Soft 3D Reconstruction for View Synthesis

ERIC PENNER, Google Inc.
LI ZHANG, Google Inc.

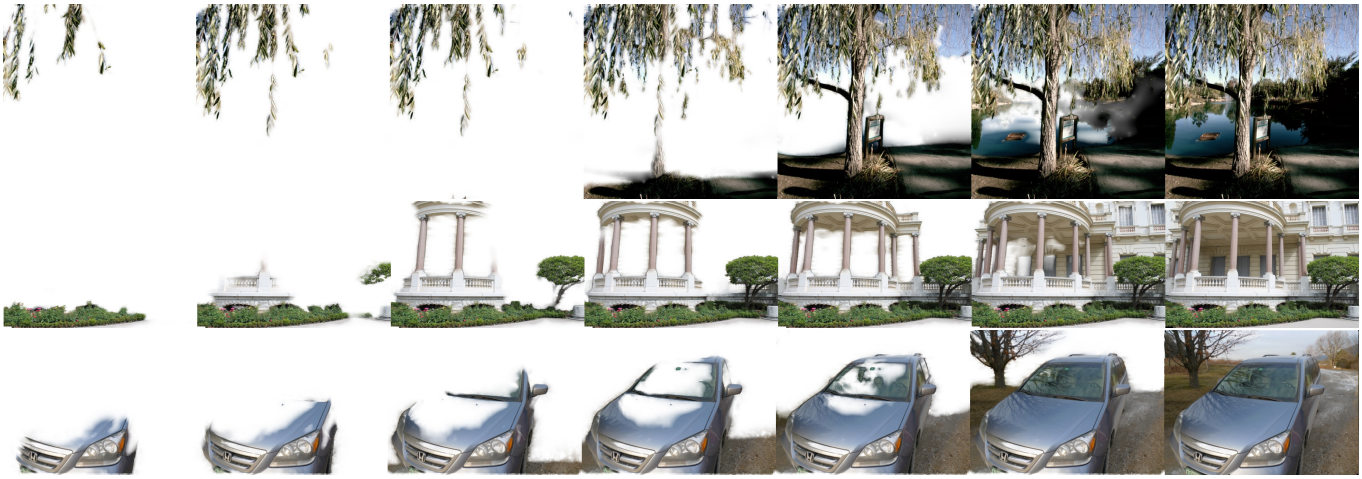

Fig. 1. Progress of rendering virtual views of difficult scenes containing foliage, wide baseline occlusions and reflections. View ray and texture mapping ray visibility is modelled softly according to a distribution of depth probabilities retained from the reconstruction stage. This results in soft edges, soft occlusion removal, partial uncertainty of depth in textureless areas, and soft transitions between the dominant depths in reflections.

We present a novel algorithm for view synthesis that utilizes a soft 3D reconstruction to improve quality, continuity and robustness. Our main contribution is the formulation of a soft 3D representation that preserves depth uncertainty through each stage of 3D reconstruction and rendering. We show that this representation is beneficial throughout the view synthesis pipeline. During view synthesis, it provides a soft model of scene geometry that provides continuity across synthesized views and robustness to depth uncertainty. During 3D reconstruction, the same robust estimates of scene visibility can be applied iteratively to improve depth estimation around object edges. Our algorithm is based entirely on O(1) filters, making it conducive to acceleration and it works with structured or unstructured sets of input views. We compare with recent classical and learning-based algorithms on plenoptic lightfields, wide baseline captures, and lightfield videos produced from camera arrays.

CCS Concepts: • **Computing methodologies** → *Computational photography*; *Reconstruction*; *Visibility*;

Additional Key Words and Phrases: View Synthesis, 3D Reconstruction

ACM Reference Format:
Eric Penner and Li Zhang. 2017. Soft 3D Reconstruction for View Synthesis. *ACM Trans. Graph.* 36, 6, Article 235 (November 2017), 11 pages. https://doi.org/10.1145/3130800.3130855

Authors' addresses: Eric Penner, Google Inc. Li Zhang, Google Inc.

0730-0301/2017/11-ART235
https://doi.org/10.1145/3130800.3130855

1 INTRODUCTION

With digital cameras continually decreasing in size and cost, and increasing demand for more immersive content (e.g. for VR/AR headsets), many new cameras and camera 'rigs' are being developed to capture images and video from several viewpoints simultaneously [Anderson et al. 2016; Cabral 2016]. At the same time, near ubiquitous digital cameras from cell phones and drones, paired with automatic camera calibration algorithms [Snavely et al. 2006], can also quickly create rich collections of photographs of interesting scenes. With such rich data, a natural goal is to immerse the user in a 3D environment of real photographed content, allowing them to see in stereo (separate views for each eye) and even to move around freely (six degrees of freedom).

Invariably, however, there is a limit to the density of input images that one can capture, necessitating a trade-off between capture coverage and capture density. View synthesis algorithms can provide virtual views of photographed content, but in this case we face a visual Turing test; people are hard wired to notice anomalies in natural images, and natural images have many properties that are simply not modelled by naive image-based rendering (IBR) methods, such as soft edges, view-dependent lighting, and translucency. The result is that the majority of rendering techniques produce roughly the correct result, but contain jarring rendering artifacts, including tearing at object boundaries, aliasing artifacts, temporal discontinuities, and ghosting.

These difficulties have led to another approach, which is to forgo explicit modelling of the problem and instead use machine learning to implicitly model complex 3D interactions using many training examples. While this direction is very promising and ever evolving,

Fig. 2. Left: Synthesized view (and average rendered depth) where we force each input to view to select a single depth before rendering (as is typically done during depth refinement). Because of the wide baseline and difficult foreground content, there are more votes in the background than in the foreground, making depth refinement unstable. Right: Our soft view synthesis algorithm produces plausible results. We also degrade gracefully to only partial ghosting as certainty drops very low at the rim of the cup.

less constrained networks that try to model everything tend to produce blurry or low-resolution results, while more customized networks start to resemble classical algorithms in various respects, such as incorporating plane-sweeps [Flynn et al. 2016] or depth selection [Kalantari et al. 2016]. Furthermore, since current ML approaches for view synthesis are trained to produce final rendered images rather than consistent 3D reconstructions, it is not yet clear how to accelerate these approaches for real-time rendering, or to produce other effects such as mixing computer generated content with photographed content.

The key contributions of this paper are as follows. We propose a new image based rendering framework that retains depth uncertainty from a local 3D reconstruction of the scene to final color synthesis, and demonstrate how traditional stereo and 3D reconstruction concepts and operations can fit inside this framework. Furthermore, we show that by designing a system end-to-end for view synthesis (rather than using an existing outputs such as a 3D meshes or point clouds), we can produce improved synthesis results that challenge existing methods for a wide variety of inputs (such as views from plenoptic cameras, camera array videos, and wide baseline image captures).

At the heart of our method is a soft model of scene geometry that retains uncertainty in the form of a distribution over depth values. From this we formulate soft visibility functions that can be used in both 3D reconstruction itself (to refine depth estimates at object boundaries), as well as throughout view synthesis (for both view rays and texture mapping rays). We model this uncertainty in both windowed cost aggregation as well as free-space reasoning from multiple depth maps. While modelling depth and scene visibility softly in final rendering may seem counter intuitive (most objects are opaque, after all), we show that preserving this uncertainty results in smooth and plausible renderings of difficult content, textureless regions and soft object edges.

2 RELATED WORK

View synthesis and stereo depth estimation have a long and overlapping history in both computer vision and graphics. Image based rendering (IBR) approaches typically use proxy geometry to synthesize views, while stereo and multi-view stereo (MVS) algorithms reconstruct geometry from images. View synthesis algorithms usually require both to provide virtual views from raw images alone. Our approach is a new view synthesis algorithm that performs a local soft 3D reconstruction of the scene. As such we review view synthesis algorithms which use automatically generated proxy geometries, and multi-view stereo approaches in the context of view synthesis.

2.1 View Synthesis

Early work set the foundations of the commonly used IBR framework, which involves supplying or estimating proxy geometry followed by reprojection and blending texture between nearby views [Chen and Williams 1993; Debevec 1996; McMillan and Bishop 1995]. Unstructured lumigraphs [Buehler et al. 2001] specifies the 'eight desirable properties' for IBR algorithms, such as continuity, proxy geometry, and epipole consistency. Many different geometry representations have been developed to handle and regularize the geometry estimation problem, such as polyhedral models [Debevec 1996], planar approximations [Sinha et al. 2009], silhouette warps [Chaurasia et al. 2011], and super pixel meshes [Chaurasia et al. 2013]. More recent work has focused on difficult cases such as planar reflections [Kopf et al. 2013; Sinha et al. 2012]. Machine learning models have been also been trained to model a local geometry for each output view from many training examples. Either a probability for each depth [Flynn et al. 2016] or a single depth [Kalantari et al. 2016] is inferred, along with an inferred color from the depth corrected input images which resolves texture occlusions.

The contribution of our approach is to provide a robust geometry and visibility representation that is both consistent and continuous, providing robustness to artifacts in difficult cases. While [Sinha et al. 2009] regularizes depth by assigning depths to a set of planes, our approach allows for arbitrary 3D shapes. Unlike human-assisted approaches such as [Chaurasia et al. 2011], our approach is completely automatic. Approaches that use hard 3D reconstructions such as [Chaurasia et al. 2013] require additional steps to fill in missing depths from the 3D reconstruction, and suffer from hard tears at depth boundaries and occlusions. Our geometry retains partial depth ambiguity to plausibly handle difficult cases, and our geometry itself can be linearly interpolated smoothly into virtual views, with occlusions and soft image edges being explicitly modelled. While methods such as [Hasinoff et al. 2006; Szeliski and Golland 1999] also provide soft edges by solving for mattes on source view depth maps edges, these methods do not provide a method to transition smoothly between input views, and do not account for difficult content where input depth maps may each be erroneous and/or disagree with each other. Our geometry itself is interpolated across views to provide continuity, and we utilize many neighboring views to provide robustness against difficult content.

2.2 Multi-View Stereo and 3D Reconstruction

Since multi-view stereo and 3D reconstruction is such a large field, we refer the reader to [Furukawa and Hernández 2015] for a review

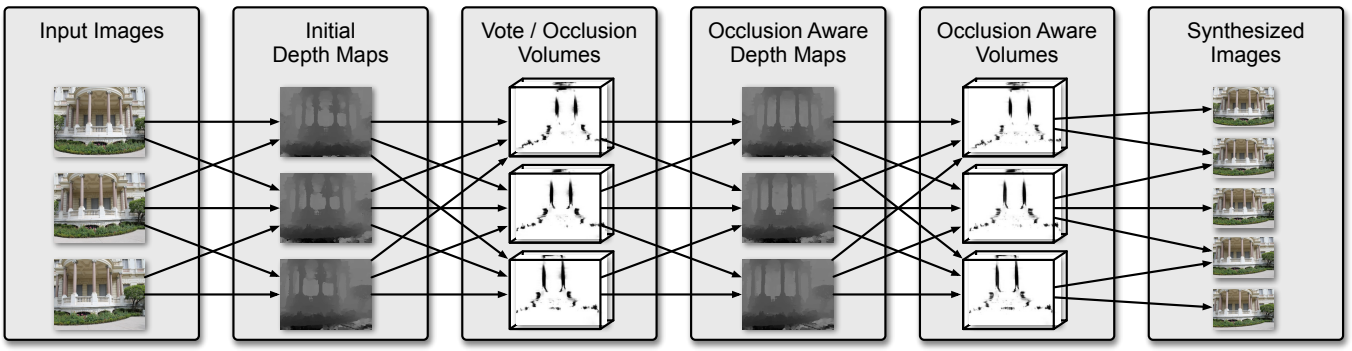

Fig. 3. Overview of our soft 3D reconstruction and view synthesis algorithm. We start by computing depth maps using a fast local stereo method. Our soft 3D reconstruction consists of volumetric vote volumes that model uncertainty from stereo and depth refinement (voting) operations. This representation is refined iteratively by providing a soft visibility estimate for per-pixel view-selection in a subsequent pass of stereo. Finally, we use the same soft representation directly to model ray visibility and occlusion during view-synthesis.

of work in this area. We focus here on a brief overview and recent work which relates to our method.

A minimal stereo setup consists of only two views from pre-calibrated cameras, while automatic calibration (provided by algorithms like [Snavely et al. 2006]) or calibrated camera 'rigs' [Anderson et al. 2016; Cabral 2016] can supply many views. Since individual matching costs are very noisy, a standard method is to aggregate matching costs to improve depth quality. Global optimization methods such as [Felzenszwalb and Huttenlocher 2006; Kolmogorov and Zabih 2001; Sun et al. 2003] seek to minimize a global cost function that also optimizes for depth smoothness, while 'local' methods attempt to achieve the same more efficiently (or even at real-time rates) using patch-based O(1) edge-aware filtering kernels such as [Hosni et al. 2011; Lu et al. 2013; Ma et al. 2013].

A unique problem in MVS is pixel-wise view selection. Using a larger neighborhood of views can provide better minima in cost curves, but implies more occlusions, which can corrupt matching costs on background objects. A very common heuristic is to use the best half of matching costs [Kang et al. 2001]. These can be selected as the minimum set, by selecting the best left/right half [Kang et al. 2001] (when the cameras follow a line), or extending to the best half-space derived from image edges [Wang et al. 2015] (when the cameras form a 2D lightfield). Other methods use probabilistic models of the scene visibility combined with smoothness assumptions such as [Strecha et al. 2004], or jointly model pixel-wise view-selection along with the depth by estimating local pairwise photo-consistency between the reference and candidate images [Zheng et al. 2014].

Our work builds on local methods for two view stereo [Hosni et al. 2011; Lu et al. 2013; Ma et al. 2013], to build a soft projective 3D reconstruction of the scene in the target view for view synthesis. By adding a second 'occlusion aware' refinement stage, and retaining a soft representation for rendering of difficult cases, we both improve depth quality and reduce common artifacts caused by incorrect depth estimation. While iterative methods exist for estimating visibility masks as part of stereo, such as [Strecha et al. 2004; Zheng et al. 2014], our method is the first we are aware of that utilizes a soft local model of visibility derived from neighboring views. We find this provides a surprisingly effective visibility estimate for pixel-wise view selection during stereo, and can also be efficiently aggregated using O(1) filtering methods.

3 OVERVIEW

Our full view synthesis algorithm is illustrated in Figure 3. The key to our algorithm is a soft volumetric reconstruction of the scene geometry which we perform in projective volumes in each input view. We estimate depth using a fast local stereo method for each input view, from which we reconstruct a soft volume that incorporates uncertainty during stereo and the fusion of many depth maps. The remainder of the paper is broken into sections which describe each of the steps in our algorithm:

- **Initial depth estimation.** We compute depth maps for all input views using a small set of N neighbors and a fast local stereo method, which is described in Section 4.
- **Soft 3D Reconstruction.** For each input view, we reconstruct the scene geometry using a discretized projective volume representation. Each voxel encodes a confidence value that a surface exists at that voxel, and incorporates uncertainty from stereo cost aggregation as well as surface/free-space confidence from voting. The confidence values are accumulated from a larger neighborhood of M >N views. This is described in Section 5.
- **Soft View Synthesis.** To explicitly model continuity across virtual views, we smoothly interpolate this soft geometry from the nearest K source views into the virtual view. To texture map the geometry as we render, we use soft visibility functions calculated for the input views as texture mapping weights (in addition to our geometry interpolation function). We describe this in Section 6.
- **Occlusion aware depth estimation.** Finally, we show how our initial stereo depth estimation can also be improved iteratively using the same representation, by using soft visibility as per-pixel view selection weights in another pass of stereo depth estimation. This is described last, in Section 7.

Throughout this paper, since we deal with many 3D projections between images (or 'views'), we use a few simple conventions for brevity. A function $F(x, y, z)$ or $G(x, y)$ is assumed to take place in a *reference* view's 2D or 3D coordinates. Image coordinates x and y are assumed to be image pixel coordinates, while z is defined by a plane sweep which is sampled linearly in $1/z$ in the reference view. When referring to the result of such a function for a *neighbor* view k, we use $F_k(x'_k, y'_k, z'_k)$ or $G_k(x'_k, y'_k)$. Here the x'_k, y'_k, z'_k denote that a 3D point has been transformed into camera k's coordinates as follows:

$$(x'_k, y'_k, z'_k) = C_k(C^{-1}_{\text{reference}}(x, y, z)), \tag{1}$$

where C_k and $C^{-1}_{\text{reference}}$ are the forward and inverse transforms for view k and the reference view respectively.

We also omit image sampling methods for brevity. In practice, when sweeping over all 3D coordinates to compute such a function, we use 2D homography warps and bilinear interpolation for 2D images, full 3D projections and tri-linear interpolation for volume sampling, and full 3D projections with percentage-closer-filtering [Reeves et al. 1987] when interpolating a depth test result is required.

4 INITIAL DEPTH ESTIMATION

Our initial depth estimation method is a 'local' method similar to [Hosni et al. 2011] which filters matching costs using an $O(1)$ edge-aware filter kernel, followed by winner-takes-all depth selection. More formally, we compute a depth map $D()$ for a view given several neighbor views as follows:

$$E_{\text{raw}}(x, y, z) = \sum_{k \in N} E_k(x, y, z) \tag{2}$$

$$E(x, y, z) = \sum_{(\hat{x}, \hat{y}) \in W} w(x, y, \hat{x}, \hat{y}) E_{\text{raw}}(\hat{x}, \hat{y}, z) \tag{3}$$

$$D(x, y) = \arg\min_{z \in Z} E(x, y, z), \tag{4}$$

where N is a set of neighbor views (up to 4 for 1D captures and 8 for 2D captures), W is an edge-aware filter window which we aggregate efficiently using the $O(1)$ guided-filter [He et al. 2010], and Z is a set of fronto-parallel planes defined by a plane sweep in the reference view. E_k is the raw matching cost between the reference view and a neighboring view k, which is sum-of absolute-differences (SAD) based.

One unique aspect to our cost and vote filtering is a pyramidal variant of the guided-filter. To allow for a larger aggregation window in textureless areas, we compute costs and votes from an image pyramid, filter each level, and then blend each lower level into the next level according to the variance in the higher level (which is already computed by the guided filter). To blend a lower level into the next, we perform a linear step blend between std-dev of 3.0 and 10.0 in the higher level. In datasets where minor pose misalignment is possible (such as datasets solved using structure-from-motion (SfM)), we blend stereo cost levels using a constant factor of 0.5, as we find lower levels provide some robustness to minor misalignment. Since our approach adapts the kernel size to be larger in textureless areas, we use only an 8x8 guided-filter window.

5 SOFT 3D RECONSTRUCTION

In this section we discuss how we reconstruct soft projective volumes of scene geometry in each reference view using neighboring view depth maps, and how we interpret this geometry to formulate visibility/occlusion functions for use by view synthesis and iterative stereo refinement. We first make several observations as motivating goals for our reconstruction approach described below.

(1) Each depth map pixel actually supplies two pieces of information: An estimate of where a surface exists, and an estimate that no surface exists in front of it (free space).
(2) Conversely, each depth map pixel supplies no information about what lies *behind* its estimated depth. Both 1 and 2 have uncertainty that arises due to the imperfect nature of passive depth estimation.
(3) One type of uncertainty is between neighboring pixels within a single image. For example, over textureless regions and near depth discontinuity boundaries, we often have multiple depth values within a nearby neighborhood in the image. We can model this local uncertainty in the depth by considering each pixel's depth as a distribution originating from its neighborhood in the depth map (with weight of each depth proportional to similarity in pixel color as well as spatial distance), rather than a single depth.
(4) Another type of uncertainty is across different views. Specifically, when we fuse multiple depth maps from different views, they do not always agree with each other. We can model this cross-view uncertainty by aggregating the consensus at each point in space, which we call a vote volume below.
(5) We need to model visibility/occlusion of a ray travelling through our geometry. Visibility should also be modelled softly, respecting the uncertainty in depth described above.

To retain depth uncertainty discussed in goals 3 and 4, we use vote-volumes to aggregate and store an entire depth distribution for each image pixel. Our vote volumes are formulated very similar to our stereo cost volumes in section 4, and use the same set of planes defined by a stereo plane sweep. To initially incorporate uncertainty arising from disagreement between depth maps (goal 4 above), we define vote-value and vote-confidence functions. A depth map initially votes that a surface exists at its depth, that no surface exists in front of it, and abstains from voting at all behind its depth:

$$VoteVal_{\text{raw}}(x, y, z) = \begin{cases} 1 & z = D(x, y) \\ 0 & otherwise \end{cases} \tag{5}$$

$$VoteConf_{\text{raw}}(x, y, z) = \begin{cases} 1 & z <= D(x, y) \\ 0 & otherwise. \end{cases} \tag{6}$$

To incorporate goal 3, we use a technique similar to constant-time weighted median filtering [Ma et al. 2013] and filter these votes using the same kernel used to filter our stereo costs:

$$VoteVal(x, y, z) = \sum_{(\hat{x}, \hat{y}) \in W(x, y)} w(x, y, \hat{x}, \hat{y}) VoteVal_{\text{raw}}(\hat{x}, \hat{y}, z) \tag{7}$$

$$VoteConf(x,y,z) = \sum_{(\hat{x},\hat{y})\in W(x,y)} w(x,y,\hat{x},\hat{y}) VoteConf_{\text{raw}}(\hat{x},\hat{y},z), \tag{8}$$

where $W(x,y)$ is the filter window for a pixel and $w()$ is the edge-aware kernel weight. Again we can compute this efficiently using the $O(1)$ guided-filter. The result of applying this filter is to spread each source depth to cover all pixels in the stereo patch, resulting in a depth distribution for each image pixel with the resulting voting weight distributed according to the stereo patch edge-aware kernel.

Now to support votes from many views, we can repeat this process several times (one for each voting neighbor view), accumulate the results, and normalize total votes by total number of voters (accumulated confidence) to form a *surface consensus* volume:

$$Consensus'(x,y,z) = \frac{\sum_{k\in M} VoteVal_k(x'_k,y'_k,z'_k)}{\sum_{k\in M} VoteConf_k(x'_k,y'_k,z'_k)}, \tag{9}$$

where M is our voting neighborhood (up to 10 for 1D captures and 25 for 2D captures). We note that while this achieves our goals and filters each vote with the same weights as its stereo patch, this requires caching all filtered volumes (or redundantly filtering many volumes if we do this from scratch each time). In practice, to reduce memory consumption we use a slightly modified filtering approach:

$$Consensus_{raw} = \frac{\sum_{k\in M} VoteVal_{\text{raw}_k}(x'_k,y'_k,z'_k)}{\sum_{k\in M} VoteConf_{\text{raw}_k}(x'_k,y'_k,z'_k)} \tag{10}$$

$$Consensus(x,y,z) = \sum_{(\hat{x},\hat{y})\in W(x,y)} w(x,y,\hat{x},\hat{y}) Consensus_{\text{raw}}(\hat{x},\hat{y},z). \tag{11}$$

The subtle difference here is that we use only one guide image and filter only one volume rather than one for every contributing depth map, which means that we do not filter each depth estimate with its exact stereo patch weights. For foreground objects these values should logically be very similar, so the main difference with this approximation is that the background surfaces (with respect to the single guide image) are filtered with the wrong edge-aware kernel.

Finally, we observe that vote confidence trends toward zero in the background due to more views becoming occluded. To prevent this from amplifying noise, we normalize equation 10 only when there is sufficient confidence (in practice we use a threshold of half the number of voting depth maps), and we allow *Consensus* to fade to zero below this threshold.

5.1 Visibility/Compositing Functions.

The consensus volume is what we use as our geometry representation in sections 6 and 7. However, before we can make use of the geometry we need to texture-map it and measure the visibility/occlusion of rays traveling through it. We start by defining ray visibility simply as the clamped summation of all prior consensus along a ray:

$$SoftVis(x,y,z) = max(0, 1 - \sum_{\hat{z}\in Z,\hat{z}<z} Consensus(x,y,\hat{z})). \tag{12}$$

We note that this is identical to 'coverage' visibility functions discussed in deep shadow maps [Lokovic and Veach 2000]. We find this is more appropriate than 'standard' alpha compositing in our case (which would result in an exponential decay), as we would like our ray visibility to approach zero as the probability that the ray is occluded approaches one (i.e. all depth estimates agree that ray has hit something).

Modifying this function can tune the effect of weakly reconstructed geometry on our output images. For example, we can apply a multiplier to terminate view rays and/or occlude textures faster. While we have not experimented with modifying this function, we note that our consistency step below has the effect of multiplying consensus by a small factor (since each depth map votes in a small interval).

5.2 Consistency

We find enforcing *full* consistency between all depth maps is quite difficult and unstable in practice. For example, the weighted median filter [Ma et al. 2013] is unstable when the number of foreground votes are close to the number of background votes. On the other hand, a simple vote threshold is also unstable when foreground votes hover around that threshold, and a low threshold risks emphasizing noise. This is exacerbated by different depth maps using different ray directions, resulting in the same 3D point having very different vote histograms in different views.

For this reason, as described above, we try to respect *all* depth estimates from many surrounding depth maps. However, we do wish to remove obvious depth *outliers* which do not agree with any other view. To achieve this, we simply add a small interval around our votes (to allow for one-off votes to still be consistent) and then subtract one from the total votes accumulated.

6 SOFT VIEW SYNTHESIS

Our soft view synthesis algorithm resembles volume rendering with projectively texture mapped volumetric geometry, and is illustrated in Figure 4. We use a small set K of neighbor input images, depending on the type of interpolation (e.g. 2/4 for linear/bilinear interpolation), and interpolation weights $W_k(x,y)$ for each input view. $W_k(x,y)$ is a single value per view in the case of structured input views (defined by the linear/bilinear interpolation function). For unstructured input views, we define $W_k(x,y)$ as a low resolution image per input view, as follows:

$$W_k(x,y) = e^{-\text{Dist}(\text{Ray}_k(x,y),O_{synth})^2/b^2}, \tag{13}$$

where O_{synth} is the optical center of the synthesized view, and b is the average nearest-neighbor baseline (distance between cameras) for the input dataset. $W_k()$ can also be used to fade to zero at image borders (to hide seams), but since all our images are color calibrated, we did not find this necessary.

We calculate consensus volumes for all input views using equation (11) and interpolate one for the virtual view using our view weights $W_k(x,y)$. We then compute soft visibility for each input view using equation (12) and use input view visibility as an additional texture mapping weight (to handle occlusion), while the interpolated consensus volume provides smoothly interpolated geometry

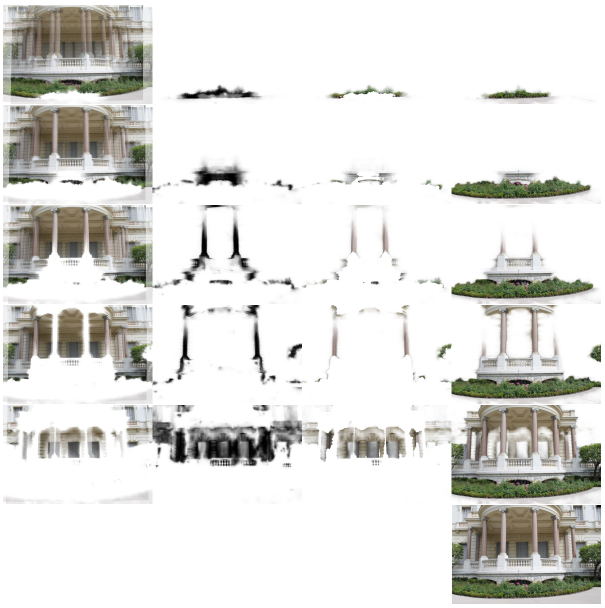

Fig. 4. Progress of rendering a frame using soft view synthesis. First column: Input images warped to a reference view depth, with occlusions masked. Second column: Votes for the current depth, that incorporates depth and free-space estimates from all neighbor images. Third column: First column modulated by the second column to create a color slice. Fourth column: Accumulation of all color slices up to this depth.

for the current frame. We then traverse through the virtual-view's consensus volume in slices, computing and accumulating color for each non-zero consensus along our view rays. A ray terminates when its visibility reaches zero (summed prior consensus equals one). Colors are in turn interpolated from neighbor views using the interpolation function modulated by per-view visibility.

More formally, the color at a 3D point is computed from the input images $I_k()$:

$$Color_{\text{synth}}(x,y,z) = \frac{\sum_k SoftVis_k(x'_k,y'_k,z'_k)W_k(x'_k,y'_k)I_k(x'_k,y'_k)}{\sum_k SoftVis_k(x'_k,y'_k,z'_k)W_k(x'_k,y'_k)}. \tag{14}$$

The geometry for the synthesized view is smoothly interpolated from our input views using our weights $W_k()$:

$$Consensus_{\text{synth}}(x,y,z) = \sum_k W_k(x'_k,y'_k)Consensus_k(x'_k,y'_k,z'_k). \tag{15}$$

The color slices are then weighted by consensus and accumulated until ray visibility reaches zero:

$$CC(x,y,z) = min(Consensus_{\text{synth}}(x,y,z), SoftVis_{\text{synth}}(x,y,z)). \tag{16}$$

$$I_{\text{synth}}(x,y) = \frac{\sum_{z\in Z} Color_{\text{synth}}(x,y,z)CC(x,y,z)}{\sum_{z\in Z} CC(x,y,z)}, \tag{17}$$

where $I_{\text{synth}}()$ is our synthesized output image. $CC()$ is consensus clamped to the remaining ray visibility and $Consensus()$ and $SoftVis()$ are defined in equations (11) and (12), respectively. We normalize by the total amount of votes accumulated along the ray in the event the ray's visibility does not reach zero before it exits the volume.

6.1 Real-Time Rendering

Since our algorithm produces a 3D representation with a simple interpretation of ray visibility, it is conducive to real-time rendering, even on very limited hardware such as mobile devices. To prototype real-time rendering we modified our algorithm to pre-compute one or multiple $Color()$ volumes in set locations (depending on the user exploration allowed), followed by GPU accelerated rendering of this alpha blended content. In this case we invoke the $Color()$ function with a higher number of input views. As we accumulate more views we accumulate colors in occluded areas, allowing for a larger range of movement from the volume's optical center. We then render content in front-to-back order using GL-SRC-ALPHA-SATURATE blend mode to replicated our visibility function described above (note this works only for integer render targets where alpha saturates).

These color volumes are typically quite sparse and can be much more compactly stored as meshes with textures. As a simple initial approach, we simply tiled the volume and dropped empty tiles, which increased performance by 5-10× over brute force volume rendering and can render at solid 60Hz on the Google Pixel smart phone using 128 layers. A color volume only provides content within a single projective volume, so to provide a larger range of 3D movement we expand the projective volume where color is accumulated, effectively 'uncropping' the volume. Even larger movement in space can be handled by blending in screen space.

7 OCCLUSION AWARE DEPTH ESTIMATION

We have also found soft visibility useful in refining the quality of our rendered depth edges, by providing approximate occlusion awareness even in wide baseline settings. A difficult problem in MVS depth estimation is handling occlusions. Including more neighbor images from wider baselines helps to disambiguate the correct surface in cost curves, but wider baselines introduce occlusions which corrupts costs around depth edges. A typical result is 'edge fattening', where a low texture background next to a foreground object assumes the foreground depth. This occurs because the correct background depth has an invalid high matching cost, while the foreground depth is still low due to the low texture of the background.

It is challenging to solve this problem in two view stereo using matching costs alone, as only one view can see the occluded surface. In MVS however, we can still find the correct surface using only matching costs so long as we can correctly identify and use the correct subset of views for each pixel (those that lack occlusions). We define this idealistic matching cost energy function as:

$$E_{ideal}(x,y,d) = \frac{\sum_k E_k(x,y,d)HardVis_k(x'_k,y'_k,d'_k)}{\sum_k HardVis_k(x'_k,y'_k,d'_k)}, \tag{18}$$

where E_k is the matching cost for neighbor view k and $HardVis()$ is a hard visibility function corresponding to the correct surface. The problem of course is that supplying $HardVis()$ is a chicken-egg problem, since visibility is dependent on the depth map $D()$, which is in turn dependent on the ideal cost itself:

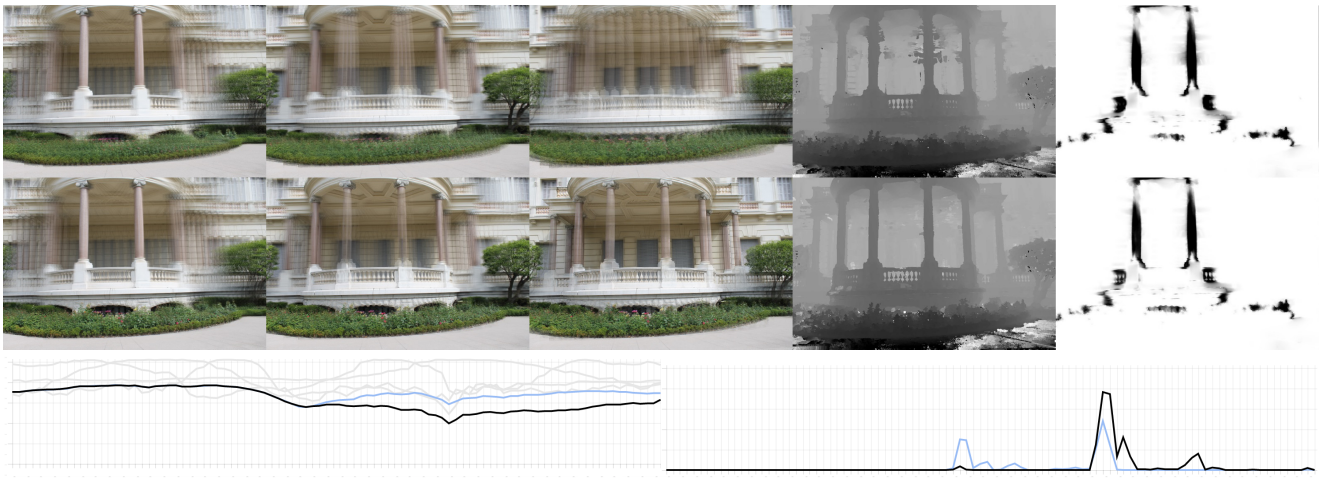

Fig. 5. Occlusion awareness. Top row: a plane sweep shows the posts coming clearly into focus, but not the wall behind them (due to occlusions). This results in corrupted depths around the posts. Consistent votes filtered as in [Ma et al. 2013] still show some corruption (because it occurs consistently in most views). Middle row: each view is weighted by per pixel soft visibility from the first pass. The posts fade out and the background comes into focus with the unoccluded views. The depth corruption is gone in both depths and the vote volume slice. Bottom Left: The first pass (blue) matching cost never makes it below an incorrect foreground depth. On the second pass (black), occluded costs are weighted by visibility and the cost curve makes it to a new absolute minimum. Bottom Right: Vote curves for the same pixel show that votes are now more consistent (more area under the black curve) and the foreground peak has been eliminated.

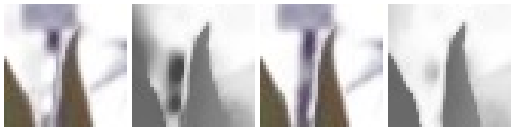

Fig. 6. Synthesized view and average depth without/with occlusion aware stereo on 'Leaves' dataset from [Kalantari et al. 2016]. We are able to find the correct depth on the second pass.

$$D(x,y) = \arg\min_{z} E_{ideal}(x,y,z). \tag{19}$$

Typical methods of handling this problem also have some drawbacks. The most common approach of using the best half of costs to approximate the non-occluded costs [Kang et al. 2001; Wang et al. 2015] introduces new ambiguities even for foreground objects. Repetitive texture, for example, might be clear when costs are averaged from two neighbors but becomes ambiguous again when the best of the two is taken. An even more simple ambiguous case is shown in Figure 7.

Our solution to the visibility problem is an iterative algorithm that uses soft visibility estimated in the first pass to provide occlusion awareness in the next pass. Since we have developed a robust estimate to visibility which enforces a certain level of consistency and incorporates uncertainty between views, we simply replace *HardVis*() in the ideal matching cost function, with our soft visibility function *SoftVis*():

$$E_{ours}(x,y,d) = \frac{\sum_k E_k(x,y,d) SoftVis_k(x'_k, y'_k, d'_k)}{\sum_k SoftVis_k(x'_k, y'_k, d'_k)}. \tag{20}$$

In practice, we find that this converges surprisingly quickly, only requiring one extra iteration of stereo to converge. Intuitively, many invalid depths are removed by consistency, while remaining fattened edges are attenuated heavily by edge-aware vote filtering, which improves the visibility functions further for the next pass. Furthermore, since these artifacts tend to be caused by large occluders, the first stereo pass can be computed at a lower resolution (in x, y and z) and still provide almost all the benefit.

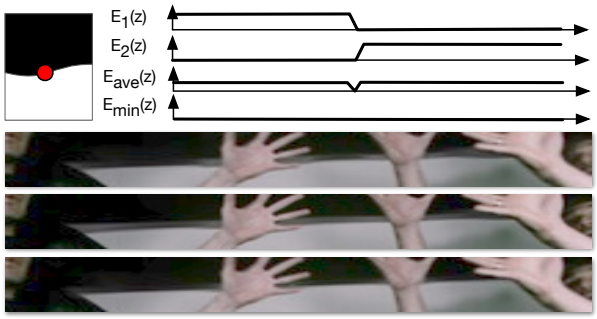

Fig. 7. Above: $E_1(z)$ and $E_2(z)$ are toy matching cost curves for top/bottom views of an image edge. The average cost $E_{ave}()$ has an ideal matching cost curve, while taking the best half $E_{min}()$ is ambiguous in this case. Below: The top synthesized view shows ghosting on the wall around foreground objects due to incorrect depth estimation. In the middle, taking the best half of costs as in [Kang et al. 2001] makes the ghosting worse rather than better in this case because the edge becomes ambiguous. On the bottom, a second pass of occlusion aware stereo improves the result.

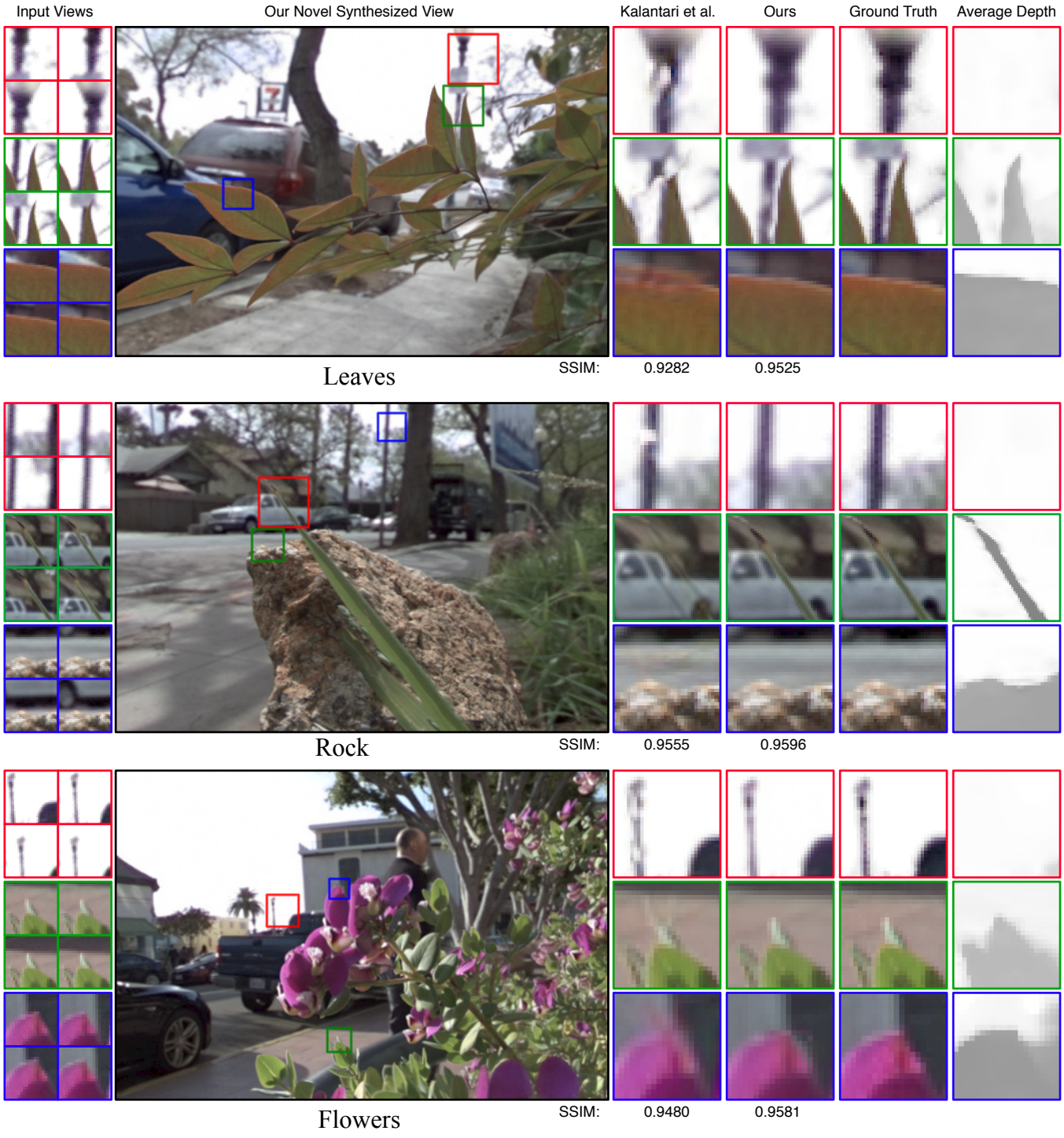

Fig. 8. Comparison of our approach with the method of [Kalantari et al. 2016] on the most difficult Lytro lightfield datasets from their paper. We also include a visualization of the soft depth used to render our result as the average depth of contributing colors. Since [Kalantari et al. 2016] compares favorably to [Zhang et al. 2016] and [Wanner and Goldluecke 2014] (using depth estimation of [Tao et al. 2013], [Wang et al. 2015] and [Jeon et al. 2015]) we refer to the reader to [Kalantari et al. 2016] for further comparisons with these methods. Our method is also temporally continuous, which we show in the supplementary video.

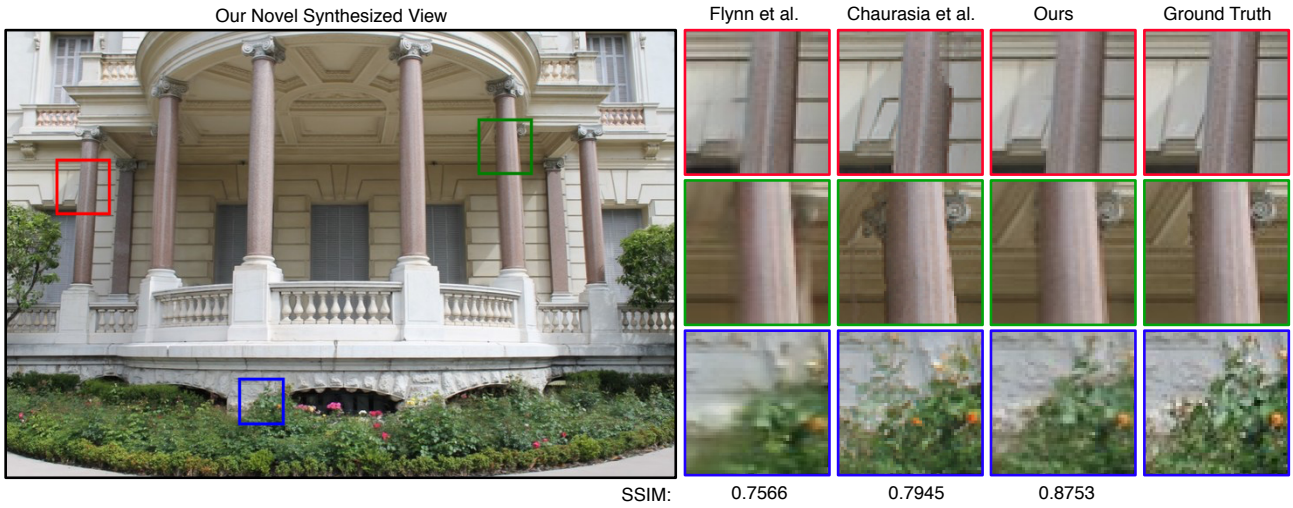

Fig. 9. Comparison of our approach with the methods of [Chaurasia et al. 2013] and [Flynn et al. 2016] on wide baseline images from [Chaurasia et al. 2013]. We note that our method is also temporally continuous across frames, which we show in the supplementary video.

	Kalantari et al.	Ours	Ours (2 Pass)
Flowers1	0.9480	0.9562	**0.9581**
Flowers2	0.9507	0.9591	**0.9616**
Cars	0.9700	0.9702	**0.9705**
Rocks	0.9555	0.9584	**0.9595**
Leaves	0.9282	0.9488	**0.9525**
Average	0.9505	0.9585	**0.9604**

Table 1. SSIM [Wang et al. 2004] scores for the 5 Lytro datasets shown in [Kalantari et al. 2016]. We note that the original images are quite dark, so we tone adjust all images as shown in Figure 8 which lowers scores slightly for both methods. Note that our method is only slightly better on datasets with low depth complexity, but significantly better on datasets cited as being the most difficult (lots of occluders closer to the camera).

8 RESULTS

Our method works across a wide variety of inputs, including structured and unstructured input images, as well as narrow and wide-baseline captures. Since methods intended for plenoptic cameras do not typically apply to unstructured wide-baseline captures (and vice versa), we report our comparisons based on camera configuration.

8.1 Plenoptic Cameras

We first compare our results on interpolation of Lytro images from [Kalantari et al. 2016] in Figure 8 and show our SSIM scores in Table 1. Since [Kalantari et al. 2016] compares favorably [Zhang et al. 2016] and [Wanner and Goldluecke 2014] (using depth estimation of [Tao et al. 2013], [Wang et al. 2015] and [Jeon et al. 2015]), we refer the reader to [Kalantari et al. 2016] to see further comparisons with these methods.

As shown, while the two methods are comparable in areas with low parallax, our method produces consistently better results for thin objects and around occlusions. In the first (red) inset for each dataset, we show thin background objects being occluded by 'floating' background pixels. In the second (green) inset, we show thin foreground objects ghosting or disappearing. In the last (blue) inset, we show color bleeding from the foreground onto the background. In all of these cases our result closely matches the ground truth.

We suspect that some of these problems are caused by performing plane sweeps in a virtual view, which is more ambiguous for thin objects since both the foreground and background can be visible in the source views. Color bleeding may be caused by the CNN making mistakes while resolving occlusions using only the warped source views (with no knowledge of the occluding geometry itself). In contrast, our approach models occlusion explicitly during texture mapping, robustness is improved by requiring consistency, and our soft depth edge alignment is improved by our second occlusion-aware stereo pass.

In terms of performance, we use our CPU implementation for evaluation, which takes 91 seconds to perform both reconstruction (10 seconds) and rendering (80 seconds for 64 frames) on a 6 core hyperthreaded CPU. This is almost an order of magnitude faster than [Kalantari et al. 2016] despite their approach using a powerful GPU to invoke their neural network.

8.2 Camera Arrays

We also show how our method handles wider baseline camera arrays with difficult content in Figure 2 and 10 as well as in the supplementary video. This camera array is a 5x5 array of cameras spaced approximately 6.5cm apart and synchronized with a shared global shutter. The camera poses are not perfectly regular, so they are solved using a method similar to [Snavely et al. 2006]. We also align colors as a preprocess using a method similar to [HaCohen et al. 2011]. As seen in Figure 10, increasing the number of stereo neighbors improves quality to a point, but a combination of both

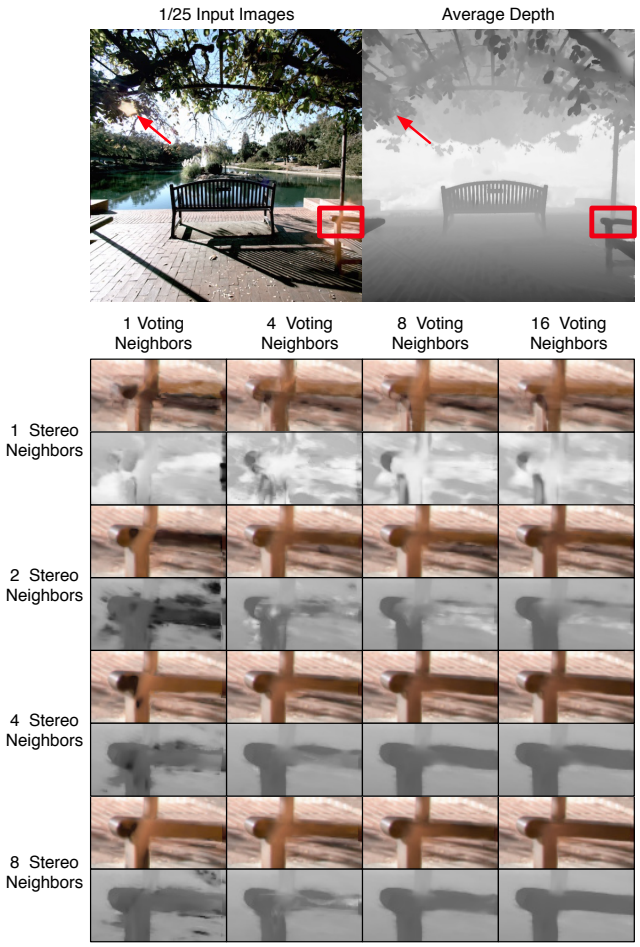

Fig. 10. Quality improvement from our soft reconstruction approach on a dataset with difficult content. Top Left: One input image of a 5x5 dataset containing many difficult aspects such as foliage, thin objects, and strong lens flares. Top right: Our average depth from a nearby synthesized frame showing high quality soft depth. Bottom: Close-ups on our reconstruction of the chair as we increase both stereo and voting neighbors. While stereo alone improves the result, only voting from many neighbors overcomes corruption from the lens flares that obscure the chair and leaves.

stereo neighbors and voting neighbors produces a strong improvement over stereo alone. The soft depth edges in the Figure actually represent a weighted combination of multiple depths, hence there is no depth edges or warping visible in the synthesized views. As seen in Figure 2, our method also produces much better results than a simple depth map refinement using the same number of voting views.

8.3 Unstructured Cameras

Next, we compare our method with the methods of [Chaurasia et al. 2013] and [Flynn et al. 2016] on an unstructured wide baseline dataset in Figure 9. As shown in the insets and SSIM scores, our method produces higher quality results with less artifacts.

Since [Chaurasia et al. 2013] solve a small linear system to regularize the transform of each super-pixel, cracks sometimes appear as neighboring super pixels are assigned to different or incorrect depths. In addition, since each image uses different geometry, these differences must be reconciled with a combination of depth based as well as view-based weighting, which in results in temporal discontinuities.

Our rendering model is similar to [Flynn et al. 2016] in that their neural network architecture implicitly models a probability over all depths. However, because colors from the input images are merged by a neural network without knowledge of the scene geometry, occlusions and difficult cases (such as the shrubs) become blurry to due excessive uncertainty. In contrast, our scene depth distribution comes from stereo estimation, depths need to be consistent in at least two views, and occlusions are removed softly using the same geometry used for rendering.

8.4 Lightfield Video

We also demonstrate how our method performs at producing light-field videos. In this case we use a synchronized camera array in a 5x4 configuration. Since our method utilizes depth maps from many views (all 20 in this case), it is robust to temporal discontinuities in any given depth map, with the final result being temporarily smooth despite our algorithm operating on individual frames. This is in contrast to global depth regularization methods that may fluctuate heavily across frames. Since temporal smoothness is difficult to illustrate on paper, we refer to reader to the supplementary video to see these results.

9 LIMITATIONS

We have demonstrated our method on a diverse set of challenging use cases, however, as with any approach our method does have limitations. As our method is view based, we reconstruct geometry directly in projective view volumes, with memory increasing linearly with depth precision. This depth precision limits the amount of free view-point movement away from the source views, after which depth quantization artifacts will appear. Our method also does not perform 'spill suppression' on our soft foreground edges, which can result in narrow color bleeding of the background color onto foreground edges. An important take away from our work is that even fast, local stereo methods can produce high quality view synthesis results. However, we assume a relative large number of input views to perform voting, and as less input views are used, depth regularization using a global optimization may improve results further relative to our local stereo method.

10 DISCUSSION AND FUTURE WORK

We have presented a novel algorithm for soft 3D reconstruction and view synthesis which uses the same vote-based representation at every stage. In depth estimation, it provides visibility estimates to perform per-pixel view selection and improve depth edges, while in synthesis we interpolate this local geometry into output frames and use it to provide soft visibility weights for texture mapping. Our approach improves on the state of the art on synthesis for narrow baseline light-field cameras, while improving further with

access to more views in unstructured camera arrays, for both stills and temporally smooth video. Moreover our algorithm is suitable for real-time rendering applications by pre-computing color+alpha meshes.

In the future we would like perform better 'spill suppression' on our soft foreground edges. We hope to utilize our 3D reconstruction here as well, since this provides us with a good idea of the background color (reducing the number of unknowns in the classical alpha matting problem). We would also like to GPU accelerate our reconstruction algorithm, which is embarrassingly parallel but currently implemented on the CPU. This could enable more mobile applications, as well as potentially real-time reconstruction from live source videos. Finally, we would like to investigate combining our algorithm with neural network architectures. By providing geometrical knowledge wherever possible (such as consistency and visibility), but still allowing a neural network to solve difficult problems (e.g. matching and filling occluded areas), we hope to combine the best of both classical and machine learning methods.

ACKNOWLEDGMENTS

We thank our colleges and collaborators at Google and all authors of prior work for providing their results on several datasets. In particular we would like to thank Marc Comino, Daniel Oliveira, Yongbin Sun, Brian Budge, Henri Astre, Vadim Cugunovs, Jiening Zhan and Kay Zhu for contributing and reviewing code; Matt Pharr and his team for supplying imagery; David Lowe, Matthew Brown, Aseem Agarwala, Matt Pharr and Siggraph Asia reviewers for their feedback on the paper.

REFERENCES

Robert Anderson, David Gallup, Jonathan T. Barron, Janne Kontkanen, Noah Snavely, Carlos Hernández, Sameer Agarwal, and Steven M. Seitz. 2016. Jump: Virtual Reality Video. *ACM Trans. Graph.* 35, 6, Article 198 (Nov. 2016), 198:1–198:13 pages.

Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. 2001. Unstructured Lumigraph Rendering. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. ACM, New York, NY, USA, 425–432.

Brian K Cabral. 2016. Introducing Facebook Surround 360: An open, high-quality 3D-360 video capture system. (2016). https://code.facebook.com/posts/1755691291326688

Gaurav Chaurasia, Sylvain Duchene, Sorkine-Hornung, and Olga Drettakis George. 2013. Depth Synthesis and Local Warps for Plausible Image-based Navigation. *ACM Trans. Graph.* 32, 3, Article 30 (July 2013), 30:1–30:12 pages.

Gaurav Chaurasia, Olga Sorkine-Hornung, and George Drettakis. 2011. Silhouette-Aware Warping for Image-Based Rendering. *Computer Graphics Forum* 30, 4 (2011).

Shenchang Eric Chen and Lance Williams. 1993. View Interpolation for Image Synthesis. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '93)*. ACM, 279–288.

Paul E. Debevec. 1996. *Modeling and Rendering Architecture from Photographs*. Ph.D. Dissertation. University of California at Berkeley, Computer Science Division, Berkeley CA.

Pedro F. Felzenszwalb and Daniel P. Huttenlocher. 2006. Efficient Belief Propagation for Early Vision. *Int. J. Comput. Vision* 70, 1 (Oct. 2006), 41–54.

John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. 2016. DeepStereo: Learning to Predict New Views From the World's Imagery. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Yasutaka Furukawa and Carlos Hernández. 2015. Multi-View Stereo: A Tutorial. *Foundations and Trends in Computer Graphics and Vision* 9, 1-2 (2015), 1–148.

Yoav HaCohen, Eli Shechtman, Dan B. Goldman, and Dani Lischinski. 2011. Non-rigid Dense Correspondence with Applications for Image Enhancement. In *ACM SIGGRAPH 2011 Papers (SIGGRAPH '11)*. ACM, New York, NY, USA, Article 70, 70:1–70:10 pages.

Samuel W. Hasinoff, Sing Bing Kang, and Richard Szeliski. 2006. Boundary matting for view synthesis. *Computer Vision and Image Understanding* 103, 1 (2006), 22–32.

Kaiming He, Jian Sun, and Xiaoou Tang. 2010. Guided Image Filtering. In *Proceedings of the 11th European Conference on Computer Vision: Part I (ECCV'10)*. Springer-Verlag, Berlin, Heidelberg, 1–14.

Asmaa Hosni, Michael Bleyer, Christoph Rhemann, Margrit Gelautz, and Carsten Rother. 2011. Real-time Local Stereo Matching Using Guided Image Filtering. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME 2011)*. 1–6. Vortrag: IEEE International Conference on Multimedia and Expo (ICME), Barcelona, Spain; 2011-07-11 – 2011-07-15.

Hae-Gon Jeon, Jaesik Park, Gyeongmin Choe, Jinsun Park, Yunsu Bok, Yu-Wing Tai, and In So Kweon. 2015. Accurate Depth Map Estimation From a Lenslet Light Field Camera. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Nima Khademi Kalantari, Ting-Chun Wang, and Ravi Ramamoorthi. 2016. Learning-Based View Synthesis for Light Field Cameras. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)* 35, 6 (2016).

Sing Bing Kang, Richard Szeliski, and Jinxiang Chai. 2001. Handling Occlusions in Dense Multi-view Stereo.. In *CVPR (1)*. IEEE Computer Society, 103–110.

V. Kolmogorov and R. Zabih. 2001. Computing visual correspondence with occlusions using graph cuts. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, Vol. 2. 508–515 vol.2.

Johannes Kopf, Fabian Langguth, Daniel Scharstein, Richard Szeliski, and Michael Goesele. 2013. Image-based Rendering in the Gradient Domain. *ACM Trans. Graph.* 32, 6, Article 199 (Nov. 2013), 199:1–199:9 pages.

Tom Lokovic and Eric Veach. 2000. Deep Shadow Maps. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 385–392.

Jiangbo Lu, Hongsheng Yang, Dongbo Min, and Minh N. Do. 2013. Patch Match Filter: Efficient Edge-Aware Filtering Meets Randomized Search for Fast Correspondence Field Estimation. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '13)*. IEEE Computer Society, Washington, DC, USA, 1854–1861.

Ziyang Ma, Kaiming He, Yichen Wei, Jian Sun, and Enhua Wu. 2013. Constant Time Weighted Median Filtering for Stereo Matching and Beyond. In *Proceedings of the 2013 IEEE International Conference on Computer Vision (ICCV '13)*. IEEE Computer Society, Washington, DC, USA, 49–56.

Leonard McMillan and Gary Bishop. 1995. Plenoptic Modeling: An Image-based Rendering System. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '95)*. ACM, New York, NY, USA, 39–46.

William T. Reeves, David H. Salesin, and Robert L. Cook. 1987. Rendering Antialiased Shadows with Depth Maps. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*. ACM, New York, NY, USA, 283–291.

Sudipta Sinha, Drew Steedly, and Rick Szeliski. 2009. Piecewise Planar Stereo for Image-based Rendering, In Twelfth IEEE International Conference on Computer Vision (ICCV 2009).

Sudipta N. Sinha, Johannes Kopf, Michael Goesele, Daniel Scharstein, and Richard Szeliski. 2012. Image-based Rendering for Scenes with Reflections. *ACM Trans. Graph.* 31, 4, Article 100 (July 2012), 100:1–100:10 pages.

Noah Snavely, Steven M. Seitz, and Richard Szeliski. 2006. Photo Tourism: Exploring Photo Collections in 3D. In *ACM SIGGRAPH 2006 Papers (SIGGRAPH '06)*. ACM, New York, NY, USA, 835–846.

C. Strecha, R. Fransens, and L. Van Gool. 2004. Wide-baseline stereo from multiple views: A probabilistic account. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, Vol. 1. I–552–I–559 Vol.1.

Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. 2003. Stereo Matching Using Belief Propagation. *IEEE Trans. Pattern Anal. Mach. Intell.* 25, 7 (July 2003), 787–800.

Rick Szeliski and Polina Golland. 1999. Stereo Matching with Transparency and Matting. *International Journal of Computer Vision* 32/1 (May 1999), 45âĂŞ61.

Michael W. Tao, Sunil Hadap, Jitendra Malik, and Ravi Ramamoorthi. 2013. Depth from Combining Defocus and Correspondence Using light-Field Cameras. *International Conference on Computer Vision (ICCV)*.

Ting-Chun Wang, Alexei Efros, and Ravi Ramamoorthi. 2015. Occlusion-aware depth estimation using light-field cameras.. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *Trans. Img. Proc.* 13, 4 (April 2004), 600–612.

Sven Wanner and Bastian Goldluecke. 2014. Variational light field analysis for disparity estimation and super-resolution. 36 (2014), 606–619.

F. L. Zhang, J. Wang, E. Shechtman, Z. Y. Zhou, J. X. Shi, and S. M. Hu. 2016. PlenoPatch: Patch-based Plenoptic Image Manipulation. *IEEE Transactions on Visualization and Computer Graphics* PP, 99 (2016), 1–1.

Enliang Zheng, Enrique Dunn, Vladimir Jojic, and Jan-Michael Frahm. 2014. PatchMatch Based Joint View Selection and Depthmap Estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.