Scalable Inside-Out Image-Based Rendering

Peter Hedman[1] Tobias Ritschel[1] George Drettakis[2] Gabriel Brostow[1]
[1]University College London [2]Inria

Figure 1: *Images from our method rendered in 1080p at 55 Hz on an Nvidia Titan X GPU. Input is an RGB-D video and 298 high-quality photos of 'Dr Johnson's house', London. With no wheelchair access to this floor, curators were keen to have their rooms digitized.*

Abstract

Our aim is to give users real-time free-viewpoint rendering of real indoor scenes, captured with off-the-shelf equipment such as a high-quality color camera and a commodity depth sensor. Image-based Rendering (IBR) can provide the realistic imagery required at real-time speed. For indoor scenes however, two challenges are especially prominent. First, the reconstructed 3D geometry must be compact, but faithful enough to respect occlusion relationships when viewed up close. Second, man-made materials call for view-dependent texturing, but using too many input photographs reduces performance. We customize a typical RGB-D 3D surface reconstruction pipeline to produce a coarse global 3D surface, and local, per-view geometry for each input image. Our tiled IBR preserves quality by economizing on the expected contributions that entire groups of input pixels make to a final image. The two components are designed to work together, giving real-time performance, while hardly sacrificing quality. Testing on a variety of challenging scenes shows that our inside-out IBR scales favorably with the number of input images.

Keywords: **Concepts:** •**Computing methodologies** → **Image manipulation;** *Computational photography;*

,

1 Introduction

The ability to capture and reproduce virtual versions of real indoor places is essential for many applications, such as virtual navigation of real-estate, museums, games, and safety training. While it is possible to obtain 3D reconstructions [Choi et al. 2015] and display them with texture [Waechter et al. 2014], this lacks view-dependent effects. Even with perfect geometry this approach looks artificial, as highlights are baked in or missing completely. Gradually, image-based rendering (IBR) is becoming an effective way to achieve both realism and interactivity. High-quality results already exist for small objects [Lensch et al. 2003] and for outdoor environments [Fehn 2004; Goesele et al. 2010; Chaurasia et al. 2013]. A key to recent success in IBR is the use of *per-view input image information*, such as custom meshes and super-pixel over-segmentation, which preserve depth boundaries even with imperfect 3D reconstructions.

Indoor scenes present specific challenges. While capturing outdoor scenes or views around an object involves "outside-in" viewing (see [Wei et al. 2014] for globally consistent 3D in that setting), interiors typically require an "inside-out" approach. For the latter, capturing similar numbers of photos yields far less overlap in scene coverage. This creates much larger parallax and many occlusions at close distances, resulting in two challenges: the need for higher-quality 3D reconstruction, and for capturing significantly more input photos, slowing down rendering when using per-view information.

We propose a new indoor IBR algorithm addressing both these challenges by combining indoor-friendly depth sensors and multi-view stereo (MVS) for improved reconstruction, and a scalable rendering algorithm which uses mesh simplification and tiling to accelerate free-viewpoint IBR of indoor scenes.

Depth sensors allow easy 3D acquisition of indoor scenes, but have several limitations, e. g., depth maps that do not align well with image edges, and low resolution compared with RGB cameras. Depth-sensor fusion algorithms [Newcombe et al. 2011; Nießner et al. 2013; Zhou et al. 2013; Choi et al. 2015] provide high-quality 3D reconstruction, but can fail in flat regions such as walls, ceilings and the floor or when viewing incandescent lights. In contrast, MVS

can achieve better results for textured zones and lights, and allows high-quality alignment of meshes with image edges. We carefully combine these two modalities to accurately align a global depth-sensor mesh to separately-captured images, specifically addressing – and significantly increasing – indoor IBR quality and speed.

One way to categorize previous free-viewpoint IBR methods is whether they use a global mesh [Buehler et al. 2001; Eisemann et al. 2008; Goesele et al. 2010], or, for more recent approaches, forward-projection of per-view information [Zitnick and Kang 2007; Kopf et al. 2013; Chaurasia et al. 2013; Ortiz-Cayon et al. 2015]. Per-view information compensates for insufficient accuracy of the global mesh, which can lack entire regions or contains incorrect geometry. For indoor scenes, depth sensors can estimate a globally "consistent" mesh [Newcombe et al. 2011; Choi et al. 2015], but still have errors at object boundaries, producing visual artifacts during IBR. Per-view information is still needed, but the number of images required for high quality is very high.

There are two key challenges for IBR of indoor scenes: combine the global 3D mesh with per-view geometry and render high-quality novel views using this geometry, without degrading performance. Our approach creates high-quality per-view meshes optimized to align with depth and image edges, which we subsequently simplify. Our rendering algorithm introduces a specialized tiling data structure, which compactly references the per-view data. We introduce an adaptive blending cost to combine several input views, and derive upper bounds of the cost for each tile. We use this cost to prioritize view-dependent data used to render the novel view, further accelerating rendering. In summary we present two main contributions:

- Careful merging of multi-modal sensor data for global scene reconstruction, which makes it easy to capture indoor scenes for IBR, followed by an algorithm to produce per-view meshes that respect view-specific depth and color discontinuities.
- A fast rendering algorithm allowing high-quality free-viewpoint IBR for indoor scenes, which uses a blending approach based on conservative cost bounds for quality, and uses mesh simplification together with tiled rendering to decouple speed from the number of input images.

We show results on a variety of complex indoor scenes, demonstrating that our approach provides high-quality results and can scale to a higher number of input images, compared to previous work.

2 Previous Work

Here we discuss the most closely related previous work in reconstruction and rendering. We refer to the excellent survey by Shum et al. [2008] for the broader context of IBR.

Geometry Reconstruction for IBR Several recent IBR methods [Goesele et al. 2010; Eisemann et al. 2008; Chaurasia et al. 2013; Ortiz-Cayon et al. 2015] use multi-view stereo (MVS) [Furukawa and Ponce 2010; Goesele et al. 2007] to reconstruct the 3D geometry of the scene, starting only with a sequence of photographs as input. In most cases, these methods use outdoor scenes as examples. Individual objects can be reconstructed at real-time rates with sufficient accuracy for IBR using visual hulls [Matusik et al. 2000]. Indoor scenes have been digitized for IBR, e. g., Sinha et al. [2009] focused on piecewise-planar geometry, while Furukawa et al. [2009] modeled shape using the Manhattan-world assumption. With manual intervention and a carefully crafted user interface, Sankar et al. [2012] reconstructed plausible indoor scenes, captured even on a mobile device.

For indoor scenes, consumer-level depth sensors [Newcombe et al. 2011; Henry et al. 2012; Choi et al. 2015] can be used to recover 3D information. Elaborate hardware has been used to methodically capture large interior spaces, room by room, producing data that is hard to use in interactive systems [Bahmutov et al. 2006]. In the Sun 3D data set [Xiao et al. 2013], 3D reconstruction is done using Structure from Motion and RGB-D images, but does not use digital camera images, which can result in imperfect reconstructions especially at depth silhouettes. Using stereo cues to improve depth sensor images is a well studied topic, surveyed by Nair et al. [2013]. These methods seldom fuse multiple depth images, and refine depth using precalibrated binocular stereo. In contrast, our per-view refinement method works with fused depth images and unstructured multi-view capture.

Existing 3D reconstruction methods suffer from various inaccuracies, including missing or spurious additional geometry, misalignment between geometry and image boundaries, and excessively smooth or noisy geometry; these result in distracting visual artifacts for IBR [Stich et al. 2011]. Combining textures from multiple views on a reconstructed object is challenging [Lensch et al. 2001]. Reconstruction-based solutions to these problems include warping images to match the geometry [Zhou and Koltun 2014], hiding seams between images while texturing [Waechter et al. 2014], or super-resolution approaches [Goldlücke et al. 2014]. In contrast, we carefully craft the 3D reconstruction to align RGB images with a global mesh, typically created by a depth sensor, then build per-view meshes to capture image and depth discontinuities with the accuracy needed for IBR. Large laser-scanned scenes can be rendered efficiently by converting them into local geometries, either into meshes to be rasterized [Arikan et al. 2014] or depth maps displayed with ray tracing [Arikan et al. 2016]. These methods apply texture by hiding seams between high-resolution photographs projected onto the local geometry, but only in Lambertian scenes. While they focus on performance, our approach *refines* global geometry into per-view geometries to improve quality and is designed to reproduce view-dependent effects such as highlights.

Image-based Rendering From the outset, Image-Based Rendering methods (e. g., [McMillan and Bishop 1995; Levoy and Hanrahan 1996; Gortler et al. 1996]) capture the entire visual information of the scene using photographs. The Light Field [Levoy and Hanrahan 1996] and the Lumigraph [Gortler et al. 1996] represent the dense information of light rays in a scene using different parameterizations. The Lumigraph introduced the idea of using geometry to assist in the generation of novel views. View-dependent texture mapping assumes the existence of a global mesh and all appearance variation is stored in textures [Debevec et al. 1998]. Surface light fields are another parametrization of a light field for scenes containing opaque surface with view-dependent appearance [Wood et al. 2000]. There have been several methods that compress and render such data efficiently [Chen et al. 2002; Vanhoey et al. 2013], while Gigaray lightfields [Birklbauer et al. 2013] use caching and tiling to accelerate computation. These methods operate on different data than ours: typically a single object or small baseline image sequences. High-quality IBR can be achieved through dense capture and by restricting the virtual camera to move on a plane [Aliaga et al. 2002] or to follow a track [Uyttendaele et al. 2004]. Our method supports free-viewpoint navigation and renders novel views that are substantially different from the input images.

The Unstructured Lumigraph Rendering (ULR) algorithm uses a set of complex weights [Buehler et al. 2001] to overcome limitations related to lack of geometry, input camera position, orientation and resolution. If the ULR approach is used per pixel, the cost of visiting all input pixels for each output pixel is prohibitive. Floating Textures overcome ghosting in ULR-like algorithms by using optical flow to improve blending, and also improve visibility [Eisemann et al. 2008]. ULR allows some degree of "free-viewpoint navigation"

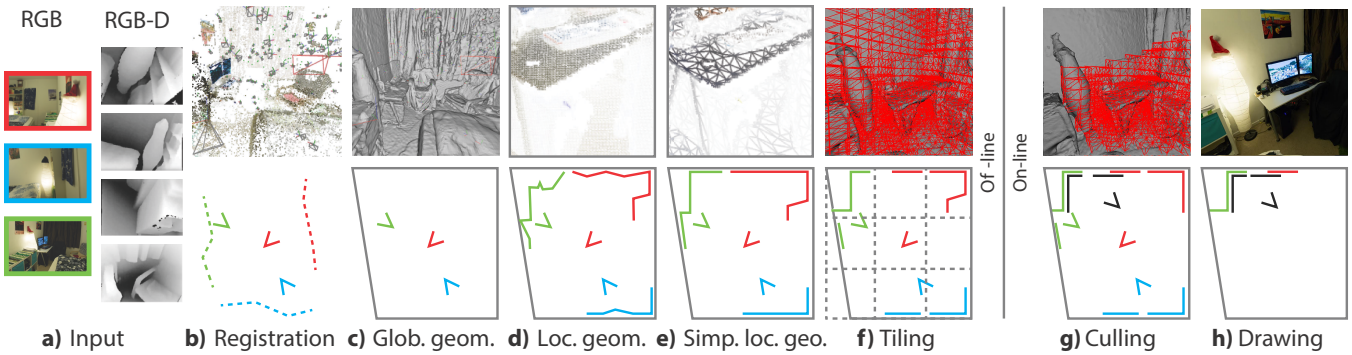

Figure 2: *Overview of our algorithm:* (a) *The user captures high-quality inside-pointing-outward RGB photos, shown here as three different camera poses. Separately, they also record a low-resolution RGB-D video by walking around the scene.* (b) *The high-resolution photos are used to reconstruct a sparse 3D point cloud, to which the depth-maps are registered.* (c) *All the depth maps are fused into a global geometry, which smooths away many important details, but captures a consensual surface approximation.* (d) *Re-using the high-res photos, we compute a high-quality local mesh for each one.* (e) *The many local meshes are simplified to reduce triangle count.* (f) *The global geometry is partitioned into tiles (in red) that organize the scene's visibility with respect to input views.* (g) *At run-time, the per-view meshes are culled, leaving only those predicted as relevant for rendering a novel view (pictured as a black camera).* (h) *The relevant per-view geometry is rendered.*

based on the global mesh which is used to *back-project* pixels in the novel view into the input images. Recently, Pujades et al. [2014] present an interesting analysis of ULR by modeling the uncertainty in geometric reconstruction, but it is not adapted to our real-time rendering context. Alternative solutions include Ambient Point Clouds [Goesele et al. 2010], which degrade image quality with a "non-photorealistic" look when reconstruction is unreliable, guided by epipolar constraints. However, in all the above cases imprecise or incomplete geometry and misalignment with input images results in significant visual artifacts in IBR, which worsen for indoor scenes.

A different class of approaches is based on the idea of using view-dependent information that preserves image discontinuities. These reduce artifacts at occlusion boundaries, by *forward-projecting* the over-segmented input images into the novel view [Zitnick and Kang 2007; Chaurasia et al. 2013; Ortiz-Cayon et al. 2015]. Bhat et al. [2007] use forward projection to enhance an existing video, allowing them to enforce temporal coherence and hide seams between input views using offline graph cuts. Kopf et al. [2013] forward project gradients and solve a Poisson problem to generate the image in the novel view, enabling smooth interpolation of reflections. Forward projection-based methods [Chaurasia et al. 2013; Ortiz-Cayon et al. 2015] produce high-quality free-viewpoint navigation mainly in outdoor scenes, typically when looking at a "backdrop" such as a building facade or a wall. For indoor scenes, forward-warping per-pixel geometry of the many input views required quickly becomes the computational bottleneck, given that a one-megapixel input image becomes a one-million vertex mesh.

A different – but direct – approach is to produce a novel view as an optimization [Fitzgibbon et al. 2005], such that the novel-view image matches the statistics of a set of examples. This approach has been demonstrated to work well for small changes of viewpoint and for dense image sets; it is however unclear how well the method adapts to the more challenging scenario we consider, with much larger changes in viewpoint and interactive rates. Initial results indicate that Deep Learning could be applicable to IBR [Flynn et al. 2016], but in its current state this method suffers from blurring and high computational cost. Commercially backed systems like Google Jump and Facebook Surround 360 are emerging, which achieve high-quality results by limiting the user to follow pre-defined paths. This keeps quality very high and makes on-set capture relatively easy; extending to our free-viewpoint context is much more challenging.

Finally, the quality of IBR critically depends on the quality and quantity of the images used. Many redundant or useless images can sometimes be worse than a smaller number of well-chosen views. Our tiled rendering approach is related to the view planning problem, applied to IBR [Vázquez et al. 2003], which can also be solved by putting a user into the loop [Davis et al. 2012]. We develop a compact and efficient solution, combining a pre-processing phase with a fast run-time step.

3 Overview

Our goal is to achieve real-time free-viewpoint image-based rendering of indoor scenes, using a collection of high-resolution digital color photographs and RGB-D video from a consumer-level depth sensor as input (Fig. 2a). Unlike traditional methods for RGB-D reconstruction [Newcombe et al. 2011; Dai et al. 2016] and texturing [Zhou and Koltun 2014; Waechter et al. 2014], we aim to reproduce view-dependent appearance such as highlights, and to render accurate images even in regions where a global 3D reconstruction of the scene has missing or inaccurate data.

Accurately aligning the photographs with the RGB-D images (Fig. 2b) and creating a global 3D reconstruction of the scene (Fig. 2c) is necessary for consistent rendering. However, these steps alone are not sufficient to achieve high quality with back-projection IBR (e.g., [Buehler et al. 2001; Eisemann et al. 2008]), since object boundaries in the photographs seldom align with the global geometry. Rather than relying on improved global geometry, we make a deliberate trade-off: To respect object boundaries we sacrifice global agreement and create per-view geometry for each high-resolution photograph (Fig. 2d). Now, we can render novel high-quality images by forward projecting per-view geometries into the novel view, and blending them using view-dependent blend weights.

Even with high-quality geometry, hundreds of input photographs are required to faithfully reproduce the view-dependent appearance of indoor scenes during free view-point navigation, given the many occlusions and large parallax at close distances. However, iterating over all photographs when rendering novel views is very expensive, especially for forward projection where each photograph corresponds to a per-view mesh with more than a million vertices. To overcome this problem, we present a culling strategy with three elements: Simplified per-view geometry (Fig. 2e), a tiling data structure to only render potentially visible geometry (Fig. 2f), and worst-case cost bounds to avoid rendering per-view geometry likely

to be discarded during view-dependent blending (Fig. 2g).

Combining the insights above, our approach works as follows: During offline pre-processing we first use Structure-from-Motion to align the photographs with the RGB-D images in the same coordinate system and create global geometry using off-the-shelf surface reconstruction (Fig. 2b & 2c, Sec. 4.1). Next, we create per-view geometry that aligns well with input image edges. We use the global geometry to guide an iterative region-growing multi-view stereo method that creates per-view depth maps for each input photograph (Fig. 2d, Sec. 4.2). We then convert the depth maps into a simplified mesh representation suitable for rendering (Fig. 2e, Sec. 4.3). Finally, we store the resulting meshes in a tiling data structure to implement our culling strategy (Fig. 2f, Sec. 5.2). During run-time, we query the data structure to find tiles of relevant per-view geometry needed to form the novel view (Fig. 2g, Sec. 5.2), and derive bounds on the blending cost per tile to allow prioritization. Then, we forward project these tiles into the novel view and blend them using adaptive blend weights that automatically balance blurring and banding (Fig. 2h, Sec. 5.1). We now proceed to explain the 3D reconstruction (Sec. 4) and the rendering (Sec. 5) phases in detail.

4 3D Reconstruction

In the absence of perfectly accurate global geometry, we instead seek global geometry with sufficient quality to serve as initialization for local per-view reconstruction. In general, image-based rendering methods strive to be robust against imperfect 3D geometry. However, some problems are difficult to fix with rendering alone, such as big holes, "flying" geometry and disjoint reconstructions.

Indoor scenes are harder than outdoor in many ways, but they do allow for the use of Kinect-style depth sensors. We use such a depth camera, along with a high-resolution digital color camera because taken together, they will allow us to reconstruct both flat-looking parts of scenes (e. g., ceilings and walls) and highly textured areas. In practice, good global reconstructions are still elusive. Scene *completeness* is one major challenge, and *accuracy* is the other. We found that global reconstructions under- or (less often) overestimate the volume of scene elements, so they fail to align with edges in the input views. Broadly speaking, global reconstruction is designed to reward the average agreement between the estimated 3D shape and all views. But even small inaccuracies in the camera poses conspire to erode 3D shapes. Our goal of free-viewpoint IBR is less forgiving about missed details, but quite compatible with global inconsistencies. We thus separate reconstruction into two goals: finding a global 3D structure that brings all data into the same coordinate system, and constructing local per-view geometry, which aligns well with image edges in just the nearby input images.

To achieve this, we first use the RGB-D images to reconstruct a consistent global geometry of the scene (Sec. 4.1), which we then refine into view-specific depth maps for each high-resolution camera using multi-view stereo (Sec. 4.2). Finally, we convert the depth maps into simplified meshes amenable to efficient rendering (Sec. 4.3).

4.1 Global Geometry

In this step, we align the RGB and RGB-D images and consolidate the depth information into one consensual surface geometry. We found this procedure to work best for our input data, in terms of scene completeness. However, other geometry reconstruction methods (e. g., BundleFusion [Dai et al. 2016], developed in parallel) might be worthwhile replacements, once available.

RGB-to-RGB-D Registration We estimate relative camera poses between the high-resolution photographs and the color component of the RGB-D images using standard Structure-from-Motion (SfM) software [Moulon et al. 2013], including radial distortion correction of the high-resolution photographs. A fundamental limitation with SfM is that it is unable to estimate the scale of the scene. Fortunately, it does provide us with the 3D locations of feature points and their 2D projections into the images (specifically the RGB-D images). Using this information, we align the SfM reconstruction with the (metric) scale of the RGB-D images: We use linear regression in a RANSAC loop to find the metric scaling factor that aligns the depths of feature points, projected into the RGB-D images, with the values stored in their depth maps.

Surface Reconstruction We project all pixels in the RGB-D images out into a unified point cloud. Note that any point between an RGB-D camera and the surfaces it can see (determined by its depth pixels) is likely to be an outlier; we call this a *visibility conflict*. Inspired by Furukawa and Ponce [2010], we use this insight to design a visibility filter that removes invalid points from the point cloud: If a point causes more than $n_{\max}$ visibility conflicts, we remove it from the cloud. As the depth captured by an RGB-D camera is less accurate further away, $n_{\max}$ is made a decreasing function of depth z in meters, i. e., $n_{\max} = \lfloor 10\text{m}/z \rfloor$. Finally, we estimate normals for the remaining points in the cloud using RANSAC plane-estimation, and fuse them into a global mesh using Screened Poisson Surface Reconstruction [Kazhdan and Hoppe 2013].

4.2 Local Per-view Geometry

For high-quality IBR, pixel-accurate alignment of depth with luminance edges is much more important than the precise depth value [Stich et al. 2011; Chaurasia et al. 2013]. We achieve this by creating a different depth map for every input view, which might deviate from the global mesh, but respects image edges.

We use the global mesh created in the previous step to guide the creation of these high-quality per-input-view depth maps, which we then use for rendering novel views. For this purpose we employ a region-growing multi-view stereo method to align the global geometry with edges in the high-resolution images. We designed this method to exploit the global mesh in three ways: 1) to provide a good initialization of the surfaces in the scene, 2) to derive a reliable reconstruction cost function for optimization and 3) to determine visibility when selecting views for evaluating photo-consistency.

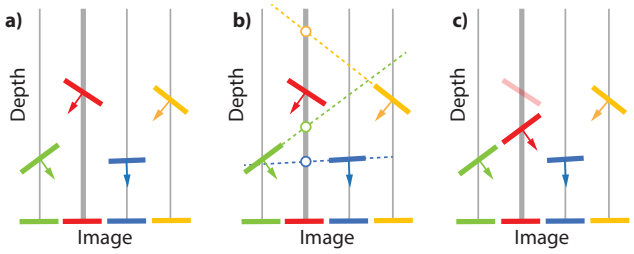

Figure 3: *The spatial propagation scheme of normal diffusion MVS. Starting from pixels associated with planes (depths and normals)* (a)*, we update the plane at a pixel (in red) by first extending the planes of neighboring pixels* (b)*. We replace the plane at a pixel if any of the extended planes has a lower reconstruction cost* (c)*. This is performed for all pixels in the image and repeated for 10 iterations.*

The objective for our region-growing method is to find a plane (i. e., a depth and a normal) with a low reconstruction cost (defined below) for each pixel in an input image. To this end, we initialize the planes with the global geometry and iteratively refine them using the spatial propagation scheme from normal diffusion multi-view

stereo (ND-MVS) [Galliani et al. 2015]. At every iteration, we create a set of plane candidates for each pixel by extending the planes of its neighbors, Fig. 3. Then, we replace the plane at a pixel with the candidate that has the lowest reconstruction cost. In all experiments, we perform 10 iterations of propagation. Please see Galliani et al. [2015] for exact details of the propagation scheme.

Our reconstruction cost $c(i)$ incorporates prior information from the global mesh with multi-view photo-consistency. More precisely,

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

is a sum of a photo-consistency, mesh, and smoothness terms.

We evaluate the photo-consistency $c_{\mathrm{p}}(i)$ at a pixel by reprojecting a small (3×3) patch around the pixel via the candidate plane i into other images, where we compare both the colors and image gradients using the formulation in [Galliani et al. 2015]. We use the global mesh to reliably select these other images. First, we rank the images based on the overlap they have with the input image when both are projected onto the global mesh. Then, we reproject the patch into the top six images not occluded by the global mesh.

The mesh prior $c_{\mathrm{m}}(i)$ encourages candidate planes to stay close to the global mesh $\mathcal{M}$. This is important for texture-less regions, where photo-consistency does not provide any extra information. Formally, where $\mathbf{p}_i$ is the world-space location of the candidate plane and $\mathbf{m}$ is a point on the global mesh,

$$c_{\mathrm{m}}(i) = \min_{\mathbf{m} \in \mathcal{M}} \frac{|\mathbf{p}_i - \mathbf{m}|^2}{(50\ \mathrm{mm})^2}. \tag{2}$$

The smoothness term $c_{\mathrm{s}}(i)$ encourages the resulting depth map to be smooth for neighboring pixels i and j of similar color. Formally,

$$c_{\mathrm{s}}(i) = \sum_{j \in \mathcal{N}(i)} \exp\left(\frac{-|c_i - c_j|^2}{2\sigma_{\mathrm{c}}^2}\right) \min\left(0.1, \frac{|\mathbf{p}_i - \mathbf{p}_j|_2^2}{(100\ \mathrm{mm})^2}\right), \tag{3}$$

where $\mathcal{N}(i)$ is the set of neighboring pixels to i and c_i and c_j are the RGB colors at pixel i and j. In all experiments, we set $\sigma_{\mathrm{c}} = 0.1$.

As a post-process, we remove pixels at unusual depths with the visibility filter described in Sec. 4.1, using $n_{\max} = 8$. Finally, we use single-view information to align the depth maps to image edges where multi-view reconstruction fails. We filter each depth map with a cross-bilateral weighted median filter [Ma et al. 2013] (guided by color, $\sigma_c = 0.1$) with a 4 px standard deviation.

4.3 Representation

Instead of storing full per-pixel per-input view depth maps, we create a compressed, per-view polygonal mesh for faster drawing and reduced storage. To this end, we convert every pixel in the depth map to a 3D vertex and form a grid mesh by connecting neighboring vertices with triangles. To avoid connecting foreground and background layers at occlusion boundaries, we do not create triangles at edges where the depth difference between the neighboring pixels is larger than 10 cm. We call those edges *splits* in the mesh.

Meshes made from depth maps have saw-tooth artifacts at discontinuities, which worsen for surfaces at glancing angles. To avoid this, we replace the depth of pixels at splits by a cross-bilateral average of their neighborhood depths, guided by both RGB and depth.

Finally, off-the-shelf mesh simplification [Garland and Heckbert 1997] is used to reduce the number of triangles in the mesh. We run the simplification until the mesh has an edge longer than 75 pixels in its input view, which typically yields a 50× to 100× reduction in the number of triangles. We flag vertices next to a split as boundary conditions that should not be altered during simplification. As can be seen in Fig. 9, this preserves crisp occlusion boundaries.

5 Rendering

We can now use the simplified per-view meshes for indoor IBR by blending many input views. To achieve high quality, including view-dependent effects such as highlights, we introduce an adaptive blending cost (Sec. 5.1). Directly rendering the per-view meshes is however prohibitively expensive, so we introduce two acceleration methods (Sec. 5.2). First, we present a spatial tiling data structure, limiting the per-view meshes used to render a given novel view. The number of tiles drawn is further reduced using a priority scheme, by deriving a bound on the blending cost for each tile. Details of our GPU rendering implementation (Sec. 5.3) conclude this section.

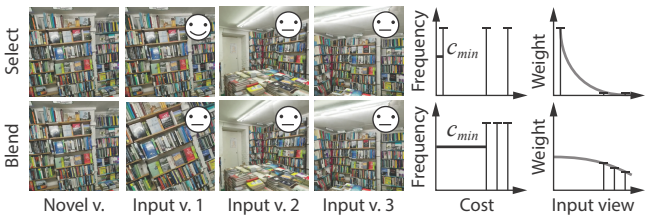

Figure 4: *Adaptive bandwidth selection for two cases: One with a single good input view* (top) *and the other with many mediocre ones* (bottom). *In the "select" case, the first view matches the novel view well and has a low IBR cost. With a low minimum cost the filter bandwidth becomes narrow, assigning most of the blend weight to the best view. In the "blend" case, all views have a high cost. Consequently, the minimum cost is high and the filter bandwidth becomes wide, blending all input views with nearly equal weights.*

5.1 Blending

We form the novel view by locally blending the input images. Choosing appropriate blend weights is key to capturing view-dependent effects while minimizing ghosting artifacts. We do this by first resolving visibility of the novel view. Then, we compute a per-pixel cost for every input view. Finally, we reconstruct the novel view by adaptively blending the visible input views based on this cost.

Resolving Visibility As we form the final image by blending multiple per-view geometries, we cannot resolve visibility with a traditional depth test that only recovers the front-most surface. Instead we use a fuzzy depth test (detailed in Sec. 5.3), allowing us to consider all per-view geometries that represent the same surface, even if they do not perfectly align.

Cost We derive an IBR cost that prioritizes which input views to blend for every pixel in the novel view. This is a function $c_{\mathrm{IBR}}(\mathbf{y}_i, \mathbf{y}_n, \mathbf{x})$ of the camera position $\mathbf{y}_i$ for the input view i, the camera position $\mathbf{y}_n$ for the novel view and a location $\mathbf{x}$ on the per-view geometry. Similarly to ULR [Buehler et al. 2001] our cost is a weighted sum of two terms $c_{\mathrm{IBR}}(\mathbf{y}_i, \mathbf{y}_n, \mathbf{x}) = (1-\gamma)c_{\mathrm{a}} + \gamma c_{\mathrm{d}}$, defined as follows:

$$c_{\mathrm{a}} = \alpha(\mathbf{y}_i \to \mathbf{x} \to \mathbf{y}_n) \qquad \text{(Angle term)} \tag{4}$$

$$c_{\mathrm{d}} = \max\left(0, 1 - \frac{||\mathbf{y}_n - \mathbf{x}||}{||\mathbf{y}_i - \mathbf{x}||}\right) \qquad \text{(Distance term)} \tag{5}$$

The *angle term* is the angle between the direction vectors from $\mathbf{x}$ towards the camera locations $\mathbf{y}_i$ and $\mathbf{y}_n$. This helps to reproduce view-dependent effects, as it prioritizes input views observing the scene from viewpoints similar to the novel view. The *distance term* depends on the ratio of the distance between the current position $\mathbf{x}$ and the input view $\mathbf{y}_i$ resp. the novel view $\mathbf{y}_n$. This reduces blur

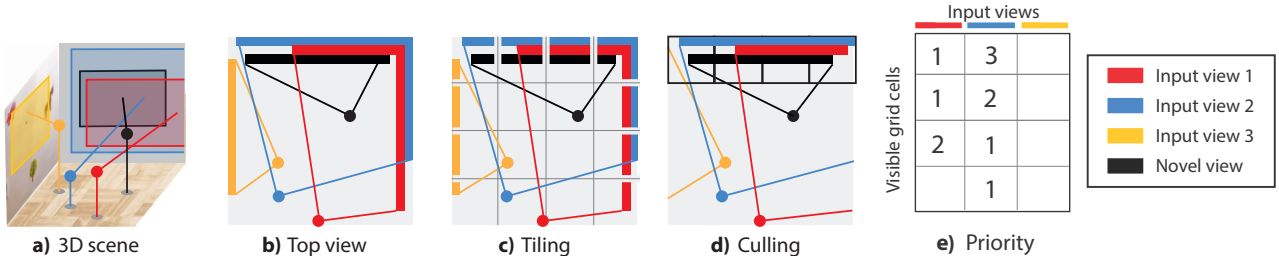

Figure 5: *The tiling procedure, for a 3D scene with a single room* (a)*, acquired from three input views* (yellow, red, blue) *to rendered in the novel view (in black). With a top-down perspective* (b)*, we see that the blue and red input cameras can be seen in the black novel view, while the yellow one does not. Our method partitions the scene into a grid and splits the geometry into tiles* (c)*. Now, tiles (e. g., all yellow ones and part of red and blue) not visible in the novel view get removed* (d)*. The table in* (e) *shows the priority of each input view tile (columns) for all grid cells visible in the novel view (rows). Table entries are blank wherever the input view does not have a tile in the corresponding grid cell.*

(undersampling) in the image by penalizing input views far from the surfaces visible in the novel view. Input views closer than the novel view are not penalized, and we correct for oversampling using mip maps and anisotropic filtering. In all experiments, we set $\gamma = 0.1$.

Bandwidth Selection We form the novel view by blending together the input views using blend weights determined by IBR cost. Specifically, we compute the blend weight $w_i = \exp(-c_i/\sigma)$ for an input view i by applying an exponential filter kernel to the IBR cost $c_i = c_{\text{IBR}}(\mathbf{y}_i, \mathbf{y}_n, \mathbf{x})$. This implies a trade-off illustrated in Fig. 4. We can choose a wide filter bandwidth σ resulting in blurring (i. e., giving all views a similar weight) or a narrow bandwidth implying banding (i. e., giving only one view a very large weight). Our key observation is that no single weighting function can capture two conditions that happen in practice: At one extreme, if many similar but incorrect views are available we need to select a wide bandwidth, i. e., to blur many images. Going for a single, but imperfect view would produce a sharp, but wrong result. At the other extreme, one view is much better than the others. Here, only this input view should be selected and others should not receive any weight as blurring many views would spoil the good one.

Similarly to variable kernel density estimation, we achieve this trade-off by letting the filter bandwidth depend on the local costs of the input views. We adaptively set the filter bandwidth by scaling the filter bandwidth with the IBR cost c_{min} of the best novel view. In other words $\sigma = \sigma_k c_{min}$, where σ_k (set to 0.33 in all experiments) controls how quickly we transition from blending the input views to selecting the best one.

5.2 Tiled rendering

Even with the simplified mesh representation, drawing the per-view geometry of every input view into the novel view is slow. The key idea here is to avoid iterating over all input views when forming the novel view. We achieve this by operating on *tiles*, i. e., entire groups of input triangles and novel-view pixels. Similar ideas have been used in real-time graphics with tile-based shading [Olsson and Assarsson 2011].

The idea is shown in Fig. 5. Consider an example of four views (black, red, blue and orange) observing a room as seen in Fig. 5a. The black camera is the novel view, the others are input views. Fig. 5b is a top-view of the same scene. For every pixel in the novel view (black), we need to compute IBR weights for all input views (yellow, red and blue) so we can determine which images to fetch RGB values from. With forward warping, this means that all triangles from each input view need to be drawn into the novel view.

To reduce the computational cost, we partition the scene into a regular 3D grid as shown in Fig. 5. As a pre-process (Fig. 5c), we associate all triangles in the input views with the grid cells they intersect. We refer to the unique pairing of a grid cell with an input view as an *input tile*, i. e., the collection of triangles from a single input view that intersect a grid cell. At run-time, we save effort by only rendering input tiles visible to the novel-view (tiling and culling). In Fig. 5d, we see that this allows us to ignore the yellow view altogether as well as parts of the red and blue views.

However, because of our view-dependent blending, only a few visible input tiles will actually contribute to the final image. We therefore strive to render a small subset of the input tiles that ensures a low IBR cost for all pixels in the novel view. To achieve this, we sort the input tiles according to their worst-case IBR cost and draw only the best ones in each visible grid cell. The exact number varies between grid cells, as we stop drawing input tiles when we determine that a cell has been sufficiently covered. Predicting the worst-case IBR cost is part of our contribution, which focuses the rendering effort to the input tiles that actually affect the final image.

We next discuss all steps in detail.

Tiling and Culling We store the input tiles in a 3D grid with a resolution of $32 \times 32 \times 32$ that covers the scene. During preprocessing, we create the input tiles by conservatively voxelizing the per-view geometry into the grid. As a consequence, each triangle in the input geometry may belong to multiple input tiles.

At run-time, we need to find the grid cells visible to the novel view. This is done in two passes. First, we draw the coarse global geometry into the framebuffer of the novel view. Second, we process all framebuffer pixels in parallel and mark all grid cells they intersect as visible using an atomic operation. To account for the mismatch between the coarse global geometry and the per-view input geometry, we perform this intersection conservatively by inflating each pixel in the framebuffer into a cube with a side length of 5 cm.

Priorization To select which input tiles to render, we sort them within each visible grid cell according to their worst-case IBR cost, i.e., an upper bound for $c_{\text{IBR}}(\mathbf{y}_i, \mathbf{y}_n, \mathbf{x})$ between the input view $\mathbf{y}_i$ and the novel view $\mathbf{y}_n$ where $\mathbf{x}$ is allowed to be anywhere in the grid cell. Concretely, $\mathbf{x}$ is inside an axis-aligned bounding box $(\mathbf{x}_{\min}, \mathbf{x}_{\max})$, see Fig. 6a. Using separately computed upper bounds for the angle term $c_{\text{a}} \leq c_{\text{a}}^{max}$ and the distance term $c_{\text{d}} \leq c_{\text{d}}^{max}$, we form the upper bound $c_{\text{IBR}}(\mathbf{y}_i, \mathbf{y}_n, \mathbf{x}) \leq (1-\gamma)c_{\text{a}}^{max} + \gamma c_{\text{d}}^{max}$.

To find an upper bound for $c_{\text{a}} = \alpha(\mathbf{y}_i \to \mathbf{x} \to \mathbf{y}_n)$, we exploit that the angles in a triangle sum to π radians. Consider the triangle between $\mathbf{x}, \mathbf{y}_i$ and $\mathbf{y}_n$. We find lower bounds $\alpha_{\min}(\mathbf{y}_{\text{i}})$ and $\alpha_{\min}(\mathbf{y}_{\text{n}})$

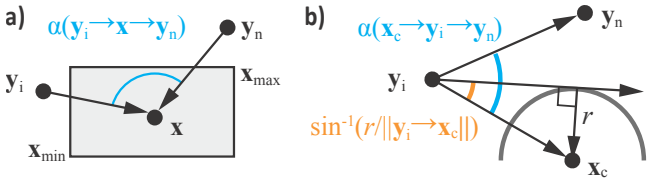

Figure 6: *We derive the upper bound on the angle $\alpha(\mathbf{y}_i \rightarrow \mathbf{x} \rightarrow \mathbf{y}_n)$, where $\mathbf{y}_i$ is an input view, $\mathbf{y}_n$ the novel view and $\mathbf{x}$ an arbitrary point in a grid cell* (a, Eq. 6). *We find the bound implicitly through lower bounds for the other angles in the triangle. Here, we use the bounding sphere around the grid cell (center $\mathbf{x}_c$, radius r) to bound $\alpha(\mathbf{x} \rightarrow \mathbf{y}_i \rightarrow \mathbf{y}_n)$ (b,* Eq. 7).

for the other two angles in the triangle and form the upper bound:

$$c_{\mathrm{a}}^{max} = \pi - \alpha_{\min}(\mathbf{y}_{\mathrm{i}}) - \alpha_{\min}(\mathbf{y}_{\mathrm{n}}). \quad (6)$$

Fig. 6b shows how we compute the lower bound $\alpha_{min}(\mathbf{y}_i)$ for one of the other angles $\alpha(\mathbf{x} \rightarrow \mathbf{y}_i \rightarrow \mathbf{y}_n)$. First, we convert the bounding box $(\mathbf{x}_{\min}, \mathbf{x}_{\max})$ to its encompassing bounding sphere centered at $\mathbf{x}_c$ with the radius $\mathbf{r}$. Now,

$$\alpha_{\min}(\mathbf{y}_{\mathrm{i}}) = \alpha(\mathbf{x}_{\mathrm{c}} \rightarrow \mathbf{y}_{\mathrm{i}} \rightarrow \mathbf{y}_{\mathrm{n}}) - \sin^{-1}\left(\frac{r}{||\mathbf{y}_{\mathrm{i}} - \mathbf{x}_{\mathrm{c}}||}\right). \quad (7)$$

We derive an upper bound for the distance term c_{d} based on the maximum distance $d_{\max}(\mathbf{y})$ and minimum distance $d_{\min}(\mathbf{y})$ between a point $\mathbf{y}$ and a bounding box $(\mathbf{x}_{\min}, \mathbf{x}_{\max})$. Using $d_{\min}(\mathbf{y}_n)$ as lower bound for the numerator and $d_{\max}(\mathbf{y}_i)$ as an upper bound for the denominator, we see that

$$c_{\mathrm{d}}^{max} = \max\left(0, 1 - \frac{d_{\min}(\mathbf{y}_{\mathrm{n}})}{d_{\max}(\mathbf{y}_{\mathrm{i}})}\right). \quad (8)$$

Drawing Once all input tiles have been sorted, we determine how many are to be rendered from each grid cell. Ideally, we would conserve bandwidth by rendering as few input tiles as possible. However, this may result in holes in the novel view; the triangles in an input tile are not guaranteed to cover a grid cell, e. g., due to splits (Sec. 4.3) or image boundaries. We refer to an input tile without any of these discontinuities as *complete*.

For each grid cell, we render the sorted input tiles until three complete tiles or a maximum of 12 input tiles have been rendered. We found 12 to be a good trade-off between completeness and compactness. In our experiments this typically reduces the number of drawn primitives to fewer than 10 % of the original per-view meshes.

5.3 Implementation

Four geometry passes are required to render a novel view. Deferred shading cannot be used to avoid multiple passes, as the color of every output pixel is determined by blending several input primitives.

As described in Sec. 5.2, the first pass renders the global geometry to find the visible grid cells. These cells are downloaded to the CPU, which prioritizes the input tiles to render in the remaining three passes. We store the geometry associated with all input tiles in a single large vertex buffer object, allowing us to perform the other passes using a single `glDrawIndirect` OpenGL instruction.

The final three passes implement the blending algorithm described in Sec. 5.1. First, we recover the front-most surface with a depth pre-pass, which enables us to perform a fuzzy depth test: In the remaining two passes any surface fragment far enough ($10\,cm$) from the front-most surface is discarded. The second pass finds the minimum IBR cost c_{min} of all input views that project onto each pixel. This is required for the per-pixel filter bandwidth σ explained in Sec. 5.1. Finally, the last pass forms the image, blending together the input views using the bandwidths computed in the previous pass.

6 Experiments

We compare our approach to existing baseline algorithms, using published code wherever possible. Most existing rendering systems were meant for outside-in scenes, so results are presented on our own data. Here and in the supplemental video, we provide qualitative comparisons of our reconstruction and rendering phases, and quantitative results for speed tests.

Data Sets We captured our own scenes as we needed both high-quality photos and depth-maps with good coverage of all interesting surfaces in a variety of indoor environments. We captured 150-300 RAW photos using a digital camera (Sony NEX-C3 at 1228×816 or Canon EOS 550D at 1296×864). We also recorded an RGB-D video using the Asus Xtion PRO depth-camera, and sub-sampled in time to yield 150-450 640×480 RGB-D images. In a pre-processing step, images were color-harmonized using Adobe Lightroom.

In Fig. 1, 7, 8 and the supplemental video we show results in the following scenes:

CREEPY ATTIC: A small ($5 \times 4 \times 4$ m^3) attic in an old building containing textiles and artwork. There is a prominent object (doll on a chair) in the middle of the room, which clearly shows the need for per-input view local geometry since under- or overestimated global geometry results in clear IBR artifacts.

DORM ROOM: A small bedroom scene ($4 \times 4 \times 5$ m^3) in a student dorm with prominent textureless walls.

MUSEUM: A preserved house from the 17th century. This is a large scene ($16 \times 6 \times 5$ m^3) exhibiting textureless walls, glossy tabletops and paintings behind mirror-reflective glass.

PLAYROOM: A medium-sized ($6 \times 6 \times 5$ m^3) livingroom cluttered with toys for young children. Aside from large, textureless surfaces this scene contains many small geometric details that cannot easily be captured using geometry reconstruction alone.

BOOK SHOP: A large ($11 \times 9 \times 5$ m^3) basement in a book shop with difficult occlusion characteristics (aisles separated by bookshelves). The books present a challenge for reconstruction and rendering as they are often non-diffuse and textured with high-frequency details.

READING CORNER: A small ($7 \times 4 \times 4$ m^3) reading corner in an academic book store. The scene contains a leather chair with strong view-dependent effects.

Rendering Comparison In the accompanying video and Fig. 8 We compare our algorithm to [Ortiz-Cayon et al. 2015], and also to ULR [Buehler et al. 2001] with improved visibility akin to floating textures [Eisemann et al. 2008]. Compared to [Ortiz-Cayon et al. 2015], our mesh-based approach tends to preserve complex shape boundaries better since super-pixels are warped independently in their method. This is illustrated in the left side of Fig. 8, where the arm and leg of the doll show severe ghosting using [Ortiz-Cayon et al. 2015]. Compared to ULR, our approach preserves the shape of detailed objects (the doll, middle of Fig. 8) or hard-to-capture objects like the lamp (Fig. 8 on the right).

Geometry Comparison In Fig. 9 we compare the geometry reconstructions from different components of our system. We see that the global reconstruction alone is complete but tends to over- or underestimate the size of foreground objects. We include ND-MVS [Galliani et al. 2015] (Local only) as an upper bound for the

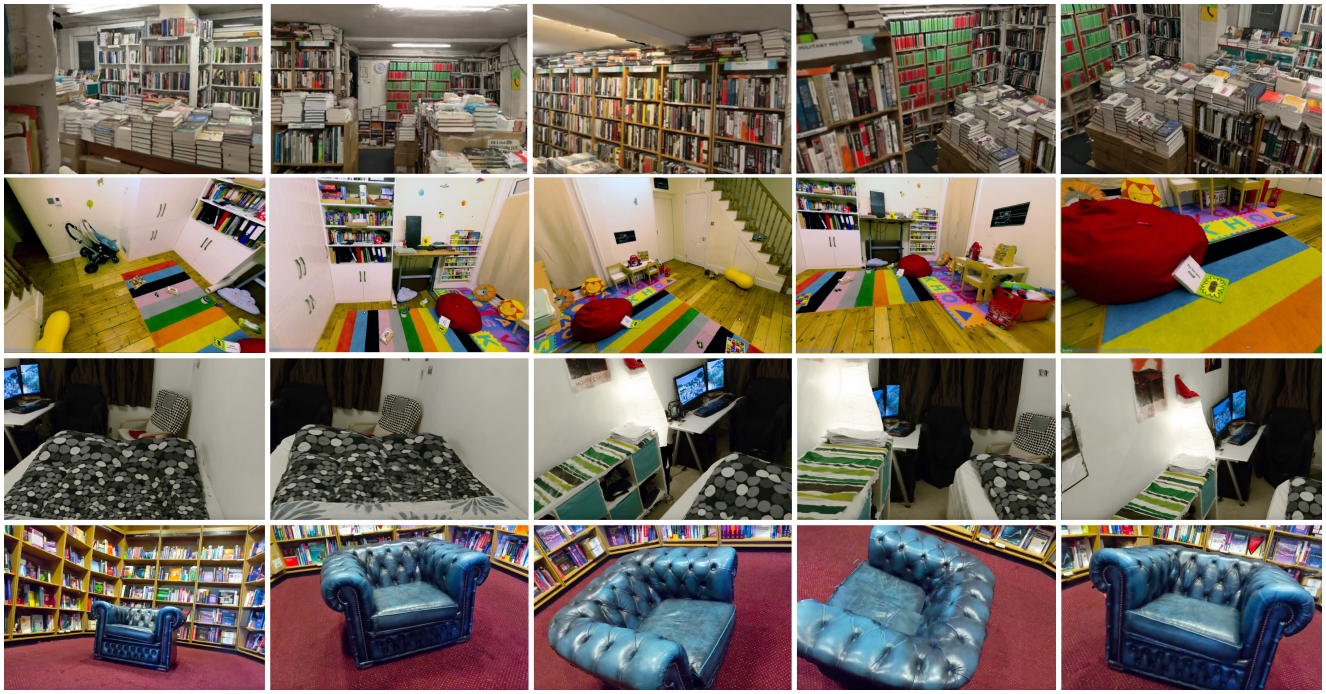

Figure 7: *Example results from the supplemental video (top to bottom):* BOOKSHOP, PLAYROOM, DORM ROOM *and* READING CORNER.

Figure 8: *Two example differences between ULR [Buehler et al. 2001] versus our approach, and Ortiz-Cayonet al. [2015] and our approach.*

quality which can be achieved using local geometry reconstruction alone, i. e., without the mesh prior and the improved visibility tests in Sec. 4.2. This method uses stochastic search to find the planes in the scene as it does not rely on global geometry for initialization. In general it preserves details better than the global reconstruction, but is often incomplete (shown as white) as textureless regions are difficult to reconstruct using multi-view stereo. The geometry produced by our local per-view refinement (Sec. 4.2) is complete and aligns well with image edges. Note how it corrects occlusion boundaries grossly misestimated in the global reconstructions (circled in black): The pillar (Fig. 9a); the cloth (Fig. 9b); the chair legs (Fig. 9c and 9d); consistent vertical edges of a bookshelf (Fig. 9e) and the back of the armchair (Fig. 9f). Correctly reconstructing these boundaries is important for IBR, as getting them wrong causes foreground color to be displayed on background geometry or vice-versa.

Performance Tbl. 1 shows show statistics of our scenes and rendering. We measured the average frame time at 1080p in each scene as the virtual camera moves along a predetermined path. Without tiling, rendering takes 31-60 ms on our high-end machine (Desktop PC; Nvidia GTX Titan X), and 209-388 ms on the low-end machine (Laptop; Nvidia GTX 660M). Tiling provides a speedup of $1.6\times$ to $3\times$ depending on the scene and hardware. Interestingly, tiling reduces the number of rendered triangles more drastically than frame time. This is explained by the overhead from the tiling process itself, particularly from prioritization; the only CPU component of our rendering algorithm. Fig. 11 shows how the performance of our tiled algorithm scales with the number of input images, demonstrating a sub-linear trend where the frame time plateaus after 150 images.

7 Conclusion and Future Work

We have presented a new IBR method that first builds a high-quality global mesh from a depth sensor, and then registers high-resolution photos to this mesh. We then build per-input-view meshes guided by the global reconstruction. Our approach combines a global/local representation of the scene to introduce a novel IBR algorithm which introduces a new blending approach, a compact representation of the per-input-view meshes and uses a tiled rendering algorithm which scales well with the number of input images.

Limitations Our approach is ultimately limited by the quality of the initial capture; evidently SfM needs to succeed for all the input images for our method to work. We suffer from the same limitations as all 3D reconstruction methods, and in particular for scenes with glass, which are problematic for modern depth cameras [Choi et al. 2016]. Even in easier cases, feedback during capture would be helpful, since unobserved regions are usually completed as large blobby surfaces by the Poisson reconstruction. Capture problems

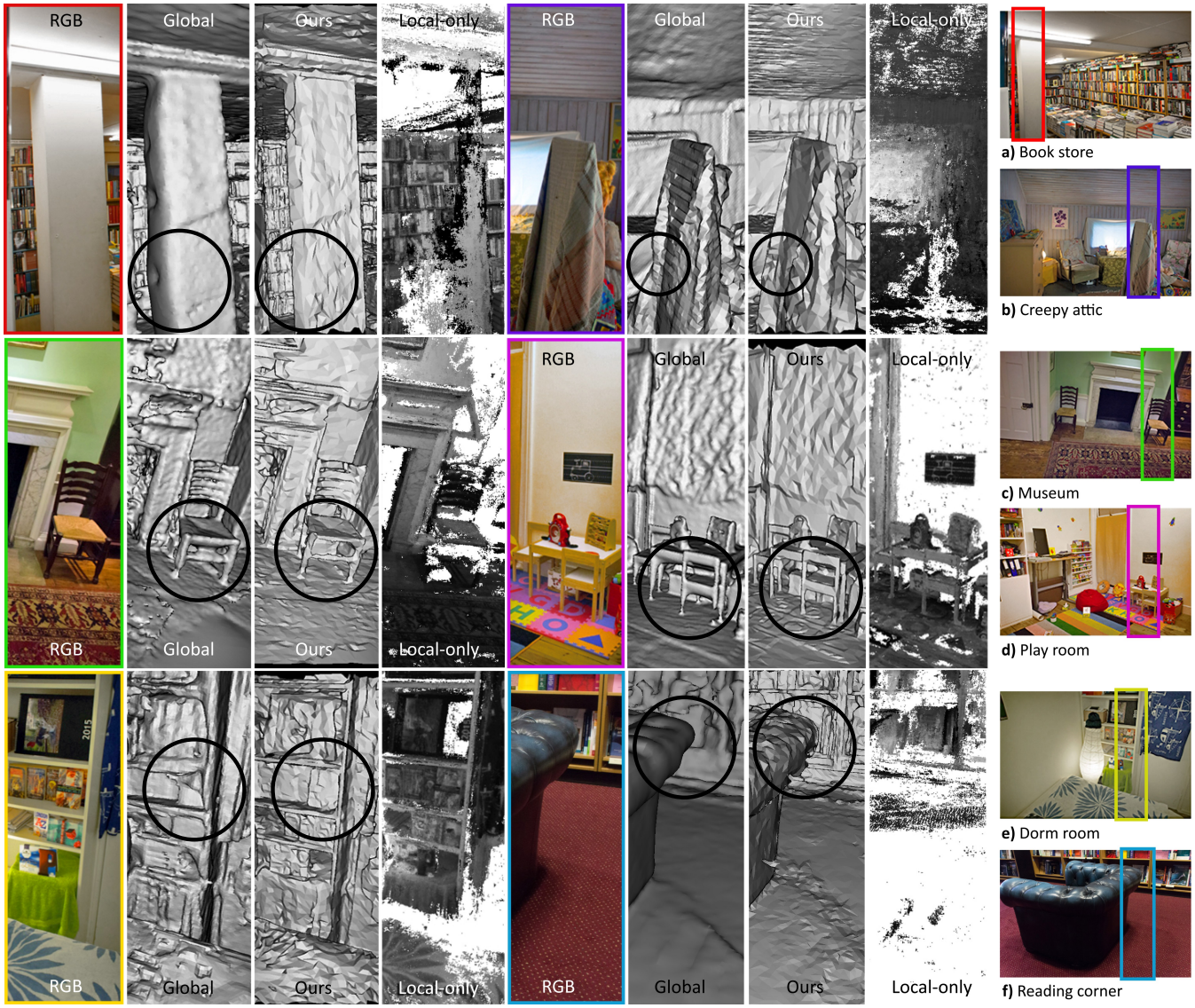

Figure 9: *Different geometry reconstructions. In each scene we compare in the highlighted inset: High-quality RGB; Global reconstruction (Sec. 4.1); Our local-global reconstruction (Sec. 4.2); ND-MVS [Galliani et al. 2015] (Local-only), an upper bound for local-only reconstruction quality. Our local-global approach combines the best of both; complete reconstructions that align well with image edges (see the black circles).*

could possibly be addressed with specialized on-set feedback about the capture progress, in the spirit of [Davis et al. 2012]. We show typical artifacts due to incorrect reconstruction in Fig. 10.

Handling much larger scenes with thousands of input photographs is currently a challenge. A clever disk and main memory caching scheme is required. Finally, our method is designed for relatively wide-baseline data sets. For light-field like densities, the CPU overhead for treating the tiling structure could become a bottleneck.

Future Work Our tiling approach could be adapted to superpixel-based methods [Chaurasia et al. 2013; Ortiz-Cayon et al. 2015], improving their performance and quality. Our approach is well-suited to indoor scenes, which present specific difficulties, but are well-adapted to depth-sensor capture. However, it is very likely that our rendering method will benefit outdoor scenes, especially in the context of wide-angle viewing, e. g., for head-mounted displays, which requires a large number of input images.

Acknowledgements

Thanks to Dr Johnson's Museum, Judd Books, Waterstones on Gower Street, and Kathryn Emmett-Brostow for being so accommodating with our filming. Thanks to Rodrigo Ortiz-Cayon for computing the comparisons. The authors are grateful for the support of EU project CR-PLAY (no 611089) `www.cr-play.eu`, EPSRC EP/K023578/1, and project SEMAPOLIS (ANR-13-CORD-0003).

References

ALIAGA, D. G., FUNKHOUSER, T., YANOVSKY, D., AND CARLBOM, I. 2002. Sea of images. In *Vis.*, IEEE, 331–338.

ARIKAN, M., PREINER, R., SCHEIBLAUER, C., JESCHKE, S., AND WIMMER, M. 2014. Large-scale point-cloud visualization through localized textured surface reconstruction. *IEEE Trans. Vis. Comput. Graphics 20*, 9, 1280–1292.

Table 1: *Evaluation of our tiled IBR in six scenes with different levels of complexity. In each scene, we record the number of primitives needed when using our tiled method, and contrast it to a brute-force method rendering all local meshes. We render at 1080p and measure frame time in milliseconds, both on a high-end Machine A (Desktop with a Nvidia GTX Titan X) and a low-end Machine B (Laptop with a Nvidia GTX 660M).*

Scene	Input		Tris/frame		Time					
					Machine A			Machine B		
	RGB	Depth	Tiling	No tiling	Tiling	No tiling	Speed-up	Tiling	No tiling	Speed-up
READING CORNER	166	167	0.70 M	6.30 M	16.0 ms	31.1 ms	1.9×	109.0 ms	209.6 ms	1.9×
CREEPY ATTIC	223	373	0.32 M	6.70 M	15.1 ms	35.3 ms	2.3×	66.60 ms	219.6 ms	3.3×
DORM ROOM	201	312	0.35 M	7.90 M	15.4 ms	45.9 ms	3.0×	69.80 ms	289.7 ms	4.1×
PLAYROOM	231	409	0.40 M	8.40 M	15.4 ms	40.8 ms	2.7×	81.70 ms	269.7 ms	3.3×
BOOK SHOP	271	438	0.90 M	14.7 M	22.1 ms	55.9 ms	2.5×	140.0 ms	330.5 ms	2.4×
MUSEUM	298	450	0.60 M	16.8 M	18.2 ms	60.5 ms	3.3×	119.4 ms	388.2 ms	3.3×

Figure 10: *Two typical limitations of our approach: low-resolution texture because cameras that saw these books were far away* (left)*, and missing geometry* (right) *where the cameras did not look.*

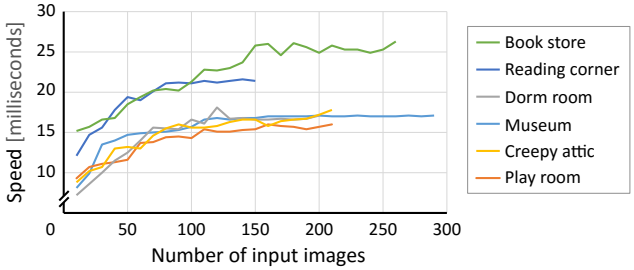

Figure 11: *Performance scaling at 1080p with Nvidia GTX Titan X. We fix the camera position in each scene and measure performance of our tiled rendering as the number of input images increases.*

ARIKAN, M., PREINER, R., AND WIMMER, M. 2016. Multi-depth-map raytracing for efficient large-scene reconstruction. *IEEE Trans. Vis. Comput. Graphics 22*, 2, 1127–1137.

BAHMUTOV, G., POPESCU, V., AND MUDURE, M. 2006. Efficient large scale acquisition of building interiors. *Comp. Graph Forum 25*, 3, 655–662.

BHAT, P., ZITNICK, C. L., SNAVELY, N., AGARWALA, A., AGRAWALA, M., COHEN, M., CURLESS, B., AND KANG, S. B. 2007. Using photographs to enhance videos of a static scene. In *EGSR*, Eurographics Association, 327–338.

BIRKLBAUER, C., OPELT, S., AND BIMBER, O. 2013. Rendering gigaray light fields. *Comp. Graph. Forum 32*, 2, 469–478.

BUEHLER, C., BOSSE, M., MCMILLAN, L., GORTLER, S., AND COHEN, M. 2001. Unstructured lumigraph rendering. In *SIGGRAPH*, ACM, 425–432.

CHAURASIA, G., DUCHENE, S., SORKINE-HORNUNG, O., AND DRETTAKIS, G. 2013. Depth synthesis and local warps for plausible image-based navigation. *ACM Trans. Graph. 32*, 3, 30:1–30:12.

CHEN, W.-C., BOUGUET, J.-Y., CHU, M. H., AND GRZESZCZUK, R. 2002. Light field mapping: efficient representation and hardware rendering of surface light fields. *ACM Trans. Graph. 21*, 3, 447–456.

CHOI, S., ZHOU, Q.-Y., AND KOLTUN, V. 2015. Robust reconstruction of indoor scenes. In *CVPR*, IEEE, 5556–5565.

CHOI, S., ZHOU, Q.-Y., MILLER, S., AND KOLTUN, V. 2016. A large dataset of object scans. *arXiv 1602.02481*.

DAI, A., NIESSNER, M., ZOLLHÖFER, M., IZADI, S., AND THEOBALT, C. 2016. BundleFusion: Real-time globally consistent 3D reconstruction using online surface re-integration. *arXiv 1604.01093*.

DAVIS, A., LEVOY, M., AND DURAND, F. 2012. Unstructured light fields. *Comp. Graph. Forum 31*, 2, 305–314.

DEBEVEC, P., YU, Y., AND BORSHUKOV, G. 1998. Efficient view-dependent image-based rendering with projective texture-mapping. In *Rendering Workshop*, Springer, 105–116.

EISEMANN, M., DE DECKER, B., MAGNOR, M., BEKAERT, P., DE AGUIAR, E., AHMED, N., THEOBALT, C., AND SELLENT, A. 2008. Floating textures. *Comp. Graph. Forum 27*, 2, 409–418.

FEHN, C. 2004. Depth-image-based rendering (DIBR), compression, and transmission for a new approach on 3D-TV. In *SPIE 5291*, 93–104.

FITZGIBBON, A., WEXLER, Y., AND ZISSERMAN, A. 2005. Image-based rendering using image-based priors. *Int. J. Comput. Vision 63*, 2, 141–151.

FLYNN, J., NEULANDER, I., PHILBIN, J., AND SNAVELY, N. 2016. DeepStereo: Learning to predict new views from the world's imagery. In *CVPR*, IEEE, 5515–5524.

FURUKAWA, Y., AND PONCE, J. 2010. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell. 32*, 8, 1362–1376.

FURUKAWA, Y., CURLESS, B., SEITZ, S. M., AND SZELISKI, R. 2009. Reconstructing building interiors from images. In *ICCV*, IEEE, 80–87.

GALLIANI, S., LASINGER, K., AND SCHINDLER, K. 2015. Massively parallel multiview stereopsis by surface normal diffusion. In *ICCV*, IEEE, 873–881.

GARLAND, M., AND HECKBERT, P. S. 1997. Surface simplification using quadric error metrics. In *SIGGRAPH*, ACM, 209–216.

GOESELE, M., SNAVELY, N., CURLESS, B., HOPPE, H., AND SEITZ, S. M. 2007. Multi-view stereo for community photo collections. In *ICCV*, IEEE, 1–8.

GOESELE, M., ACKERMANN, J., FUHRMANN, S., HAUBOLD, C., KLOWSKY, R., STEEDLY, D., AND SZELISKI, R. 2010. Ambient point clouds for view interpolation. *ACM Trans. Graph. 29*, 4, 95:1–95:6.

GOLDLÜCKE, B., AUBRY, M., KOLEV, K., AND CREMERS, D. 2014. A super-resolution framework for high-accuracy multiview reconstruction. *Int. J. Comput. Visionn 106*, 2, 172–191.

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The lumigraph. In *SIGGRAPH*, ACM, 43–54.

HENRY, P., KRAININ, M., HERBST, E., REN, X., AND FOX, D. 2012. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *Int. J. Rob. Res. 31*, 5, 647–663.

KAZHDAN, M., AND HOPPE, H. 2013. Screened poisson surface reconstruction. *ACM Trans. Graph. 32*, 3, 29:1–29:13.

KOPF, J., LANGGUTH, F., SCHARSTEIN, D., SZELISKI, R., AND GOESELE, M. 2013. Image-based rendering in the gradient domain. *ACM Trans. Graph. 32*, 6, 199:1–199:9.

LENSCH, H. P., HEIDRICH, W., AND SEIDEL, H.-P. 2001. A silhouette-based algorithm for texture registration and stitching. *Graph. Models 63*, 4, 245–262.

LENSCH, H., KAUTZ, J., GOESELE, M., HEIDRICH, W., AND SEIDEL, H.-P. 2003. Image-based reconstruction of spatial appearance and geometric detail. *ACM Trans. Graph. 22*, 2, 234–257.

LEVOY, M., AND HANRAHAN, P. 1996. Light field rendering. In *SIGGRAPH*, ACM, 31–42.

MA, Z., HE, K., WEI, Y., SUN, J., AND WU, E. 2013. Constant time weighted median filtering for stereo matching and beyond. In *ICCV*, IEEE, 49–56.

MATUSIK, W., BUEHLER, C., RASKAR, R., GORTLER, S. J., AND MCMILLAN, L. 2000. Image-based visual hulls. In *SIGGRAPH*, ACM, 369–374.

MCMILLAN, L., AND BISHOP, G. 1995. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH*, ACM, 39–46.

MOULON, P., MONASSE, P., AND MARLET, R. 2013. Adaptive structure from motion with a contrario model estimation. In *ACCV*, Springer, 257–270.

NAIR, R., RUHL, K., LENZEN, F., MEISTER, S., SCHÄFER, H., GARBE, C. S., EISEMANN, M., MAGNOR, M., AND KONDERMANN, D. 2013. A survey on time-of-flight stereo fusion. In *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*. Springer, 105–127.

NEWCOMBE, R. A., IZADI, S., HILLIGES, O., MOLYNEAUX, D., KIM, D., DAVISON, A. J., KOHI, P., SHOTTON, J., HODGES, S., AND FITZGIBBON, A. 2011. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, IEEE, 127–136.

NIESSNER, M., ZOLLHÖFER, M., IZADI, S., AND STAMMINGER, M. 2013. Real-time 3D reconstruction at scale using voxel hashing. *ACM Trans. Graph. 32*, 6, 169:1–169:11.

OLSSON, O., AND ASSARSSON, U. 2011. Tiled shading. *J. Graphics, GPU, and Game Tools 15*, 4, 235–251.

ORTIZ-CAYON, R., DJELOUAH, A., AND DRETTAKIS, G. 2015. A Bayesian approach for selective image-based rendering using superpixels. In *3DV*, IEEE, 469–477.

PUJADES, S., DEVERNAY, F., AND GOLDLUECKE, B. 2014. Bayesian view synthesis and image-based rendering principles. In *CVPR*, IEEE, 3906–3913.

SANKAR, A., AND SEITZ, S. 2012. Capturing indoor scenes with smartphones. In *UIST*, ACM, 403–412.

SHUM, H.-Y., CHAN, S.-C., AND KANG, S. B. 2008. *Image-based rendering*. Springer.

SINHA, S. N., STEEDLY, D., AND SZELISKI, R. 2009. Piecewise planar stereo for image-based rendering. In *ICCV*, IEEE, 1881–1888.

STICH, T., LINZ, C., WALLRAVEN, C., CUNNINGHAM, D., AND MAGNOR, M. 2011. Perception-motivated interpolation of image sequences. *ACM Trans. Applied Perception 8*, 2, 11:1–11:25.

UYTTENDAELE, M., CRIMINISI, A., KANG, S. B., WINDER, S., SZELISKI, R., AND HARTLEY, R. 2004. Image-based interactive exploration of real-world environments. *IEEE Comput. Graph. Appl. 24*, 3, 52–63.

VANHOEY, K., SAUVAGE, B., GÉNEVAUX, O., LARUE, F., AND DISCHLER, J.-M. 2013. Robust fitting on poorly sampled data for surface light field rendering and image relighting. *Comp. Graph. Forum 32*, 6, 101–112.

VÁZQUEZ, P.-P., FEIXAS, M., SBERT, M., AND HEIDRICH, W. 2003. Automatic view selection using viewpoint entropy and its application to image-based modelling. *Comp. Graph. Forum 22*, 4, 689–700.

WAECHTER, M., MOEHRLE, N., AND GOESELE, M. 2014. Let there be color! Large-scale texturing of 3D reconstructions. In *ECCV*. Springer, 836–850.

WEI, J., RESCH, B., AND LENSCH, H. P. 2014. Multi-view depth map estimation with cross-view consistency. In *BMVC*, BMVA Press.

WOOD, D. N., AZUMA, D. I., ALDINGER, K., CURLESS, B., DUCHAMP, T., SALESIN, D. H., AND STUETZLE, W. 2000. Surface light fields for 3D photography. In *SIGGRAPH*, ACM, 287–296.

XIAO, J., OWENS, A., AND TORRALBA, A. 2013. SUN3D: A database of big spaces reconstructed using SfM and object labels. In *ICCV*, IEEE, 1625–1632.

ZHOU, Q.-Y., AND KOLTUN, V. 2014. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Trans. Graph. 33*, 4, 155:1–155:10.

ZHOU, Q.-Y., MILLER, S., AND KOLTUN, V. 2013. Elastic fragmens for dense scene reconstruction. In *Proc. ICCV*, IEEE, 473–480.

ZITNICK, C. L., AND KANG, S. B. 2007. Stereo for image-based rendering using image over-segmentation. *Int. J. Comput. Vision 75*, 1, 49–65.