# Hybrid Image-based Rendering for Free-view Synthesis

SIDDHANT PRAKASH, Université Côte d'Azur and Inria, France
THOMAS LEIMKÜHLER, Université Côte d'Azur and Inria, France
SIMON RODRIGUEZ, Université Côte d'Azur and Inria, France
GEORGE DRETTAKIS, Université Côte d'Azur and Inria, France

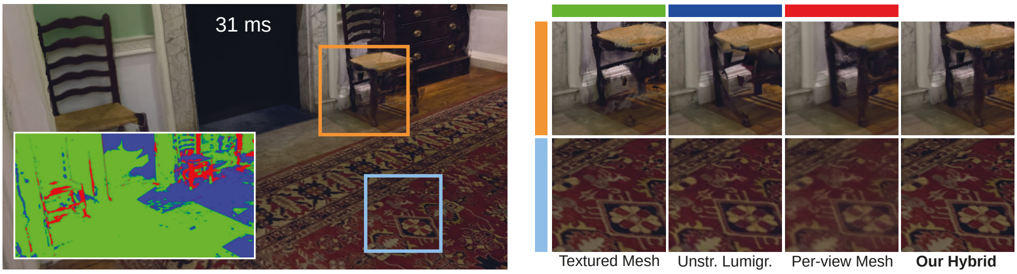


Fig. 1. Our method analyzes the geometric and photometric consistency of a casually captured scene and for each novel-view pixel selects the best rendering algorithm (color mask, Red: Per-view Mesh; Blue: Unstructured Lumigraph; Green: Textured Mesh). We address color seams, view-dependent effects, missing geometry (orange inset, top), and texture sharpness (light blue inset, bottom), while retaining interactive frame rates.

Image-based rendering (IBR) provides a rich toolset for free-viewpoint navigation in captured scenes. Many methods exist, usually with an emphasis either on image quality or rendering speed. In this paper we identify common IBR artifacts and combine the strengths of different algorithms to strike a good balance in the speed/quality tradeoff. First, we address the problem of visible color seams that arise from blending casually-captured input images by explicitly treating view-dependent effects. Second, we compensate for geometric reconstruction errors by refining per-view information using a novel clustering and filtering approach. Finally, we devise a practical hybrid IBR algorithm, which locally identifies and utilizes the rendering method best suited for an image region while retaining interactive rates. We compare our method against classical and modern (neural) approaches in indoor and outdoor scenes and demonstrate superiority in quality and/or speed.

CCS Concepts: • **Computing methodologies → Image-based rendering**; *Image processing*; Rasterization; Texturing.

Additional Key Words and Phrases: interactive rendering, image harmonization, uncertainty

**ACM Reference Format:**
Siddhant Prakash, Thomas Leimkühler, Simon Rodriguez, and George Drettakis. 2021. Hybrid Image-based Rendering for Free-view Synthesis. *Proc. ACM Comput. Graph. Interact. Tech.* 4, 1 (May 2021), 20 pages. https://doi.org/10.1145/3451260

Authors' addresses: Siddhant Prakash, Université Côte d'Azur and Inria, Sophia-Antipolis, France, siddhant.prakash@inria.fr; Thomas Leimkühler, Université Côte d'Azur and Inria, Sophia-Antipolis, France, thomas.leimkuhler@inria.fr; Simon Rodriguez, Université Côte d'Azur and Inria, Sophia-Antipolis, France, simon.rodriguez@inria.fr; George Drettakis, Université Côte d'Azur and Inria, Sophia-Antipolis, France, george.drettakis@inria.fr.

## 1 INTRODUCTION

Freely and *interactively* navigating in virtual representations of captured 3D scenes has widespread applications, e.g., urban planning, training in virtual reality, games or augmented reality. Recent Structure from Motion (SfM) [Snavely et al. 2006] and Multi-View Stereo (MVS) [Goesele et al. 2007] solutions allow the creation of a textured 3D mesh of a scene with only photos or video as input. However, these meshes have only diffuse appearance, and often have geometric reconstruction errors, degrading realism. Image-Based Rendering (IBR) [Shum et al. 2008] and recent neural rendering methods [Tewari et al. 2020] address some of these issues, using multiple photos to compensate for geometric errors and to render view-dependent effects such as glossy appearance. However, current methods still suffer from visual artifacts and can be slow.

We target a new IBR algorithm that addresses previous artifacts, focusing on scenes with wide-baseline capture, while allowing free-viewpoint navigation far from the input views at *interactive frame rates*. The first artifact we consider occurs when blending multiple photos of a given scene for IBR: each photo can be captured with different camera parameters (exposure, white balance, etc.) or lighting may change during capture. As a result, multiple views of a given diffuse surface do not blend well in a novel IBR view, creating visible seams. Previous methods fail to satisfactorily harmonize our wide-baseline datasets [HaCohen et al. 2013; Huang et al. 2017], and are not designed to preserve view-dependent effects, such as glossy material appearance. We want to identify these regions and preserve them in the images blended during IBR. The second artifact is related to geometric errors. SfM/MVS generate a *global mesh*. The fusion step of MVS is based on photoconsistency among views. Many hard cases such as non-diffuse surfaces, small or thin structures etc., are not well reconstructed resulting in visual artifacts for IBR methods using the global mesh [Buehler et al. 2001; Eisemann et al. 2008]. One effective solution is to use *per-view* information [Chaurasia et al. 2013; Hedman et al. 2018] which is not always multi-view consistent, but can capture missing geometric details. This information is valid close to input views, but can result in artifacts in distant novel views, depending on the blending strategy.

Finally, there is a tradeoff between rendering speed and image quality in all algorithms used to display multi-view datasets. The textured mesh is much faster than IBR, but is only valid in diffuse, well reconstructed regions. IBR algorithms using the global mesh geometry can reproduce glossiness, but suffer in badly reconstructed regions (ghosting, blurring), where per-view solutions are better. To provide the best speed/quality tradeoff, we need to identify each case, and efficiently combine the advantages of each method.

Our method addresses these issues. First, we propose a new color harmonization approach, where we use a textured mesh as a view-independent basis for diffuse harmonization, and *photometric variance* to identify view-dependent regions. We then re-inject these regions into the input images using techniques akin to digital photomontage [Agarwala et al. 2004]. This results in modified input images with diffuse regions that can be seamlessly blended in IBR while *preserving view-dependent* content. Second, we analyze the depth errors of view-dependent meshes [Hedman et al. 2018], and propose a new filtering approach based on clustering of depth of the different meshes, followed by spatial filtering. Finally, we propose a new *hybrid IBR* method by analyzing the strengths and weaknesses of different rendering algorithms. We identify regions where the global mesh is sufficient by estimating geometric uncertainty and use our filtering-based per-view mesh IBR algorithm for uncertain regions. Our method provides a good quality/speed tradeoff, even compared to deep-learning based solutions such as Deep Blending [Hedman et al. 2018].

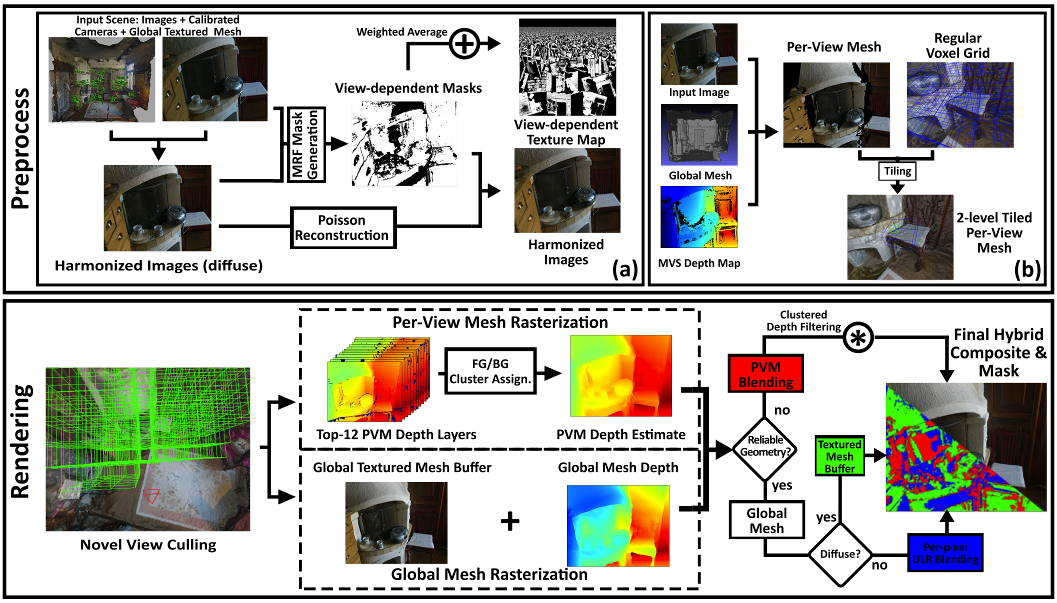


Fig. 2. The first step of our method takes an IBR scene and reduces color seam artifacts by harmonizing input images in a pre-processing step to obtain harmonized images *(a)*. Next, we compute per-view meshes (PVMs) similar to [Hedman et al. 2018] resulting in our 2-level tiled PVMs *(b)*. For rendering, we start by rasterizing the global mesh and PVMs to get a global and per-view depth estimate. We compare the global and per-view depth maps to decide if PVM rendering should be used instead of the global mesh. Finally, when rendering with the global mesh, we check if a pixel is diffuse using the view-dependent texture map obtained in *(a)* and use the textured mesh buffer for the final color, else we perform per-pixel ULR blending and use the blended color.

In summary, we present three main contributions:

- A new multi-view image harmonization algorithm that removes color differences in diffuse regions and *re-injects view-dependent effects* such as glossy appearance in the harmonized input images.
- A new blending algorithm for per-view meshes based on depth clustering and spatial filtering.
- A new, efficient hybrid IBR method that strikes a good balance in the speed/quality tradeoff.

We show results on several indoors and outdoors scenes, and provide comparisons with previous IBR and neural rendering algorithms, clearly illustrating the speed and/or quality superiority of our method. Our entire *hybrid IBR* pipeline is summarized in Fig. 2. The source code, supplemental materials and datasets can be found here: https://repo-sam.inria.fr/fungraph/hybrid-ibr/

## 2 RELATED WORK

We discuss previous work on harmonizing images for blending and stitching applications, related to our harmonization solution. Then we give an overview of the most closely related image-based and neural rendering algorithms.

## 2.1 Image Harmonization

Seamless blending is a long-standing problem in computer vision and graphics. Many applications such as panoramic image stitching, texture synthesis, photogrammetry along with image-based rendering require seamless blending.

Implicit methods such as Kim and Pollefeys [2008] and Goldman [2010] were proposed to prevent seam artifacts due to unknown vignetting, exposure and camera response functions causing appearance variation in multi-image datasets. These methods work best for 2D image operations on images captured with specific constraints but can be difficult to adapt to multi-view datasets. Zhang et al. [2016] propose a solution for 3D datasets which recovers per-image exposure and per-vertex radiance of the mesh to radiometrically calibrate the scene. Color transfer methods have also been used to harmonize images. Moulon et al. [2017] use common color segments between a reference and a target image and solve a linear optimization to recover gain/bias factors per image handling shutter speed and aperture variations. Transfer of image color statistics using a reference image was also explored in HaCohen et al. [2011] using an extension of the PatchMatch algorithm to find Non-Rigid Dense Correspondence (NRDC) regions and then fit a parametric color model for global color transfer from a reference image to the target image. This method was extended by HaCohen et al. [2013] for multi-image datasets. While these methods provide good color harmonization, they fail on large datasets such as ours, which have insufficient overlapping regions between distant images and in general are not designed to correctly handle view-dependent effects.

Multi-view mesh texturing solutions implicitly address the problem of color harmonization. Lempitsky and Ivanov [2007] use a Markov Random Field (MRF) [Kwatra et al. 2003] that incorporates a pairwise penalty for color difference at seams of model polygons. Follow-up work [Waechter et al. 2014; Zhou and Koltun 2014] extends this by formulating an optimization problem having a data term for view selection and smoothness term for color correction. Recently, Huang et al. [2017] used the NRDC color model to obtain seamless textures by solving an optimization to recover a parametric curve per image and adjusting the intensities accordingly. Commercial solutions [Jancosek and Pajdla 2011; Reality 2018] produce diffuse textures with similar properties. We use a harmonized textured mesh as a basis for our solution, but we *preserve* pixels in the input images corresponding to *view-dependent effects*.

## 2.2 Image-Based Rendering

Rendering a scene from captured images has a long history in computer graphics. Early work [Chen 1995; Chen and Williams 1993; McMillan and Bishop 1995] explored view interpolation between captured or rendered images to synthesize novel views. The seminal work of Levoy and Hanrahan [1996] and Gortler et al. [1996] introduced the idea of sampling the plenoptic function to warp input views into novel views using light fields and the Lumigraph respectively, with structured, grid-like capture. Debevec et al. [1998] presented one of the first mesh-based rendering solutions by performing view-dependent texture mapping efficiently on geometry built with user input. Unstructured Lumigraph Rendering (ULR) [Buehler et al. 2001] allowed IBR for casual capture by blending input images based on a geometric proxy. With advances in structure-from motion (SfM) and multi-view stereo (MVS), using explicit geometry has become a viable option for IBR. We call the geometry reconstructed by MVS the *global mesh*.

Snavely et al. [2006] proposed a system for camera calibration and viewing to enable virtual tours of captured locations through crowd-sourced images. Further improvements focused on rectifying the global mesh to model floating geometry [Eisemann et al. 2008] or using ambient point clouds for view interpolation [Goesele et al. 2010]. Despite these improvements, the global mesh still suffers from poor reconstruction in many parts of the scene, e.g., non-diffuse and transparent surfaces that are not photoconsistent, or small, thin and/or repetitive structures that are hard for MVS.

One class of solutions to these issues is to use *per-view* information for each input photo. Superpixel-based depth map refinement and synthesis was used by Chaurasia et al. [2013] together with shape-preserving warps to the novel views, while per-view meshes from refined depth maps were used in Inside Out [Hedman et al. 2016] to rectify reconstruction errors. In both cases, geometric details incorrectly reconstructed or even missing in the global mesh are corrected. We build on Inside Out per-view meshes to propose a new approach which handles uncertainty in local depth when seen from novel views. Reflections are a special case of view-dependent effects, that have been explicitly handled in previous work [Kopf et al. 2013; Rodriguez et al. 2020; Sinha et al. 2012] using layered rendering or semantics of scene content. Our hybrid algorithm takes inspiration from these ideas.

Ortiz-Cayon et al. [2015] presented an approach to determine whether a simple homography can selectively replace the more expensive warps of the superpixel-based solution of Chaurasia et al. [2013] in a pre-process. We introduce a hybrid rendering algorithm that is more powerful in that it selects the rendering algorithm for each pixel and each novel view.

Deep Blending [Hedman et al. 2018] improved the per-view geometry of Inside Out by combining high-quality depth maps [Schönberger et al. 2016] with a more complete (but possibly inaccurate) global mesh [Jancosek and Pajdla 2011; Reality 2018]. In addition, depth discontinuities and mesh edges are treated carefully, resulting in higher quality meshes compared to Inside Out. The method then learns blend weights to correct remaining artifacts, and, notably, to get rid of erroneous and floating geometry.

Deep Blending is one of the early *neural rendering* [Tewari et al. 2020] solutions. Thies et al. [2018] separates diffuse and specular components of a scene and learns the view-dependent effects explicitly. Recently, Riegler and Koltun [2020] use a recurrent neural network to learn warping and blending input views to novel views separately; we compare to this method in Sec. 7.

Other neural rendering methods aim to synthesize novel views by learning implicit scene representations [Flynn et al. 2019; Meshry et al. 2019; Mildenhall et al. 2019; Thies et al. 2019] or explicit volumetric representations [Lombardi et al. 2019; Martin-Brualla et al. 2021; Mildenhall et al. 2020; Sitzmann et al. 2019; Srinivasan et al. 2019; Zhou et al. 2018]. Most of these methods focus either on capturing a single object, or require small-baseline capture that allows only small motion around input views. Importantly performance of most of these methods is typically in (tens of) seconds or even minutes per frame, precluding their use for *interactive* applications. As reported by [Koltun 2020], the typical runtime for novel-view synthesis using [Martin-Brualla et al. 2021; Mildenhall et al. 2020] is 1-2 minutes and roughly 1 second using [Riegler and Koltun 2020]. Our method takes around 30 ms on average and can provide frame-rates as high as 50Hz (20 ms).

Our solution, based on improved image harmonization, spatial filtering and a hybrid rendering algorithm avoids the use of a neural network altogether, achieving a good balance between speed and quality. Our hybrid renderer runs at much faster frame rates than all current neural rendering algorithms, allowing *interactive* free-viewpoint navigation of captured scenes.

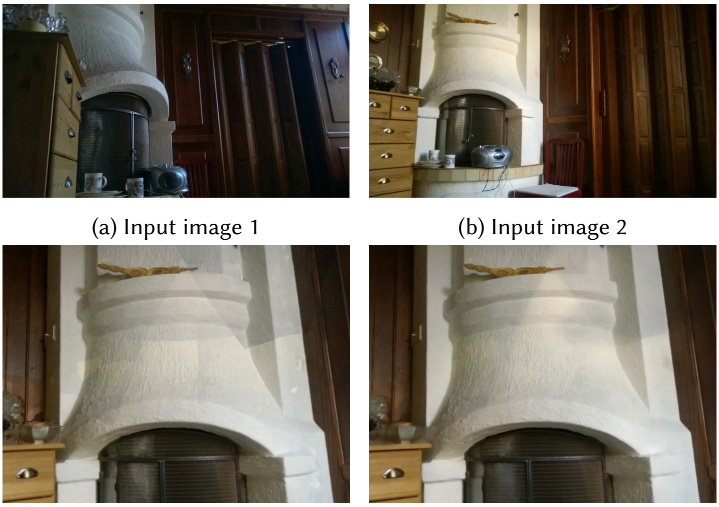
## 3 ANALYSIS AND MOTIVATION

As discussed in the introduction, we address three major issues for wide-baseline IBR: 1. Eliminating visual artifacts due to differences in diffuse surface appearance due to capture errors, while *preserving* desirable view-dependent effects. 2. Removing residual artifacts due to geometric reconstruction errors using per-view geometry. 3. Dealing with the quality-speed tradeoff by carefully selecting between different algorithms.

We will analyze each issue below, including the strengths and weaknesses of different IBR algorithms, helping us develop our new hybrid solution. In particular, we consider Inside Out [Hedman et al. 2016] and Deep Blending [Hedman et al. 2018] that use per-view meshes (PVM) to compensate for geometric inaccuracies. In addition to the deep-learning approach to blending, Hedman et al. [2018] introduced a baseline heuristic method to blend PVMs without a neural network; we refer to this method as PVM-H from now on.

### 3.1 Harmonization with View-Dependent Effects

When capturing multi-view datasets there can be subtle changes in illumination due to internal settings of the camera – even when some exposure settings are fixed – or changes in lighting during the capture session. These differences in input images show up as discontinuities or *seams* in novel views during IBR, when blending pixels from different input images. Note that learned weights in the recent Deep Blending [Hedman et al. 2018] algorithm reduce such seams, but residual artifacts remain, as shown in Fig. 3. Previous methods (Sec. 2) remove the view-dependent effects we want to preserve.


(a) Input image 1

(b) Input image 2

(c) PVM-H

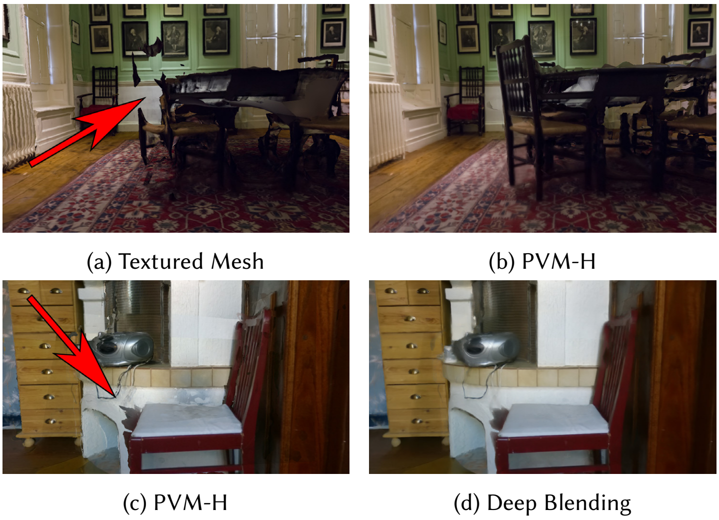
(d) Deep Blending

Fig. 3. Top: The chimney has different colors due to exposure differences in the two images. The color differences on the metal handles should be preserved. Bottom left: Novel view with PVM-H has significant seams. Bottom right: Deep Blending reduces the seams, but residual artifacts remain.

Our goal is to achieve results on par or even better than Deep Blending in terms of removing seams. We do this by *harmonization of diffuse* surfaces while explicitly *identifying and preserving* view-dependent effects (Sec. 4).

### 3.2 Compensating for Geometric Reconstruction Errors

Early image-based rendering algorithms used the global mesh which suffers from reconstruction errors, typically due to the uncertainty in the depth maps computed by multi-view stereo algorithms, and in the subsequent fusion step. PVM-based methods [Hedman et al. 2018, 2016] reduce artifacts due to such errors. This is clearly illustrated in Fig. 4, where the chair is missing from the global textured mesh (4a), but present in the rendering using PVM-H (4b).



(a) Textured Mesh

(b) PVM-H

(c) PVM-H

(d) Deep Blending

Fig. 4. Top left: The textured mesh is missing the chair. Top right: Per-view meshes are able to reconstruct the missing detail. Bottom: An extreme case (left), where a PVM creates visual artifacts, that even the learned weights of Deep Blending (right) cannot correct.

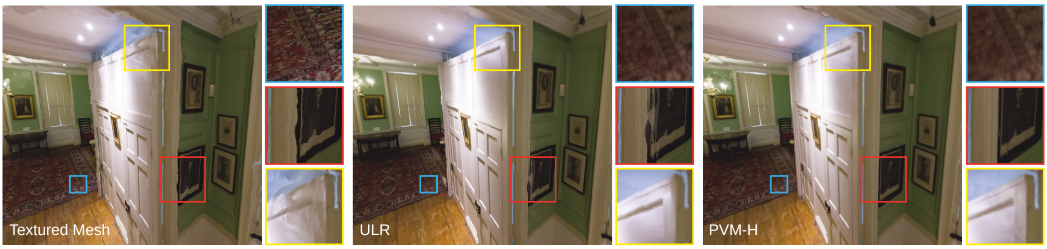


Fig. 5. Textured mesh preserves sharp details in diffuse regions that are lost due to blending in ULR and PVM-H (blue inset). ULR corrects minor geometric errors in textured mesh typically in glossy regions (yellow inset). Large geometric errors, such as ones due to occlusion edges, are rectified by per-view geometry improving PVM-H rendering quality over ULR (red inset).

PVMs can correct erroneous geometric reconstructions in the global mesh, but are not necessarily multi-view consistent. As a result, when moving away from the corresponding input view, PVMs can introduce artifacts. An extreme case is shown in Fig. 4c, where a PVM from the front of the chair extends incorrectly. Deep Blending (Fig. 4d) tries to resolve these issues by assigning a low blend weight to PVMs responsible for artifacts. However, in such cases, the weighting strategy is insufficient. Resolving such uncertainty in depth by improving the pre-processing step for PVMs [Hedman et al. 2018] is hard because the input images do not capture the required depth information in any view.

To summarize, PVMs can correct for missing and erroneous geometry of the global mesh, but can also result in artifacts far from the input views. These observations guide our approach to improve per-view mesh IBR by *estimating uncertainty* and using a novel filtering method for view synthesis.

### 3.3 Quality-Speed Tradeoff for IBR

We consider three algorithms for displaying multi-view content: the textured mesh (TM), produced by any MVS system, per-pixel unstructured lumigraph rendering (ULR) [Buehler et al. 2001] and finally per-view mesh rendering (PVM). The discussion here of PVM is valid both for the PVM-H approach [Hedman et al. 2018] and our improved solution (Sec. 5). We analyze the tradeoffs of these algorithms, motivating our hybrid rendering solution (Sec. 6); we start with a cost breakdown of different steps for each algorithm (Tbl. 1).

Table 1. Average cost of each step for the different rendering algorithms (see Sec. 5 for details).

| Steps/Algorithm | TM | ULR | PVM-H | DB |
|---|---|---|---|---|
| Global Geometry | 0.5 *ms* | 0.5 *ms* | 0.5 *ms* | 0.5 *ms* |
| Voxel Lookup | - | - | 3.4 *ms* | 3.1 *ms* |
| Tiles Sorting | - | - | 1.3 *ms* | 1.5 *ms* |
| PV Depth Pass | - | - | 0.6 *ms* | 0.6 *ms* |
| Blend Cost Compute | - | - | 3.6 *ms* | - |
| Mosaic Generate | - | - | - | 5.4 *ms* |
| Blending | - | 4.5 *ms* | 0.2 *ms* | 51.6 *ms* |
| **Total** | 0.5 *ms* | 5.0 *ms* | 9.6 *ms* | 62.7 *ms* |

*Textured Mesh.* Rendering a scene with TM incurs negligible cost on modern GPUs using a single fetch operation per fragment. The other two algorithms also rasterize a g-Buffer from the global mesh: ULR uses the depth per fragment to lookup color in input images, and PVM uses the depth to determine visible PVMs; we can thus render TM at every frame at no extra cost. Novel views are very stable with TM, but the rendered images can lack realism in regions with poor reconstruction, and for non-diffuse objects. On the positive side, TM – unlike all IBR methods – does not blend multiple input images which can avoid blur in some cases.

*Unstructured Lumigraph.* The original ULR algorithm [Buehler et al. 2001] triangulates the image and blends images over this image-space mesh. With high-quality MVS meshes, we use deferred shading where ULR weights are applied at each visible point of the global mesh to blend a subset of input images (we preselect the 4 best using ULR weights and then blend these). This algorithm was used as a baseline comparison in several previous IBR solutions [Hedman et al. 2018; Mildenhall et al. 2019; Rodriguez et al. 2020].

Per-pixel ULR preserves view-dependent effects, since for each pixel in the novel view, the algorithm re-projects and blends colors from a subset of input cameras, i.e., a highlight moves from its position in one input view to another. It also reduces artifacts due to minor reconstruction errors in TM as shown in Fig. 5 (yellow inset). Interestingly, this happens in regions which exhibit glossiness that in turn introduces error in reconstruction. Thus rendering these pixels with ULR over TM improves quality both in terms of reconstruction error and view-dependent appearance. In addition, we see in Tbl. 1 that this version of ULR is faster than PVM.

ULR can also treat distant backgrounds, where MVS systems often create large textureless triangles. As previously noted [Rodriguez et al. 2020], PVMs are unsuitable for such regions, because they are sparsely covered in input images and would be wasteful.

*Per-View Meshes.* As discussed in Sec. 3.2, PVMs improve image quality by correcting erroneous geometry of the global mesh, but can result in significant artifacts far from input views. We make the empirical observation that for a given novel view, a majority of PVMs provide the correct depth. This suggests a new depth filtering strategy that can reduce some of the artifacts due to

PVMs. From Tbl. 1 we see that PVMs are slower than ULR but are still 6 times faster than the full deep-learning pipeline.

These observations guide the design of our hybrid IBR algorithm (Sec. 6 and Fig. 7).

## 4 IMAGE HARMONIZATION

In wide baseline multi-view datasets, images may have significant appearance variation between views for the same region, (Fig. 6a), as explained in Sec. 3.1. We will utilize multi-view consistency to harmonize appearance while explicitly preserving the view-dependent effects, in two steps. We give an overview of the steps involved in Fig. 2a. First we harmonize the input images as if they were entirely diffuse, using the textured mesh as a baseline for color correction. Next, we explicitly identify view-dependent regions that we want to preserve, and re-inject them into the harmonized images similar to digital photomontage [Agarwala et al. 2004]. The final processed images are correctly harmonized in diffuse regions, while preserving desirable view-dependent effects such as glossy or specular surfaces (see Fig. 6).

### 4.1 Diffuse Harmonization

We use the textured mesh as a baseline for color correction, since it provides a stable common reference point for appearance across input views without view-dependent effects. The global mesh texture provides a base color for each vertex and is typically free from incorrect color seams, serving as a good baseline for image harmonization. We experimented with different texturing methods, e.g., Waechter et al. [2014] and a simple approach using ULR weights to blend textures [Bonopera et al. 2020]. We obtained best results using the textured mesh produced by RealityCapture [Reality 2018], which we use for all our tests. It should be noted that our solution can work with Delaunay tetrahedralization meshes reconstructed using any off-the-shelf solution such as COLMAP [Schönberger et al. 2016] and textured using any open-source texturing method.

We project each vertex in the cameras from which it is visible, compute the per-channel standard deviation of color intensities corresponding to each pixel associated with the vertex and store it as the vertex color, in the *variance mesh*.

Recovering a single exposure factor per-image, as done in previous work, does not capture color variations due to changes in lighting in localized image regions. Instead we compute a per-pixel, per-channel *scaling factor* for each input image, correcting each pixel to be close to the corresponding texture value, and applying a Gaussian kernel for smoothness. We perform the following operation to obtain the output pixel value $I_{out}^{\text{lin}}$:

$$I_{out}^{\text{lin}} = \frac{G * T^{\text{lin}}}{G * I^{\text{lin}}} I^{\text{lin}},$$

where $G$ denotes a Gaussian kernel (filter size 3% of dominant image resolution), $*$ denotes convolution, $T^{\text{lin}}$ is the texture, $I^{\text{lin}}$ the linearized input image and all operations are applied independently for each color channel, and in linear color space to ensure meaningful image transformations, by inverting gamma correction.

We leave out saturated pixels with an intensity in the highest or lowest 2% range from the computation. Additionally, we leave out pixels with color variance greater than 10% of the color range, identified by back-projecting the variance mesh to input images.

### 4.2 Re-injecting View-Dependent Pixels

Input images are now harmonized, removing seams when blending for novel view synthesis. However, desirable view-dependent effects such as glossy highlights are also lost. We re-inject this information in two steps. First, we identify regions with view-dependent effects by computing a

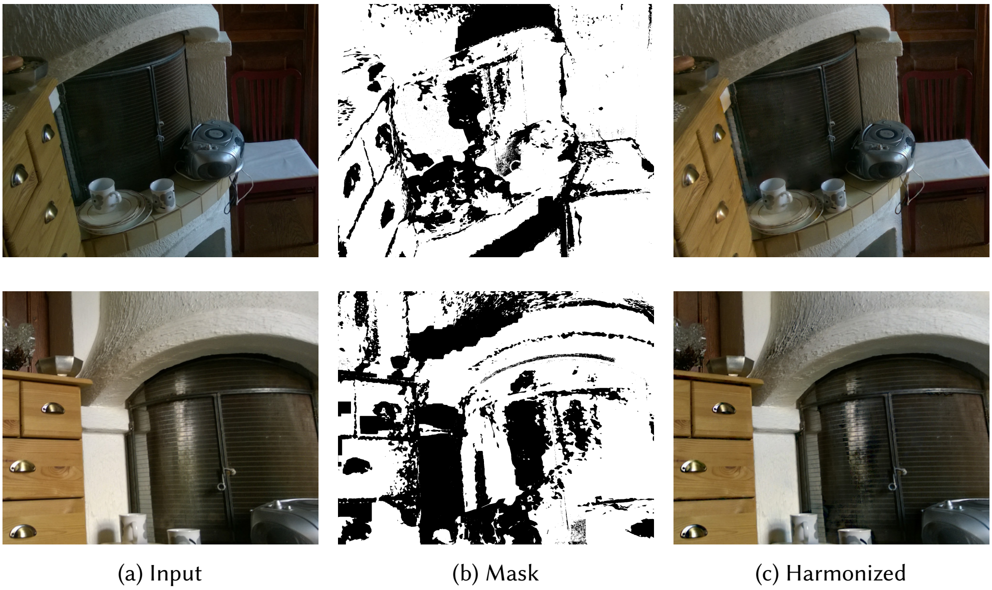

(a) Input (b) Mask (c) Harmonized

Fig. 6. Harmonization: From an input image (a) exhibiting significant appearance variation, we obtain harmonized images (c). A mask (b) alleviates specularity suppression.

harmonization mask for each input image. Second, for each view-dependent region in each input image, we re-introduce the original image statistics.

To compute the harmonization mask we start by formulating a graph-cut based energy minimization problem [Kwatra et al. 2003] to recover a *binary mask*. The mask indicates if the pixel should be changed to contain view-dependent information or remain unchanged in the harmonized image. We use a method similar to Agarwala et al. [2004]; details are provided in supplemental. The mask obtained from the MRF is combined (binary "and") with the regions which were pre-identified as saturated and highly view-dependent in Sec. 4.1.

Once the regions are identified, we perform Poisson blending [Pérez et al. 2003] to avoid seams on region boundaries. We use diffuse harmonized images as a guide to re-introduce the gradients of the original image in the regions identified (in black) using the binary mask (Fig. 6b). We see that shiny materials (fireplace protection, cassette player, drawer handles) are marked as view-dependent. The final harmonized image for a given pair of images is shown in Fig. 6c.

## 5 PER-VIEW MESH-BASED RENDERING

In this section we address artifacts due to PVM rendering that typically occur far from the input views (Sec. 3.2), by introducing a two-level voxel structure and a spatial filtering step.

### 5.1 Storage and Acceleration

Previous methods [Hedman et al. 2018, 2016] store PVMs as *tiles* in a voxel structure, allowing fast rendering and camera selection for blending. PVMs are "sliced" in voxels, based on spatial location; the slice and voxel are called a *tile*. Deep Blending [Hedman et al. 2018] tries to address depth ambiguity using learned weights. Instead, we identify erroneous input and either blend with a low weight or discard it altogether.

Our initial observation is that the granularity of the original tiles is too large for an effective elimination strategy. As a consequence, we introduce a two-level hierarchy, providing finer granularity: We use a coarse grid of resolution $32^3$ to fetch visible voxels in a novel view, and a fine grid to slice the per-view meshes. The fine grid subdivides each coarse grid cell using a sub-sampling factor of 4 per dimension. The process of generating per-view meshes is summarized in Fig. 2b. For a detailed explanation of the steps involved, we refer to Section 4 from Hedman et al. [2016].

Our PVM rendering starts by rasterizing the global mesh, and marking all voxels intersected as visible. For each visible voxel at the fine scale we rank the input tiles to obtain a sorted list of the top-12 input tiles per voxel similar to [Hedman et al. 2016]. We then rasterize all 12 tiles in parallel using indirect layered rendering.

### 5.2 Clustered Depth Filtering

Most rasterized tiles tend to have a correct depth, but occasionally outliers occur. We address this problem building on the assumption that a majority of tile depths is in fact correct. The intuition is to separate the foreground from the background since the PVMs contain either correct previously unreconstructed geometry (Fig. 4b), or exhibit incorrect "over-reconstructed" geometry (Fig. 4c).

First, we find two depth estimates per pixel using $k$-means clustering - the potential foreground and background. We obtain a blended color $c_{i,k}$ per pixel $i$ and cluster $k$, by calculating ULR weights [Buehler et al. 2001] for the candidate colors within the cluster. These assign higher weight to views that are closer to the novel view in position and viewing direction.

We now need to determine how to weigh the two clusters for rendering. In areas with significant geometric reconstruction errors, clustering may be inconsistent across neighboring pixels. We address this by introducing a spatial filter over the depth candidates to keep depth in uncertain regions locally consistent. Intuitively, we encourage pixels to maintain the same depth in a local neighborhood, weighted by proximity. Given a novel view with two depth candidates $k$ per pixel $i$, we estimate weights $W_{i,k}$ taking the spatial neighborhood $\Omega_i$ around pixel $i$ into account:

$$W_{i,k} = \sum_{j \in \Omega_i} \sum_{k'} G_{i,j;\sigma} \exp(-\alpha(d_{j,k'} - d_{i,k})) \# N_{k'}. \tag{1}$$

Here, $\#N_k$ is the number of depth candidates for cluster $k$, and $d_{i,k}$ is the mean depth of pixel $i$ in cluster $k$. Further, $G_{i,j;\sigma}$ is a spatial Gaussian kernel with standard deviation $\sigma$. Intuitively, we count the number of cluster assignments modulated by a weight that depends on proximity both in image space and in depth, where the parameter $\alpha$ steers the depth weight falloff. The final color $c_i$ is obtained via:

$$\bar{c}_i = \frac{\sum_k W_{i,k}^{\gamma} c_{i,k}}{\sum_k W_{i,k}^{\gamma}}$$

The parameters $\sigma$, $\alpha$, and $\gamma$ are determined using a parameter sweep (Sec. 7). This filtering approach eliminates many of the incorrect PVM artifacts observed in PVM-H, and in some cases improves over Deep Blending [Hedman et al. 2018].

## 6 HYBRID RENDERING

Our hybrid rendering algorithm first renders the novel view using textured mesh (TM), since this is required by all methods (ULR and PVM). We then select which algorithm should be used for each pixel, building on the analysis presented in Sec. 3.3.

### 6.1 Hybrid Rendering Algorithm

Our hybrid approach follows the flow chart shown in Fig. 7. We first decide whether a pixel is outside the PVM grid bounds; in this case it is rendered with ULR. We then decide whether a pixel

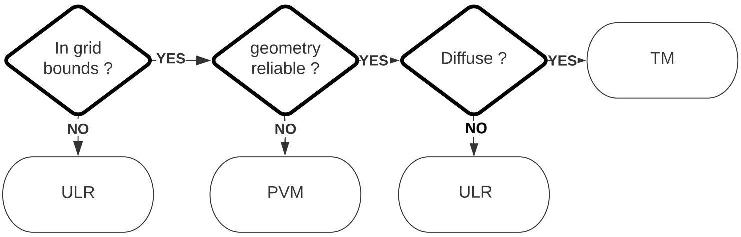


Fig. 7. Hybrid Algorithm Selection Flow Chart

can be rendered with the global mesh based on *geometric uncertainty*. Regions with high geometric uncertainty should be rendered with PVM, and others with the global mesh. For pixels that can be rendered with the global mesh, ULR should be preferred in view-dependent regions. To find these, we use a measure of *photometric uncertainty* for a given pixel in the novel view. Uncertain pixels are assumed to be view-dependent and thus are rendered with ULR while pixels with low uncertainty are assumed to be diffuse and are rendered with TM. We next provide details on how we compute geometric and photometric uncertainty.

*Geometric Uncertainty:* To measure the geometric uncertainty at a pixel we compare PVM depth with the global mesh. Specifically, any fragment which exceeds the absolute depth difference by $3cm$ is rendered with PVM. A threshold of $3cm$ works best empirically for all our datasets. Depth differences lower than this threshold imply the global depth is close enough to per-view depth and the region can be reliably rendered using the global mesh.

Since PVM tiles have multiple depths, we use the clustering step described above (Sec. 5.2) to determine which depth to use for this comparison. If $d_{fg}$ is the depth of the foreground cluster and $d_{bg}$ the background, we estimate depth using a voting strategy:

$$d = (\#(d_{bg}) >= \tau_{depth} * \#(d_{fg})) \ ? \ d_{bg} : d_{fg} \tag{2}$$

where # is the size of each cluster, and $\tau_{depth}$ the depth threshold.

*Photometric Uncertainty:* For the remaining pixels that can be rendered with the global mesh, our main criterion is whether they correspond to a diffuse region or not. We have already computed this information as part of the image harmonization method (Sec. 4.2). Specifically, we have a binary mask of view-dependent regions for each input image. We use these masks to obtain a texture map indicating view-dependent regions over the global mesh by computing the weighted average of the mask values accumulated per-vertex, which we call *view-dependent texture*. The texture map is computed in a pre-process after image harmonization. We perform per-pixel ULR for pixels determined as highly view-dependent (i.e., variance > $\tau_{var}$) and use TM for the rest.

Parameters $\tau_{var}$ and $\tau_{depth}$ are determined using a parameter sweep on a synthetic scene (Sec. 7).

## 6.2 Rendering

The rendering loop proceeds as follows:

```
1. Rasterize global TM, select voxels for tile selection, mask background pixels if out of grid bounds.
2. Rasterize PVMs: rank tiles and create up to 12 layers per pixel.
3. Clustering and Blending Shader: perform clustering step, create mask deciding PVM/TM or ULR.
4. **If** ULR **then**
       ULR blend shader: apply per-pixel ULR
5. **Else if** PVM **then**
```

Spatial Filter Shader: output filtered color

6. Blur the mask between algorithms and composite

We blur the mask with a kernel of size $\sigma_{blur}$, also determined by our parameter sweep (Sec. 7). The mask is visualized in Fig. 8, last row. We implement our hybrid rendering algorithm in our C++ framework using OpenGL/GLSL. The source code along with all pre-processing utilities can be found here: https://gitlab.inria.fr/sibr/projects/hybrid_ibr

## 7 RESULTS AND EVALUATION

We provide comparisons of our rendering method with baseline algorithms as well as current state-of-the-art neural rendering methods, using published codes (see supplemental for details). We also perform baseline comparison and ablation studies to show the gain in quality achieved with harmonization as well as PVM-H rendering. We provide a supplemental video, and a supplemental webpage which provides all comparisons available.

*Parameter evaluation.* To select values for parameters for the filtering and hybrid rendering steps we selected 22 novel views in a synthetic scene. We computed the ground truth using path tracing and performed a sweep of acceptable values selecting the average of the values that gave the lowest error. The values are $\sigma = 3$, $\alpha = 7$, $\gamma = 7$ for the filter (Sec. 5.2), $\tau_{depth} = 0.83$, $\tau_{var} = 0.82$ (Sec. 6.1), and finally $\sigma_{blur} = 4$ (Sec. 6.2).

*Datasets:* We present our results on 6 datasets from [Hedman et al. 2016]: Dr Johnson, Hugo, Library, Playroom, Creepy Attic and Ponche, 1 dataset from [Knapitsch et al. 2017; Riegler and Koltun 2020]: Train, 1 new indoor dataset that we captured: Salon, and 1 synthetic dataset: Synthetic Attic. We present 6 indoor and 3 outdoor scenes, showing the versatility of our method.

### 7.1 Rendering Results & Comparisons

Fig. 8 shows the comparison between our hybrid rendering and the following algorithms (see supplemental webpage and video):

- Textured Mesh(TM): Global mesh textured with [Jancosek and Pajdla 2011; Reality 2018].
- Per-pixel ULR as described in Sec. 3.3.
- Inside Out: SfM+MVS variant of [Hedman et al. 2016] used as a baseline in Hedman et al. [2018].
- Deep Blending: Method of Hedman et al. [2018] with learned blend weights.
- Free View Synthesis (FVS): learning-based method [Riegler and Koltun 2020].

We also compare with Neural Radiance Fields (NeRF), a neural scene representation method [Mildenhall et al. 2020] for the Salon scene. Directly using the full set of images did not give good results, so we present a best-effort attempt by selecting a subset of images that gave the best results.

Our method corrects erroneous geometry compared to TM while maintaining sharp details in diffuse regions. Compared to ULR we often reduce ghosting and blurring artifacts as well as prominent harmonization artifacts due to blending. Our two-level PVM with filtering improves visual quality compared to Inside Out. Compared to neural methods of Deep Blending and FVS, our method provides more stable and often sharper results, removing most of the visible color artifacts encountered due to neural rendering. Note however that FVS was trained on the very different Tanks and Temples dataset [Knapitsch et al. 2017].

In terms of pure rendering quality, Deep Blending is arguably better in many cases, but at more than 2.5 times slower frame rate (see below). Our approach provides good balance in the quality/speed tradeoff, and often is better at maintaining high frequencies.

As noted in Sec. 3.2, PVMs can create visible artifacts in extreme cases which can be hard to solve even using neural networks. Our two-level PVMs with filtering provide significant improvement in

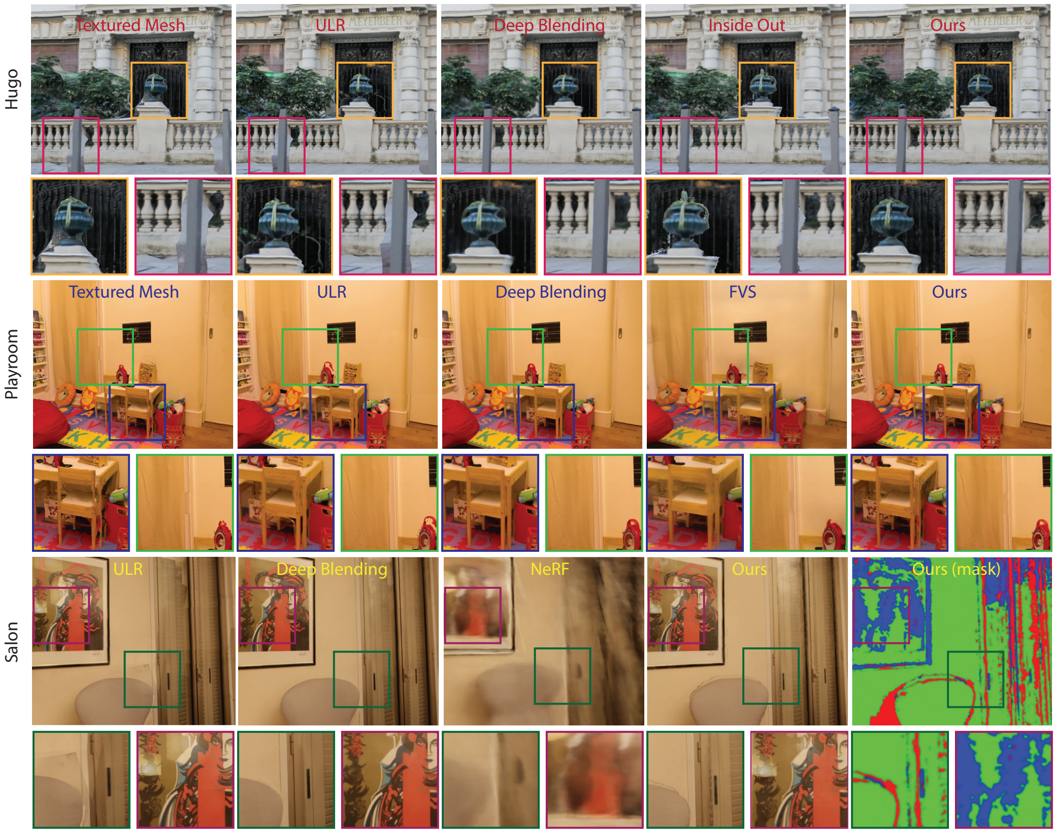


Fig. 8. Comparison of novel view rendering results in Hugo, Playroom, and Salon. We show comparison against different classical IBR (Textured Mesh, Unstructured Lumigraph [Buehler et al. 2001], Inside Out [Hedman et al. 2016] ) and neural rendering (Deep Blending [Hedman et al. 2018], NeRF [Mildenhall et al. 2020], Free-view Synthesis [Riegler and Koltun 2020]) approaches. Our method achieves better quality by removing many common IBR artifacts such as color seams (bottom row, dark green inset), view-dependent effects preservation (bottom row, magenta inset), missing geometry (middle row), over-reconstructed geometry (top row, pink inset), texture sharpness and ghosting artifacts (top row, orange inset).

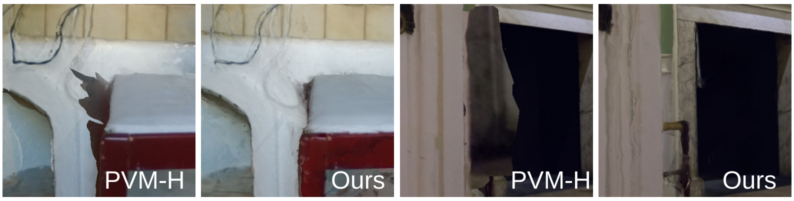


Fig. 9. Comparing our solution with PVM-H in Library (left) and Dr Johnson (right).

such hard cases. Fig. 9 compares our PVM blending algorithm with PVM-H method of [Hedman et al. 2018]. Our algorithm is able to get rid of most such cases and provides much cleaner and sharper edges.

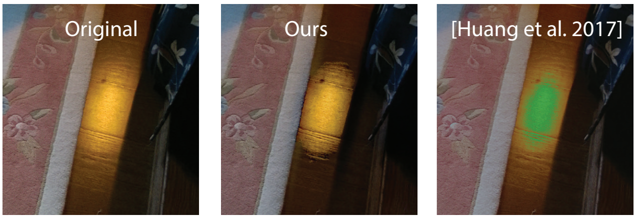


Fig. 10. Evaluation of Image Harmonization on Library.

*Image Harmonization: Baseline comparison.* As discussed in Sec. 2, many methods perform color correction to avoid visible seams in the textures. We implement the optimization-based solution of Huang et al. [2017] for color correction as a baseline comparison to our method. The method recovers a single parametric curve per-channel per-input image and scales the intensity of every pixel in the image based on its recovered curve. While this helps achieve harmonized color in most diffuse regions, the regions which exhibit specular highlights result in color artifacts, see Fig. 10.

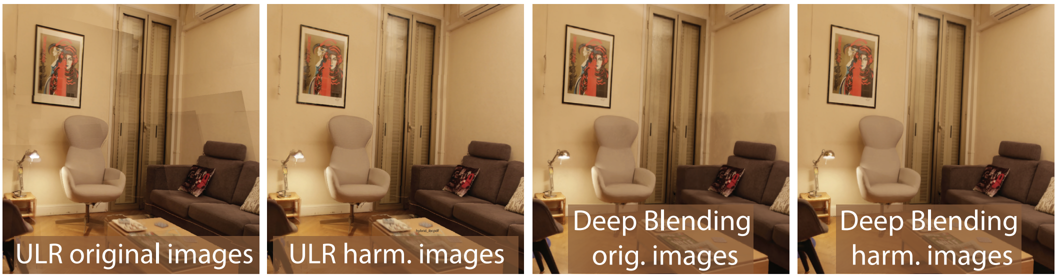


Fig. 11. Ablation results of Image Harmonization on Salon.

*Harmonization for ULR and Deep Blending.* Harmonization can be useful for other algorithms. We show results using our harmonized and original images for ULR and Deep Blending in Fig. 11. Basic Deep Blending tries to harmonize the images by learning the blend weights such that they result in fewer seams but fails to completely remove it, while our image harmonization removes the seams completely.

### 7.2 Performance

We ran tests on a desktop machine, with Intel Xeon Gold 5218 2.30GHz Processor and Quadro RTX 5000 GPU and a laptop with an Intel Core i9-8950HK 2.90GHz Processor and NVIDIA GeForce RTX 2080 Max-Q GPU. All timings are on the desktop unless stated otherwise.

Table 2. Average cost of each step for our rendering algorithm for Ponche & Library.

| Steps | Ponche | Library | Steps | Ponche | Library |
|---|---|---|---|---|---|
| 1. Voxel lookup | 3.06 *ms* | 3.11 *ms* | 5. Spatial Filtering | 1.99 *ms* | 1.29 *ms* |
| 2. Tile Sort | 9.83 *ms* | 4.29 *ms* | 6. ULR Blend | 0.94 *ms* | 1.35 *ms* |
| 3. Per-View Depth Pass | 5.30 *ms* | 4.00 *ms* | 7. Masking | 2.01 *ms* | 1.34 *ms* |
| 4. Cluster & Blend | 5.42 *ms* | 5.07 *ms* | **Total** | **28.55 ms** | **20.45 ms** |

*Runtime Statistics:* Table 2 provides a runtime breakdown of different steps of our algorithm. We provide additional runtime comparisons on the Desktop and Laptop machines and pre-processing time statistics in our supplemental material. Pre-processing for the harmonization takes between 15 min to 2-3 hours, depending on the number of images in the dataset and their resolution; the MRF step is the most expensive.

### 7.3 Quantitative Evaluation

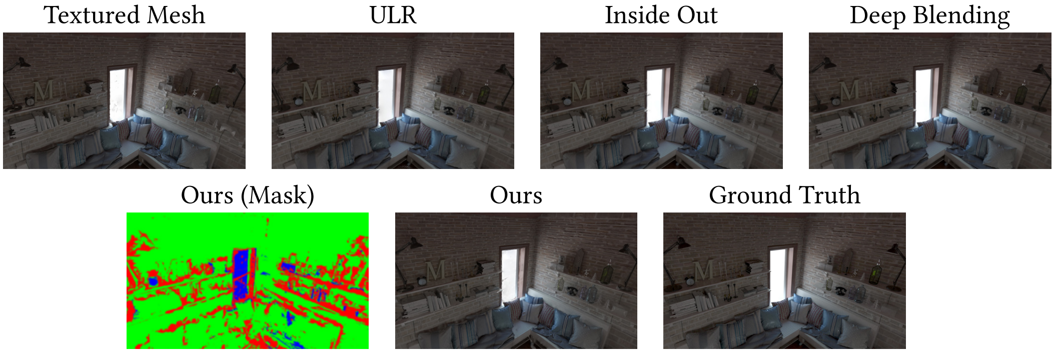


Fig. 12. Comparisons with path-traced ground truth rendering for each method on Synthetic Attic dataset. Top Row - Baselines: Textured Mesh [Reality 2018], Unstructured Lumigraph (ULR) [Buehler et al. 2001]; Previous methods: Inside Out [Hedman et al. 2016], Deep Blending [Hedman et al. 2018]; Bottom row - Ours per-pixel algorithm mask (Red: Per-view Mesh (PVM); Blue: Unstructured Lumigraph (ULR); Green: Textured Mesh (TM)); ours and the path-traced ground truth.

For completeness, we present quantitative evaluations on a synthetic and a real scene. As discussed below, current quantitative metrics are not very successful at identifying the kind of visual artifacts created by free-viewpoint IBR algorithms.

The Synthetic Attic dataset is shown in Fig. 12. We show path-traced ground truth and comparisons with our method as well as previous approaches. We also show the algorithm selection mask for the given input view which indicates the algorithms used for each pixel. Notice how our hybrid algorithm is able to identify the occlusion edges as the regions where most geometric errors occur.

We performed a quantitative analysis on the real-world dataset Ponche, by holding out 10% of the input views for rendering. For ULR, all test images were left out. For Inside Out, Deep Blending, and our method the corresponding held-out per-view meshes are not used. However, it should be noted that during the global mesh reconstruction and texturing the held-out images were used.

We also provide a quantitative comparison with path-traced ground truth images on the Synthetic Attic scene in Table 3. All test images were unseen during the entire pre-processing (including camera calibration and reconstruction) for the synthetic dataset.

Table 3 reports quantitative error of our method compared to previous IBR algorithms. We show structural dissimilarity (DSSIM) [Loza et al. 2006], E-LPIPS perceptual metric [Kettunen et al. 2019] and Peak Signal-to-Noise Ratio (PSNR).

The quantitative results indicate that, for the real scene Ponche, we have achieved our goal of having equivalent, if not – slightly – better quality than previous methods, but with faster compute time. While we perform best with respect to the E-LPIPS and PSNR metrics in the Ponche real-world dataset, Deep Blending [Hedman et al. 2018] performs best across the 3 metrics in the synthetic scene. According to DSSIM, ULR performs better than both our and Deep Blending in the real-world

Table 3. Quantitative comparison of different algorithms over images rendered in held-out fashion for real dataset Ponche while using ground truth path-traced images of novel views for synthetic dataset Synthetic Attic.

| IBR Algorithms | Ponche | | | Synthetic Attic | | |
|---|---|---|---|---|---|---|
| | E-LPIPS ↓ | DSSIM ↓ | PSNR ↑ | E-LPIPS ↓ | DSSIM ↓ | PSNR ↑ |
| TM | 0.0296 | 0.150 | 21.26 | 0.0225 | 0.160 | 25.29 |
| ULR | 0.0243 | **0.123** | 21.59 | 0.0234 | 0.147 | 26.38 |
| InsideOut | 0.0290 | 0.129 | 21.78 | 0.0221 | 0.143 | 27.90 |
| Deep Blending | 0.0279 | 0.132 | 22.04 | **0.0203** | **0.137** | **28.24** |
| Ours | **0.0239** | 0.138 | **22.06** | 0.0218 | 0.154 | 26.85 |

dataset; This is in contrast to what we perceive from the video sequences (please see supplemental) as ULR tends to have numerous color seams, ghosting, and blurring artifacts compared to all other methods. In addition, the differences between algorithms are generally very small, again in contrast to visual evidence. Given these inconsistencies and unreliability, quantitative comparisons are of questionable utility in this context, underlining the need for further research on perceptual metrics specific to different types of IBR artifacts – both at global and local scales.

## 8 LIMITATIONS AND CONCLUSIONS

We analyzed common IBR artifacts and used the insights obtained to propose a hybrid rendering algorithm that runs at interactive rates, with a good balance between speed and quality. Specifically, we have shown how to *(i)* perform effective image harmonization while preserving view-dependent effects, *(ii)* utilize and fuse per-view information to correct for geometric reconstruction errors, and *(iii)* combine the strengths of three different algorithms in a novel hybrid algorithm.

*Limitations.* While we show that we indeed can leverage the strengths of the individual algorithms, residual artifacts remain due to their weaknesses. Our hybrid approach often chooses TM, that can result in deformed lines (see Playroom sequence, 0:16) or over-reconstructed geometry on object edges (see Salon sequence, 0:12 and supplemental video 0:05). ULR can be prone to ghosting artifacts when the silhouette of objects is not reconstructed accurately; our filtered PVMs occasionally exhibit (semi-)transparent foreground regions, whenever our cluster assignment assumptions are violated. Further, we do not explicitly enforce temporal consistency in our PVM module. Extending our depth filtering approach to the temporal domain could be a direction for future work, but would require a more elaborate treatment of fuzzy depth.

*Future Work.* While we focused on existing, interactive rendering algorithms, we believe that our hybrid design in principle allows the exploration of other combinations of existing or future algorithms. When neural rendering becomes sufficiently fast, it could provide missing high-quality details for specific image regions our framework is able to identify through simple yet effective heuristics.

## ACKNOWLEDGMENTS

The authors would like to thank Peter Hedman and Julien Philip for initial discussions and tests. The research was funded by the ERC Advanced grant FUNGRAPH No 788065 (https://project.inria.fr/fungraph/). The authors are grateful to Inria Sophia Antipolis - Méditerranée "Nef" computation cluster and the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

## REFERENCES

Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. 2004. Interactive Digital Photomontage. In *ACM SIGGRAPH 2004 Papers* (Los Angeles, California) *(SIGGRAPH '04)*. Association for Computing Machinery, New York, NY, USA, 294–302. https://doi.org/10.1145/1186562.1015718

Sebastien Bonopera, Peter Hedman, Jerome Esnault, Siddhant Prakash, Simon Rodriguez, Theo Thonat, Mehdi Benadel, Gaurav Chaurasia, Julien Philip, and George Drettakis. 2020. sibr: A System for Image Based Rendering. https://sibr.gitlabpages.inria.fr/

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 (SIGGRAPH '01)*. Association for Computing Machinery, New York, NY, USA, 425–432. https://doi.org/10.1145/383259.383309

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. https://doi.org/10.1109/3DV.2015.59

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), 30.

Shenchang Eric Chen. 1995. QuickTime VR: An Image-Based Approach to Virtual Environment Navigation. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. Association for Computing Machinery, New York, NY, USA, 29–38. https://doi.org/10.1145/218380.218395

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

Paul Debevec, Yizhou Yu, and George Borshukov. 1998. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. In *Rendering Techniques '98*, George Drettakis and Nelson Max (Eds.). Springer Vienna, Vienna, 105–116.

M. Eisemann, B. De Decker, M. Magnor, P. Bekaert, E. De Aguiar, N. Ahmed, C. Theobalt, and A. Sellent. 2008. Floating Textures. *Computer Graphics Forum* 27, 2 (2008), 409–418. https://doi.org/10.1111/j.1467-8659.2008.01138.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2008.01138.x

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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

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. https://doi.org/10.1145/1833349.1778832

M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. 2007. Multi-View Stereo for Community Photo Collections. In *2007 IEEE 11th International Conference on Computer Vision*. 1–8. https://doi.org/10.1109/ICCV.2007.4408933

Daniel B Goldman. 2010. Vignette and exposure calibration and compensation. *IEEE transactions on pattern analysis and machine intelligence* 32, 12 (2010), 2276–2288.

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 (SIGGRAPH '96)*. Association for Computing Machinery, New York, NY, USA, 43–54. https://doi.org/10.1145/237170.237200

Yoav HaCohen, Eli Shechtman, Dan B Goldman, and Dani Lischinski. 2011. Non-rigid dense correspondence with applications for image enhancement. *ACM transactions on graphics (TOG)* 30, 4 (2011), 1–10.

Yoav HaCohen, Eli Shechtman, Dan B Goldman, and Dani Lischinski. 2013. Optimizing color consistency in photo collections. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–10.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. 2018. Deep Blending for Free-Viewpoint Image-Based Rendering. *ACM Trans. Graph.* 37, 6, Article 257 (Dec. 2018), 15 pages. https://doi.org/10.1145/3272127.3275084

Peter Hedman, Tobias Ritschel, George Drettakis, and Gabriel Brostow. 2016. Scalable inside-out image-based rendering. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 231.

Jingwei Huang, Angela Dai, Leonidas J Guibas, and Matthias Nießner. 2017. 3Dlite: towards commodity 3D scanning for content creation. *ACM Trans. Graph.* 36, 6 (2017), 203–1.

M. Jancosek and T. Pajdla. 2011. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR 2011*. 3121–3128. https://doi.org/10.1109/CVPR.2011.5995693

Markus Kettunen, Erik Härkönen, and Jaakko Lehtinen. 2019. E-LPIPS: Robust Perceptual Image Similarity via Random Transformation Ensembles. arXiv:1906.03973 [cs.CV]

Seon Joo Kim and Marc Pollefeys. 2008. Robust radiometric calibration and vignetting correction. *IEEE transactions on pattern analysis and machine intelligence* 30, 4 (2008), 562–576.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and Temples: Benchmarking Large-Scale Scene Reconstruction. *ACM Trans. Graph.* 36, 4, Article 78 (July 2017), 13 pages. https://doi.org/10.1145/3072959.3073599

Vladlen Koltun. 2020. Towards Photorealism. (2020). https://youtu.be/Rd0nBO6--bM 42nd German Conference on Pattern Recognition (DAGM GCPR 2020), the 25th International Symposium on Vision, Modeling and Visualization (VMV 2020) and the 10th Eurographics Workshop on Visual Computing for Biology and Medicine (VCBM 2020).

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), 199.

Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. 2003. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 277–286.

V. Lempitsky and D. Ivanov. 2007. Seamless Mosaicing of Image-Based Texture Maps. In *2007 IEEE Conference on Computer Vision and Pattern Recognition.* 1–6. https://doi.org/10.1109/CVPR.2007.383078

Marc Levoy and Pat Hanrahan. 1996. Light Field Rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '96).* Association for Computing Machinery, New York, NY, USA, 31–42. https://doi.org/10.1145/237170.237199

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. 2019. Neural Volumes: Learning Dynamic Renderable Volumes from Images. *ACM Trans. Graph.* 38, 4, Article 65 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3323020

A. Loza, L. Mihaylova, N. Canagarajah, and D. Bull. 2006. Structural Similarity-Based Object Tracking in Video Sequences. In *2006 9th International Conference on Information Fusion.* 1–6. https://doi.org/10.1109/ICIF.2006.301574

Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. 2021. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. arXiv:2008.02268 [cs.CV]

Leonard McMillan and Gary Bishop. 1995. Plenoptic Modeling: An Image-Based Rendering System. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95).* Association for Computing Machinery, New York, NY, USA, 39–46. https://doi.org/10.1145/218380.218398

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/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

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 Trans. Graph.* 38, 4, Article 29 (July 2019), 14 pages. https://doi.org/10.1145/3306346.3322980

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. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 405–421.

Pierre Moulon, Pascal Monasse, Romuald Perrot, and Renaud Marlet. 2017. OpenMVG: Open Multiple View Geometry. In *Reproducible Research in Pattern Recognition*, Bertrand Kerautret, Miguel Colom, and Pascal Monasse (Eds.). Springer International Publishing, Cham, 60–74.

Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson Image Editing. In *ACM SIGGRAPH 2003 Papers.* Association for Computing Machinery, New York, NY, USA, 313–318. https://doi.org/10.1145/1201775.882269

Capturing Reality. 2018. RealityCapture reconstruction software. https://www.capturingreality.com/Product.

Gernot Riegler and Vladlen Koltun. 2020. Free View Synthesis. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 623–640.

Simon Rodriguez, Siddhant Prakash, Peter Hedman, and George Drettakis. 2020. Image-Based Rendering of Cars Using Semantic Labels and Approximate Reflection Flow. *Proc. ACM Comput. Graph. Interact. Tech.* 3, 1, Article 6 (April 2020), 17 pages. https://doi.org/10.1145/3384535

Johannes L. Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 501–518.

Heung-Yeung Shum, Shing-Chow Chan, and Sing Bing Kang. 2008. *Image-based rendering.* Springer Science & Business Media.

Sudipta N. Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. 2012. Image-based rendering for scenes with reflections. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 100–1.

Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Niessner, Gordon Wetzstein, and Michael Zollhofer. 2019. DeepVoxels: Learning Persistent 3D Feature Embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

Noah Snavely, Steven M. Seitz, and Richard Szeliski. 2006. Photo Tourism: Exploring Photo Collections in 3D. In *ACM SIGGRAPH 2006 Papers* (Boston, Massachusetts) *(SIGGRAPH '06)*. Association for Computing Machinery, New York, NY, USA, 835–846. https://doi.org/10.1145/1179352.1141964

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/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

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. Zollhöfer. 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. 2019. Deferred Neural Rendering: Image Synthesis Using Neural Textures. *ACM Trans. Graph.* 38, 4, Article 66 (July 2019), 12 pages. https://doi.org/10.1145/3306346.3323035

Justus Thies, Michael Zollhöfer, Christian Theobalt, Marc Stamminger, and Matthias Nießner. 2018. IGNOR: Image-guided Neural Object Rendering. *CoRR* abs/1811.10720 (2018). arXiv:1811.10720 http://arxiv.org/abs/1811.10720

Michael Waechter, Nils Moehrle, and Michael Goesele. 2014. Let There Be Color! Large-Scale Texturing of 3D Reconstructions. In *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars (Eds.). Springer International Publishing, Cham, 836–850.

Edward Zhang, Michael F Cohen, and Brian Curless. 2016. Emptying, refurnishing, and relighting indoor spaces. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 1–14.

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:1805.09817 [cs.CV]