Scalable Image-based Indoor Scene Rendering with Reflections

JIAMIN XU, State Key Lab of CAD&CG, Zhejiang University, China
XIUCHAO WU, State Key Lab of CAD&CG, Zhejiang University, China
ZIHAN ZHU, State Key Lab of CAD&CG, Zhejiang University, China
QIXING HUANG, University of Texas at Austin, USA
YIN YANG, Clemson University, USA
HUJUN BAO, State Key Lab of CAD&CG, Zhejiang University, China
WEIWEI XU*, State Key Lab of CAD&CG, Zhejiang University, China

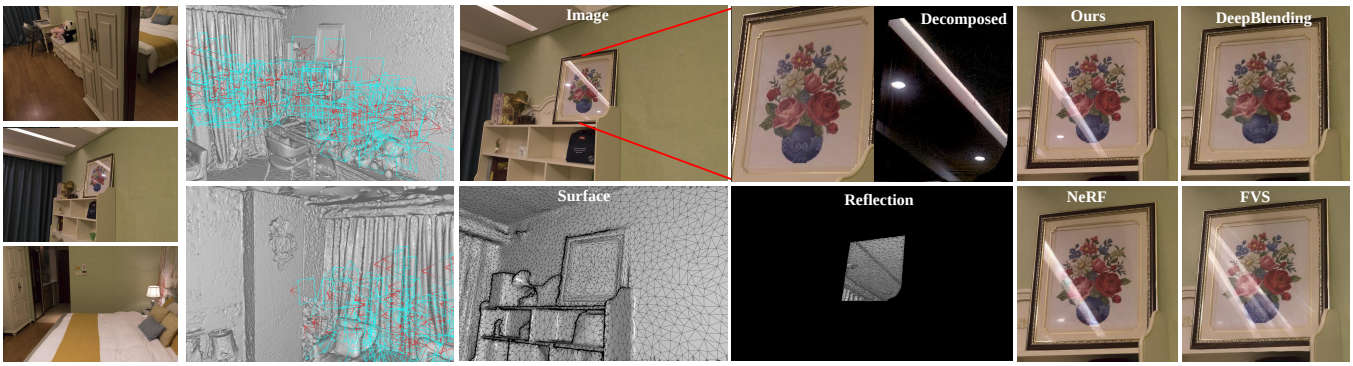


Fig. 1. This figure illustrates the proposed two-layer representation and its rendering result with reflections. Decomposed images: the input image inside the frame is decomposed into surface and reflection layer images. In this example, state-of-the-art view synthesis methods, such as DeepBlending [Hedman et al. 2018], NeRF [Mildenhall et al. 2020], and FVS [Riegler and Koltun 2020], render images with blurred reflections or without reflections at a novel viewpoint. In contrast, our image-based rendering pipeline can achieve a 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. We make substantial progress towards three sub-problems in IBR, namely, depth and reflection reconstruction, view selection for temporally coherent view-warping, and smooth rendering refinements. First, we introduce a global-mesh-guided alternating optimization algorithm that robustly extracts a two-layer geometric representation. The front and back layers encode the RGB-D reconstruction and the reflection reconstruction, respectively. This representation minimizes the image composition error under novel views, enabling accurate renderings of reflections. Second, we introduce a novel approach to select adjacent views and compute blending weights for smooth and temporal coherent renderings. The third contribution is a supersampling network with a motion vector rectification module that refines the rendering results to improve the final output's temporal coherence. These three contributions together lead to a novel system that produces highly realistic rendering results with various reflections. The rendering quality outperforms state-of-the-art IBR or neural rendering algorithms considerably.

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

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

ACM Reference Format:
Jiamin Xu, Xiuchao Wu, Zihan Zhu, Qixing Huang, Yin Yang, Hujun Bao, and Weiwei Xu. 2021. Scalable Image-based Indoor Scene Rendering with Reflections. *ACM Trans. Graph.* 40, 4, Article 60 (August 2021), 14 pages. https://doi.org/10.1145/3450626.3459849

*Corresponding author

Authors' addresses: Jiamin Xu, superxjm@yeah.net, State Key Lab of CAD&CG, Zhejiang University, China; Xiuchao Wu, wuxiuchao@zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, China; Zihan Zhu, zihan.zhu@zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, China; Qixing Huang, huangqx@cs.utexas.edu, University of Texas at Austin, USA; Yin Yang, yin5@clemson.edu, Clemson University, USA; Hujun Bao, bao@cad.zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, China; Weiwei Xu, xww@cad.zju.edu.cn, State Key Lab of CAD&CG, Zhejiang University, China.

0730-0301/2021/8-ART60 $15.00
https://doi.org/10.1145/3450626.3459849

1 INTRODUCTION

Image-based rendering (IBR) algorithms have been applied to synthesize photo-realistic images at novel viewpoints for indoor scenes, crucial to immersive virtual reality applications, such as free-viewpoint navigation of real-estate or museums. IBR is challenging because

realistic images have sophisticated view-dependent features. Examples include occlusions, sharp highlights, and reflections. High-fidelity IBR relies on reconstructing and encoding these features either explicitly or implicitly.

Existing IBR approaches fall into two categories. The first category utilizes layered representations to encode each input image. Examples include layered depth images (LDI) [Shade et al. 1998], multi-plane images (MPI) [Flynn et al. 2019; Zhou et al. 2018], multi-spherical images (MSI) [Broxton et al. 2020], and two-layer representation [Sinha et al. 2012]. These representations encode occlusions and view-dependent features in IBR explicitly. The IBR results depend on the reconstruction quality of these layered representations and effective algorithms to blend them smoothly under continuously changing viewpoints. An alternative approach for IBR is to train neural networks, such as neural radiance fields [Mildenhall et al. 2020] and the deep view synthesis network [Xu et al. 2019], to model the scene structure from sampled images implicitly. These networks have the potential to render reflective surfaces realistically. However, it is unclear whether neural networks can capture all necessary view-dependent visual features. For example, it is still challenging for existing deep neural networks to model sharp edges of reflections, as shown in Fig. 1. Moreover, it remains computationally expensive to train these networks for large-scale indoor scenes.

This paper introduces a novel scalable IBR algorithm, which applies to large indoor scenes with reflections. Our approach combines the strengths of both categories of approaches. Specifically, we use a two-layer representation to capture view-dependent visual features. We also leverage the power of deep neural networks to rectify artifacts after rendering and blending two-layer representations at novel viewpoints. Our approach allows us to render and blend layer-based representations at a lower resolution and then employ a deep neural network to perform supersampling, which outputs continuous and high-resolution images. The advantage of rendering with lower-resolution images is that it can save texture storage and allow us to sample the images of an indoor scene densely, which is beneficial to the quality of blending results. This paper introduces a novel blending scheme with considerably improved spatial and temporal smoothness. Moreover, the supersampling network is adapted from the network in [Wang et al. 2020], and we add a motion vector rectification module to promote temporal smoothness when performing supersampling.

Our approach is also motivated from recent progress on large-scale geometry reconstruction, from RGB images [Furukawa and Ponce 2010; Hartley and Zisserman 2004; Schonberger and Frahm 2016] and RGBD images [Dong et al. 2019; Xu et al. 2017]. State-of-the-art approaches can even reconstruct good approximations of mirrors and glass (c.f. [Whelan et al. 2018]). Our IBR pipeline fuses RGB-based and RGBD-based reconstructions to obtain a global geometric reconstruction of the underlying scene. When re-projected under the camera pose of each input image, this reconstruction provides an effective initialization for computing accurate layered representations. In particular, we also show how to formulate an optimization problem to refine the layered decomposition jointly.

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 various reflections, and the rendering quality is superior to state-of-the-art IBR or neural rendering algorithms.

2 RELATED WORK

IBR applies across 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]. We refer to [Shum and Kang 2000; Zhang and Chen 2003] for comprehensive surveys of IBR and [Tewari et al. 2020] for recent advances. This section reviews the literature closely related to our work.

2.1 IBR with Geometry

Geometry information is mainly used to warp images to novel viewpoints in IBR. The representation of the scene 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 using multi-view stereo (MVS) algorithms [Furukawa and Ponce 2010; Goesele et al. 2007; Hosni et al. 2011]. The reconstructions can guide view warping and view blending for novel view synthesis [Chaurasia et al. 2011; Goesele et al. 2010; Ortiz-Cayon et al. 2015]. Chaurasia et al. [2013] utilized super-pixels as constraints to obtain per-pixel depth. It significantly reduces the image warping artifacts along occlusion edges. For indoor scenes, the Manhattan-world assumption is exploited to reconstruct piece-wise 3D planes from the input images for IBR of indoor scenes [Furukawa et al. 2009; Sinha et al. 2009].

In [Hedman et al. 2016], the reconstructed global geometry is refined at each view for aligning edges of the depth channel and the RGB channels. The resulting per-view meshes can handle large occlusions and motion parallax in IBR. Afterward, Hedman et al. [2018] proposed to combine two different MVS reconstructions for per-view depth refinement and train a deep neural network to blend images warped with per-view meshes to reduce ghosting artifacts. These two approaches can reproduce view-dependent effects to some extent. However, they can not handle reflections because of blending artifacts, e.g., when only using reflective surface geometry to warp images. Our per-view surface layer construction algorithm is inspired by these two works. However, our approach exploits a two-layer mesh representation to render indoor scenes with reflections. Moreover, our supersampling network is trained to fuse blended images temporally to improve the rendering result, which is also different from the network in [Hedman et al. 2018].

2.2 Layered Representation and Reflection Decomposition

Layered representations are widely used to handle occlusions and capture high-frequency reflections in IBR. Pioneering work on the layered representation used in IBR is layered depth images

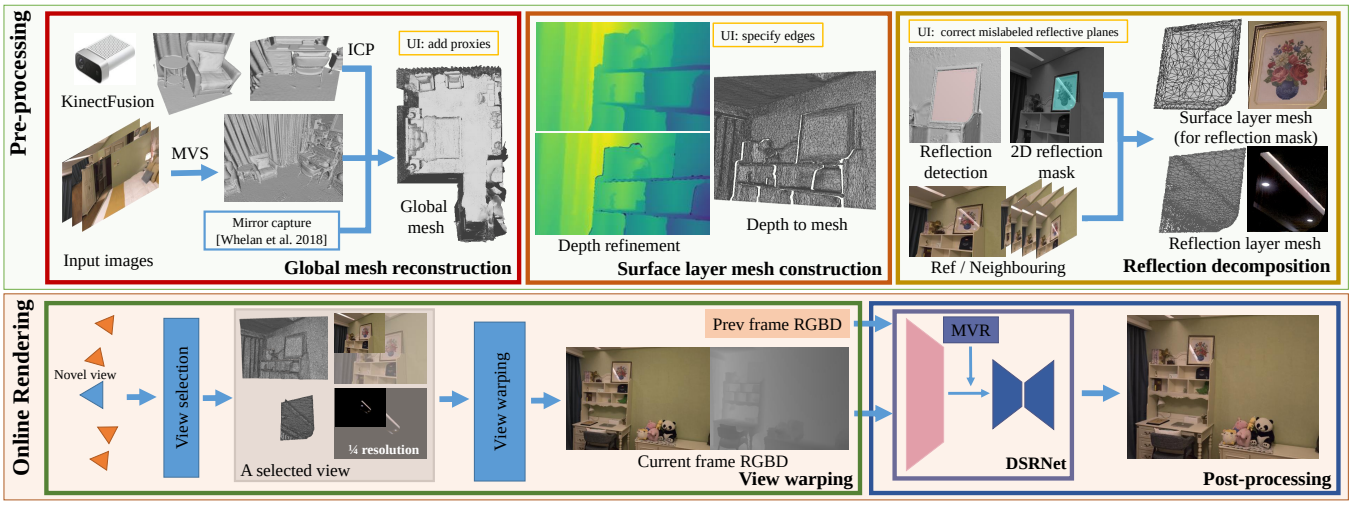


Fig. 2. The pipeline of our IBR approach. UI: User interaction.

(LDI) [Shade et al. 1998]. LDI is a projective volume at a specific viewpoint that stores the scene geometry inside the volume to handle large occlusions. Penner et al. [2017] constructed projective volumes with additional depth uncertainty information for captured images and achieved high-quality view synthesis results at the occlusion edges. In [Hedman et al. 2017], 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, where MSI was first computed at each frame by extending the deep neural network in [Xu et al. 2019] and simplified thereafter to multi-layer meshes. This method can handle view-dependent effects, but the target is to allow as-large-as-possible viewpoint movement in VR videos.

Our two-layer mesh representation is mostly related to the reflection decomposition approach in [Sinha et al. 2012]. Kopf et al. [2013] proposed to render reflections in the gradient domain. Rodriguez et al. [2020] constructed a two-layer representation, i.e., a background layer and a car window layer, to handle the IBR of car windows' reflections using reflective flows. They integrated semantic labels to reconstruct the ellipsoid approximation of the curved car windows. 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]. Recently, deep learning-based reflection decomposition methods are explored widely [Li et al. 2020; Liu et al. 2020b; Yang et al. 2018]. However, without a geometric structure of the scene, it is hard to guarantee the quality of the decomposition results. In contrast, we leverage the reconstructed global mesh to robustly compute the color and geometry of reflection layers to render reflections.

2.3 Deep Learning-based IBR

Given the captured images, deep learning-based IBR methods are capable of learning multi-scale features as a scene representation to facilitate IBR, such as end-to-end deep stereo for unstructured view interpolation [Flynn et al. 2016], deep view synthesis for sparse images captured under controlled conditions [Xu et al. 2019], MPI [Mildenhall et al. 2019; Srinivasan et al. 2019; Xu et al. 2019; Zhou et al. 2018], neural textures [Thies et al. 2019a], and neural volumes [Lombardi et al. 2019]. Coordinate-based multilayer perceptrons (MLP) have been applied to learn an implicit function to represent a 3D scene by minimizing the similarity between rendered and captured images at the same viewpoints [Sitzmann et al. 2019]. Mildenhall et al. [2020] trained a MLP that takes 3D coordinates and a viewing ray as inputs to encode radiance fields effectively, termed neural radiance fields (NeRF). However, the training and testing of the NeRF networks are time-consuming. Hence, Liu et 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 to fuse the image features for novel view synthesis. Riegler et al. [2020] 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 2021]. In [Meshry et al. 2019], a factored representation of a scene, including point cloud, semantic segmentation, and latent appearance codes, is used to render the scene with different appearance.

2.4 Deep Learning for Image and Video Super-resolution

Deep learning-based image super-resolution (SR) methods range from the CNN-based methods to approaches using generative adversarial networks (GANs) [Dong et al. 2014; Ledig et al. 2017; Rakotonirina and Rasoanaivo 2020]. A comprehensive survey on deep

learning-based image super-resolution methods can be found in Wang et al. [Wang et al. 2020]. In video SR, temporal coherence is achieved by integrating motion compensation modules into the SR neural network. Recent approaches include multi-resolution spatial transformer modules in VESPCN [Caballero et al. 2017], sub-pixel motion compensation layers in SPMCVSR [Tao et al. 2017], pyramid, cascading and deformable (PCD) alignment modules in EDVR [Wang et al. 2019], and recurrent networks to accelerate the frame warping in video SR [Fuoli et al. 2019; Haris et al. 2019; Sajjadi et al. 2018].

In the 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. The key contribution of this work is a motion vector rectification module that can refine the geometry-based correspondences between consecutive frames to improve the SR results. This module effectively reduces warping errors induced by imprecise layered reconstructions.

3 OVERVIEW

Our IBR approach consists of a pre-processing stage and an online rendering stage to render the planar reflections of indoor scenes realistically (See Fig. 2). The goal of the pre-processing stage is to reconstruct the two-layer mesh representation for each view, guided by the reconstructed global mesh **G**. In this paper, we perform geometry reconstruction by adding the strengths of MVS reconstruction and RGBD-based reconstruction to obtain a high-quality global mesh, as shown in the dark red box in Fig. 2.

The two-layer representation for an input image **I** consists of a surface layer mesh that encodes the non-reflective regions of an image and another layer that encodes the reflective regions. Reconstruction of the surface layer starts with rendering the global mesh **G** at the given viewpoint of **I** to obtain a depth image **D**, and then refine **D** to align depth and color images (Fig. 2: dark orange box). This alignment reduces tearing-apart and ghosting artifacts in IBR. In contrast to using bilateral median filters [Hedman et al. 2018, 2016], we integrate surface normal information in both depth edge detection and refinement to assist the edge alignment. The surface layer mesh is finally constructed according to the refined depth image **D** (Sec. 4.2). In the reflection decomposition step (Fig. 2: dark yellow box), we detect reflective planes for each input image using a multi-view consistency cue. For images that contain detected planar reflective planes, the meshes and textures for the surface layer and the reflection layer of these planes are obtained by solving an alternating optimization problem (Sec. 4.3). To reduce the memory cost, the textures for two-layer meshes are stored at $\frac{1}{4}$ of the resolution of our rendering result.

The second stage of our IBR pipeline, online rendering, aims to generate view synthesis results according to the pre-computed two-layer mesh representation for each image. This stage has two steps. The first step performs view warping (Fig. 2: dark green box), in which we introduce a novel view selection and blending approach that ensures the smoothness of the view-synthesis when changing the camera pose. The details are explained in Sec. 5.1. The second step applies a deep neural network (named DSRNet) to refine the view warping results, as shown in the dark blue box in Fig. 2. This network increases the resolution of view-synthesis and performs visual rectifications as a post-processing step, such as anti-aliasing and reducing the ghosting effects caused by imprecise two-layer reconstructions. The details are explained in Sec. 5.2.

Note that our system also allows users to create proxies for hard-to-reconstruct light sources and correct reflective plane detection errors. To facilitate the alignment of depth and color edges, we also allow users to draw lines to indicate occlusion edges when they occur in the regions of near-constant color. The details of reflective plane detection and user interactions can be found in supp. material.

4 PER-VIEW TWO-LAYER MESH CONSTRUCTION

This section presents the preprocessing stage, which reconstructs a two-layer representation for each input image. This stage assumes a global mesh reconstruction **G** of the underlying scene that is not necessary to be precise.

4.1 Global Mesh Reconstruction

To reconstruct the global mesh **G**, we utilize both color images captured by a `Canon EOS 60D` digital single-lens reflex camera and RGBD images captured by a `Microsoft Kinect4`. The camera is hand-held in most cases, and it is mounted on a tripod when capturing reflective surfaces such that the photographer is not in the reflection. Our global mesh is first constructed using MVS software `RealityCapture` [CapturingReality 2016], as in [Hedman et al. 2018]. We convert the captured raw images (at the resolution of $6,000$ by $4,000$) into 16-bit `tiffs` to calculate camera poses more accurately and uniformly sample 500~800 images from the captured images according to their timestamp to accelerate the 3D reconstruction process. Second, during the indoor scene scanning, for some textureless objects or planar areas, we also scan them using the Kinect4 camera. We capture a few RGBD sequences corresponding to different objects or parts of the scene, and each sequence is fused into a mesh using the KinectFusion algorithm [Newcombe et al. 2011]. Then we register the fused meshes into the global mesh reconstructed by RealityCapture using the iterative closest point (ICP) method, which forms the final global mesh. The initial pose of each RGBD sequence is obtained by computing the camera pose of its first RGB image in MVS.

The geometry of mirrors and their masks in an image are obtained by the method developed by Whelan et al. [2018], but we simplify their hardware by removing the SLAM cameras. The captured color images with AprilTags are also fed into RealityCapture software to compute their camera poses.

4.2 Surface Layer Mesh Construction

We first project the global mesh **G** using the camera pose of the input image **I** to obtain an initial depth image **D**. As **G** is not precise, there are misalignments between depth and color edges. They are corrected in two steps, i.e., depth edge detection and refinement.

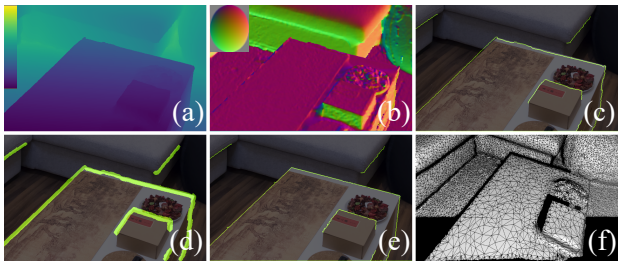


Fig. 3. Depth refinement to align depth and color edges. (a) Initial depth map. (b) Visualization of the normal map. (c) Misalignment between depth and color edges. (d) Regions between depth and color edges. (e) Refined depth edges. (f) Constructed surface mesh. Lines in yellowgreen indicate the depth edges. Please zoom-in to view the details.

4.2.1 Depth Edge Detection. For each pixel i in the depth image, we first calculate its vertex position $\mathbf{v}_i$ and normal $\mathbf{n}_i$ according to its depth d_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

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

If dt_{ij} exceeds a threshold λ, then we label ij as a depth edge (See Fig. 3(c)). We set $\lambda = 0.01\cdot\max(1, \min(d_i, d_j))$, accounting for large discretization errors for pixels with large depth. After obtaining depth edges, we generate a depth refinement mask in the neighborhood of the depth edges and refine the depth of the input image within this mask. As shown in Fig. 3(d), the depth mask is given by the fronto-parallel square patch of side length $4cm$ at each depth edge pixel in the current view.

4.2.2 Depth Refinement. We proceed in two steps to refine the pixel-wise depth. Similar to [Hedman et al. 2018], for each initial depth image, we apply COLMAP [Schönberger et al. 2016] to recover the detailed, pixel-wise depth. To enhance stability, we only do COLMAP-based refinement inside the refinement mask. For each pixel in the mask, COLMAP runs the pixel-wise view selection based multi-view stereo algorithm to obtain the depth. Specially, it first runs a photometric stage to optimize photo consistency, followed by a geometric stage that takes multi-view geometric consistency into account. Then, we remove misaligned depth edge pixels based on comparing the photometric and geometric stage results. If the photometric stage depth and the geometric stage depth of one pixel differ by more than 5% of the geometric stage depth, we discard the depth of this pixel.

The second step adapts edge-aware 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 [Dollár and Zitnick 2015; He et al. 2019] as follows (Fig. 3(e)):

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

where the pixels in $\mathcal{A}_i$ are the 4-closest neighbors outside the refinement mask of i computed using the geodesic distance $d_g(p_i, p_j)$, and $w_g(i,j) = \exp(-d_g(i,j))$; $\hat{d}_i^j$ is the depth induced from intersecting the line of sight through pixel i and the plane defined according to the 3D position and normal at pixel j. As the interpolation considers color edges, the resulting images usually possess aligned depth and color edges. In a few cases, if no color edge is close to an occlusion edge in almost constant color regions, we also allow users to specify edges on color images.

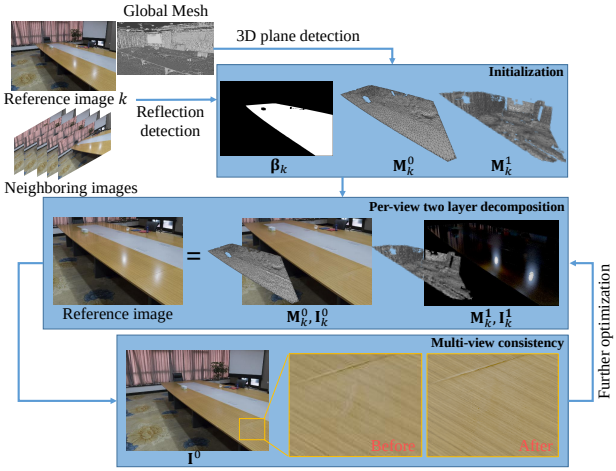


Fig. 4. The pipeline of reflection decomposition algorithm.

4.2.3 Converting Refined Depth to a Surface Mesh. We initialize a triangle mesh by converting each pixel square into two triangles. We then apply [Garland and Heckbert 1997] to simplify the mesh to the final per-view surface mesh. Moreover, we preserve mesh edges along depth occlusion edges by increasing their quadric error by a factor of 400~500 in mesh simplification to facilitate edge anti-aliasing in the rendering. After simplification, the vertex number of per-view mesh is less than 40,000 on average.

Our per-view depth refinement strategy is similar to [Hedman et al. 2018]. However, we found that performing 2D edge-aware interpolation for misaligned pixels is empirically more stable than the bilateral median filter in our experiments.

4.3 Reflection Decomposition

The reflection decomposition of a reference image $\mathbf{I}_k$ recovers four quantities: the surface layer image $\mathbf{I}_k^0$ and mesh $\mathbf{M}_k^0$ and the reflection layer image $\mathbf{I}_k^1$ and mesh $\mathbf{M}_k^1$. We enforce the linear image composition rule in [Sinha et al. 2012] as a constraint in the reflection decomposition:

$$\mathbf{I}_k(\mathbf{u}) = \mathbf{I}_k^0(\mathbf{u}) + \beta_k(\mathbf{u})\mathbf{I}_k^1(\mathbf{u}) \quad (3)$$

where $\mathbf{u}$ indicates a pixel of $\mathbf{I}_k$. Note that $\beta_k(\mathbf{u})$ is the pre-computed mask that indicates if a pixel $\mathbf{u}$ belongs to the reflection regions or not. To simplify the notations, we concatenate $\mathbf{I}_k^0$ and $\mathbf{I}_k^1$ as $\mathbf{I}_k^{0,1}$. Likewise, $\mathbf{M}_k^{0,1}$ is the concatenation of $\mathbf{M}_k^0$ and $\mathbf{M}_k^1$.

Since the number of variables is much larger than the number of known pixel RGB values of $\mathbf{I}_k$, we introduce additional constraints to regularize the decomposition. The constraints fall into two categories. The first category consists of regularization constraints

for $\mathbf{M}_k^{0,1}$, such as smoothness regularization and the regularization induced from the global mesh $\mathbf{G}$. The second category utilizes consistency constraints between $\mathbf{I}_k$ and a set of neighboring images $\mathcal{N}_k$ (defined below). These consistency constraints are imposed at three levels, i.e., the two layers and the composite image. Note that our approach also utilizes the neighboring images to rectify reflected highlights with saturated intensities that break the composition rule in Eq. 3.

Our reflection decomposition approach alternates between optimizing the decomposition of each input image and aggregating constraints from neighboring images to enforce multi-view consistency. In the following, we first describe the initialization step. We then introduce the decomposition step and the aggregation step.

4.3.1 Initialization. We initialize the per-image reflection mask β_k by projecting the detected global reflective plane onto the image $\mathbf{I}_k$. The surface and reflection layer meshes $\mathbf{M}_k^{0,1}$ are initialized using the refined depth and the planar-reflection geometry inside the reflection mask, respectively (please see our supp. material).

Next, we define the set of neighboring images $\mathcal{N}_k$ by rendering the reflection layer mesh $\mathbf{M}_k^1$ in the other images and then checking the depth overlap (depth difference < $0.05 * \min(\text{depth})$) in β_k. The images with more than 30% depth overlap are sorted according to the camera pose distance. We keep the top six images to form $\mathcal{N}_k$. Our approach also utilizes an image flow $\omega_{k'}(u, \mathbf{M}_k^0)$ between image $\mathbf{I}_k$ and each neighboring image $\mathbf{I}_{k'}$. It is computed by rendering $\mathbf{M}_k^0$ with the camera pose associated to $\mathbf{I}_{k'}$.

The quality of the image flow $\omega_{k'}(u, \mathbf{M}_k^0)$ depends on the quality of $\mathbf{M}_k^0$. During the initialization step, we refine $\omega_{k'}$ using a non-rigid 2D warping function $\mathbf{F}_{k'}$. $\mathbf{F}_{k'}$ is defined on a 40×30 2D grid that uniformly covers the image, resulting in 41×31 control vertices. It is represented by the offset vectors $\mathbf{f}_{k'}^l$ associated to the control vertices $\mathbf{v}_{k'}^l$ of the grid, where l is the index through the set of control vertices. The warping function $\mathbf{F}_{k'}$ for a pixel $\mathbf{u}$ is as follows:

$$\mathbf{F}_{k'}(\mathbf{u}) = \mathbf{u} + \textstyle\sum_l \theta_{k'}^l(\mathbf{u})\, \mathbf{f}_{k'}^l \tag{4}$$

where $\theta_{k'}^l$ are the weight functions of bilinear interpolation using the control vertices of the grid cell that contains u. With this setup, we use the image flow $\mathbf{F}_{k'}\left(\omega_{k'}\left(\mathbf{u}, \mathbf{M}_k^0\right)\right)$ to aggregate information across neighboring views during the initialization phase. Note that we use $\omega_{k'}\left(\mathbf{u}, \mathbf{M}_k^0\right)$ at subsequent iterations.

The offset vectors $\mathbf{f}_{k'}^l$ are obtained by minimizing

$$E = \textstyle\sum_{\mathbf{u}} \|\mathbf{I}_{k'}\left(\mathbf{F}_{k'}\left(\omega_{k'}\left(\mathbf{u}, \mathbf{M}_k^0\right)\right)\right) - \mathbf{I}_k(\mathbf{u})\|^2 + w_r \textstyle\sum_l \|\mathbf{f}_{k'}^l\|^2 \tag{5}$$
$$+ w_a \textstyle\sum_t \left\| \mathbf{v}_1^t - \left(\mathbf{v}_2^t + \alpha \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \left(\mathbf{v}_3^t - \mathbf{v}_2^t\right)\right)\right\|^2$$

where $w_r = 1.2$ and $w_a = 0.5$ in our experiments. The first term ensures the color consistency between $\mathbf{I}_{k'}$ and $\mathbf{I}_k$ with respect to the image flow $\mathbf{F}_{k'}\left(\omega_{k'}\left(\mathbf{u}, \mathbf{M}_k^0\right)\right)$. The second term regularizes the deformation. The third term is the as-rigid-as-possible (ARAP) regularization introduced in [Igarashi et al. 2005], where α is the grid aspect ratio. To compute this term, we split each grid cell into two triangles, where $\mathbf{v}_1^t$, $\mathbf{v}_2^t$ and $\mathbf{v}_3^t$ represent three vertices of each triangle t. Their deformed positions can be obtained by substituting the corresponding $\mathbf{v}_{k'}^l$s into Eq. 4. We perform 5 Gauss-Newton iterations to obtain the 2D warping field. Fig. 5 illustrates the influence of 2D warping to the initialization of $\mathbf{I}_k^0$.

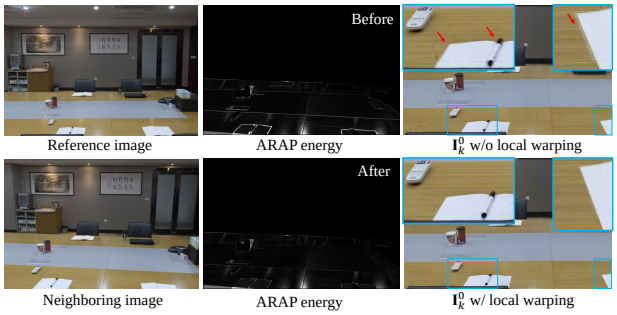


Fig. 5. An example of a 2D warping result. Left: input images. Middle: The pixel difference is reduced after warping. Right: The decomposed foreground surface image after the first iteration of the alternating optimization algorithm.

Given $\mathcal{N}_k$ and $\omega_{k'}(u, \mathbf{M}_k^0)$, we initialize

$$\mathbf{I}_k^0(\mathbf{u}) = \min\left(\left\{\mathbf{I}_{k'}\left(\mathbf{F}_{k'}\left(\omega_{k'}\left(\mathbf{u}, \mathbf{M}_k^0\right)\right)\right) | k' \in \mathcal{N}_k \bigcup k\right\}\right). \tag{6}$$

We then set $\mathbf{I}_k^1 = \mathbf{I}_k - \mathbf{I}_k^0$. Note that the min operator is used to provide a meaningful value for $\mathbf{I}_k^0(\mathbf{u})$ while prioritizing that $\mathbf{I}_k^1(\mathbf{u})$ is non-negative.

4.3.2 Two-layer Decomposition. We solve the following optimization problem to optimize $\mathbf{M}_k^{0,1}$ and $\mathbf{I}_k^{0,1}$:

$$\underset{(\mathbf{R},\mathbf{T})_k^1, \mathbf{M}_k^{0,1}, \mathbf{I}_k^{0,1}}{\operatorname{argmin}} \quad E_d + \lambda_s E_s + \lambda_p E_p + \lambda_{mv} E_{mv}, \quad \text{s.t. } \mathbf{I}_k^{0,1}(\mathbf{u}) \in [0..1], \tag{7}$$

where $(\mathbf{R}, \mathbf{T})_k^1$ are the rotation and translation transformations for $\mathbf{M}_k^1$ respectively. We set $\lambda_s = 0.04$, $\lambda_p = 0.01$. The weight λ_{mv} is set to 0 initially and then 0.05 after the aggregation step according to multi-view consistency. Below we introduce the formulation of each term.

Data Term. The data term E_d measures the difference between the image $\tilde{\mathbf{I}}_{k'}$ composed by rendering two-layer images at a neighboring viewpoint k' according to $\mathbf{M}_k^{0,1}$ and the captured image $\mathbf{I}_{k'}$:

$$E_d = \sum_{k' \in \mathcal{N}_k \bigcup k} \sum_{\mathbf{u}} \|\tilde{\mathbf{I}}_{k'}(\mathbf{u}) - \mathbf{I}_{k'}(\mathbf{u})\|^2 \tag{8}$$

$$\tilde{\mathbf{I}}_{k'}(\mathbf{u}) = \mathbf{I}_k^0\left(\omega_{k'}^{-1}\left(\mathbf{u}, \mathbf{M}_k^0\right)\right) + \beta_k\left(\omega_{k'}^{-1}\left(\mathbf{u}, \mathbf{M}_k^0\right)\right) \mathbf{I}_k^1\left(\omega_{k'}^{-1}\left(\mathbf{u}, \mathbf{M}_k^1\right)\right),$$

where $\omega_{k'}^{-1}$ represents the inverse warping function $\omega_{k'}$.

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

$$E_s = \sum_{\mathbf{u}} \left(e^{-\nabla \mathbf{I}_k^{0,1}(\mathbf{u})} \|\nabla \mathbf{I}_k^{0,1}(\mathbf{u})\|^2\right) + \sum_{\mathbf{v}} \|\mathbf{H}\mathbf{M}_k^{0,1}(\mathbf{v})\|^2 \tag{9}$$

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. This term regularizes $\mathbf{I}_k^{0,1}$ to make the optimization stable:

$$E_p = \sum_{\mathbf{u}} \left(\|\mathbf{I}_k^0(\mathbf{u})\|^2 \right) + \sum_{\mathbf{u}} \left(\|\mathbf{I}_k^1(\mathbf{u})\|^2 \right) \tag{10}$$

Optimization. To handle the large number of optimization variables in $\mathbf{I}_k^{0,1}$ and $\mathbf{M}_k^{0,1}$, we develop an alternating optimization algorithm to minimize the objective function Eq. 7. Specifically, 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 the other set of variables.

4.3.3 Multi-view Consistency. After performing two-layer decomposition for each input image, we aggregate the results of neighboring images. This step can be considered a filtering step, which can enhance the decomposition of each image. To this end, we introduce two aggregated images $\tilde{\mathbf{I}}_k^0$ and $\tilde{\mathbf{I}}_k^1$ from neighboring images (defined below). When performing the next iteration of per-view two-layer decomposition, we introduce an additional regularization term E_{mv} in Eq. 7 as follows:

$$E_{mv} = \sum_{\mathbf{u}} \|\mathbf{I}_k^0(\mathbf{u}) - \tilde{\mathbf{I}}_k^0(\mathbf{u})\|^2 + \sum_{\mathbf{u}} \|\mathbf{I}_k^1(\mathbf{u}) - \tilde{\mathbf{I}}_k^1(\mathbf{u})\|^2, \tag{11}$$

Intuitively, this term ensures that $\mathbf{I}_k^0(\mathbf{u})$ and $\mathbf{I}_k^1(\mathbf{u})$ are compatible with the image flows between neighboring images and is easy to optimize. Usually, our reflection decomposition algorithm converges to a good-quality solution within two iterations.

We define $\tilde{\mathbf{I}}_k^0$ by robustly aggregating the corresponding layers among neighboring images:

$$\tilde{\mathbf{I}}_k^0(\mathbf{u}) = \text{median}\left(\left\{ \mathbf{I}_{k'}^0\left(\omega_{k'}\left(\mathbf{u}, \mathbf{M}_k^0\right)\right) | k' \in \mathcal{N}_k \bigcup k \right\} \right) \tag{12}$$

where $\mathbf{M}_k^0$ and $\mathbf{I}_{k'}^0$ are optimized in previous per-view reflection decomposition iterations. $\tilde{\mathbf{I}}_k^1$ is given by enforcing the linear composition rule in Eq. 3:

$$\tilde{\mathbf{I}}_k^1(\mathbf{u}) = \begin{cases} \mathbf{I}_k(\mathbf{u}) - \tilde{\mathbf{I}}_k^0(\mathbf{u}) & \text{if } \mathbf{I}_k(\mathbf{u}) - \tilde{\mathbf{I}}_k^0(\mathbf{u}) - \mathbf{I}_k^1(\mathbf{u}) < 0 \\ \mathbf{I}_k^1(\mathbf{u}) & \text{otherwise} \end{cases} \tag{13}$$

There are two special cases of reflections: 1) Highlights. Pixels inside highlights have saturated intensities, which can not be modeled by Eq. 3. Thus, we propose to detect the pixels with highlights to avoid the computation of E_d for these pixels. 2) Mirrors. Considering that mirrors are perfectly reflective surfaces without texture, we choose to set $\mathbf{I}_k^1 = \mathbf{I}_k$ and $\mathbf{I}_k^0 = \mathbf{0}$ for the pixels inside a mirror. The mirror plane is determined using a simplified hardware in [Whelan et al. 2018] with AprilTags. The details on how we handle these two special cases are presented in the supp. material.

5 ONLINE RENDERING

The online rendering stage generates novel-views from a moving camera. Besides ensuring the quality of an individual image, a key goal is to ensure that the rendered images are smooth. The design of our online rendering module is tailored towards both goals. Specifically, it consists of a view warping step (Sec. 5.1), which initializes the rendering, and a supersampling step (Sec. 5.2), which employs a neural network to refine the rendering result as a post-process.

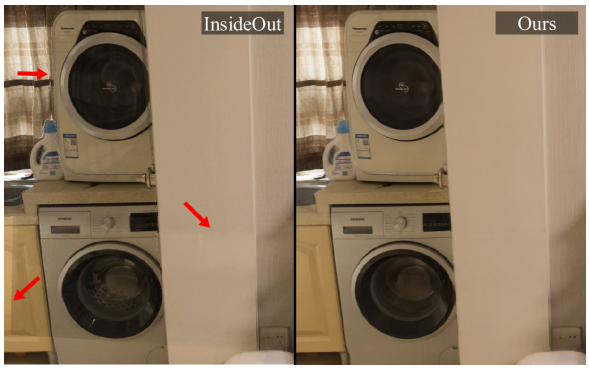


Fig. 6. Compared with InsideOut[Hedman et al. 2016], our view warping can avoid discontinuity artifacts and produce smoother image blending results.

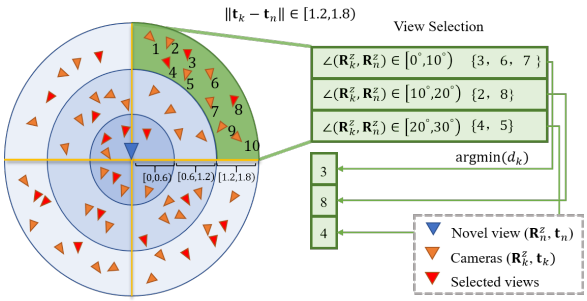


Fig. 7. 2D illustration of view selection. The view selection is done independently for each quadrant.

5.1 View Warping using Two-layer Mesh Representation

Given a new view $\mathbf{V}_n$, our view warping algorithm first selects a suitable set of relevant input images. It then renders the pre-computed two-layer meshes of each input image under this new view. Each synthesized image is given by Eq. 3. The output of this step is obtained by blending the synthesized images (c.f. [Hedman et al. 2016]). The quality of the output heavily depends on the blending strategy and relevant-view's selection strategy. We introduce a new approach that reduces discontinuities in view selection and blending weight distribution among neighboring pixels. Fig. 6 shows an example result of our approach, which improves the smoothness of the renderings. Similar to [Hedman et al. 2016], our view warping algorithm consists of three steps: view selection, fuzzy depth test and camera-pose-based view blending.

5.1.1 View Distance. For each camera pose $\mathbf{V}_k$, we denote $\mathbf{t}_k$ and $\mathbf{R}_k^z$ as its optical center and optical axis, respectively. In other words, $\mathbf{R}_k^z$ corresponds to the z axis of the camera rotation matrix. We first define a distance d_k between an input view $\mathbf{V}_k$ and the novel view

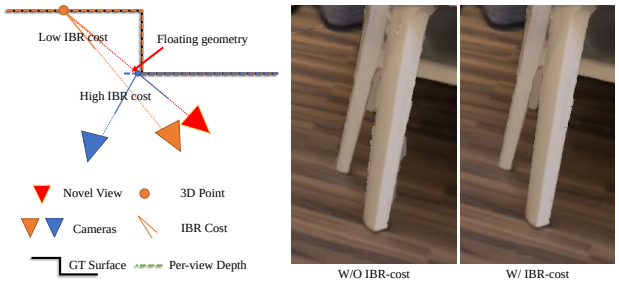


Fig. 8. The floating geometry at one pixel will be removed if its IBR-cost is high with respect to the novel view.

$\mathbf{V}_n$ to facilitate the view blending.

$$d_k = \angle\left(\mathbf{R}_k^z, \mathbf{R}_n^z\right) * \pi/180 + \lambda\|\mathbf{t}_k - \mathbf{t}_n\|/\|\mathbf{t}_n\| \tag{14}$$

where $\angle\left(\mathbf{R}_k^z, \mathbf{R}_n^z\right)$ is the angle between $\mathbf{R}_k^z$ and $\mathbf{R}_n^z$; $\|\mathbf{t}_k - \mathbf{t}_n\|$ is the distance between $\mathbf{t}_k$ and $\mathbf{t}_n$.

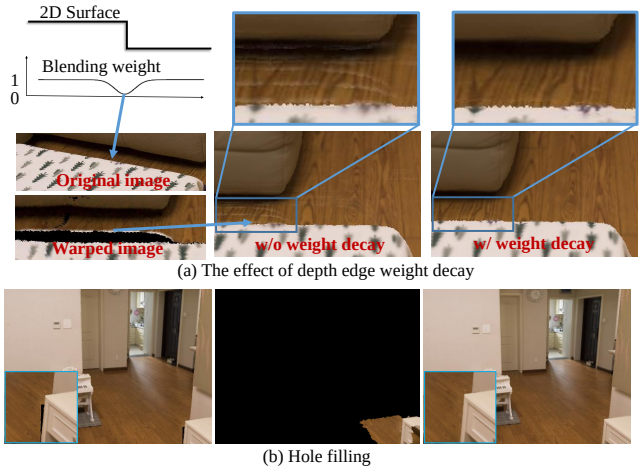
This distance focuses on the consistency of the look-at direction of each camera by only using $\mathbf{R}_k^z$ as a simplified representation of camera orientation. Empirically, we found it works well since there seldom exist the camera rotations around its optical axis. We set $\lambda = 0.1$ in our experiments.

5.1.2 View Selection. To select views surrounding $\mathbf{V}_n$ with a uniform distribution, we choose to divide the 3D space around a novel view into 72 sub-regions and select one captured view with minimum view distance d_k in each sub-region. Specifically, we first split the 3D space into eight quadrants by local coordinate xyz-axes. We then continue to divide the captured views into nine sub-regions for each quadrant based on the pairwise combination of three angle intervals and three optical center distance intervals (see Fig. 7 for the default setting).

We found that it is also important to maintain the overlap between selected views for $\mathbf{V}_n$ and those of the previous view $\mathbf{V}_{n-1}$ to enhance temporal coherence. We achieve this by delaying the removal of selected views at $\mathbf{V}_{n-1}$. Precisely, we first calculate $\angle(\mathbf{R}_n^z, \mathbf{R}_{n-1}^z)$ and $\|\mathbf{t}_n - \mathbf{t}_{n-1}\|$ between $\mathbf{V}_n$ and $\mathbf{V}_{n-1}$. These two values are then added to the maximum angle condition and maximum distance condition which are 30° and 1.8m respectively as shown in Fig. 7. Finally, the views selected at $\mathbf{V}_{n-1}$ that satisfies the updated conditions are also selected for $\mathbf{V}_n$.

5.1.3 Fuzzy Depth Test. Given the view selection result, we first generate a front-most depth map (FDMP) by rendering the meshes of selected views into the novel view. The FDMP will be used for the fuzzy depth test, to determine if a pixel contributes to the blending result. Specifically, 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.

However, floating geometries, i.e., the geometry errors, of per-view meshes will downgrade the FDMP's quality, making the depth test inaccurate. Thus, we utilize a per-pixel IBR-cost to reduce the influence of floating geometries [Hedman et al. 2018]. Specifically, we first determine the smallest IBR-cost from the 3D points projected



(a) The effect of depth edge weight decay

(b) Hole filling

Fig. 9. (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 remaining view warping result with weight decay at the hole boundary.

to the same pixel of $\mathbf{V}_n$ and then only keep those points whose difference of its IBR-cost to the smallest IBR-cost is less than a threshold (default 0.17). The minimal depth of these remaining 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]:

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

where $\mathbf{x}$ is a 3D point. As shown in Fig. 8, this new approach can reduce the influence of floating geometries, especially when some depth edges are missed in depth-color edge alignment, e.g., there are no color edges close to the depth edges.

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

$$w_k = \exp(-d_k/\delta) \tag{15}$$

where δ is set to 0.033 in our experiments. This weighting scheme favors those views close to $\mathbf{V}_n$ in view warping. Note that the blending weight is the same for surface and reflection layer images.

To avoid discontinuity artifacts, we first apply image feathering, which is 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 a 20-pixel distance to the image boundaries, which is efficient in removing discontinuities 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). The operation rectifies pixels that may contain noise, which can be warped to semantically different objects in the scene (See Fig. 9). Weight decay can reduce the weight of such pixels. Besides, warped

pixels far from the depth edge in another view will contribute substantially to the final RGB value. All the decayed weights are stored in the alpha channel of mesh textures. There may be small-area holes left after the camera-poses-based view blending, as shown in Fig. 9(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.

5.2 DSRNet

The second step of the online rendering employs DSRNet to rectify the output of the view-warping and perform supersampling to output the final result of target resolution. This step is also able to reduce artifacts when stitching view-warping results.

5.2.1 Network Architecture. The key challenge of supersampling in IBR is to improve the continuity of output. Similar to RSSNet [Xiao et al. 2020], DSRNet combines the view warping result of the previous frame $(\mathbf{I}_{n-1}, \mathbf{D}_{n-1})$ and that of the current frame $(\mathbf{I}_n, \mathbf{D}_n)$ to produce the final output $\mathbf{I}_n^{\mathrm{f}}$. As illustrated in Fig. 10, the network architecture of DSRNet combines two feature extraction towers that operate on the current frame (dark pink box) and the previous frame (dark green box) respectively. Thereafter, DSRNet fuses the extracted features together and pass it through a U-Net (dark blue box) to generate the final output $\mathbf{I}_n^{\mathrm{f}}$. Conceptually, this approach implicitly performs temporal smoothing (i.e., between consecutive frames). Note that the architecture of the U-Net and feature extraction towers are standard, which follow those of RSSNet.

The performance of DSRNet is driven by the quality of the feature module. Similar to RSSNet [Xiao et al. 2020], DSRNet first computes a motion field $\mathbf{M}_r$ that aligns pixels of the previous frame and pixels of the current frame. The key innovation of DSRNet is to define the motion vector rectification module (dark purple box) by combining multiple inputs:

$$\mathbf{M}_r = \mathbf{M}_d + \mathrm{MVR}(\mathbf{I}_n, \mathbf{I}_{n-1}, \mathbf{M}_d \odot \mathbf{I}_{n-1}, \mathbf{M}_d) \tag{16}$$

where $\mathbf{M}_d$ is the base motion vector predicted using the depth images $\mathbf{D}_{n-1}$ and $\mathbf{D}_n$. The refinement module MVR takes the view-warping results $\mathbf{I}_{n-1}$ and $\mathbf{I}_n$, $\mathbf{M}_d$, and the deformed image $\mathbf{M}_d \odot \mathbf{I}_{n-1}$ given by deforming $\mathbf{I}_{n-1}$ using $\mathbf{M}_d$, where $\odot$ denotes the deformation operation. Intuitively, this module applies a one-step flow refinement between $\mathbf{I}_n$ and $\mathbf{M}_d \odot \mathbf{I}_{n-1}$. Given that MVR is designed for motion vector fine-tuning, the output of MVR is limited to $[-5, 5]$ pixels in our experiments. DSRNet combines the features of $\mathbf{I}_n$ and the weighted features of $\mathbf{I}_{n-1}$ after applying $\mathbf{M}_r$. Similar to RSSNet, the weights are obtained by applying three trainable convolution operators on $\mathbf{I}_n$ and $\mathbf{M}_r \odot \mathbf{I}_{n-1}$.

5.2.2 Training Losses. Similar to [Xiao et al. 2020], we combine a structural similarity index (SSIM) loss and a perceptual loss to train DSRNet:

$$\mathrm{L}\left(\mathbf{I}^{\mathrm{f}}, \mathbf{I}^{\mathrm{gt}}\right) = 1 - \mathrm{SSIM}\left(\mathbf{I}^{\mathrm{f}}, \mathbf{I}^{\mathrm{gt}}\right) + w \sum_{i=1}^{5} \left\| \mathrm{conv}_i\left(\mathbf{I}^{\mathrm{f}}\right) - \mathrm{conv}_i\left(\mathbf{I}^{\mathrm{gt}}\right) \right\|_2^2$$

where $\mathbf{I}^{\mathrm{f}}$ and $\mathbf{I}^{\mathrm{gt}}$ are the network output and the captured ground-truth image respectively; conv_i is the pre-computed filter, which is a component of the perceptual loss. Weight w (default 0.1) is used to balance the two losses.

Table 1. Statistics of reconstructed indoor scenes. #Img denotes the number of total images captured in the scene. Img/Mesh Storage denotes the GPU memory storage for down-sampled texture images and two-layer meshes. Numbers in brackets indicate the memory storage for surface and reflection layer meshes. #RGBD denotes the total time in seconds of all RGBD sequences scanned in 30fps.

Scene	Area(m^2)	#Img	Img/Mesh Storage(**GB**)	#RGBD
Hotel Room	7.0 ∗ 4.4	1741	0.55 / 1.89 (1.61+0.28)	540s
Living Room 1	8.2 ∗ 6.3	2289	0.72 / 2.48 (2.17+0.31)	495s
Living Room 2	12.3 ∗ 8.1	2782	0.88 / 3.04 (2.40+0.64)	945s
Meeting Room 1	11.2 ∗ 6.5	1631	0.52 / 1.84 (1.61+0.23)	720s
Meeting Room 2	13.3 ∗ 10.4	998	0.32 / 1.22 (0.88+0.34)	585s

We use a warping loss to train the MVR module:

$$\mathrm{L}_{warp} = \mathrm{L1}\left(\mathrm{G}\left(\mathbf{I}_n\right), \mathrm{G}\left(\mathbf{M}_r \odot \mathbf{I}_{n-1}\right)\right) + \mathrm{L1}\left(\mathbf{I}_n, \mathbf{M}_r \odot \mathbf{I}_{n-1}\right)$$

where $\mathrm{L1}(\cdot)$ denotes the L1 Loss and $G(\cdot)$ is the Gaussian filter with 5×5 kernels. The Gaussian filter is used to smooth the local gradients and avoid gradient vanishing for pixels not on the color edges.

6 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 implementation details of the per-view reflection decomposition, view warping and the training of the DSRNet can be found in our supp. material. 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 an OpenGL/CUDA interop interface to directly interchange rendering buffer and network tensor data at GPU memory during online rendering. Our pipeline's average running time to render an image with 1280×960 resolution 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 Tab. 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. During rendering, the GPU memory required to store the two-layer meshes and textures of each scene is listed in the 4th column in Tab. 1. Besides, the DSRNet consumes another 1.7GB GPU memory to render the scene at 1280×960 resolution. Example pre-processing time and the DSRNet training time for the Living Room 1 scene are shown in Tab. 2. In this section, we will report the evaluation results of reflection decomposition, DSRNet, and the rendering result comparisons with state-of-the-art

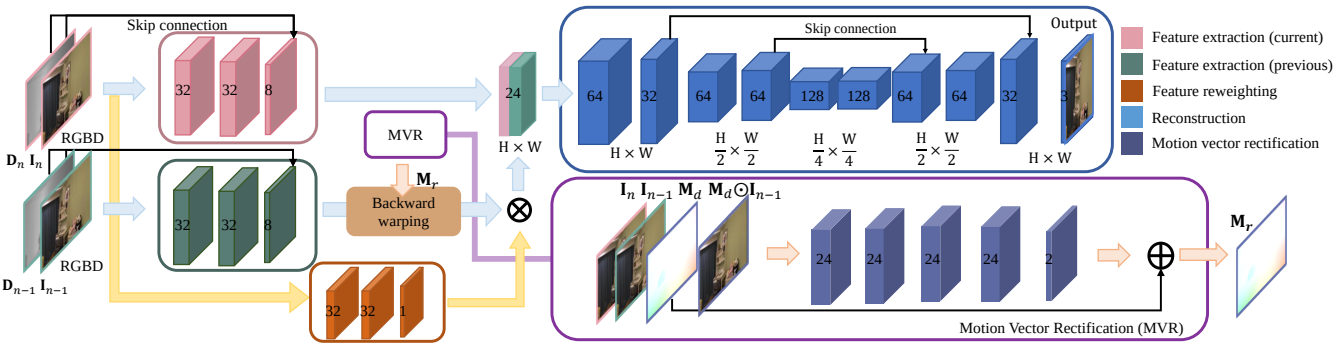


Fig. 10. Network architecture of our method. Our network consists of four modules, including feature extraction, re-weighting, reconstruction and motion vector rectification (MVR). The numbers on each network layer represent the output channels. In the reconstruction module, the height (H) and width (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.

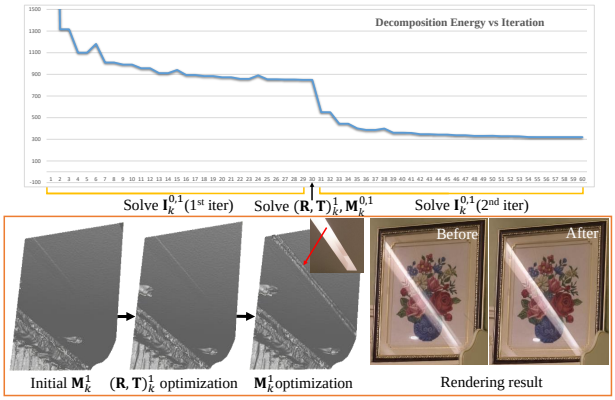


Fig. 11. Convergence curve of the alternating optimization algorithm for reflection decomposition. The rendering results before and after optimization are shown in the last two images in the second row.

Table 2. Preprocessing times for Living Room 1 in hours. SfM+MVS: the global mesh reconstruction using RealityCapture software. Per-view depth: the per-view depth refinement. Meshing: the per-view mesh simplification. Decomp.: the per-view two-layer decomposition.

Step	SfM+MVS	Per-view Depth	Meshing	Decomp.	DSRNet Training
Runtime	8h	5.5h	7h	1.5h	12h

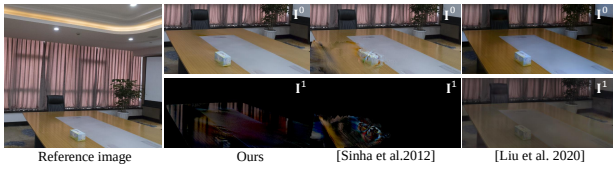


Fig. 12. Reflection decomposition comparison.

IBR methods. Please also see the accompanying video for the video comparisons.

6.1 Evaluation of the Reflection Decomposition Algorithm

Fig. 11 illustrates the convergence curve of the alternating optimization algorithm for reflection decomposition. The energy defined in Eq. 7 is optimized using the conjugate gradient (CG) method in Ceres solver [Agarwal et al. 2010], and it 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. 11 is used to emphasize the effect of the optimization of $\mathbf{M}_k^1$. It can be seen that the image rendered using the optimization result is sharp and free of misalignment artifacts in highlights that is present before the optimization. A comparison in Fig. 12 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]. Given that we do not capture 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].

In Fig. 13, 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 data term inside the detected highlight regions. The holes inside the highlights of the decomposed surface image are filled by the multi-view consistency step. Fig. 14 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

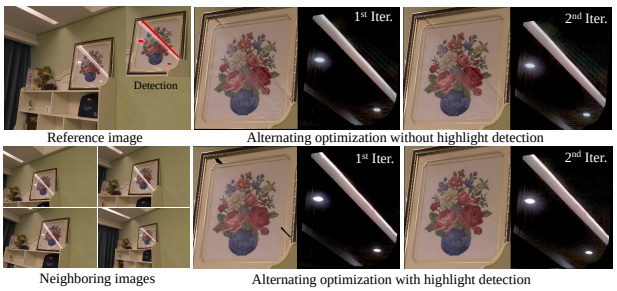


Fig. 13. Two-layer decomposition for highlights. Red regions on the top-right of the reference image indicate the detected highlights. Without highlight detection, the highlights in neighboring views will lead to spreading artifacts as shown in the decomposed foreground surface image in the top row.

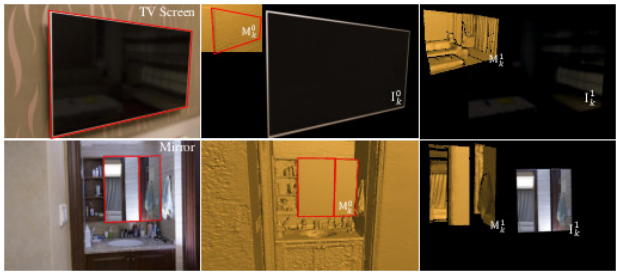


Fig. 14. Reflection decomposition results of a TV screen and a mirror.

Fig. 15. View warping vs. DSRNet. For each pair, view warping result is on the left, and the DSRNet result is on the right. The blurring and aliasing at object boundaries are effectively removed by the DSRNet.

Table 3. MVR ablation study.

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

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. 14.

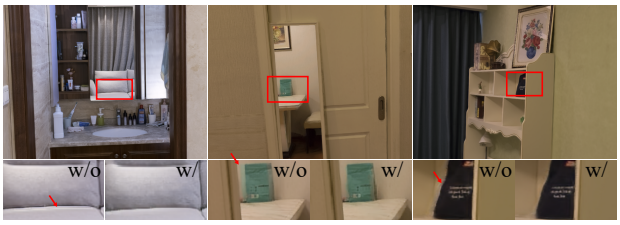


Fig. 16. The improvement of object boundary rendering quality using the MVR module. W/: with MVR. W/O: without MVR.

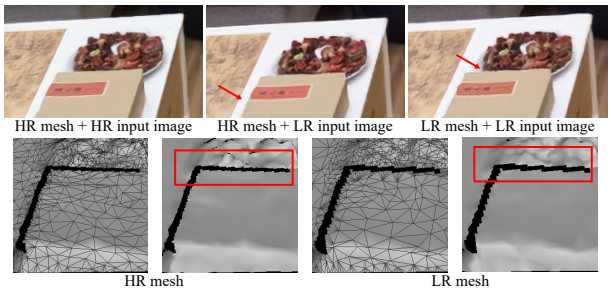


Fig. 17. The influence of mesh and input image resolution to DSRNet. HR/LR mesh: mesh constructed using high/low resolution depth map. HR/LR input image: generate high-resolution or low-resolution images with view warping. HR mesh + HR input image leads to improved rendering quality. Please also see the accompanying video for the comparison.

Table 4. Loss term 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

6.2 Evaluation of DSRNet

After training, the DSRNet can produce sharp, high-quality images at novel viewpoints. As illustrated in Fig. 15, 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. 16 illustrates that the designed MVR module is beneficial to remove the ghosting artifacts.

6.2.1 Ablation Study. We perform ablation studies to evaluate the influence of the MVR module and loss terms on the DSRNet. Tab. 3 shows that 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. 16, 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 results on the hotel room scene for this ablation study are shown in Tab. 4. The results verify that both VGG loss and L1 loss are essential to the quality of the rendered images.

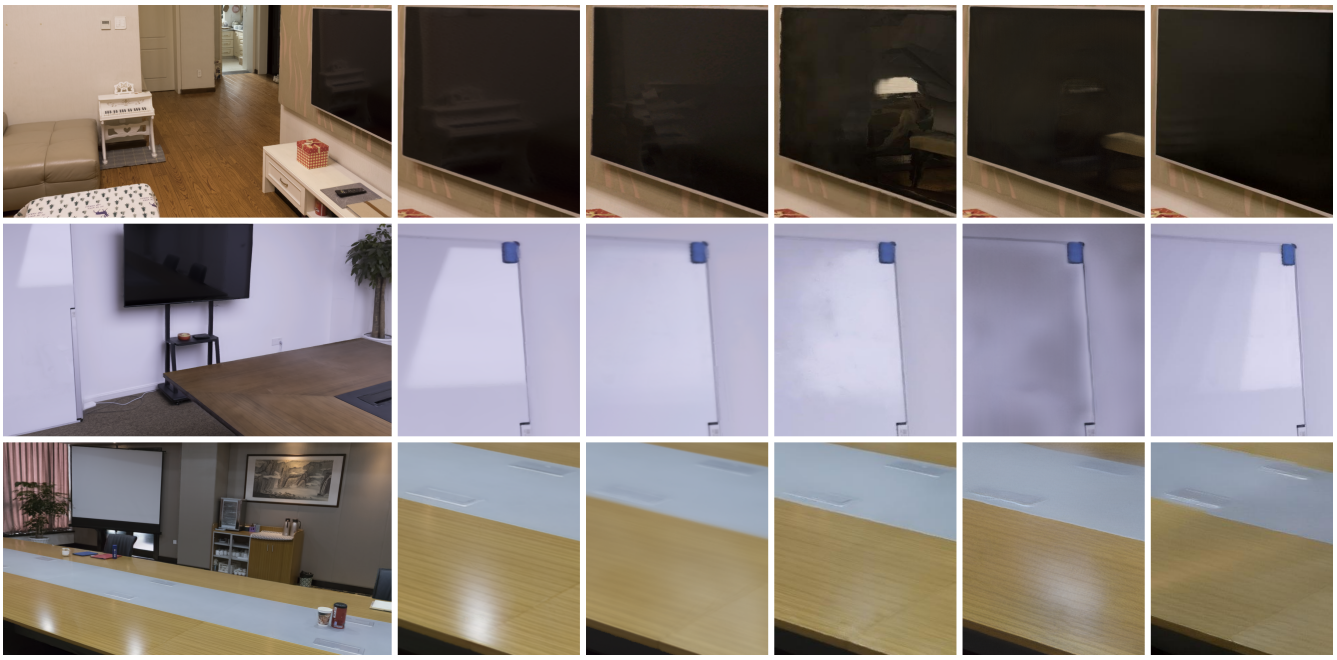


Fig. 18. Rendering result comparisons with **InsideOut** [Hedman et al. 2016], **DeepBlending** [Hedman et al. 2018] , **NRW** [Meshry et al. 2019] and **FVS** [Riegler and Koltun 2020].

Fig. 19. Comparisons with **LLFF** [Mildenhall et al. 2019] and **NeRF** [Mildenhall et al. 2020].

6.2.2 Influence of the View Warping Result on the DSRNet. As shown in Fig. 17, generating high-resolution (HR) images with per-view meshes constructed with HR images can achieve superior rendering results. The reason we choose to construct the mesh with HR images is to preserve the mesh edges on occlusion edges. Therefore, the occlusion edge details of the HR RGBD images can be better preserved, which facilitates the DSRNet in producing high-quality images. Note that the mesh constructed on low-resolution (LR) RGBD images has considerably fewer boundary edges, thereby leading to blurring or aliasing artifacts around occlusion edges. Furthermore, we found it is beneficial to recover occlusion edge details if generating HR images in view warping.

Table 5. Quantitative comparisons.

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**

6.3 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 2020]. 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 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**. Given that our DSRNet is trained for each scene to improve the rendering quality, we also fine-tune the networks of **DeepBlending** and **FVS** for the comparison. As shown in Fig. 18, 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 Tab. 5, in which our pipeline achieves the best performance over state-of-the-art methods on the validation datasets.

Fig. 19 illustrates the comparisons with **LLFF** and **NeRF**. Although these two methods can render high-quality images, it is still challenging for them to handle high-frequency signals, such as reflections and check patterns, resulting in evident blurring artifacts. In contrast, our geometry-based IBR pipeline can produce sharp images in these challenging cases.

7 CONCLUSIONS AND DISCUSSION

We have developed an IBR pipeline to render indoor scenes with reflections. It has two main technical components: global-mesh-guided robust two-layer mesh construction and a 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.

One limitation of our pipeline is that it can not handle curved 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 for a curved reflective surface using our IBR pipeline can be an alternative method to simulate its reflection. Another limitation is that our reflection decomposition algorithm needs to have enough images surrounding each reference image with translational motions to achieve high-quality decomposition results. Otherwise, we discard the decomposition results. As a result, our view warping algorithm might blur the undecomposed reflections since it is designed to favor the temporal smoothness of rendering results. In addition, our pipeline can not handle glass with both background transmission and reflection. In order to realistically render them, we might need to extend the linear composition rule used in our paper to three layers, including transmission, reflection, and possible opaque materials, such as papers or stickers, on the glass. 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 2021], into the pipeline to balance between the rendering speed and the robustness to inaccurate geometry in IBR.

ACKNOWLEDGMENTS

We thank anonymous reviewers for their professional and constructive comments. Special thanks to Yifei Li for the help on the super-resolution network. Weiwei Xu is partially supported by NSFC (No. 61732016). Qixing Huang would like to acknowledge the support from NSF IIS-2047677 and NSF HDR TRIPODS-1934932.

REFERENCES

S. Agarwal, K. Mierle, and Others. 2010. Ceres Solver. http://ceres-solver.org.
M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. Duvall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec. 2020. Immersive Light Field Video with a Layered Mesh Representation. *ACM Trans. Graph.* 39, 4 (2020), 15.
C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. 2001. Unstructured lumigraph rendering. In *ACM Trans. Graph.* 425–432.
J. Caballero, C. Ledig, A. P. Aitken, A. Acosta, J. Totz, Z. Wang, and W. Shi. 2017. Real-Time Video Super-Resolution with Spatio-Temporal Networks and Motion Compensation. In *CVPR, IEEE.* 2848–2857.
CapturingReality. 2016. Reality capture, http://capturingreality.com.
C. R. A. Chaitanya, A. S. Kaplanyan, C. Schied, M. Salvi, A. Lefohn, D. Nowrouzezahrai, and T. Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. *ACM Trans. Graph.* 36, 4, Article 98 (2017), 12 pages.
G. Chaurasia, S. Duchene, O. Sorkine-Hornung, and G. Drettakis. 2013. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph.* 32, 3 (2013), 1–12.
G. Chaurasia, O. Sorkine-Hornung, and G. Drettakis. 2011. Silhouette-Aware Warping for Image-Based Rendering. In *Computer Graphics Forum*, Vol. 30. 1223–1232.
S. E. Chen and L. Williams. 1993. View Interpolation for Image Synthesis. In *SIGGRAPH, ACM.* 279–288.
P. E. Debevec, C. J. Taylor, and J. Malik. 1996. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *SIGGRAPH, ACM.* 11–20.
M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. 1999. Implicit Fairing of Irregular Meshes Using Diffusion and Curvature Flow. In *SIGGRAPH, ACM.* 317–324.
P. Dollár and C. L. Zitnick. 2015. Fast Edge Detection Using Structured Forests. *IEEE Trans. PAMI* 37, 8 (2015), 1558–1570.
C. Dong, C. C. Loy, K. He, and X. Tang. 2014. Learning a deep convolutional network for image super-resolution. In *ECCV, Springer.* 184–199.
S. Dong, K. Xu, Q. Y. Zhou, A. Tagliasacchi, S. Xin, M. Nießner, and B. Chen. 2019. Multi-Robot Collaborative Dense Scene Reconstruction. *ACM Trans. Graph.* 38, 4, Article 84 (2019), 16 pages.
A. Edelsten, P. Jukarainen, and A. Patney. 2019. Truly next-gen: Adding deep learning to games and graphics. In *In NVIDIA Sponsored Sessions (Game Developers Conference).*
J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker. 2019. Deepview: View synthesis with learned gradient descent. In *CVPR, IEEE.* 2367–2376.
J. Flynn, I. Neulander, J. Philbin, and N. Snavely. 2016. Deepstereo: Learning to predict new views from the world's imagery. In *CVPR, IEEE.* 5515–5524.
D. Fuoli, S. Gu, and R. Timofte. 2019. Efficient Video Super-Resolution through Recurrent Latent Space Propagation. In *ICCV, IEEE Workshop.* 3476–3485.
Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. 2009. Reconstructing building interiors from images. In *ICCV, IEEE.* 80–87.
Y. Furukawa and J. Ponce. 2010. Accurate, Dense, and Robust Multiview Stereopsis. *IEEE Trans. PAMI* 32, 8 (2010), 1362–1376.
M. Garland and P. S. Heckbert. 1997. Surface Simplification Using Quadric Error Metrics. In *SIGGRAPH, ACM.* 209–216.
M. Goesele, J. Ackermann, S. Fuhrmann, C. Haubold, R. Klowsky, D. Steedly, and R. Szeliski. 2010. Ambient Point Clouds for View Interpolation. In *SIGGRAPH, ACM.* Article 95, 6 pages.
M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. 2007. Multi-View Stereo for Community Photo Collections. In *ICCV, IEEE.* 1–8.
S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. 1996. The lumigraph. In *SIGGRAPH, ACM.* 43–54.
X. Guo, X. Cao, and Y. Ma. 2014. Robust separation of reflection from multiple images. In *CVPR, IEEE.* 2187–2194.
M. Haris, G. Shakhnarovich, and N. Ukita. 2019. Recurrent Back-Projection Network for Video Super-Resolution. In *CVPR, IEEE.* 3892–3901.
R. I. Hartley and A. Zisserman. 2004. *Multiple View Geometry in Computer Vision* (second ed.). Cambridge University Press, ISBN: 0521540518.
J. He, S. Zhang, M. Yang, Y. Shan, and T. Huang. 2019. BDCN: Bi-Directional Cascade Network for Perceptual Edge Detection. In *CVPR, IEEE.* 3828–3837.
P. Hedman, S. Alsisan, R. Szeliski, and J. Kopf. 2017. Casual 3D Photography. *ACM Trans. Graph.* 36, 6, Article 234 (2017), 15 pages.

P. Hedman, J. Philip, T. Price, J. M. Frahm, G. Drettakis, and G. Brostow. 2018. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.* 37, 6 (2018), 1–15.

P. Hedman, T. Ritschel, G. Drettakis, and G. Brostow. 2016. Scalable inside-out image-based rendering. *ACM Trans. Graph.* 35, 6 (2016), 1–11.

H. Hirschmuller. 2008. Stereo Processing by Semiglobal Matching and Mutual Information. *IEEE Trans. PAMI* 30, 2 (2008), 328–341.

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

T. Igarashi, T. Moscovich, and J. F. Hughes. 2005. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.* 24, 3 (2005), 1134–1141.

J. Kopf, F. Langguth, D. Scharstein, R. Szeliski, and M. Goesele. 2013. Image-based rendering in the gradient domain. *ACM Trans. Graph.* 32, 6 (2013), 1–9.

C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, and Z. Wang. 2017. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *CVPR, IEEE*. 105–114.

M. Levoy and P. Hanrahan. 1996. Light field rendering. In *SIGGRAPH, ACM*. 31–42.

C. Li, Y. Yang, K. He, S. Lin, and J. E. Hopcroft. 2020. Single Image Reflection Removal through Cascaded Refinement. In *CVPR, IEEE*. 3565–3574.

Y. Li and M. S. Brown. 2013. Exploiting Reflection Change for Automatic Reflection Removal. In *ICCV, IEEE*.

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).

L. Liu, J. Gu, K. Z. Lin, T. S. Chua, and C. Theobalt. 2020a. Neural Sparse Voxel Fields. *NeurIPS* (2020).

Y. L. Liu, W. S. Lai, M. H. Yang, Y. Y. Chuang, and J. B. Huang. 2020b. Learning to See Through Obstructions. In *CVPR, IEEE*. 14215–14224.

S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh. 2019. Neural Volumes: Learning Dynamic Renderable Volumes from Images. *ACM Trans. Graph.* 38, 4CD (2019), 65.1–65.14.

W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. 2000. Image-Based Visual Hulls. In *SIGGRAPH, ACM*. 6.

W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. Mcmillan. 2002. Image-Based 3D Photography Using Opacity Hulls. *ACM Trans. Graph.* 21, 3 (2002), 427–437.

M. Meshry, D. B. Goldman, S. Khamis, H. Hoppe, R. Pandey, N. Snavely, and R. Martin-Brualla. 2019. Neural rerendering in the wild. In *CVPR, IEEE*. 6878–6887.

B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. 2019. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph.* 38, 4 (2019), 1–14.

B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and N. Ren. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV, Springer*.

R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, and A. W. Fitzgibbon. 2011. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*. IEEE, 127–136.

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

R. Ortiz-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.

E. Penner and L. Zhang. 2017. Soft 3D reconstruction for view synthesis. *ACM Trans. Graph.* 36, 6 (2017), 1–11.

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

J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. 2015. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR, IEEE*. 1164–1172.

G. Riegler and V. Koltun. 2020. Free View Synthesis. In *ECCV, Springer*.

G. Riegler and V. Koltun. 2021. Stable View Synthesis. In *CVPR, IEEE*.

S. Rodriguez, S. Prakash, P. Hedman, and G. Drettakis. 2020. Image-Based Rendering of Cars using Semantic Labels and Approximate Reflection Flow. *Proc. ACM Comput. Graph. Interact.* 3 (2020).

M. S. Sajjadi, Vemulapalli, and M. R., Brown. 2018. Frame-Recurrent Video Super-Resolution. In *CVPR, IEEE*. 6626–6634.

J. L. Schonberger and J. M. Frahm. 2016. Structure-from-Motion Revisited. In *CVPR, IEEE*. 4104–4113.

J. L. Schönberger, E. Zheng, J. M. Frahm, and M. Pollefeys. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *ECCV, Springer*, Vol. 9907. 501–518.

J. Shade, S. Gortler, L. He, and R. Szeliski. 1998. Layered depth images. In *SIGGRAPH, ACM*. 231–242.

H. Y. Shum and S. B. Kang. 2000. *A Review of Image-based Rendering Techniques*. Technical Report. Microsoft.

S. N. Sinha, J. Kopf, M. Goesele, D. Scharstein, and R. Szeliski. 2012. Image-based rendering for scenes with reflections. *ACM Trans. Graph.* 31, 4 (2012), 1–10.

S. N. Sinha, D. Steedly, and R. Szeliski. 2009. Piecewise planar stereo for image-based rendering. In *ICCV, IEEE*. 1881–1888.

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

P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. 2019. Pushing the boundaries of view extrapolation with multiplane images. In *CVPR, IEEE*. 175–184.

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

X. Tao, H. Gao, R. Liao, J. Wang, and J. Jia. 2017. Detail-Revealing Deep Video Super-Resolution. In *ICCV, IEEE*. 4482–4490.

N. Tatarchuk, B. Karis, M. Drobot, N. Schulz, J. Charles, and T. 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.

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

J. Thies, M. Zollhöfer, and M. Nießner. 2019b. Deferred neural rendering: Image synthesis using neural textures. *ACM Trans. Graph.* 38, 4 (2019), 1–12.

X. Wang, K. Chan, K. Yu, C. Dong, and C. C. Loy. 2019. EDVR: Video Restoration With Enhanced Deformable Convolutional Networks. In *CVPR, IEEE Workshop*. 1954–1963.

Z. Wang, J. Chen, and S. C. H Hoi. 2020. Deep Learning for Image Super-resolution: A Survey. *IEEE Trans. PAMI* (2020), 1–1.

T. Whelan, M. Goesele, S. J. Lovegrove, J. Straub, S. Green, R. Szeliski, S. Butterfield, S. Verma, R. A. Newcombe, M. Goesele, et al. 2018. Reconstructing scenes with mirror and glass surfaces. *ACM Trans. Graph.* 37, 4 (2018), 102–1.

D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle. 2000. Surface light fields for 3D photography. In *SIGGRAPH, ACM*. 287–296.

L. Xiao, S. Nouri, M. Chapman, A. Fix, D. Lanman, and A. Kaplanyan. 2020. Neural supersampling for real-time rendering. *ACM Trans. Graph.* 39, 4 (2020), 142–1.

K. Xu, L. Zheng, Z. Yan, G. Yan, E. Zhang, M. Niessner, O. Deussen, D. Cohen-Or, and H. Huang. 2017. Autonomous Reconstruction of Unknown Indoor Scenes Guided by Time-Varying Tensor Fields. *ACM Trans. Graph.* 36, 6 (2017), 15.

Z. Xu, S. Bi, K. Sunkavalli, S. Hadap, H. Su, and R. Ramamoorthi. 2019. Deep view synthesis from sparse photometric images. *ACM Trans. Graph.* 38, 4 (2019), 1–13.

T. Xue, M. Rubinstein, C. Liu, and W. T. Freeman. 2015. A computational approach for obstruction-free photography. *ACM Trans. Graph.* 34, 4 (2015), 1–11.

J. Yang, D. Gong, L. Liu, and Q. Shi. 2018. Seeing deeply and bidirectionally: A deep learning approach for single image reflection removal. In *ECCV, Springer*. 654–669.

C. Zhang and T. Chen. 2003. A survey on image-based rendering. *Signal Processing Image Communication* 19 (2003), 1–28.

T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. 2018. Stereo magnification: Learning view synthesis using multiplane images. In *SIGGRAPH, ACM*.