Deep Blending for Free-Viewpoint Image-Based Rendering

PETER HEDMAN, University College London
JULIEN PHILIP, Inria, Université Côte d'Azur
TRUE PRICE, UNC Chapel Hill
JAN-MICHAEL FRAHM, UNC Chapel Hill
GEORGE DRETTAKIS, Inria, Université Côte d'Azur
GABRIEL BROSTOW, University College London / Niantic

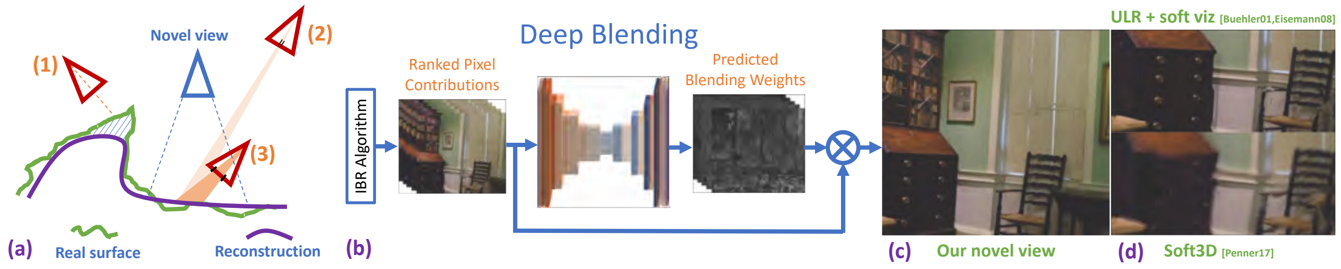

Fig. 1. Image-based rendering *blends* contributions from different input images to synthesize a novel view. **(a)** This blending operation is complex for a variety of reasons. For example, incorrect visibility (*e.g.*, for the non-reconstructed green surface in input view (1)) may result in wrong projections of an image into the novel view. Blending must also account for, *e.g.*, differences in projected resolution ((2) and (3)). Previous methods have used hand-crafted heuristics to overcome these issues. **(b)** Our method generates a set of ranked contributions (mosaics) from the input images and uses *predicted blending weights* from a CNN to perform *deep blending* and synthesize **(c)** novel views. Our solution significantly reduces visible artifacts compared to **(d)** previous methods.

Free-viewpoint image-based rendering (IBR) is a standing challenge. IBR methods combine warped versions of input photos to synthesize a novel view. The image quality of this combination is directly affected by geometric inaccuracies of multi-view stereo (MVS) reconstruction and by view- and image-dependent effects that produce artifacts when contributions from different input views are blended. We present a new deep learning approach to blending for IBR, in which we use held-out real image data to learn blending weights to combine input photo contributions. Our Deep Blending method requires us to address several challenges to achieve our goal of interactive free-viewpoint IBR navigation. We first need to provide sufficiently accurate geometry so the Convolutional Neural Network (CNN) can succeed in finding correct blending weights. We do this by combining two different MVS reconstructions with complementary accuracy vs. completeness tradeoffs. To tightly integrate learning in an interactive IBR system, we need to adapt our rendering algorithm to produce a fixed number of input layers that can then be blended by the CNN. We generate training data with a variety of captured scenes, using each input photo as ground truth in a held-out approach. We also design the network architecture and the training loss to provide high quality novel view synthesis, while reducing temporal flickering artifacts. Our results demonstrate free-viewpoint IBR in a wide variety of scenes, clearly surpassing previous methods in visual quality, especially when moving far from the input cameras.

Authors' addresses: Peter Hedman, University College London; Julien Philip, Inria, Université Côte d'Azur; True Price, UNC Chapel Hill; Jan-Michael Frahm, UNC Chapel Hill; George Drettakis, Inria, Université Côte d'Azur; Gabriel Brostow, University College London / Niantic.

0730-0301/2018/11-ART257 $15.00
https://doi.org/10.1145/3272127.3275084

CCS Concepts: • **Computing methodologies → Image-based rendering**; *Neural networks*; Reconstruction;

Additional Key Words and Phrases: Image-based rendering, Free-viewpoint, Deep learning

ACM Reference Format:
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 (November 2018), 15 pages. https://doi.org/10.1145/3272127.3275084

1 INTRODUCTION

There is increasing demand for *realistic* 3D content that is easy to capture, and that can be used for *free-viewpoint, interactive* navigation. Image-Based Rendering (IBR) can provide such realistic interactive imagery, but current methods [Hedman et al. 2016; Ortiz-Cayon et al. 2015; Penner and Zhang 2017], still suffer from many visible artifacts, especially when moving far from the input photos. Novel views are synthesized in IBR by combining warped pixels from input photos; output quality depends on the computation of visibility in the presence of *inaccurate geometry* and on the *blending* method. With only few exceptions (*e.g.*, [Li and Li 2008]), previous solutions use heuristic blending to handle geometric inaccuracies, and to correct image seams and ghosting due to view-specific differences in the combined images. Blending needs to correct for artifacts due to incorrect *occlusion edges*, *visible seams* due to *texture stretch/misalignment*, and *lack of color harmonization*, as well as view-dependent effects from highlights, different exposures, and unsuitable camera selection. These complex, often contradictory requirements have led prior work to develop case-specific, hand-crafted heuristics that always fail for some configurations.

Our main insight is that a data-driven solution is currently a better strategy to effectively satisfy these challenging requirements. We introduce a *deep blending* algorithm, leveraging convolutional neural networks (CNNs) that learn *blending weights* that will most reasonably approximate real imagery for novel view synthesis (Fig. 1).

There are several challenges to overcome in order for our *deep blending* approach to be realized. To maintain interactive framerates during rendering, we need to tightly integrate the feed-forward evaluation of the CNN in our rendering loop. To do this, we choose a network architecture with a fixed number of inputs. We generate inputs to the network by warping the input photos to the novel view and creating "mosaics" of ranked contributions from the different input views. We then choose the fixed set of best candidates for each pixel, that are subsequently input to the network for blending using learned weights. To provide sufficient training data to learn these blending weights, we leverage a dataset of 2630 input photos from 19 distinct scenes. Apart from the validation set, each photo is used as a ground truth target that the network strives to reconstruct, held-out in a round-robin fashion. Finally, to overcome alignment problems and flickering, we use a perceptually-motivated loss together with an auxiliary temporal consistency term. In IBR, each *viewpoint* in a scene is a distinct sample. We seek to generalize to novel viewpoints, never seen during training, allowing free-viewpoint rendering.

For our deep blending method to succeed, the underlying geometry we use to create the mosaics should be faithful in several regards. While Multi-View Stereo (MVS) algorithms now provide impressive 3D reconstructions, they cannot achieve the quality required for realistic IBR. A first set of MVS algorithms (*e.g.*, [Schönberger et al. 2016]) provides high-quality depth maps containing fine details, while a second set provides better globally consistent geometry, *i.e.*, they estimate a smooth connected surface, even in hard-to-reconstruct (*e.g.*, textureless) regions (*e.g.*, [Jancosek and Pajdla 2011; RealityCapture 2016]). The meshes provided by either method still result in artifacts if used directly by an IBR algorithm (*e.g.*, Buehler et al. [2001]). State-of-the-art IBR algorithms try to overcome this difficulty using *per-view geometric structures*, which may not be globally consistent but achieve good visual quality [Chaurasia et al. 2013; Hedman et al. 2016] if blended correctly.

Nevertheless, they still suffer from *geometric inconsistencies*, *outliers*, and *inaccurate occlusion edges*. We develop a new per-view geometry refinement method that overcomes the shortcomings of each of the two sets of reconstruction methods. We do this by using the information available in the one set of methods as a prior to compensate for information lacking in the other. Our solution, which is separate from our neural network, generates per-view geometry that is of sufficient quality to allow deep blending to succeed.

Overall, we present two main contributions:

- A deep convolutional neural network solution to the IBR blending problem.
- An improved per-view geometry refinement, and geometry-aware mesh simplification, that together produce high-quality source geometries that facilitate deep blending for IBR.

Our solution provides realistic image-based rendering across the variety of scenes attempted by our community so far, and it gracefully degrades in quality when the geometric reconstruction fails completely or exposure differences are too pronounced. For a majority of the scenes tested – both outdoors and indoors – our method achieves excellent quality for free-viewpoint navigation, while also being capable of achieving interactive frame-rates.

2 MOTIVATION AND PRIOR WORK

There are several significant artifacts that frequently occur in IBR, illustrated in Fig. 2. The first set are related to inaccurate geometry and occlusion edges, and result in ghosting, "floating geometry," and the incorrect projection of background onto foreground or vice versa. The second are "visible seam" artifacts that are related to texture misalignment, different resolutions, and shifts in colors for the same (or neighboring) content across input images. The third are view-dependent effects such as highlights. For highlights, we wish to preserve a smooth version where possible, and discard insufficiently sampled highlights. Finally, interactive IBR requires that output imagery be temporally stable (see supplemental video for an illustration). Here, we review prior work and discuss how existing methods have attempted to address these artifacts.

As there is a vast body of prior work on 3D geometry reconstruction, we review only the most recent works related to our approach. We also discuss related IBR algorithms, especially for blending, and recent work on machine learning techniques for blending and IBR.

2.1 Geometry Reconstruction and Occlusion Edges

Multi-view stereo algorithms (e.g., [Furukawa and Ponce 2010; Goesele et al. 2007; Jancosek and Pajdla 2011]) perform automatic 3D geometry reconstruction from unstructured photo datasets with diverse viewpoints. Approaches based on Delaunay tetrahedralization (e.g., [Jancosek and Pajdla 2011; Labatut et al. 2007; RealityCapture 2016]) are able to generate impressive 3D models from photos, even in the presence of traditionally hard cases such as large textureless regions. Similarly, Ummenhofer and Brox [2017] show that it is possible generate dense meshes from noisy multi-view stereo point clouds using a regularized signed distance field.

In parallel, other methods improve the quality of individual depth maps by discretizing the scene depths and enforcing smoothness in image-space [Hirschmuller 2006; Scharstein and Szeliski 2002]. These methods are able to produce edge-aligned geometry, which is smooth for textureless regions, albeit with visible staircasing artifacts due to discretization. Recently, Patch-Match based algorithms such as COLMAP [Schönberger et al. 2016] have been demonstrated to create the most accurate geometry in benchmark tests [Knapitsch et al. 2017; Schöps et al. 2017]. By constraining the viewpoint, layered depth panoramas offer usable geometry, but cannot represent view-dependent effects and suffer from artifacts when moving away from the point of capture [Hedman et al. 2017; Zheng et al. 2007].

These methods allow easy capture since they only require photos as input, but the accuracy of reconstructed geometry is generally insufficient to ensure realistic IBR. In contrast, our accurate per-view geometry, along with our deep blending approach, attains realistic image synthesis while maintaining the advantage of easy capture.

2.2 Image-Based Rendering and Blending

IBR can generate realistic 3D content, using only photos as input. Early approaches [Gortler et al. 1996; Levoy and Hanrahan 1996]

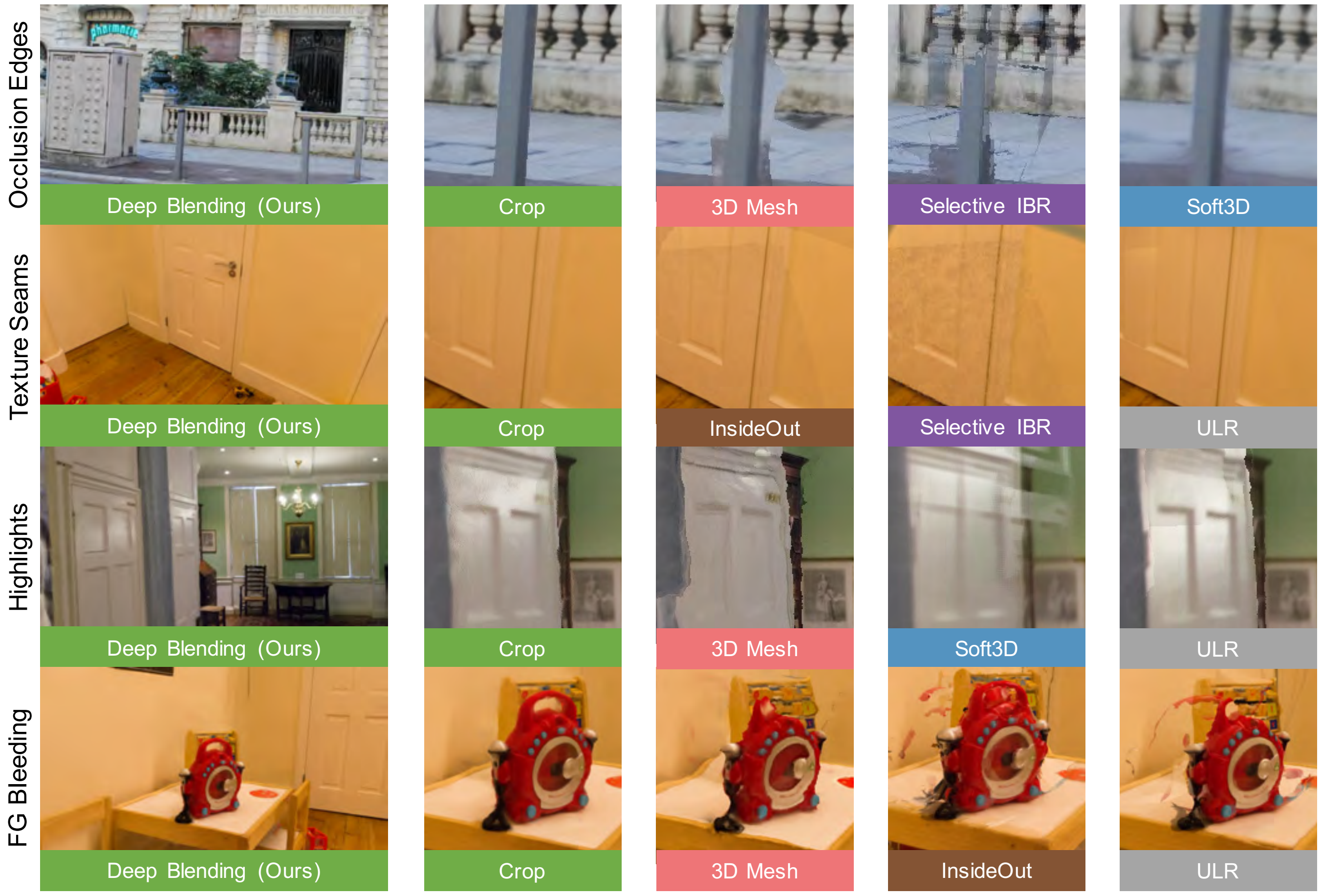

Fig. 2. Example artifacts, one per row, commonly faced by IBR methods. Each row shows an example frame produced by our method, a cropped region of this image, and the same crop for three other IBR methods. Five methods are compared: Textured Mesh-based rendering, Selective IBR [Ortiz-Cayon et al. 2015], ULR [Buehler et al. 2001] with soft visibility [Eisemann et al. 2008], InsideOut [Hedman et al. 2016], and Soft3D [Penner and Zhang 2017]. Only some methods shown per row; videos of all methods are available in the supplemental material.) **Row 1:** Occlusion edge artifacts. Our CNN-based solution can correct for imprecise object boundaries without overblurring. **Row 2:** Seams due to color harmonization errors. Deep Blending correctly harmonizes the result, whereas the other methods shown have moderate to subtle color edge artifacts. **Row 3:** Highlights from illumination. The textured mesh fails to capture this effect at all; our network blends seamlessly compared to other approaches. **Row 4:** Foreground bleeding into background. Incorrect object geometry and poor per-view depth maps can lead to severe ghosting artifacts in existing methods. Our network learns to output blend weights that overcome this behavior.

required complex capture setups, making them impractical for widespread use. Recently, commercial systems [Anderson et al. 2016][1][2] are able to deliver high-quality results by capturing data with a multi-camera rig and constraining the virtual viewpoint. Methods that can use an unstructured set of photos [Buehler et al. 2001; Heigl et al. 1999] facilitate our goal of ease-of-capture.

The Unstructured Lumigraph [Buehler et al. 2001] uses a globally consistent geometric *proxy* and blends the reprojected input images in the novel view. The importance of maintaining accurate results at *occlusion edges* was identified early, using soft visibility [Pulli et al. 1997; Zheng et al. 2009] or superpixels and alpha matting [Zitnick et al. 2004] to improve the blending. Assumptions about source image positioning have also been considered. For example, floating textures [Eisemann et al. 2008] use optical flow in short-baseline video sequences to correct for inaccurate geometry.

Davis et al. [2012] performed bilinear blending of viewpoints located approximately on the surface of a sphere around a capture subject; this assumes the source viewpoints vary smoothly along a 2D manifold, which is more restrictive than the general capture scenarios we target. Among other special capture configurations, Arikan et al. [2014; 2016] present a fast rendering and seam-hiding method for the case of high-quality diffuse scenes imaged using laser scanners.

More recently, *per-view representations* have been used to maintain accurate image edges during rendering. These include superpixels [Chaurasia et al. 2013; Ortiz-Cayon et al. 2015] or per-view meshes [Hedman et al. 2016]. In these solutions, different blending strategies have been used. We review these ([Buehler et al. 2001;

[1]https://facebook360.fb.com/facebook-surround-360/
[2] https://www.blog.google/products/google-vr/experimenting-light-fields/

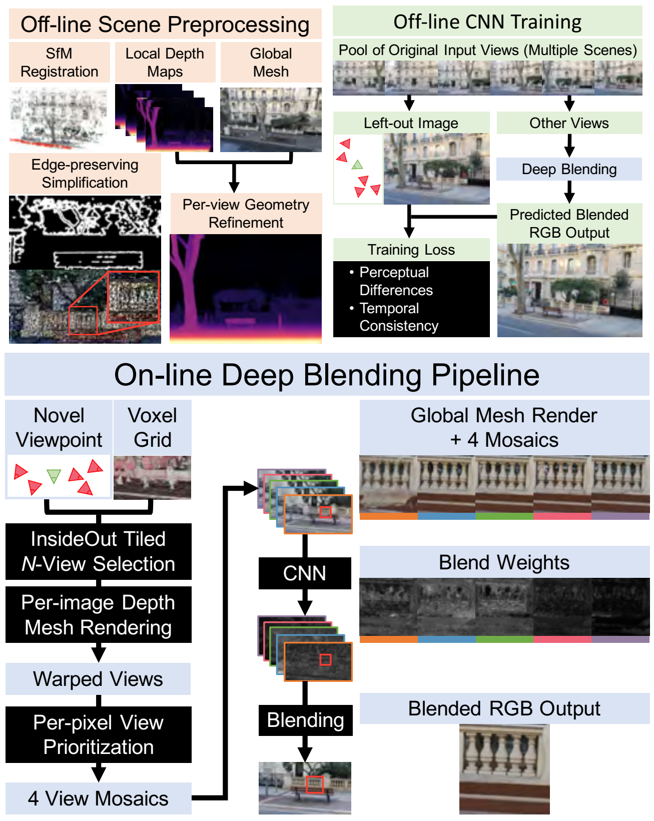
Off-line Scene Preprocessing
SfM Registration
Local Depth Maps
Global Mesh
Edge-preserving Simplification
Per-view Geometry Refinement
Off-line CNN Training
Pool of Original Input Views (Multiple Scenes)
Left-out Image
Other Views
Deep Blending
Predicted Blended RGB Output
Training Loss
• Perceptual Differences
• Temporal Consistency
On-line Deep Blending Pipeline
Novel Viewpoint
Voxel Grid
InsideOut Tiled N-View Selection
Per-image Depth Mesh Rendering
Warped Views
Per-pixel View Prioritization
4 View Mosaics
CNN
Blending
Global Mesh Render + 4 Mosaics
Blend Weights
Blended RGB Output

Fig. 3. Overview of our approach. **Top left:** Scene preprocessing entails constructing 1) 6-DoF image positions using SfM, 2) per-image depth maps using MVS, 3) a global mesh, 4) mesh-refined depth maps, and 5) simplified per-view meshes respecting occlusion edges. **Top right:** Training uses a perceptual loss to compare our pipeline's output (bottom) for a known viewpoint to the real image; a temporal consistency term penalizes differences after viewpoint perturbation. **Bottom:** Deep Blending outputs color images for novel scene viewpoints. At each output pixel, InsideOut [Hedman et al. 2016] ranks pixels in the dataset images. Our network takes 4 color *mosaics* of the top samples, plus a global mesh rendering, and outputs per-pixel blend weights. The weighted sum of inputs forms a new color image.

Chaurasia et al. 2013; Hedman et al. 2016; Kopf et al. 2014a]) when discussing our approach to blending. Soft3D [Penner and Zhang 2017] is the most recent IBR algorithm, based on a regular discretization of space using the input images and a sophisticated blending approach using a *soft* estimation of visibility. Global proxy IBR methods (e.g., [Buehler et al. 2001; Eisemann et al. 2008; Heigl et al. 1999]) are inherently limited in realism by the accuracy of the 3D reconstruction; per-view solutions score much better on this count.

All previous per-view methods have quite limited free-viewpoint capabilities, *e.g.*, due to discretization [Penner and Zhang 2017], an implicit fronto-parallel superpixel assumption [Chaurasia et al. 2013; Ortiz-Cayon et al. 2015], or – to a lesser extent – due to a variety of rendering artifacts (see Fig. 2) that occur for many existing methods, including InsideOut [Hedman et al. 2016]. Our approach follows a similar architecture to InsideOut, but enhances realism thanks to the improved per-view geometry and deep blending.

2.3 Learning for IBR and Blending

The early work on image-based priors for IBR [Fitzgibbon et al. 2005; Woodford and Fitzgibbon 2005; Woodford et al. 2007, 2006] used a form of learning to synthesize novel views, based on a dictionary of patches from the input images. More recently, Convolutional Neural Networks (CNNs) and deep learning have been applied to the novel view synthesis problem. DeepStereo [Flynn et al. 2016] learns to predict a depth and a color using separate "towers" in the network, building on traditional plane-sweep algorithms. Zhou et al. [2016], use an encoder-decoder approach to predict the flow field transforming an input image to the novel view. There has been interesting work in learning for view synthesis in the context of Light Fields [Kalantari et al. 2016; Srinivasan et al. 2017]. However, the constraints of the narrow-baseline inputs result in very different design choices and it is difficult to see how to directly apply these to our scenario of wide-baseline capture.

While these methods provided very impressive and promising results, they do not allow free-viewpoint rendering, and most of them are far from interactive. Like many deep learning methods, the wide-baseline CNN solutions [Flynn et al. 2016; Zhou et al. 2016] suffer from visual artifacts that do not provide the level of realism we seek. Our approach provides much richer input into the learning step – resulting in higher realism – by using high-quality refined geometry to provide ranked source-image mosaics to the CNN.

Finally, deep learning has been used to estimate weights for HDR blending [Kalantari and Ramamoorthi 2017], or for filtering for Monte-Carlo path tracing [Bako et al. 2017]; while these methods are similar in spirit to our approach for learning blend weights, they involve a different set of challenges requiring different solutions.

3 OVERVIEW

The core part of most IBR algorithms is the reprojection and *blending* of input images to synthesize a novel view. This blending step is a very complex operation that needs to compensate for inaccurate geometry and for artifacts induced by view- and image-dependent effects. Previous methods use complex heuristics, hand-crafted to work for specific scene configurations. However, they are generally unable to provide realistic free-viewpoint navigation for IBR. Deep learning offers a very powerful tool for coping with variable inputs, while explicitly rewarding high quality blending. Our key novelty is the introduction of a deep blending method for IBR, that synthesizes each novel view after having learned to compute the *blending weights* to combine the relevant input photos.

Deep blending for IBR poses several challenges. We first need to provide the network with the best possible geometry to reduce the amount of artifact correction required. We then need to determine the CNN architecture and adapt previous rendering algorithms to maintain interactive free-viewpoint navigation, while tightly integrating with learning. We also need to generate sufficient training data to allow the network to learn good blending weights. Finally, we need to define a loss function that minimizes rendering artifacts. We next present an overview of how we solve each of these challenges, using the pipeline shown in Fig. 3.

Our input is a set of input photographs of a scene. We first calibrate the cameras using structure from motion (SfM) [Schönberger and Frahm 2016]. Following previous work [Hedman et al. 2016], we use per-view meshes for rendering. The goal of our first step is to provide high quality per-view depth map refinement, and generate the compact meshes that respect occlusion edges as much as possible. We achieve this by combining two different MVS methods: COLMAP [Schönberger et al. 2016] – based on Patch-Match [Barnes et al. 2009] – that provides fine details in each per-view depth map, and methods based on Delaunay tetrahedralization [Jancosek and Pajdla 2011; RealityCapture 2016] that provide a smooth mesh estimate in regions where COLMAP fails. We use the complementary information from each approach as a prior for the other during our per-view mesh refinement algorithm.

Integrating a Neural Network into an interactive IBR system is challenging. To allow a per-frame interactive rendering loop that includes a CNN evaluation, we choose a U-net [Ronneberger et al. 2015] architecture, and generate a fixed set of inputs to the CNN. For rendering, we build on InsideOut [Hedman et al. 2016], which at each output pixel selects a variable number of input photos to blend into a final image (Fig. 3). In our rendering loop, we rank these per-pixel selections to generate a fixed number of *mosaics* that are blended into the novel view. Each pixel of the first mosaic contains the color value of the best selected pixel, the second mosaic contains the second best, and so on. Pixels are ranked according to an IBR cost [Hedman et al. 2016] and a new visibility term we introduce.

Training data for our supervised learning of the CNN weights is non-traditional: the same photo serves, at times, as one of the *inputs* to the mosaic-building step, or it is held-out so it can serve as the ground truth *output* that the network tries to reconstruct from mosaics of other input photos. We generate a large dataset of input images through round-robin use of this hold-out strategy, and through data augmentation. The test results we show are distinct samples (i.e., novel viewpoints) never seen in training. Finally, to achieve good visual quality, to overcome alignment issues and to reduce flickering, we use a perceptually-motivated loss [Johnson et al. 2016] and introduce a temporal coherence term. Our experiments show that the network training can be run on a general set of images from training scenes, and that performance is stable even when no images for a given testing scene were seen during training.

Our approach allows interactive rendering for many of our scenes, and provides close to photorealistic free-viewpoint navigation, even far from the input cameras. Compared to previous work, our method significantly reduces many glaring artifacts.

4 HIGH-QUALITY PER-VIEW MESHES FOR DEEP IBR

The input to our method is a set of photos calibrated with SfM [Schönberger and Frahm 2016]. From these, we can use MVS reconstruction to generate per-view depth maps, that can then be converted into per-view meshes for IBR [Hedman et al. 2016]. There are several desirable properties for these meshes: 1) they need to respect occlusion edges, and should be "cut" with no geometry straddling a depth discontinuity, thus avoiding reprojection artifacts in novel views; 2) they should have low triangle complexity to minimize the effect on frame-rate during rendering, and preferably have fewer triangles in the background. When these requirements are satisfied, visibility-related artifacts are reduced during rendering, making the task easier for our deep blending approach, while maintaining interactive frame-rates.

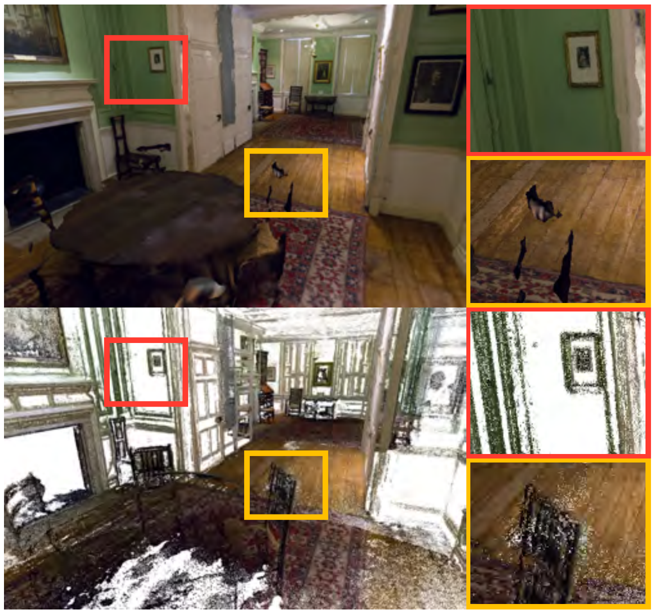

Fig. 4. **Top:** Globally consistent reconstruction with RealityCapture [2016]. **Bottom:** Same scene and viewpoint using COLMAP [Schönberger and Frahm 2016; Schönberger et al. 2016]. We can clearly see that the featureless wall is completely missing from the COLMAP depth maps, but a smooth estimate is provided by RealityCapture. Conversely, the details of the back of the chair are only present in the COLMAP depth maps.

To respect occlusion edges, we introduce a per-view refinement algorithm that fuses and combines information from two different MVS reconstruction methods that have different completeness-vs.-accuracy tradeoffs. To achieve low mesh complexity, we present a view-dependent mesh simplification method that achieves very high compression rates while respecting occlusion edges. Our solution results in significantly better preservation of occlusion edges and much lower mesh complexity overall, compared to previous methods [Hedman et al. 2016].

4.1 Per-View Geometry Refinement

To achieve good quality per-view refinement, we combine two complementary MVS methods, one with better detail accuracy, and one with better global completeness. For accurate depth map reconstruction, we use COLMAP [Schönberger et al. 2016]. COLMAP resolves small details accurately, but can break down for large textureless regions. For global meshing, we use RealityCapture [2016], which is the commercial evolution of the CMPMVS method [Jancosek and Pajdla 2011]. This method provides a smooth global mesh estimate, providing information in regions that are not reconstructed by other approaches (e.g, textureless content). The strengths and weaknesses of each method are clearly illustrated in Fig. 4.

The Patch-Match-based algorithm of COLMAP successfully finds small structures in the scene, but often tends to produce unreliable edges, since it optimizes for photoconsistency using a patch. This is a known problem in stereo algorithms, *i.e.*, the algorithm must choose between edge fattening and missing small structures (see Scharstein and Szeliski [2002], who point out that approaches focusing on

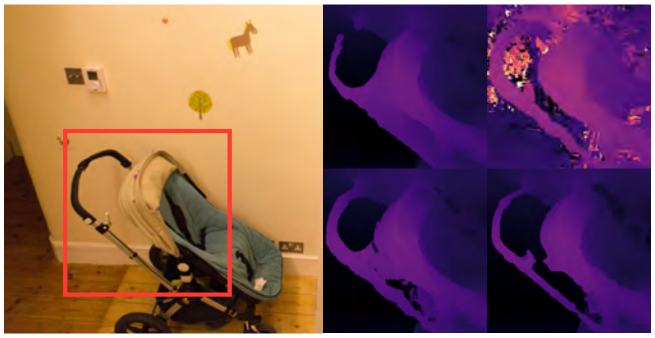
Fig. 5. Merging globally and locally accurate depth maps leads to improved occlusion edge handling and artifact minimization. **Left:** Reference image **Top middle:** Globally complete RealityCapture mesh. **Top right:** Locally accurate COLMAP depth map. **Bottom middle:** Fused COLMAP and RealityCapture depth maps. **Bottom right:** Our refined depth map.

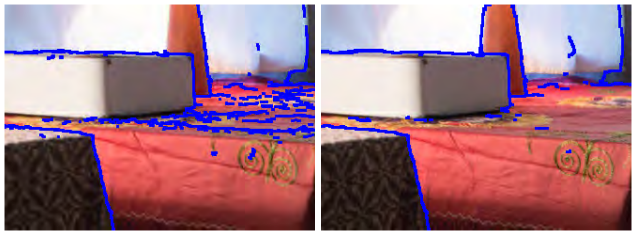
Fig. 6. **Left:** Occlusion edges, marked in blue, for a depth map using InsideOut [Hedman et al. 2016]. **Right:** Our solution provides much cleaner edges on surfaces at grazing angles.

details fail in textureless regions). However, since we already have a globally consistent geometry, we can significantly constrain the search space for our stereo optimization and still obtain reliable results by performing single-pixel optimization.

At a high level, we optimize for per-pixel photoconsistency, but use the geometries estimated from MVS as strong priors to avoid ambiguity. We first fuse the COLMAP depth maps with the RealityCapture mesh wherever COLMAP is uncertain (see Appendix A), replacing any unreliable depth by the global geometry. Then, we run a per-pixel photoconsistency optimization to refine the occlusion edges further, similar to InsideOut [Hedman et al. 2016].

Our refinement first uses Patch-Match optimization to refine pixel depths i, searching for a low photoconsistency cost while preferring that they stay close to the original depth map. We minimize the cost

$$c(i) = c_{\mathrm{p}}(i) + \alpha_{\mathrm{nn}} c_{\mathrm{nn}}(i), \tag{1}$$

where the photoconsistency cost $c_{\mathrm{p}}(i)$ [Hedman et al. 2016] emphasizes both color and image-gradient differences, and $c_{\mathrm{nn}}(i) = d(i, D)^2 / depth(i)^2$ is the depth-normalized square distance to the nearest neighbor in the original depth map D. We balance the terms with $\alpha_{\mathrm{nn}} = 4 \times 10^{-4}$.

We then apply a bilateral median filter on the depth. We stop this optimization after two iterations (two backwards-forwards passes of patch match optimization and two median filter iterations), as the change is negligible after this for all our test scenes.

The resulting depth maps both preserve small features and provide reliable information in textureless regions. In Fig. 5, we show how our approach combines the advantages of both sources of 3D.

4.2 Occlusion Edges and Meshing Simplification

The refinement step above produces a dense depth map which we use to create per-view meshes for IBR. We now need to simplify the per-view meshes as much as possible while preserving occlusion edges, to minimize the effect on rendering performance. We do this by first identifying occlusion edges, then "cutting" the mesh to preserve them, followed by a mesh simplification step.

4.2.1 Meshes from Depth maps. First, we need to reliably detect triangles at occlusion edges. Noisy detection often creates isolated components which degrades simplification. Previous work used a simple 3D distance threshold [Hedman et al. 2016] for this purpose. We found that the min/max aspect ratio measure from [Pébay and Baker 2003] more reliably detects degenerate triangles at occlusion edges. To avoid over-cutting background surfaces and surfaces at grazing angles, we also account for depth and the slope of the underlying surface when detecting occlusion edges.

We connect each pixel to its neighbors with a quadrilateral. This results in two different possible pairs of triangles. We keep the pair which minimizes ρ, the maximum aspect ratio of the two triangles. Quads with a large ρ value are likely to be at occlusion edges, since this value corresponds to a large difference in depth. A naive solution would thus be to discard quads with ρ above a threshold. For surfaces seen from grazing angles, this removes too many triangles as the depth varies greatly. We would also discard too many small foreground quads for rendering, since our measure is scale-invariant and we want to keep closely seen objects as clean as possible. To overcome these issues, we modulate ρ by two terms. For grazing angle triangles, we estimate a smooth normal for each pixel and modulate ρ by the dot product between the normal and direction to the viewpoint. We clamp this modulation to avoid a more than 10 times reduction. For foreground pixels, we also modulate ρ by a second term, which is the disparity at the pixel normalized by the median disparity over the image. This modulation is clamped between 50% and 200% (see Appendix B for details). Examples of occlusion edge cuts using our method are shown in Fig. 6.

4.2.2 Mesh Simplification. Once occlusion edges have been cut, we create a full-resolution mesh that we simplify based on the edge-collapse method of Garland and Heckbert [Garland and Heckbert 1997]. There is a rich literature on view-dependent simplification for polygonal models, including, *e.g.*, screen-size error thresholds and silhouette preservation [Luebke and Erikson 1997], or sophisticated methods to avoid folding triangles [El-Sana and Varshney 1999]. The solution we present here is simpler, since we target the specific case of per-view meshes for multi-view photo datasets.

We construct our per-view meshes to have uniformly sized triangles in image-space. Compared to earlier approaches [Hedman et al. 2016], which strive for uniformly sized triangles in 3D, this better preserves objects close to the camera (with smaller triangles), while also reducing the number of triangles in the background. We achieve this by dividing the standard quadric simplification cost by the squared distance between the edge and the camera.

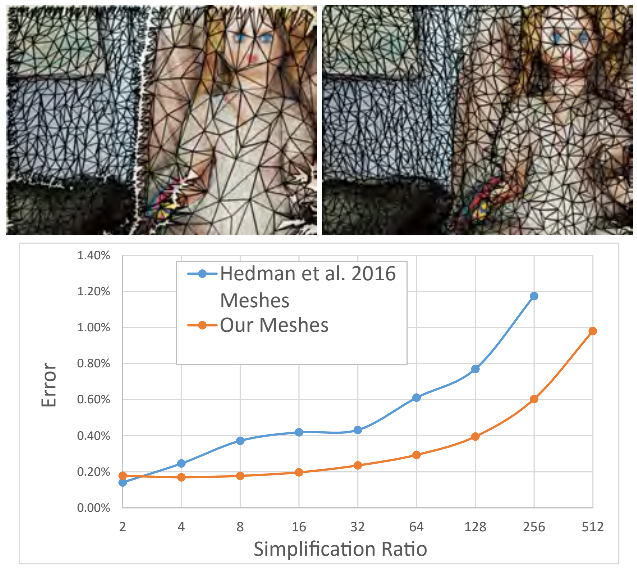

Fig. 7. **Top Left:** Simplification using InsideOut meshing [Hedman et al. 2016]. **Top Right:** Our solution, which provides higher uniformity in image space and preserves occlusion edges with good quality. Meshes were simplified by a factor of 256x. **Bottom:** Mean relative error in 3D at different simplification ratios for a set of 20 meshes from different scenes. More than 80% of meshes could not be simplified at 512x using InsideOut.

One frequent problem with quadric error metrics are “folding triangles”: we avoid this by preventing normal deviation caused by collapse over 60°. Finally, we multiply the weight of occlusion edges by the targeted simplification ratio to encourage their preservation. As we can see in Fig. 7, we can achieve extremely high simplification factors (up to 256x) with little loss in quality.

5 LEARNING TO BLEND

The blending step of IBR needs to compensate for geometric and visibility errors, and for view- and image-dependent effects. Correctly handling visibility for IBR in the presence of inaccurate geometry is a very hard problem that has plagued the field from the outset. Various hand-crafted heuristics include soft visibility [Eisemann et al. 2008; Pulli et al. 1997; Zheng et al. 2009], the “prefer background” heuristic [Chaurasia et al. 2013], per-view fuzzy depth test [Hedman et al. 2016], and the discretized soft visibility of the Soft3D approach [Penner and Zhang 2017]. Similarly, final blending weights are typically heuristics, which try to address the different sources of artifacts. These include the angle and distance terms for ULR [Buehler et al. 2001], the more sophisticated texture-stretch heuristics of Hyperlapse [Kopf et al. 2014b] – requiring an offline graphcut/Poisson blending step – or the adaptive bandwidth of InsideOut [Hedman et al. 2016]. Despite attempts at a more rigorous Bayesian formulation [Pujades et al. 2014], reliably achieving high-quality blending effectively remains elusive.

Deep learning has demonstrated the ability to perform very complex image transformation tasks and is thus an ideal candidate for solving the IBR blending problem. We train a convolutional neural network (CNN) to generate *blending weights* that are then used to combine warped (or reprojected) contributions from different input images. Our goal is to evaluate the feed-forward CNN in an interactive renderer, imposing strict constraints on network architecture, input layers, and the renderer. Generating sufficient training data to find the weights is also challenging. Finally, care must be taken when defining the training loss: we want to produce images that are as realistic as possible, avoiding temporal artifacts such as flickering.

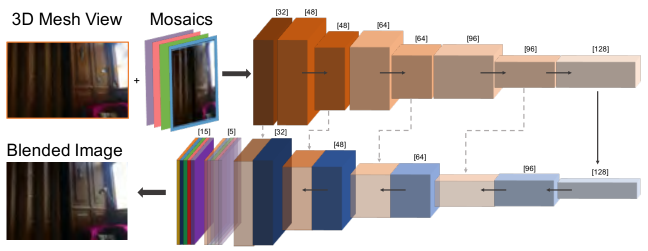

Fig. 8. DeepBlending network architecture. An RGB view of the global mesh is concatenated channel-wise with 4 RGB mosaic images and fed into a fully convolutional U-net-type architecture. Each successive block is generated via a 3×3 convolution followed by a ReLU activation; the number of channels for each block is shown in brackets. Dotted lines represent “skip connections,” where features in the down-convolution portion are concatenated to features in the up-convolution portion. Decreases in spatial resolution are made by convolving with a stride of two, and increases use nearest-neighbor upsampling before the convolution is applied. A softmax activation is applied to obtain per-pixel blend weights. The novel image is obtained by taking the per-pixel weighted sum of the 5 network inputs.

5.1 Network Architecture and Rendering Algorithm

Our goal of tight integration in the rendering loop imposes strict requirements in terms of network architecture, and precludes involved solutions such as Recurrent Neural Networks [Gregor et al. 2015]. In addition, we need a CNN that is capable of handling image transformations in a multi-scale fashion. For these reasons, we choose a small U-net architecture [Ronneberger et al. 2015] with skip connections, which has been demonstrated to work well on image transformation tasks [Zhu et al. 2017].

The network architecture is shown in Fig. 8. This architecture has a receptive field of 63 pixels at the bottleneck, in theory giving it the power to correct for artifacts that are at most this size.

This network architecture requires a fixed number of inputs, and we choose to provide it with five images, one global mesh render and 4 source-image mosaics, that need to be blended. Similar to InsideOut [Hedman et al. 2016], our rendering algorithm uses a spatial acceleration data structure to select, at each output pixel, pixels in multiple input images that contribute to the novel view.

On a high level, InsideOut uses a voxelized representation of the scene to accelerate view selection, where each voxel indexes relevant triangles from the per-view meshes. First, it finds the visible voxels for a novel viewpoint, and ranks the input images associated with each visible voxel according to their potential usefulness. Next, the triangles for the top-ranked images at every voxel are rendered into the novel viewpoint, which provides a set of unique input images for each pixel in the output view. In the end, each output pixel will have 4 to 12 color values rendered into it, depending on the views selected in the corresponding voxel.

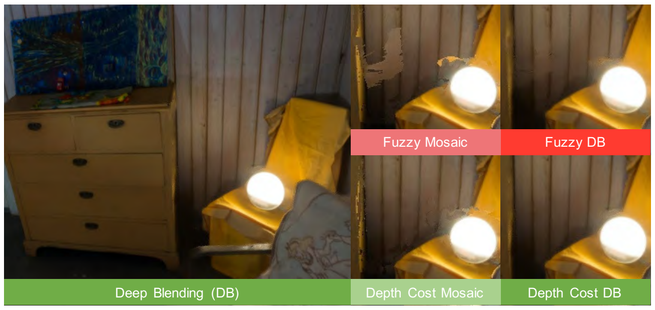

Fig. 9. Our depth cost reduces the effect of "floating geometry" that occurs with a fuzzy depth test [Hedman et al. 2016]. **Left:** Full view of our output. **Middle:** Top-ranked mosaic per cost. **Right:** Network outputs.

Since we have chosen to fix the number of inputs to our CNN, we need to modify the rendering algorithm to "pack" this information into four separate layers that will then be fed to the network for blending. We do this by creating four *mosaics* formed by re-ranking the set of sampled source pixels on a per-output-pixel basis. We rank source pixels using the IBR cost of InsideOut, but modulate the cost with an additional depth component (see Appendix C), effectively replacing the fuzzy depth test that tends to encourage the appearance of "floating" per-view geometry. We see the effect of this new cost function in Fig. 9. For each pixel, we store the four most highly ranked contributions in the four mosaics. In the case of missing geometry, we fill holes in the input mosaics with a rendered view of the textured global mesh. These four mosaics, together with a view of the global mesh, form the input layers to our network. This is illustrated in Fig. 10.

We experimented with using extra input channels (depth, normal, heuristic blend weights), but these did not provide a conclusive improvement. This is probably because these inputs have already been used to form the ranked input mosaics, so the extra layers simply contain redundant information already implicitly encoded in the input layers and their relative order.

5.2 Training data

We train and test our network on a number of different scenes. Note that in our experiments, we show results both with and without images from the test scene being included in the training data. However, the actual results consist only of data samples – the novel viewpoints – that were never seen in training.

In previous work [Ortiz-Cayon et al. 2015; Waechter et al. 2017], *input* photos are used as ground truth images, in a held-out or leave-one-out strategy for error evaluation. We adopt a similar approach, but to succeed in training our CNN, we require a large number of images with sufficient variety of scene content. For this, we collected a total of 19 scenes from different sources: 7 scenes from Chaurasia et al. [Chaurasia et al. 2013], 4 scenes from Hedman et al. [Hedman et al. 2016], 1 scene from the Eth3D benchmark [Schöps et al. 2017], and 7 new scenes which we have captured ourselves. Each scene contains between 12 and 418 input images, with resolution varying from 1228×816 to 2592×1944. There are 5 indoors scenes, and 5 scenes containing a significant amount of vegetaion. In total, we have 2630 images, and we use data augmentation to mitigate the risk

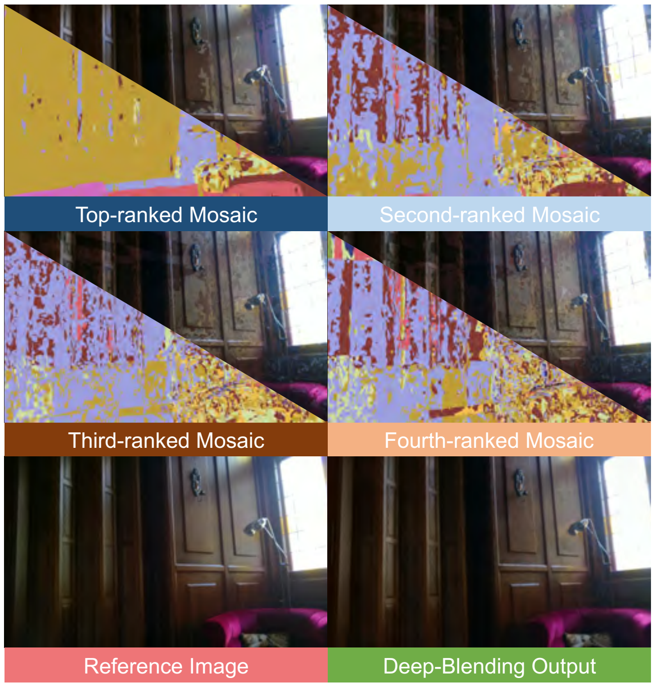

Fig. 10. Our network takes as input ranked mosaics generated from a set of warped candidate views. For each pixel, the candidates are ranked based on their expected blending contribution, and 4 color-image mosaics are formed from the top 4 rankings. Example mosaics are shown in the first two rows. The top right halves show the color mosaic, while the bottom left halves show colormaps of the selection, with each input shown in a different color. Weighted blending outputs from our network (bottom right) are trained by minimizing their difference with real images (bottom left). Our network also blends an RGB view of the global mesh (not shown).

of overfitting by taking random 256x256 crops of the images and performing random rotations and horizontal/vertical flips. We use 90% of the images in each scene for training, leaving the remaining 10% as a validation set.

We also experimented with data augmentation for the mosaics themselves, and generated mosaics with blend weights computed for a different camera location. This did not ensure temporal smoothness, but instead suppressed highlights.

5.3 Training loss

The goal of the training is to determine how well the held-out novel view resembles the input photograph. We face two main challenges: dealing with slight misalignments due to warping and encouraging temporal coherence.

Since our geometry reconstruction is not perfect, we may never obtain an accurate match between the warped input images and the held-out ground truth. In particular, the surface may have slight reconstruction error, and there can be an offset between the warped mosaics and the reference photograph or differences in texture resolution and stretch. Together, these effects mean that we cannot form the reference photograph as a weighted average of the input

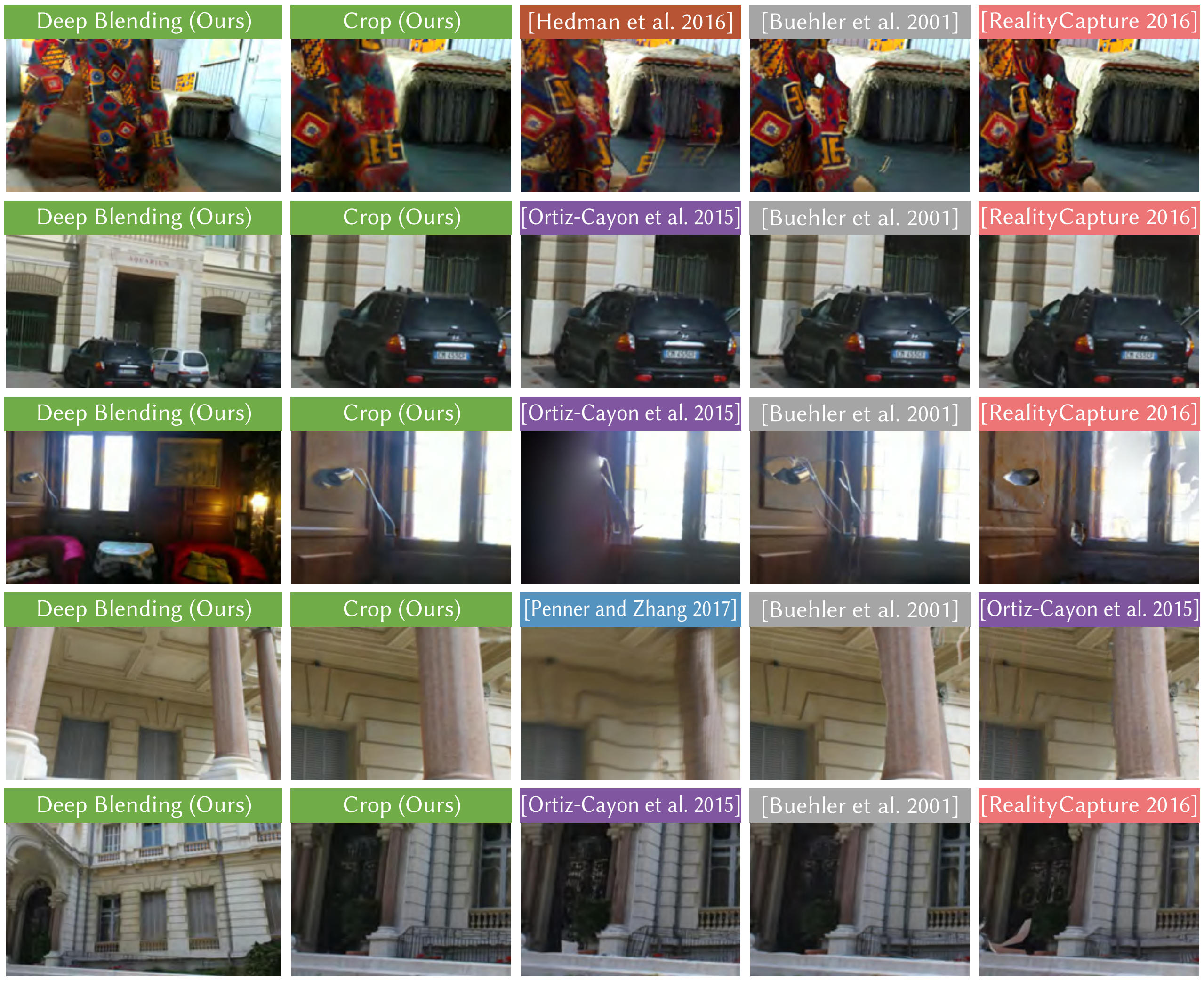

Fig. 11. Results from 5 scenes. Left: Full novel view from our solution, followed by a crop using our method. The remaining three columns show previous methods as in Fig. 2. In most cases, errors due to inaccurate geometry result in visual artifacts such as ghosting, incorrect edge reconstruction as well as blur.

layers. This makes training with an image loss such as L_1 or SSIM ill-conditioned.

Instead, we build on work developed for style transfer, where "perceptual" image differences have proven their effectiveness, based on pretrained VGG16 network features [Johnson et al. 2016]. For a given novel image I_N and a reference image I_R, our loss $\mathcal{L}$ is:

$$\mathcal{L}(I_N, I_R) = |I_N - I_R| + |\mathrm{VGG16}_{relu12}(I_N) - \mathrm{VGG16}_{relu12}(I_R)| + |\mathrm{VGG16}_{relu22}(I_N) - \mathrm{VGG16}_{relu22}(I_R)| \quad (2)$$

where $\mathrm{VGG16}_{relu*}$ are the feature activations at different scales of a VGG16 network pretrained on ImageNet.

With the formulation above, the network is able to create high-quality single frames. However, for difficult regions in the scene, the network struggles to find a good solution and the weights tend to oscillate from frame to frame. To disambiguate this case, we encourage the network to produce temporally stable results with an auxiliary loss, based on a small window of camera motion. As training data, we generate small 2-frame clips, which start from a reference camera pose and move in a random direction. To keep camera motion relatively small, we set frame-to-frame displacement to be 25% of the average baseline. During training, we run the network twice to generate outputs for both the frames in the clip.

The final temporally consistent loss function we use is as follows:

$$\mathcal{L}_T(I_N, I_R) = \mathcal{L}(I_N^t, I_R) + 0.33 * \mathcal{L}(I_N^{t-1}, \mathcal{W}_f(I_N^t)), \quad (3)$$

where $\mathcal{W}_f(I)$ is the warped image I using optical flow. Optical flow is estimated as the average flow of the 4 input mosaics, ignoring global geometry. We can compute this flow exactly for each mosaic, since we have the corresponding geometry and render using OpenGL.

We also tested a loss including image gradients, which in theory should discourage seams due to color harmonization problems.

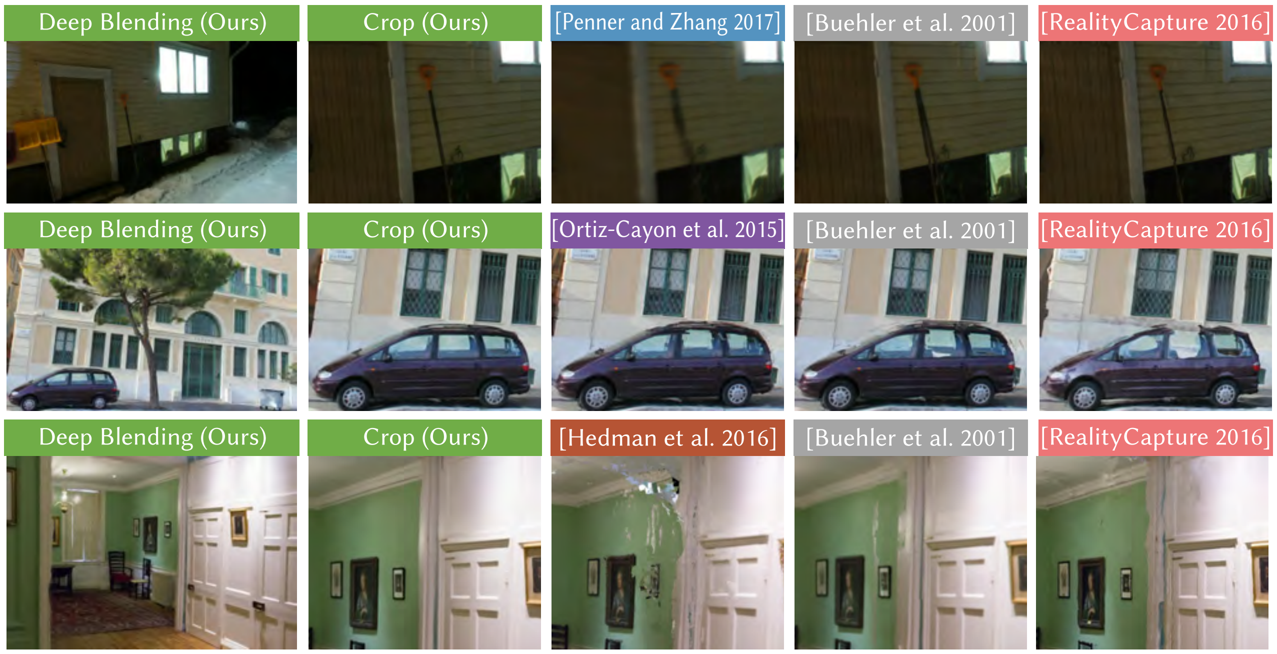

Fig. 12. Results continued, presentation as in Fig. 11. In some cases, our method can recover from large geometric errors (e.g., car in the middle row).

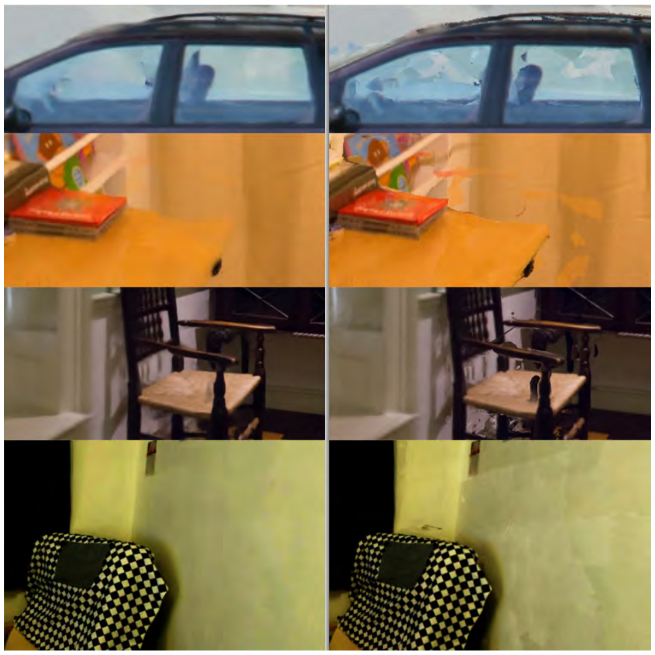

Fig. 13. Our deep blending network vs heuristic blending, both using our refined per-view meshes. **Left:** Our full method, showing the difference in quality due to Deep Blending. **Right:** Our heuristic blend cost (see Sec. 5.1).

However, this instead encouraged outliers and slowed down convergence. We did not experiment with adversarial losses, which is an interesting future work direction; however, maintaining temporal smoothness would probably be very challenging. We also experimented with a temporal window as input to the network. Interestingly, this did not improve convergence, possibly because the network is unable to correct the parallax between the inputs.

6 IMPLEMENTATION AND EVALUATION

We implemented our system in C++ and OpenGL, combined with the C++ interface of TensorFlow [Abadi et al. 2015] for runtime blend evaluation. Our rendering pipeline is based on InsideOut [Hedman et al. 2016] with the depth component modification to the IBR cost computation (see Sec. 5.1).

To maximize the pipeline's potential for interactive rendering times, we implemented custom TensorFlow operations that directly copy the input and output OpenGL textures to and from the network in on-device GPU transfers, making use of the OpenGL/CUDA interop interface available in the CUDA version 9.0 library. A custom CUDA kernel was implemented to increase the speed of 2D spatial upsampling; all other network components used TensorFlow's native operations with cuDNN 7.1 [Chetlur et al. 2014].

6.1 Results and Comparisons

In Figs. 11 and 12, we show results from 8 scenes, comparing our method with other end-to-end IBR systems:

RealityCapture: The textured mesh from [RealityCapture 2016].
Selective IBR: The superpixel IBR approach by [Ortiz-Cayon et al. 2015] using the RealityCapture mesh as input geometry.
ULR: Unstructured Lumigraph Rendering [Buehler et al. 2001] with soft visibility [Eisemann et al. 2008] and the RealityCapture mesh as the geometry proxy.
Soft3D: The novel view synthesis algorithm by [Penner and Zhang 2017], using their custom soft 3D reconstructions.
InsideOut: The indoor IBR system by [Hedman et al. 2016] using their custom depth-sensor based 3D reconstructions.

Except for InsideOut, which uses the original reconstructions with depth-sensor data, all methods use the same camera registration and calibration produced by COLMAP. The full set of videos with paths from all these scenes and comparisons with other methods is in supplemental material. We show Soft3D only in 4 scenes, as it has difficulty handling the very unstructured capture of our data and comparisons in more scenes would not be very informative.

Quantitative comparison for IBR is always a difficult endeavor. We provide a quantitative comparison using the Virtual Rephotography approach [Waechter et al. 2017], using the shiftable L1 error metric of [Hedman et al. 2017]. For this, and all following quantitative evaluations, we have used the images from paths of a subset of 5 scenes, namely CreepAttic, Museum-1, NightSnow (rows 1,4 and 6 in Fig. 11), DrJohnson (row 3 in Fig. 12), and Hugo-1 (Fig. 3). It is important to note that this evaluation favors previous methods, since most of the time images captured are quite close to other input views (by definition, to provide overlap for SfM and MVS); our method excels in viewpoints further from the inputs, but meaningful rephotography would require capture of photos specifically for this purpose. The results of this comparison are shown in Fig. 18.

For the results in Fig. 11 and 12, we included all scenes in the training. We also show results with networks trained without the scene being shown (Fig. 16). We see that there is little visible difference between the two, which is also confirmed by the numerical results shown in Fig. 15 (right).

6.2 Evaluation of Deep Blending vs. Geometric Refinement

We evaluate the relative effect of each of our steps, namely deep blending vs. our improved geometric refinement. We implemented an IBR algorithm similar to [Hedman et al. 2016], with two variables: InsideOut per-view refinement vs. our refinement, and InsideOut-style blending using our depth-modulated cost (which we call *heuristic blending* from now on) vs. Deep Blending, for a total of four configurations. For a fair comparison with our refinement, InsideOut refinement is performed using only the RealityCapture mesh.

In Fig. 13, we compare heuristic blending vs. deep blending using our refinement, isolating the effect of blending only, and observe there are several visible artifacts that are corrected by our solution. In Fig. 14, we show the effect of using our refinement vs. InsideOut refinement, but using Deep Blending for both, isolating the effect of refinement only. We also show InsideOut refinement with heuristic blending that is clearly worse; this is confirmed quantitatively in the leave-one-out rephotography [Waechter et al. 2017] results in Fig. 15. These graphs plot the “shiftable L1 error” [Hedman et al. 2017] – the minimum L1 distance for a 7×7 color patch around each ground-truth image pixel, compared with all output patches that have been shifted up to ± 2 pixels in x and/or y around the source pixel. Interestingly, the performance is similar in the three cases where either or both of our improvements are used. This indicates that while Deep Blending can correct for bad refinement for the rephotography test (*i.e.*, for views not too far from the input photos), the relative effect of blending and refinement is equally important. As can be seen in the supplemental material and video, the relative visual importance of each can be scene- and context-dependent, and our complete solution is generally better visually, especially far from the input views.

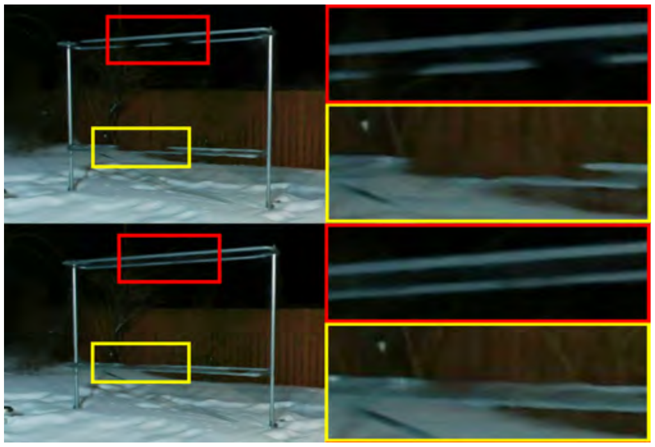

Fig. 14. Different geometry refinement approaches with our deep blending network. **Top:** Using the geometry refinement from [Hedman et al. 2016]. **Bottom:** Our solution, which better recovers thin structures.

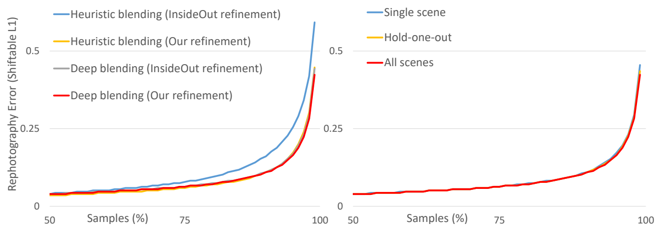

Fig. 15. Rephotography comparison, *i.e.*, how well different approaches can reconstruct held-out input images. The graphs show the cumulative distribution of errors over five scenes (Creepy Attic, Dr Johnson, Museum-1, Hugo-1, and Night Snow), *i.e.*, the percentage of pixels (x-axis) with an error smaller than a threshold (y-axis), starting at the median error. **Left, ablation:** We compare our deep blending with a heuristic weighted average, and our geometry refinement with the earlier refinement approach from [Hedman et al. 2016]. **Right, generalization (evaluated over the same 5 scenes):** We compare a single network trained on all 19 scenes with (1) **hold-one-out** networks that never saw the test scene during training, and (2) **single-scene** networks that only saw the test scene during training.

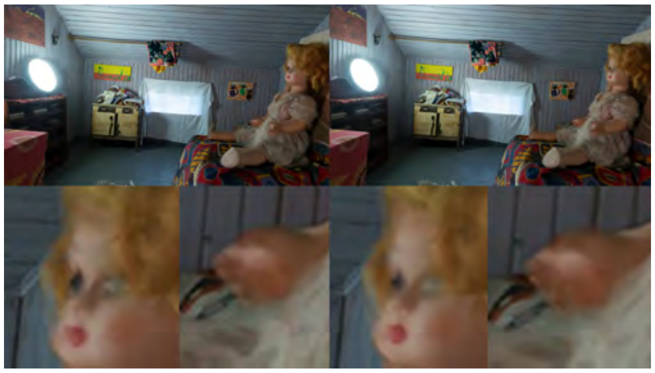

Fig. 16. **Left:** novel view computed with all scenes used for training, **Right:** same view rendered with a network that never saw the scene during training. The result is similar.

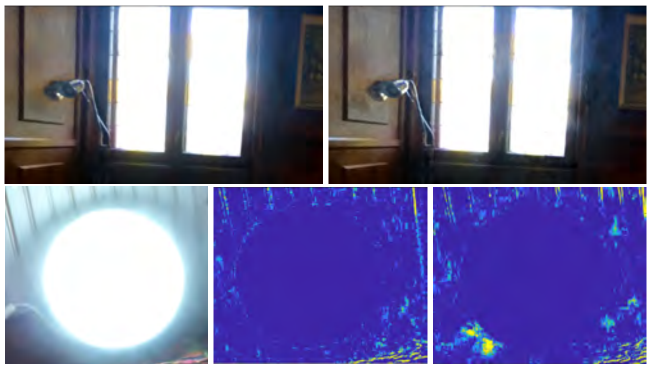

Fig. 17. **Top Left:** Full method, with the VGG loss. **Top Right:** Same view without the VGG loss. **Bottom Left:** Frame t. **Bottom Middle:** Our full method (10x temporal difference between frame t and t+1) using our full method. **Bottom Right:** Our method, no temporal loss (10x temporal difference between frame t and t+1).

6.3 Network Evaluation

We also perform ablation tests on the learning process. In our supplemental material, we provide a large number of videos showing the performance for different ablations in data or method. We have investigated several aspects:

- **Training scenes.** In Figs. 15-16 and supplemental material, we show the effect of running our network in 5 scenes it did not see during training (hold-one-out). The results are quite similar to training with the scene. We also test the effect of only training with images from one scene (single-scene), which in Fig. 18 does not show a strong numerical difference, but tends to reduce temporal stability (see supplemental material).
- **Loss.** In Fig. 17, we show the effect of our perceptual (VGG) loss vs. an L_1 loss. Our supplemental videos further show this difference in performance. We also include results of our network trained without the temporal loss; these exhibit strong inter-frame intensity flicker in many of the sequences.
- **Image regression.** We tested a variant of the network that directly regresses (predicts) the output image rather than blend weights. Similarly to [Bako et al. 2017], we observe that directly regressing the output image works well, but network convergence is much slower.
- **Number of mosaics.** We show supplementary results using only the top 1 or 2 mosaics as network input. We observe diminishing return with more mosaics – each additional mosaic contributes to smaller details, particularly along surface boundaries. Having more mosiacs does, however, lead to smoother view transitions.
- **Training data size.** We show supplemental videos after training with a random 30% subset of our data. Surprisingly, the qualitative performance is fairly constant. This is likely because the blend weight prediction function has complex variation throughout an image, meaning that even a reduced number of training images will still well cover the space of blend weight idiosyncracies that need to be learned.

6.4 Performance

We evaluate the runtime performance of our interactive rendering pipeline at a resolution of 1280×720 px on a system with a 3.47 GHz Intel Xeon processor and an NVIDIA GTX 1080Ti GPU. (Note that the videos in our supplemental are rendered offline at a resolution of 1920 × 1080 px.) Table 1 shows average per-frame execution times of our interactive pipeline over several scenes. Runtimes are specific to the scene and are not driven only by the deep blending network component. For smaller scenes, especially indoor scenes, our implementation achieves greater than 30fps performance, which falls off gradually as scene complexity increases. For large outdoor scenes, we observe that the primary bottleneck is in the voxel-wise camera selection, which in our current implementation is tied to a fixed-size spatial subdivision. An adaptive or hierarchical approach is an interesting future direction for this issue.

Preprocessing times for our pipeline varied from scene to scene but generally finished overnight. We treat network training separately, which takes two days, but does not necessarily have to be re-run for each new scene (see Figs. 15-16). Example processing times for the Creepy Attic scene are shown in Table 2.

6.5 Limitations

Our method gracefully degrades when there is very little 3D geometry from the initial 3D reconstruction; we show an example in Fig. 19. Our method will also tend to flicker when the differences in exposure are very large and inconsistent (*e.g.*, the Bridge scene in the supplemental materials). The network occasionally creates blur, typically in regions with missing geometry or where difficult decisions need to be made (highlights, resolution mismatches etc.). Nonetheless, in the vast majority of the 19 scenes, our method outperforms all previous solutions, although in some cases one artifact is traded for another (e.g., blur instead of seams). We discuss possible avenues to address these limitations in future work, below.

7 CONCLUSIONS

We have presented Deep Blending for IBR, demonstrating that it is possible to learn blending weights for free-viewpoint navigation at interactive display rates.

In future work, we would like to address the problems outlined above. To reduce blur, one possible avenue would be the use of an adversarial loss [Isola et al. 2017]; it is unclear how well such an approach would deal with temporal coherence. To reduce flickering for very large exposure inconsistencies would probably require an effective pre-processing step for color harmonization, while respecting differences due to view-dependent materials.

Achieving real-time performance, especially in the context of stereo viewing (*i.e.*, at least 90 fps), requires performance improvements. This can partly be addressed with upgraded hardware: A dual-GPU system would render the views for both eyes in parallel, and next generation GPUs provide further speedups. There are several possible avenues of research to bridge the remaining gap, *e.g.*, using fewer mosaics, more efficient networks architectures [Howard et al. 2017], model compression [Kim et al. 2015; Molchanov et al. 2016], or porting the blending network to GLSL [Nalbach et al. 2017].

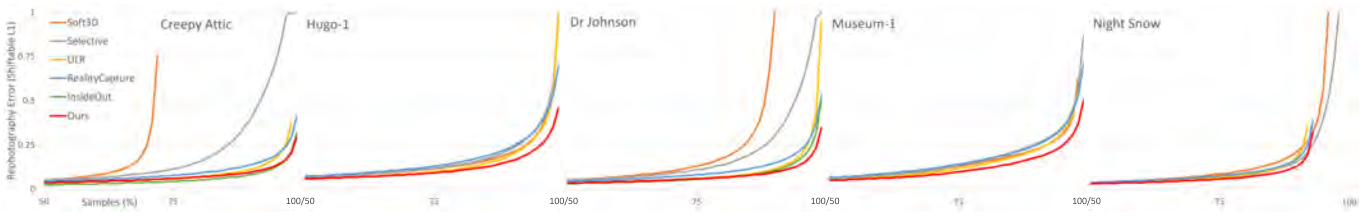

Fig. 18. Rephotography comparison, *i.e.*, how well different approaches can reconstruct held-out input images. With the shiftable L1 difference [Hedman et al. 2017], lower is better. The graphs show the cumulative distribution of errors, *i.e.*, the percentage of pixels (x-axis) with an error smaller than a threshold (y-axis), starting at the median error. Here, we see that some scenes are harder than others, and that our approach is effective for a higher % of pixels. In this experiment, we only use hold-one-out networks that did not see the test scene during training.

ACKNOWLEDGMENTS

This research was financially supported by the Rabin Ezra scholarship fund, NSF CNS-1405847, NERC NE/P019013/1, and the EU H2020 EMOTIVE project (http://www.emotiveproject.eu). We thank Clément Godard for helpful input on network training and different loss functions, Akash Bapat for conversations regarding depthmap and global geometry improvements, and Adrien Bousseau for insightful comments.

Table 1. Average runtimes (ms/frame) over 100 frames for our IBR system.

Scene	Non-network Runtime	Total Runtime	Scene	Non-network Runtime	Total Runtime
Museum-1	6.2	26.2	Hugo-1	14.1	35.6
Creepy Attic	7.1	26.8	Night Snow	15.3	33.0
Dr Johnson	12.4	33.6	Boats	19.7	47.6

Table 2. Preprocessing times for Creepy Attic (249 images at 1228 × 816) using a 4-core Intel Core i7 processor and an NVIDIA Titan X GPU.

Component	Runtime	Component	Runtime
COLMAP SfM	1h	Depth map Refinement	0.75h
COLMAP MVS	5.25h	Meshing	1.5h
RealityCapture MVS	0.5h	Network Training	37h

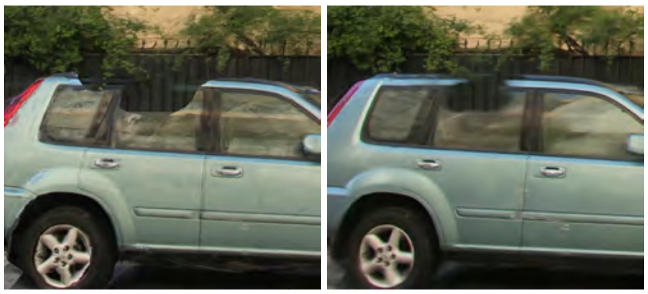

Fig. 19. **Left:** From the rendering of the mesh, we see that part of the geometry of the car is completely missing. **Right:** Despite significant visual improvement, our method cannot hallucinate the missing geometry.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Software available from tensorflow.org.

Robert Anderson, David Gallup, Jonathan T. Barron, Janne Kontkanen, Noah Snavely, Carlos Hernandez Esteban, Sameer Agarwal, and Steven M. Seitz. 2016. Jump: Virtual Reality Video. *ACM Transactions on Graphics (TOG)* 35, 6 (2016).

Murat Arikan, Reinhold Preiner, Claus Scheiblauer, Stefan Jeschke, and Michael Wimmer. 2014. Large-scale point-cloud visualization through localized textured surface reconstruction. *IEEE Trans. Vis. Comput. Graphics* 20, 9 (2014), 1280–1292.

Murat Arikan, Reinhold Preiner, and Michael Wimmer. 2016. Multi-Depth-Map Raytracing for Efficient Large-Scene Reconstruction. *IEEE Trans. Vis. Comput. Graphics* 22, 2 (2016), 1127–1137.

Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony DeRose, and Fabrice Rousselle. 2017. Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 97.

Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 24.

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured lumigraph rendering. In *SIGGRAPH*.

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.

Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. 2014. cuDNN: Efficient Primitives for Deep Learning. In *Proceedings of the NIPS Workshop on Deep Learning and Representation Learning*.

Abe Davis, Marc Levoy, and Fredo Durand. 2012. Unstructured light fields. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 305–314.

Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson de Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. 2008. Floating Textures. *Comp. Graph. Forum* (2008).

Jihad El-Sana and Amitabh Varshney. 1999. Generalized view-dependent simplification. In *Computer Graphics Forum*, Vol. 18. Wiley Online Library, 83–94.

Andrew Fitzgibbon, Yonatan Wexler, and Andrew Zisserman. 2005. Image-based rendering using image-based priors. *International Journal of Computer Vision* 63, 2 (2005), 141–151.

John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. 2016. Deepstereo: Learning to predict new views from the world's imagery. In *Computer Vision and Pattern Recognition (CVPR)*. 5515–5524.

Yasutaka Furukawa and Jean Ponce. 2010. Accurate, Dense, and Robust Multi-View Stereopsis. *IEEE Trans. PAMI* (2010).

Michael Garland and Paul S Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 209–216.

Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M Seitz. 2007. Multi-view stereo for community photo collections. In *International Conference on Computer Vision (ICCV)*.

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*. ACM, 43–54.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623* (2015).

Kaiming He, Jian Sun, and Xiaoou Tang. 2010. Guided image filtering. In *European Conference on Computer Vision (ECCV)*. Springer, 1–14.

Peter Hedman, Suhib Alsisan, Richard Szeliski, and Johannes Kopf. 2017. Casual 3D photography. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 234.

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.

Benno Heigl, Reinhard Koch, Marc Pollefeys, Joachim Denzler, and Luc Van Gool. 1999. Plenoptic modeling and rendering from image sequences taken by a hand-held camera. In *Mustererkennung 1999*. Springer, 94–101.

Heiko Hirschmuller. 2006. Stereo Vision in Structured Environments by Consistent Semi-Global Matching. In *Computer Vision and Patter Recognition (CVPR)*. 2386–2393.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint abs/1704.04861* (2017).

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-To-Image Translation With Conditional Adversarial Networks. In *CVPR*.

Michal Jancosek and Tomás Pajdla. 2011. Multi-view reconstruction preserving weakly-supported surfaces. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 3121–3128.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision (ECCV)*. Springer, 694–711.

Nima Khademi Kalantari and Ravi Ramamoorthi. 2017. Deep high dynamic range imaging of dynamic scenes. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 144.

Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. 2016. Learning-based view synthesis for light field cameras. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 193.

Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. 2015. Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications. *arXiv preprint abs/1511.06530* (2015).

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. 2017. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 78.

Johannes Kopf, Michael F. Cohen, and Richard Szeliski. 2014a. First-person Hyper-lapse Videos. *ACM Transactions on Graphics (TOG)* 33, 4, Article 78 (July 2014), 10 pages.

Johannes Kopf, Michael F Cohen, and Richard Szeliski. 2014b. First-person hyper-lapse videos. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 78.

Patrick Labatut, Jean-Philippe Pons, and Renaud Keriven. 2007. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *International Conference on Computer Vision (ICCV)*. IEEE, 1–8.

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

Wenfeng Li and Baoxin Li. 2008. Joint Conditional Random Field of multiple views with online learning for image-based rendering. In *Computer Vision and Pattern Recognition*. IEEE.

Zhengqi Li and Noah Snavely. 2018. MegaDepth: Learning Single-View Depth Prediction from Internet Photos. In *Computer Vision and Pattern Recognition (CVPR)*.

David Luebke and Carl Erikson. 1997. View-dependent simplification of arbitrary polygonal environments. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 199–208.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2016. Pruning Convolutional Neural Networks for Resource Efficient Transfer Learning. *arXiv preprint abs/1611.06440* (2016).

Oliver Nalbach, Elena Arabadzhiyska, Dushyant Mehta, Hans-Peter Seidel, and Tobias Ritschel. 2017. Deep Shading: Convolutional Neural Networks for Screen-Space Shading. 36, 4 (2017).

Rodrigo Ortiz-Cayon, Abdelaziz Djelouah, and George Drettakis. 2015. A Bayesian Approach for Selective Image-Based Rendering using Superpixels. In *International Conference on 3D Vision (3DV)*. Lyon, France. https://hal.inria.fr/hal-01207907

Philippe Pébay and Timothy Baker. 2003. Analysis of triangle quality measures. *Math. Comp.* 72, 244 (2003), 1817–1839.

Eric Penner and Li Zhang. 2017. Soft 3D reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 235.

Sergi Pujades, Frédéric Devernay, and Bastian Goldluecke. 2014. Bayesian view synthesis and image-based rendering principles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3906–3913.

Kari Pulli, Hugues Hoppe, Michael Cohen, Linda Shapiro, Tom Duchamp, and Werner Stuetzle. 1997. View-based rendering: Visualizing real objects from scanned range and color data. In *Rendering techniques? 97*. Springer, 23–34.

CapturingReality RealityCapture. 2016. RealityCapture. http://capturingreality.com

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-assisted Intervention (MICCAI)*. Springer, 234–241.

Daniel Scharstein and Richard Szeliski. 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision* 47, 1-3 (2002), 7–42.

Johannes Lutz Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *European Conference on Computer Vision (ECCV)*.

Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. 2017. A Multi-View Stereo Benchmark with High-Resolution Images and Multi-Camera Videos. In *Computer Vision and Pattern Recognition (CVPR)*.

Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. 2017. Learning to synthesize a 4d rgbd light field from a single image. In *International Conference on Computer Vision (ICCV)*, Vol. 2. 6.

Benjamin Ummenhofer and Thomas Brox. 2017. Global, Dense Multiscale Reconstruction for a Billion Points. *International Journal of Computer Vision* 125, 1 (2017), 82–94.

Michael Waechter, Mate Beljan, Simon Fuhrmann, Nils Moehrle, Johannes Kopf, and Michael Goesele. 2017. Virtual rephotography: Novel view prediction error for 3D reconstruction. *ACM Transactions on Graphics (TOG)* 36, 1 (2017), 8.

Oliver Woodford and Andrew W Fitzgibbon. 2005. Fast image-based rendering using hierarchical image-based priors. In *BMVC*, Vol. 1. 260–269.

Oliver J Woodford, Ian D Reid, and Andrew W Fitzgibbon. 2007. Efficient new-view synthesis using pairwise dictionary priors. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1–8.

Oliver J Woodford, Ian D Reid, Philip HS Torr, and Andrew W Fitzgibbon. 2006. Fields of Experts for Image-based Rendering.. In *BMVC*, Vol. 3. 1109–1108.

Ke Colin Zheng, Alex Colburn, Aseem Agarwala, Maneesh Agrawala, David Salesin, Brian Curless, and Michael F Cohen. 2009. Parallax photography: creating 3d cinematic effects from stills. In *Proceedings of Graphics Interface 2009*. Canadian Information Processing Society, 111–118.

Ke Colin Zheng, Sing Bing Kang, Michael F Cohen, and Richard Szeliski. 2007. Layered depth panoramas. In *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1–8.

Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. 2016. View synthesis by appearance flow. In *European Conference on Computer Vision (ECCV)*. Springer, 286–301.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. 1 (2017).

C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. 2004. High-quality video view interpolation using a layered representation. In *ACM transactions on graphics (TOG)*, Vol. 23. ACM, 600–608.

APPENDIX A DEFINITION OF DEPTH MAP UNCERTAINTY

COLMAP runs in two stages: photometric and geometric. The photometric stage only optimizes photoconsistency during reconstruction. This results in outliers and noise for ambiguous regions such as textureless areas. In the geometric stage, a joint optimization for photoconsistency is performed, but the resulting depth maps are tested so they agree with each other in space. This correctly removes ambiguous regions, but as observed Li and Snavely [2018] (Supplemental Fig 1), it sometimes erodes foreground objects.

We solve this problem by considering a pixel to be uncertain if its photometric depth and geometric depth consistency differs by more than 5%. As a final step, we smooth our uncertainty definition, by applying a small guided filter [He et al. 2010] to the thresholded image (10 pixels wide, with a threshold of 2%).

APPENDIX B OCCLUSION EDGE CUTTING THRESHOLD

To detect pixel-precise occlusion edges we need a local measure that is scale invariant, sensitive to strong gradients in depth maps and that measures geometry quality. The aspect ratio of a triangle, defined by $q(t) = \frac{\|t\|_\infty}{h(t)}$ where $\|t\|_\infty$ represents the length of the longest side and $h(t)$ the height of the triangle with respect to this side, is such a measure. For each pixel we consider the quad formed by neighbors. This quad can be meshed by two configurations of two triangles. We define ρ as being the minimizer over the configurations of the maximum of $q(t)$ between the two triangles.

This measure guarantees good occlusion edge detection and good geometry in most cases as pixels with a large ρ correspond to stretched triangles configuration with inhomogeneous depth variation. Nonetheless for surfaces seen at grazing angle this measure alone, as the ones used in InsideOut [Hedman et al. 2016] or Casual3D [Hedman et al. 2017], leads to over-detection. To overcome this shortcoming we modulate ρ by:

$$d_P = max(0.1, |cos(\vec{v} \cdot \vec{n})|)$$

where v is the view direction. We also modulate ρ by a second term

$$d_D = min(2, max(0.5, D/D_{\text{mean}}))$$

where D is disparity. This term allows to cut more in the background as we have less certainty about the estimated depth and our method is more robust to the lack of per-view geometry than to false geometry. Finally occlusion edges pixels are defined as those with $\gamma = \rho * d_P * d_D$ larger than the threshold $\tau = 25$ using a hysteresis threshold to avoid instabilities for edges with γ close to τ.

APPENDIX C DEPTH COMPONENT FOR IBR COST

Similarly to a standard fuzzy depth test [Hedman et al. 2016], we first compute the depth d_{min} of the closest surface for each pixel. Our depth cost reaches its minimum depth at d_{min}, increasing linearly with the depth d_i of the current sample. To ensure that the background will not be completely discarded in the presence of floating geometry, we saturate our cost at a distance of $2d_{min}$. More specifically,

$$w_{\text{depth}} = \text{clamp}_{[0,1]}\left(\frac{d_i - d_{min}}{d_{min}}\right) \quad (4)$$