mathbf{t}_n - \mathbf{x}) * \pi/180 + \max(0, 1 - \|\mathbf{t}_n - \mathbf{x}\|/\|\mathbf{t}_k - \mathbf{x}\|) \quad (12)$$

where $\mathbf{x}$ is a 3D point. As shown in Fig. 6, this modification can reduce the influence of floating geometry, especially when depth edges might be missed in depth-color edge alignment, if there is no color edges close to the depth edges.

View blending: We render per-view textured two-layer mesh of selected views at $\mathbf{V}_n$ and then blend the rendered surface and reflection images separately. The fuzzy depth test is first used to remove hidden triangles before blending, and the blending weight for a selected view $\mathbf{V}_k$ is defined as follows:

$$w_k = \exp(-d_k/\delta) \quad (13)$$

where δ is set to 0.033 in our implementation. This weighting scheme favors those views close to $\mathbf{V}_n$ in view warping, and the blending



(a) The effect of depth edge weight decay

(b) Hole filling

Fig. 7. (a) We decay the weights near occlusion edges to improve the smoothness of view blending. (b) We leverage tile-based rendering [Hedman et al. 2016] to render the pixels inside a hole, and also blend the rendering result with the rest view warping result with weight decay at the hole boundary.

weight is the same for surface and reflection layer images. To avoid discontinuity artifact, we first apply image feathering, a weight decay operation often used in image stitching, near the warped image boundaries [Szeliski 2006]. It is achieved by decreasing the blending weight w_k smoothly to 0 within 20-pixel distance to the image boundaries, which is efficient to remove discontinuity caused by color variation among images. Second, we also exploit weight decay to decrease the weight of pixels near depth edges (± 5 pixels distance to the depth edges along edge gradients), since these pixels might contain noise and be warped to semantically different objects in the scene, as shown in Fig. 7. The weight decay can reduce the weight for such pixels, and those warped pixels far from the depth edge in another view will contribute more to the final RGB value. All the decayed weights are stored in the alpha channel of mesh textures.

There might be small-area holes left after the camera-poses-based view blending, as shown in Fig. 7(a). For pixels inside holes, we leverage the tile-based rendering method in [Hedman et al. 2016] to render the voxels and their eight neighbors that intersect with the surface of these pixels.

6 DSRNET

While per-view depth refinement can significantly improve mesh quality, there are still inaccurate geometries after edge-aware interpolation. As a result, the rendering results of the two-layer view warping algorithm inevitably contain a few artifacts, such as ghosting or zigzag at object boundaries. We thus develop the DSRNet that non-trivially adapt the real-time, super-sampling network (RSSNet) in [Xiao et al. 2020] to improve the quality and resolution of rendered images at a novel view $\mathbf{V}_n$. The overall structure of our DSRNet is similar to RSSNet, which also has three modules: feature extraction, re-weighting, and reconstruction. However, the DSRNet differs from RSSNet in two aspects. First, we add a motion vector refinement (MVR) module to correct the correspondence errors caused by inaccurate geometry before re-weighting. It can improve

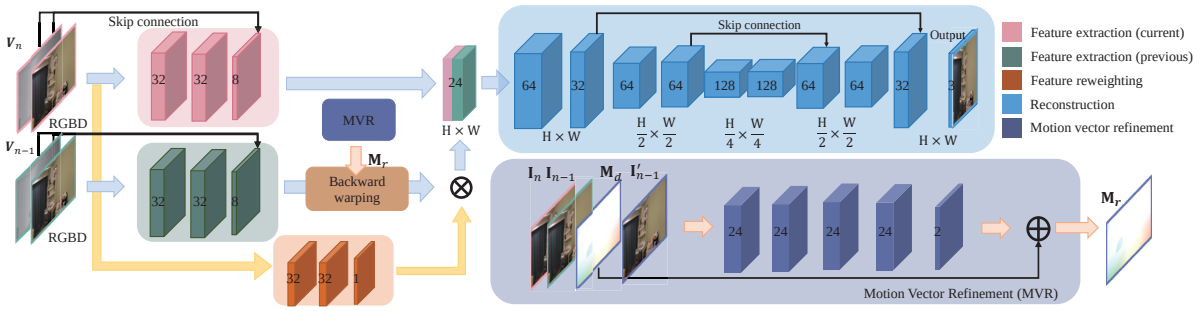


Fig. 8. Network architecture of our method. Our network consists of four modules, including feature extraction, re-weighting, reconstruction and motion vector refinement(MVR). The numbers on each network layer represent the output channels. In reconstruction module, the height (H) and width (W) of output features are marked under corresponding network layers. The kernel size is 3×3 at all layers excepts the first layer of MVR, whose kernel size is 5×5 instead.

the correspondence between the previous view $\mathbf{V}_{n-1}$ and $\mathbf{V}_n$. Consequently, the output image quality is improved. Second, although the DSRNet allows us to store textures of $1/4$ resolution, namely $H/2 \times W/2$, of the rendered image of resolution $H \times W$ to save the memory storage, we found there would be more aliasing artifacts near occlusion edges if we render $1/4$ resolution image in two-layer view warping algorithm as the input of the DSRNet. Hence, we choose to construct per-view meshes using high-resolution (HR) images and preserve mesh edges as-much-as possible on the occlusion edges during mesh simplification. Meanwhile, during rendering, with $1/4$ resolution textures, we render blurred images of the resolution $H \times W$ using two-layer view warping algorithm, which reduces aliasing in the DSRNet output further.

Network Architecture: Fig. 8 shows the architecture of our network. For the three modules in the original RSSNet [Xiao et al. 2020], such as feature extraction, re-weighting, and reconstruction modules, the convolutional kernel size is 3×3 at all layers. The feature extraction module has three convolutional layers, and each of them is followed by a nonlinear activation ReLU layer. The re-weighting module also has three convolutional layers. However, it uses Tanh as the activation function at its last layer, which is followed by a scaling operation to map the output values from $(-1, 1)$ to $(0, 10)$. The reconstruction module is a 10-layer U-net with three scales [Ronneberger et al. 2015]. In the reconstruction module, we use max-pooling for downsampling and bilinear interpolation for upsampling.

Due to the existence of floating or inaccurate geometry at each view, simply warping the color image of the previous frame $\mathbf{I_{n-1}}$ to $\mathbf{V}_n$ by the per-view mesh might lead to inaccurate correspondence, resulting in the degradation of the output image. Therefore, the MVR module is designed to refine the depth-based motion vector $\mathbf{M_d}$ used to warp $\mathbf{I_{n-1}}$ before the re-weighting module. This module takes depth-based motion vector $\mathbf{M_d}$, color image of current frame $\mathbf{I_n}$ (rendered at $\mathbf{V}_n$ using the two-layer IBR algorithm), previous frame $\mathbf{I_{n-1}}$, and warped previous frame $\mathbf{I'_{n-1}}$ as input, and aims to predict offsets to correct the motion vectors $\mathbf{M_d}$:

$$\mathbf{M}_r = \mathbf{M}_d + \mathrm{MVR}(\mathbf{I}_n, \mathbf{I}_{n-1}, \mathbf{I}'_{n-1}, \mathbf{M}_d) \tag{14}$$

where $\mathbf{M_r}$ are the refined motion vector. Since MVR is designed for motion vector fine-tuning, the output of MVR is limited to $[-5, 5]$ pixels by mapping original MVR output x to $\tanh(x) * 5$.

Training Losses: The training loss of our method is same as [Xiao et al. 2020], which is the weighted combination of structural similarity index (SSIM) and perceptual loss. More formally, the total training loss is as follows:

$$\mathrm{L}(\mathbf{x}, \hat{\mathbf{x}}) = 1 - \mathrm{SSIM}(\mathbf{x}, \hat{\mathbf{x}}) + w \cdot \sum_{i=1}^{5} ||\mathrm{conv}_i(\mathbf{x}) - \mathrm{conv}_i(\hat{\mathbf{x}})||_2^2 \tag{15}$$

where $\mathbf{x}$ and $\hat{\mathbf{x}}$ are the captured ground-truth images and network output respectively. Here weight w is used to balance the two losses and in most of our experiments, $w = 0.1$.

As to the MVR module, we use a warping loss to supervise its training as follows:

$$\mathrm{L}_{warp} = \mathrm{L1}\left(\mathrm{G}(\mathbf{I}_n), \mathrm{G}(\mathbf{I}'_{n-1})\right) + \mathrm{L1}\left(\mathbf{I}_n, \mathbf{I}'_{n-1}\right) \tag{16}$$

where $L1(\cdot)$ denotes the L1 Loss and $G(\cdot)$ is the Gaussian filter with 5×5 kernels. Here Gaussian Filter is used to smooth the local gradient and avoid gradient vanishing for not color edge pixels.

7 IMPLEMENTATION DETAILS

Per-view reflection decomposition: Since there should be a large number of optimization variables to represent surface and reflection layer RGBD images in Eq. 4, we apply conjugate gradient (CG) with Polak-Ribiére updates [Nocedal and Wright 2006] to alternatively solve for $\mathbf{I}_k^{0,1}$ and $\mathbf{D}_k^{0,1}$. During the step to optimize for $\mathbf{I}_k^{0,1}$, we clip the gradient of each pixel to be in the range ([0..1]). In all our experiments, we fix CG iteration number to be 30 to balance between the energy reduction and time consumption.

Two-layer IBR Rending: All the mesh textures are stored using RGBA **DXT5** compression format, which also provides the 4:1 compression ratio. The alpha channel is used to store the weight decay near image boundaries and occlusion edges and the mask of highlight pixels (1 bit packed into the 8bit alpha channel). During rendering, the highlight pixels will not composite with the surface layer since it will lead to the degradation of the rendered highlight. In this case, we ignore the surface layer color and only use the reflection layer pixels in the area with detected highlights warped to the novel view $\mathbf{V}_n$. Each per-view mesh is simplified, and we use unsigned short to store the vertex indices to reduce storage further.

SR Network Training: We train the SR network with the MVR module using $512 * 512$ random crops from the captured indoor scene images and set batch size to 8. The optimizer for the training

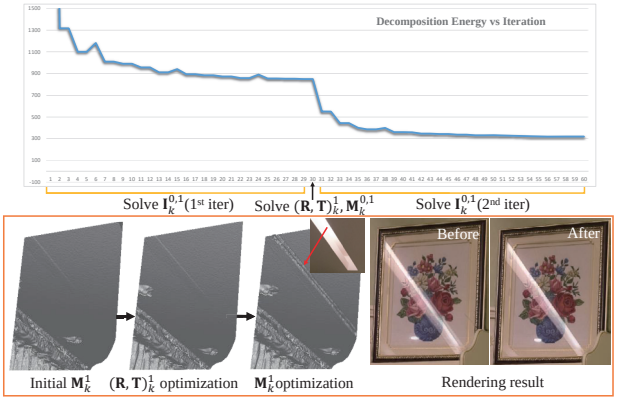


Fig. 9. Convergence curve of the alternative optimization algorithm for reflection decomposition. The rendering results before and after optimization are shown in the last two images in second row.

Scenes	Area(m^2)	#Img	Img/Mesh Storage(GB)
Hotel Room	$7.0 * 4.4$	1741	0.55 / 1.89 (1.61+0.28)
Living Room 1	$8.2 * 6.3$	2289	0.72 / 2.48 (2.17+0.31)
Living Room 2	$12.3 * 8.1$	2782	0.88 / 3.04 (2.40+0.64)
Meeting Room 1	$11.2 * 6.5$	1631	0.52 / 1.84 (1.61+0.23)
Meeting Room 2	$13.3 * 10.4$	998	0.32 / 1.22 (0.88+0.34)

Table 1. Statistics of reconstructed indoor scenes. #Img denotes the number of total images captured in the scene. Img/Mesh Storage denotes the memory storage for down-sampled texture images and two-layer meshes. Numbers in brackets indicate the memory storage for surface and reflection layer meshes respectively.

is ADAM method [Kingma and Ba 2015], and the learning rate is set to 1×10^{-4} at the beginning and decayed by a factor of 0.95 every 20 epochs. The number of epochs used to train the network is 300.

8 EXPERIMENTS

We have implemented our IBR pipeline on a desktop PC with a 4.20GHz Intel Core i7-7700K CPU and an NVIDIA RTX 2080Ti GPU. The DSRNet is trained on a GPU server with two NVIDIA RTX 2080Ti GPU cards. The forward inference of the network is accelerated by Nvidia TensorRT [Nvidia 2018] with 16-bit precision. All the per-view two-layer meshes are stored in GPU memory. We utilize OpenGL/CUDA interop interface to interchange rendering buffer and network tensor data at GPU memory directly during online rendering. Our pipeline's average running time to render an image of resolution 1280×960 is 49.1ms, including 30.7ms for view warping and 18.4ms for the DSRNet inference.

To evaluate our pipeline, we have applied it to render five reconstructed indoor scenes with different sizes, types, and reflection scenarios, including one hotel room, two living rooms, and two meeting rooms (see Table. 1). We train DSRNet separately for each scene, using 90% of the captured images as the training dataset and the remaining 10% as the validation set. In this section, we will report the evaluation results of reflection decomposition, DSRNet, and the rendering result comparisons with state-of-the-art IBR methods. Please also see the accompanying video for the video comparisons.

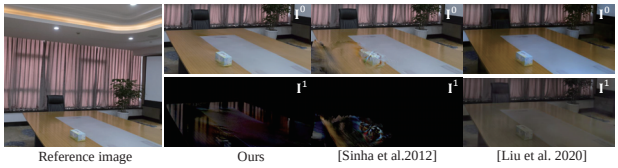


Fig. 10. Reflection decomposition comparison.

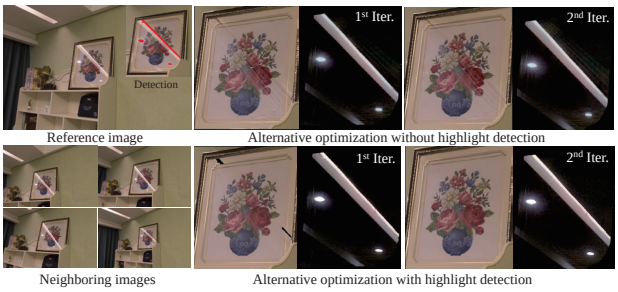


Fig. 11. Two-layer decomposition with highlight detection. Red regions on the top-right of reference image indicate the detected highlights. Without highlight detection, the highlights in neighboring views will lead to spreading artifacts as shown in the decomposed foreground surface image in top row.

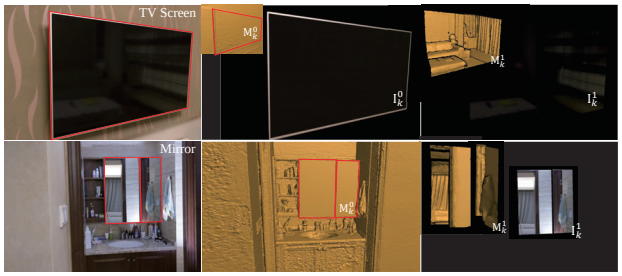


Fig. 12. Reflection decomposition results of a TV screen and a mirror.

8.1 Evaluation of the Reflection Decomposition Algorithm

Fig. 9 illustrates the convergence curve of the alternating optimization algorithm for reflection decomposition. The energy defined in Eq. 4 gradually decreases with each CG iteration when optimizing for $\mathbf{I}_k^{0,1}$ at the beginning. After 30 iterations, the algorithm continues with optimizing for $\mathbf{M}_k^{0,1}$ and $(\mathbf{R},\mathbf{T})_k^1$, leading to the further decline of the energy function. Usually, the alternating optimization algorithm converges with two outer iterations to optimize for $\mathbf{I}_k^{0,1}$ alternatively. The red arrow in Fig. 9 is used to emphasize the effect of the optimization of $\mathbf{M}_k^1$. It can be seen that the rendering result using the optimized $\mathbf{I}_k^{0,1}$ and $\mathbf{M}_k^1$ is sharp and free of misalignment artifacts in highlights caused by view warping with initialized two-layer meshes and images. A comparison in Fig. 10 shows that, with prior geometry, our reflection decomposition result is superior to the results of reflection removal algorithms based on semi-global stereo [Sinha et al. 2012] and deep-learning [Liu et al. 2020b]. We hypothesize that the failure of the algorithm in [Sinha et al. 2012] is due to the difficulty to reliably estimate the two-layer depth using semi-global stereo algorithm [Hirschmuller 2008]. Since we do not capture the images continuously as in videos, it is also challenging to estimate dense optical flows for surface and reflection layers required in [Liu et al. 2020b].

Fig. 13. View warping vs. DSRNet. For each pair, the image on the left is produced by our view warping algorithm, and the image on the right is produced by the DSRNet. The blurring and aliasing at object boundaries are effectively removed by the DSRNet.

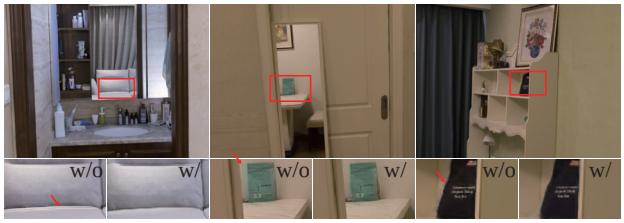


Fig. 14. The improvement of object boundary rendering quality using the MVR module. W/: with MVR. W/O: without MVR.

In Fig. 11, we show how the highlight detection influences the reflection decomposition result. If we use the linear composition rule in this case, the highlights in neighboring views will lead to artifacts in the foreground surface image, resulting in a large area of artifacts in the decomposed surface image. The artifacts are corrected after ignoring the composition energy terms inside the detected regions with highlights. The holes inside the highlights of the decomposed surface image are filled by the global consistency step. Fig. 12 shows the two-layer image and mesh construction results of a TV screen and a mirror. Since we enforce the RGB of $\mathbf{I}_k^0$ of mirrors to be zero, a color-less assumption for mirrors, we did not show black $\mathbf{I}_k^0$ for the mirror. The TV screen's depth can be scanned with Kinect4 due to its surface matte. It benefits the initialization of the surface layer mesh and helps to obtain high-quality reflection decomposition, as shown in the top-row of Fig. 12.

8.2 Evaluation of DSRNet

After training, the DSRNet can produce sharp, high-quality HR images at novel viewpoints. As illustrated in Fig. 13, although the images from view warping are blurred and have alias artifacts around edges, the image quality can be effectively enhanced by the DSRNet. Moreover, Fig. 14 illustrates that the designed MVR module is beneficial to remove the ghosting artifacts caused by inaccurate geometries.

Ablation Study:We perform ablation studies to evaluate the influence of the MVR module and loss terms on the DSRNet. As shown in Table 2, the network with the MVR module can improve PSNR values for all our reconstructed scenes and is beneficial to the improvement of the SSIM metric. In Fig. 14, we show that the ghosting artifacts indicated by the red arrows can be corrected after integrating the MVR module into the DSRNet. Moreover, we remove each loss term to evaluate its influence on the network. The evaluation

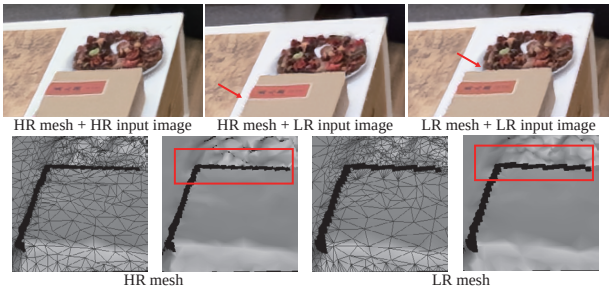


Fig. 15. The influence of mesh and input image resolution to DSRNet. HR/LR mesh: mesh constructed using high/low resolution depth map. HR/LR input image: generate high-resolution or low-resolution images with view warping. HR mesh + HR input image leads to better rendering quality. Please also see accompanying video for the comparison.

Scene	W/ MVR		W/O MVR	
	PSNR↑	SSIM↑	PSNR↑	SSIM↑
Hotel Room	**33.57106**	**0.96860**	33.20517	0.96856
Living Room 1	**31.35471**	0.96757	31.35119	**0.96785**
Living Room 2	**30.01572**	**0.95905**	29.43158	0.95892
Meeting Room 1	**30.46487**	**0.98267**	29.96584	0.98134
Meeting Room 2	**31.37820**	**0.96296**	30.73507	0.96277

Table 2. MVR ablation study

Only VGG		Only SSIM		SSIM+VGG	
PSNR↑	SSIM↑	PSNR↑	SSIM↑	PSNR↑	SSIM↑
28.13277	0.95896	33.18738	**0.96884**	**33.57106**	0.96860

Table 3. Loss term ablation study.

results on the hotel room scene for this ablation study are shown in Table 3. They verify that both VGG loss and L1 loss are essential to the quality of the rendered images.

The influence of view warping result on the DSRNet: As described in Sec. 6, although we store textures in 1/4 resolution (640x480), we choose to generate images with original HR (1280x960) resolution in view-warping to reduce aliasing and flickers near occlusion edges. In Fig. 15, we show that generating HR images with per-view meshes constructed with HR images can achieve superior rendering results. The reason we choose to construct mesh with HR images is to preserve the edges on occlusion edges. Therefore, the occlusion edge details of the HR RGBD images can be better preserved, which facilitates the DSRNet to produce high-quality images. It can be seen the mesh constructed on LR RGBD images has much fewer boundary edges, leading to blurring or aliasing artifacts around occlusion edges. Furthermore, we found it is also beneficial to recover occlusion edge details if generating HR images in view warping.

8.3 Rendering Results and Comparisons

To demonstrate the advantage of our pipeline, we compare our method against state-of-the-art view synthesis methods, such as **InsideOut** [Hedman et al. 2016], **DeepBlending** [Hedman et al. 2018], **Neural Rerendering in the Wild (NRW)** [Meshry et al. 2019], **LLFF** [Mildenhall et al. 2019], **NeRF** [Mildenhall et al. 2020] and **FVS** [Riegler and Koltun 2020a]. For fair comparisons, we use captured high-resolution images plus our constructed per-view meshes as the input of InsideOut and DeepBlending. For **NRW**, we use a

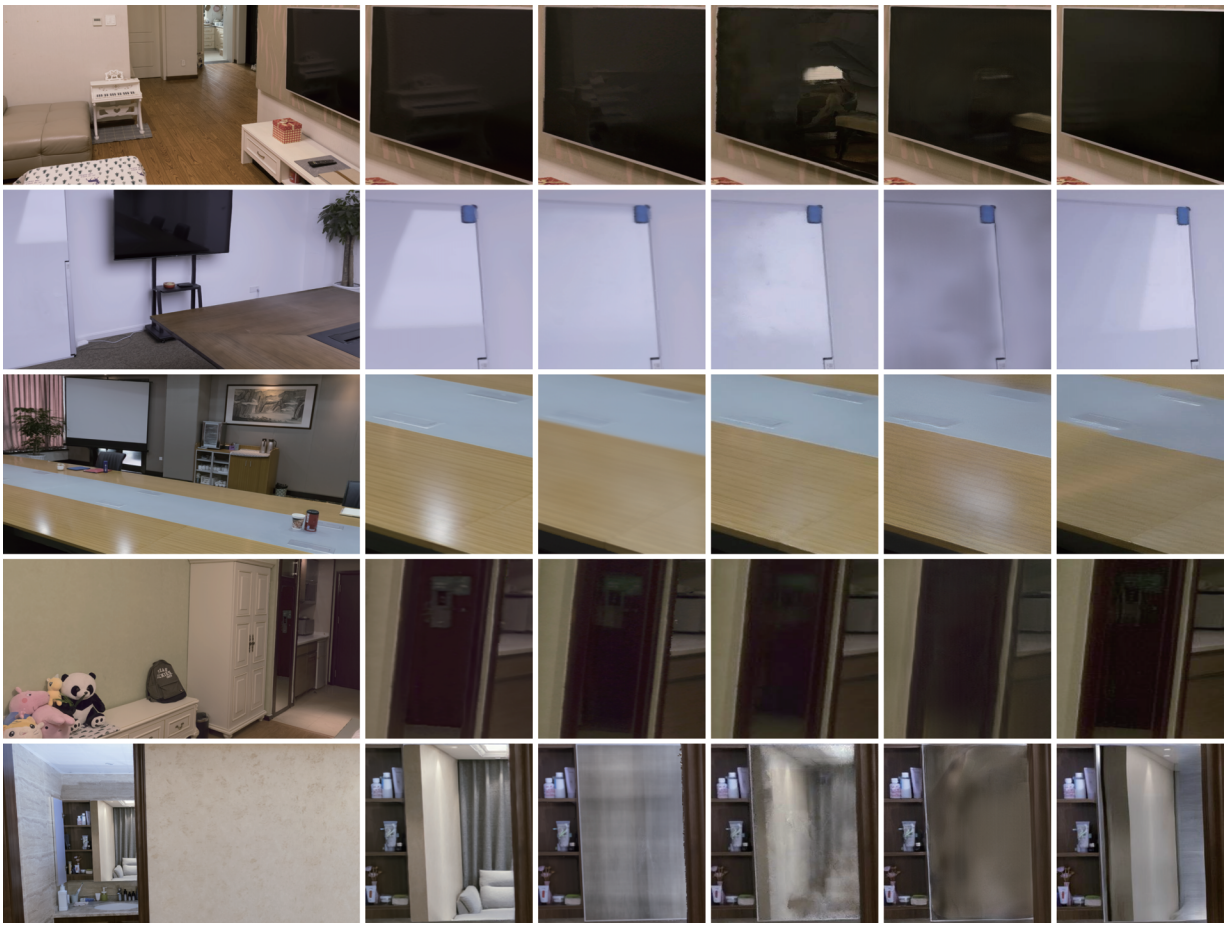


Fig. 16. Rendering result comparisons with **InsideOut** [Hedman et al. 2016], **DeepBlending** [Hedman et al. 2018] , **NRW** [Meshry et al. 2019] and **FVS** [Riegler and Koltun 2020a].

Scene	Metric	Deep Blending	Inside Out	NRW	FVS	Ours
Hotel Room	PSNR↑	32.90	31.56	31.11	27.26	**33.57**
	SSIM↑	0.881	0.880	0.831	0.835	**0.968**
Living Room 1	PSNR↑	31.33	29.99	30.13	25.54	**31.35**
	SSIM↑	0.875	0.881	0.828	0.833	**0.968**
Living Room 2	PSNR↑	29.04	29.77	27.93	25.32	**30.40**
	SSIM↑	0.828	0.827	0.785	0.808	**0.961**
Meeting Room 1	PSNR↑	29.86	29.19	25.70	24.61	**30.46**
	SSIM↑	0.926	0.934	0.875	0.871	**0.983**
Meeting Room 2	PSNR↑	**31.70**	30.27	29.57	26.38	31.38
	SSIM↑	0.865	0.871	0.802	0.809	**0.963**

Table 4. Quantitative comparisons.

textured global mesh generated by RealityCapture to render the input color and depth images. The required semantic map is obtained by segmenting the image with indoor scene class labels using the network provided by **NRW**. Furthermore, as our DSRNet is trained for each scene to improve rendering quality, we also fine-tune the networks of **DeepBlending** and **FVS** for the comparisons. As shown in Fig. 16, our method outperforms other methods on the rendering quality of reflections. With the developed reflection decomposition algorithm and the DSRNet, our system also achieves sharper rendering results. The quantitative comparisons conducted on the five reconstructed scenes are shown in Table 4, where our pipeline achieves the best performance over state-of-the-art methods on the validation datasets.

Fig. 17. Comparisons with **LLFF** [Mildenhall et al. 2019] and **NeRF** [Mildenhall et al. 2020].

Fig. 17 illustrates the comparisons with **LLFF** and **NeRF**. While these two methods can render high-quality images, it is still challenging for them to handle the high-frequency signals, such as reflections and check patterns, resulting in obvious blurring artifacts. In contrast, our geometry-based IBR pipeline can produce sharp images in these challenging cases.

9 CONCLUSIONS AND DISCUSSIONS

We have developed an IBR pipeline for the image-based rendering of indoor scenes with reflections. It has two main technical components: global-mesh-guided robust two-layer mesh construction and DSRNet based rendering pipeline to save memory storage. We also design a view-warping algorithm to produce temporally smooth images during free-viewpoint navigation as the input of DSRNet. Our pipeline can handle various types of reflections and achieve high-quality rendering results. Its running time with NVIDIA RTX 2080Ti GPU is below 50ms on average, suitable for interactive virtual reality applications.

A limitation of our current pipeline is that it can not handle curved mirrors or reflective surfaces. Empirically, a curved reflective surface can be approximated by many piece-wise triangles, and we can construct a reflection layer mesh for each triangle. However, the memory cost of this simple extension is high, and the rendering speed is substantially reduced. Rendering an environment map using our IBR pipeline for a curved reflective surface can be an alternative method to simulate its reflection. Another challenge to our pipeline is rendering the glasses that have both background transmissions and reflections. The linear composition rule used in our paper is only for reflective surface and its reflections. We might need to extend it to three layers, including transmissions, reflections, and possible opaque or transparent materials, such as papers or stickers, on the glasses, to handle the rendering of glasses. Currently, our rendering pipeline mainly works in RGB space, and the lightweight DSRNet is selected for the rendering speed. In the future, it would be interesting to investigate how to integrate feature space representation, similar to neural texture [Thies et al. 2019b] and stable view synthesis [Riegler and Koltun 2020b], into the pipeline to balance between the rendering speed and the robustness to inaccurate geometry in IBR.

REFERENCES

Michael Broxton, John Flynn, Ryan Overbeck, Daniel Erickson, Peter Hedman, Matthew Duvall, Jason Dourgarian, Jay Busch, Matt Whalen, and Paul Debevec. 2020. Immersive Light Field Video with a Layered Mesh Representation. *ACM Trans. Graph.* 39, 4, Article 86 (July 2020), 15 pages.

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 425–432.

Jose Caballero, Christian Ledig, Andrew P. Aitken, Alejandro Acosta, Johannes Totz, Zehan Wang, and Wenzhe Shi. 2017. Real-Time Video Super-Resolution with Spatio-Temporal Networks and Motion Compensation.. In *CVPR*. IEEE Computer Society, 2848–2857.

CapturingReality. 2016. Reality capture, http://capturingreality.com.

R. O. Cayon, A. Djelouah, and G. Drettakis. 2015. A Bayesian Approach for Selective Image-Based Rendering Using Superpixels. In *2015 International Conference on 3D Vision*. 469–477.

Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. *ACM Trans. Graph.* 36, 4, Article 98 (2017), 12 pages.

Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. 2013. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)* 32, 3 (2013), 1–12.

Gaurav Chaurasia, Olga Sorkine, and George Drettakis. 2011. Silhouette-Aware Warping for Image-Based Rendering. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 1223–1232.

Shenchang Eric Chen and Lance Williams. 1993. View Interpolation for Image Synthesis. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (Anaheim, CA) *(SIGGRAPH '93)*. Association for Computing Machinery, New York, NY, USA, 279–288. https://doi.org/10.1145/166117.166153

Tao Chen, Zhe Zhu, Ariel Shamir, Shi-Min Hu, and Daniel Cohen-Or. 2013. 3-Sweep: Extracting Editable Objects from a Single Photo. *ACM Trans. Graph.* 32, 6, Article 195 (Nov. 2013), 10 pages.

Paul E Debevec, Camillo J Taylor, and Jitendra Malik. 1996. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 11–20.

Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. 1999. Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. 317–324.

Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. 2014. Learning a deep convolutional network for image super-resolution. In *Computer Vision–ECCV 2014*. Springer, 184–199.

Siyan Dong, Kai Xu, Qiang Zhou, Andrea Tagliasacchi, Shiqing Xin, Matthias Nießner, and Baoquan Chen. 2019. Multi-Robot Collaborative Dense Scene Reconstruction. *ACM Trans. Graph.* 38, 4, Article 84 (July 2019), 16 pages. https://doi.org/10.1145/3306346.3322942

Andrew Edelsten, Paula Jukarainen, and Anjul Patney. 2019. Truly next-gen: Adding deep learning to games and graphics. In *In NVIDIA Sponsored Sessions (Game Developers Conference)*.

John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. 2019. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2367–2376.

John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. 2016. Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5515–5524.

Dario Fuoli, Shuhang Gu, and Radu Timofte. 2019. Efficient Video Super-Resolution through Recurrent Latent Space Propagation. In *2019 IEEE International Conference on Computer Vision Workshop (ICCVW)*. 3476–3485.

Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. 2009. Reconstructing building interiors from images. In *2009 IEEE 12th International Conference on Computer Vision*. 80–87.

Y. Furukawa and J. Ponce. 2010. Accurate, Dense, and Robust Multiview Stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 8 (2010), 1362–1376.

Michael Garland and Paul S. Heckbert. 1997. Surface Simplification Using Quadric Error Metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97)*. ACM Press/Addison-Wesley Publishing Co., USA, 209–216. https://doi.org/10.1145/258734.258849

Michael Goesele, Jens Ackermann, Simon Fuhrmann, Carsten Haubold, Ronny Klowsky, Drew Steedly, and Richard Szeliski. 2010. Ambient Point Clouds for View Interpolation. In *ACM SIGGRAPH 2010 Papers* (Los Angeles, California) *(SIGGRAPH '10)*. Association for Computing Machinery, New York, NY, USA, Article 95, 6 pages.

Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. 2007. Multi-View Stereo for Community Photo Collections.. In *ICCV*. IEEE Computer Society, 1–8.

Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. 1996. The lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 43–54.

Xiaojie Guo, Xiaochun Cao, and Yi Ma. 2014. Robust separation of reflection from multiple images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2187–2194.

Muhammad Haris, Gregory Shakhnarovich, and Norimichi Ukita. 2019. Recurrent Back-Projection Network for Video Super-Resolution. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 3892–3901.

R. I. Hartley and A. Zisserman. 2004. *Multiple View Geometry in Computer Vision* (second ed.). Cambridge University Press, ISBN: 0521540518.

Peter Hedman, Suhib Alsisan, Richard Szeliski, and Johannes Kopf. 2017. Casual 3D Photography. *ACM Trans. Graph.* 36, 6, Article 234 (Nov. 2017), 15 pages.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–15.

Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. 2016. Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–11.

H. Hirschmuller. 2008. Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 2 (2008), 328–341.

M. Jancosek and T. Pajdla. 2011. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR 2011*. 3121–3128.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2015).

Johannes Kopf, Fabian Langguth, Daniel Scharstein, Richard Szeliski, and Michael Goesele. 2013. Image-based rendering in the gradient domain. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–9.

C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. 2017. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 105–114.

Marc Levoy and Pat Hanrahan. 1996. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 31–42.

Yu Li and Michael S. Brown. 2013. Exploiting Reflection Change for Automatic Reflection Removal. In *2013 IEEE International Conference on Computer Vision (ICCV)*.

Jiaying Lin, Guodong Wang, and Rynson WH Lau. 2020. Progressive Mirror Detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3697–3705.

D. B.* Lindell, J. N. P.* Martel, and G. Wetzstein. 2020. AutoInt: Automatic Integration for Fast Neural Volume Rendering. *arXiv preprint arXiv:2012.01714* (2020).

Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020a. Neural Sparse Voxel Fields. *NeurIPS* (2020).

Yu-Lun Liu, Wei-Sheng Lai, Ming-Hsuan Yang, Yung-Yu Chuang, and Jia-Bin Huang. 2020b. Learning to See Through Obstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14215–14224.

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural volumes: Learning dynamic renderable volumes from images. *arXiv preprint arXiv:1906.07751* (2019).

Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. 2000. Image-Based Visual Hulls *(SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., USA, 6. https://doi.org/10.1145/344779.344951

Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul Beardsley, Remo Ziegler, and Leonard McMillan. 2002. Image-Based 3D Photography Using Opacity Hulls. *ACM Trans. Graph.* 21, 3 (July 2002), 427–437. https://doi.org/10.1145/566654.566599

Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. 2019. Neural rerendering in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6878–6887.

Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. 2019. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–14.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934* (2020).

Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization* (second ed.). Springer, New York, NY, USA.

Nvidia. 2017-2018. Nvidia Corporation. TensorRT. https://developer.nvidia.com/tensorrt.

Eric Penner and Li Zhang. 2017. Soft 3D reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 1–11.

N. C. Rakotonirina and A. Rasoanaivo. 2020. ESRGAN+ : Further Improving Enhanced Super-Resolution Generative Adversarial Network. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 3637–3641.

Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. 2015. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1164–1172.

C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz. 2011. Fast cost-volume filtering for visual correspondence and beyond. In *CVPR 2011*. 3017–3024.

Gernot Riegler and Vladlen Koltun. 2020a. Free View Synthesis. In *European Conference on Computer Vision*.

Gernot Riegler and Vladlen Koltun. 2020b. Stable View Synthesis. arXiv:2011.07233 [cs.CV]

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*.

Mehdi S. M. Sajjadi, Raviteja Vemulapalli, and Matthew Brown. 2018. Frame-Recurrent Video Super-Resolution. In *2018 IEEE Conference on Computer Vision and Pattern Recognition*. 6626–6634.

R. Schnabel, R. Wahl, and R. Klein. 2007. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum* 26, 2 (2007), 214–226.

J. L. Schönberger and J. Frahm. 2016. Structure-from-Motion Revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4104–4113.

Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo.. In *ECCV (Lecture Notes in Computer Science, Vol. 9907)*. Springer, 501–518.

Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. 1998. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. 231–242.

HHeung-Yeung Shum and Sing Bing Kang. 2000. *A Review of Image-based Rendering Techniques*. Technical Report. Microsoft.

Sudipta N Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. 2012. Image-based rendering for scenes with reflections. 31, 4 (2012), 1–10.

S. N. Sinha, D. Steedly, and R. Szeliski. 2009. Piecewise planar stereo for image-based rendering. In *2009 IEEE 12th International Conference on Computer Vision*. 1881–1888.

Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. 2019. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*. 1121–1132.

Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. 2019. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 175–184.

Richard Szeliski. 2006. Image Alignment and Stitching: A Tutorial. MSR-TR-2004-92.

Xin Tao, Hongyun Gao, Renjie Liao, Jue Wang, and Jiaya Jia. 2017. Detail-Revealing Deep Video Super-Resolution. In *ICCV*. IEEE Computer Society, 4482–4490.

Natasha Tatarchuk, Brian Karis, Michal Drobot, Nicolas Schulz, Jerome Charles, and Theodor Mader. 2014. Advances in Real-Time Rendering in Games, Part I (Full Text Not Available). In *ACM SIGGRAPH 2014 Courses*. Article 10, 1 pages.

A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B Goldman, and M. Zollhfer. 2020. State of the Art on Neural Rendering. *Computer Graphics Forum* 39, 2 (2020), 701–727. https://doi.org/10.1111/cgf.14022 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.14022

Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019a. Deferred Neural Rendering: Image Synthesis Using Neural Textures. *ACM Trans. Graph.* 38, 4, Article 66 (July 2019), 12 pages.

Justus Thies, Michael Zollhöfer, and Matthias Nießner. 2019b. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–12.

Zhihao Wang, Jian Chen, and S. C. H. Hoi. 2020. Deep Learning for Image Super-resolution: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), 1–1.

Zhihao Wang, Jian Chen, and Steven C. H. Hoi. 2020. Deep Learning for Image Super-resolution: A Survey. arXiv:1902.06068 [cs.CV]

Thomas Whelan, Michael Goesele, Steven J Lovegrove, Julian Straub, Simon Green, Richard Szeliski, Steven Butterfield, Shobhit Verma, Richard A Newcombe, M Goesele, et al. 2018. Reconstructing scenes with mirror and glass surfaces. *ACM Trans. Graph.* 37, 4 (2018), 102–1.

Daniel N Wood, Daniel I Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H Salesin, and Werner Stuetzle. 2000. Surface light fields for 3D photography. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 287–296.

Lei Xiao, Salah Nouri, Matt Chapman, Alexander Fix, Douglas Lanman, and Anton Kaplanyan. 2020. Neural supersampling for real-time rendering. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 142–1.

Kai Xu, Lintao Zheng, Zihao Yan, Guohang Yan, Eugene Zhang, Matthias Niessner, Oliver Deussen, Daniel Cohen-Or, and Hui Huang. 2017. Autonomous Reconstruction of Unknown Indoor Scenes Guided by Time-Varying Tensor Fields. *ACM Trans. Graph.* 36, 6, Article 202 (Nov. 2017), 15 pages. https://doi.org/10.1145/3130800.3130812

Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. 2019. Deep view synthesis from sparse photometric images. *ACM Transactions on Graphics (TOG)* 38, 4 (2019), 1–13.

Tianfan Xue, Michael Rubinstein, Ce Liu, and William T Freeman. 2015. A computational approach for obstruction-free photography. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11.

Cha Zhang and Tsuhan Chen. 2004. A survey on image-based rendering —— representation, sampling and compression. *Signal Processing: Image Communication* 19, 1 (2004), 1 – 28.

Qian-Yi Zhou and Vladlen Koltun. 2014. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–10.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. 2018. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817* (2018).