Depth from Gradients in Dense Light Fields for Object Reconstruction

Kaan Yücer[1,2] Changil Kim[1] Alexander Sorkine-Hornung[2] Olga Sorkine-Hornung[1]

[1]ETH Zurich [2]Disney Research

Abstract

Objects with thin features and fine details are challenging for most multi-view stereo techniques, since such features occupy small volumes and are usually only visible in a small portion of the available views. In this paper, we present an efficient algorithm to reconstruct intricate objects using densely sampled light fields. At the heart of our technique lies a novel approach to compute per-pixel depth values by exploiting local gradient information in densely sampled light fields. This approach can generate accurate depth values for very thin features, and can be run for each pixel in parallel. We assess the reliability of our depth estimates using a novel two-sided photoconsistency measure, which can capture whether the pixel lies on a texture or a silhouette edge. This information is then used to propagate the depth estimates at high gradient regions to smooth parts of the views efficiently and reliably using edge-aware filtering. In the last step, the per-image depth values and color information are aggregated in 3D space using a voting scheme, allowing the reconstruction of a globally consistent mesh for the object. Our approach can process large video datasets very efficiently and at the same time generates high quality object reconstructions that compare favorably to the results of state-of-the-art multi-view stereo methods.

1. Introduction

Reconstructing objects in 3D from a set of pictures is a long standing problem in computer vision, since it enables many applications, including (but not limited to) digitizing real-world objects, localization and navigation, and generating 3D content for computer graphics and virtual reality. Due to the breadth of these possible use cases, image based 3D reconstruction has been well studied, and many different solutions have been proposed, ranging from generating meshes for small objects [15] to reconstructing entire cities [1].

Despite significant research efforts, objects with thin features and fine details still pose a problem to multi-view stereo (MVS) techniques due to numerous reasons. First of all, these features occupy only a small number of pixels in the views they are visible in, making locating them difficult. They are also usually only visible in a small number of views, such that matching between different views becomes challenging. Moreover, most patch based MVS techniques miss thin features, since they require the patches on the objects to be several pixels wide, which is not always the case in real-life capture scenarios. Graph-cut based reconstruction techniques [21, 35] face difficulties with textureless thin features, because it is hard for such methods to localize the features using photoconsistency values inside a volumetric discretization, often resulting in elimination of these features in the reconstructions.

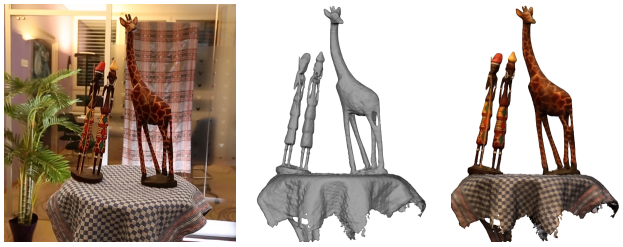

Figure 1: Given a dense light field around a foreground object, our technique reconstructs the object in 3D with its fine details. Left: An input view from the AFRICA dataset. Right: The reconstructed mesh rendered with and without the texture. Note the reconstructed details, such as the stick, the legs and the arms of the figurines.

In this work, we propose an algorithm that uses light fields with high spatio-angular sampling, such that fine features become more prominent thanks to the increased coherency and redundancy in the data. The features are visible in more views and occupy a larger amount of pixels inside the light field compared to sparsely sampled images. Increased coherency of the data also makes it possible to work directly on the pixel level without requiring patch based matching, such that thin object parts can be identified (Figure 1). At the same time, dense light fields are challenging to process due to the sheer amount of available data. Existing multi-view stereo methods evolved largely towards handling sparse, imperfect input and thus cannot handle such dense data effectively.

Recent novel approaches to light field processing have already shown an increase in the quality of the depth maps, where finer details are visible in the reconstructions [24, 37].

However, the resulting scene representations are per-view depth maps, which are not always globally consistent and require an additional surface reconstruction step to generate more widely used representations, *e.g*. triangle meshes. Furthermore, such techniques often specialize in regularly sampled light fields (such as regular 2D grids) and do not easily generalize to more casual, hand-held capture scenarios. A more recent method [40] generates pixel-accurate segmentations for casually captured light fields using pixel-level approaches. Unfortunately, segmentations can only be used to describe the objects in form of visual hulls, where concavities are missing.

Our approach makes the following contributions: Firstly, we propose an efficient and reliable technique for computing per-pixel depth values by utilizing the gradient directions inside the light field volume. In this way, depths at image edges can be computed without the need to match features or optimize for 3D patches. Secondly, we utilize a novel directional photoconsistency computation, which makes it possible to differentiate between texture and boundary edges in the images. This computation is instrumental in propagating depth values from high gradient regions to smooth areas. Lastly, we propose a robust depth aggregation technique that thrives with a large amount of depth maps and extracts precise and detailed object surfaces from the gathered information.

Every step of our technique is designed to take advantage of the highly coherent light field data. Typically, we capture an HD video containing thousands of frames to sample the light field surrounding an object. Since such dense light fields contain more samples about every scene point compared to traditional MVS techniques, we can work on pixel level and make robust decisions about the pixels independently. Most existing MVS methods cannot make use of such data in full. We designed our pipeline to be able to handle the sheer amount of data and avoid long run times or large memory requirements. These design decisions make it possible to manage light fields consisting of thousands of views efficiently and generate high quality 3D reconstructions for objects with fine details.

2. Related work

3D reconstruction from light fields. Beyond the early application of light fields in image based rendering [6, 9, 19, 26], the dense representation of the scene they attain has provided new perspectives to a variety of existing computer vision problems. The high coherency in light fields can be utilized to extract higher level information about the scene, such as understanding specular properties of objects [8], segmenting scenes into multiple components [38], computing scene flow [2], synthesizing superresolution images [34], and generating high quality per-view depth maps [24, 39]. The use of image statistics [7] and explicitly trying to identify occlusion edges [36] can enhance the quality of depth maps, especially around object boundaries. Most recently, convolutional neural networks [20] have been applied to solve the depth reconstruction problem. Similar ideas have been used to reconstruct depth maps from light fields captured with lenslet arrays [3, 27, 33]. Our aim in this paper is complementary to these works: we strive to reconstruct the 3D shape of the object using a dense light field captured from view points around the object, making use of the local gradient information to compute fast and reliable depth maps without the need for optimization techniques.

When capturing a light field from a scene, the scene points' projections to the views create trajectories depending on the 3D position of the scene point and the camera positions. Exploiting such trajectories for direct reconstruction of depth or segmentations have been a promising research direction since its first introduction by Bolles *et al*. [4], who analyzed several different camera alignments, including linear alignment. Feldmann *et al*. [10] compute point clouds by fitting analytical curves to the trajectories, but are limited to perfectly circular camera motion. Fitting lines to the epipolar plane images can reveal the depths of scene points [37], but this computation requires the camera positions to be aligned on a regular 2D grid. More recent work [40] has shown that light field gradients can be utilized to segment the foreground object from the background clutter both in 2D as segmentation masks, and in 3D, described by a visual hull, without requiring stringent capture scenarios. In our work, we extend the gradient computations to estimate depth values instead of segmentations, and the depths are then used to generate 3D surface meshes with concavities and fine details.

Multi-view stereo. The aim of multi-view stereo is to reconstruct objects in 3D using a set of images. We discuss the previous work on MVS that is most related to ours. For a more detailed survey and evaluation of MVS, we refer the reader to the comprehensive studies [13, 29].

A common approach to object reconstruction is to start with a visual hull of the object and carve it out to get the actual shape with concavities [14, 21, 31, 35]. The carving operation is done either via volumetric graph-cuts inside the visual hull volume, or by deforming the visual hull mesh via an energy optimization. In order to get the initial surface, per-image segmentations need to be computed either by hand or by background subtraction, which is time consuming with thousands of images and cluttered backgrounds. With high voxel and image resolutions, computing a global solution can become infeasible. Also, silhouette extraction is prone to errors, which requires special handling for thin objects [32].

Another general direction is to create reliable oriented point clouds by applying patch-based feature matching, and then expand around these points to cover the object [1, 15, 18]. The resulting dense point cloud is turned into a mesh by a surface reconstruction technique, *e.g*. Poisson reconstruction [23]. Thin features, which cannot be matched via feature

matching, and textureless regions pose problems, since without sufficient number of points in these areas, they cannot be accurately represented in the final mesh.

Computing per-image depth maps and merging them later in 3D space has been shown to generate high quality object reconstructions [5, 17, 28, 30]. Usually, depth maps are computed using dense correspondence matching, which is susceptible to image noise or large smooth regions without regularization, while applying regularization often destroys fine details. Further, correspondence matching and the subsequent triangulation becomes inaccurate in narrow baseline scenarios, as is the case with densely sampled light fields. In this work, we propose an efficient narrow baseline depth computation technique using light field gradients, which can be applied for each pixel independently, and a depth map aggregation technique that can merge a large number of depth maps efficiently.

3. Depth from gradient

Our goal is to reconstruct objects in 3D that are captured densely in unstructured light fields [9]. As in Davis *et al*. [9], we do not try to reparameterize the captured light field, but work directly on a sequence of images and associated camera poses. Such unstructured light fields can be easily captured at a very dense sampling rate by a video camera moving around the objects. We do not assume a particular camera path nor a machine gantry, but assume only that the camera moves roughly along a path surrounding the object to sample as much of it as possible. In a more regularly sampled light field, each scene point will leave a distinctive trajectory in the spatio-angular volume of the light field depending on the particular camera motion and the 3D position of the scene point: lines with different slopes for linear camera motion [4] or helical structures for circular camera motions [10]. The direction of such trajectories encodes the depths of scene points, and has been used to reconstruct scene depth.

As the spatio-angular sampling rate of the light field increases, the parallax between views starts getting smaller, and each 3D point's trajectory forms a more continuous curve, where the direction of the trajectory could be estimated locally (See Figure 3) without requiring any specific camera motion. The gradient direction over a few adjacent frames gives a local, linear approximation of the trajectory of each pixel, from which depths of scene points can be computed. This observation opens up the possibility to compute per-pixel depth values efficiently by solely measuring light field gradients. Since this does not require any patches or features to be computed and matched between images but uses local information only, pixel-wise depth values can be computed more efficiently. Further, since the depth estimation is done on a per-pixel basis, potentially finer object details can be revealed and computation can be trivially parallelized.

We first compute the per-view depth maps using the gradient only in high-gradient regions. The depth values are propagated within each image guided by a bi-directional photo-consistency. All depth maps are then aggregated into a voxelized 3D scene, and further refined to result in an oriented point cloud, which are turned into a triangle mesh using Poisson surface reconstruction.

3.1. Depth computation

Given a dense 3D light field L represented by a set of images $(I_1, \ldots, I_n)$, where $L(x, y, i)$ refers to the pixel $\mathbf{p} = (x, y)$ in image I_i, and their camera projection matrices $\mathbf{P}_i$, we define the light field gradient $\nabla L_{i,j}$ around $\mathbf{p}$ in image I_i and $\mathbf{q}$ in image I_j as:

$$\nabla L_{i,j}(\mathbf{p}, \mathbf{q}) = \nabla s_{i,j}(\mathbf{p}, \mathbf{q}), \tag{1}$$

where $s_{i,j}(\mathbf{p}, \mathbf{q})$ is a 2×5 image patch constructed by stacking 5-pixel-long light field segments centered at $\mathbf{p}$ in I_i and centered at $\mathbf{q}$ in I_j together. We construct $s_{i,j}(\mathbf{p}, \mathbf{q})$ by using the epipolar geometry between the views I_i and I_j: We know that the actual scene point at $\mathbf{p}$ will appear on its epipolar line ℓ in I_j. Given a reference point $\mathbf{q}$ in I_j along ℓ, we sample a 5-pixel-long segment in I_j along ℓ centered at $\mathbf{q}$ to generate $s_j(\mathbf{p}, \mathbf{q})$. $\mathbf{q}$ also corresponds to a depth value $d_{\mathbf{q}}$ for $\mathbf{p}$ as a result of epipolar geometry. We project the sampled points in I_j back to I_i using a fronto-parallel plane placed at depth $d_{\mathbf{q}}$, and sample I_i at these locations to generate $s_i(\mathbf{p}, \mathbf{q})$. Note that this computation is reliable only when I_i and I_j face similar directions. If the depth value $d_{\mathbf{q}}$ is the actual depth value for $\mathbf{p}$, then we expect $s_i(\mathbf{p}, \mathbf{q})$ and $s_j(\mathbf{p}, \mathbf{q})$ to be identical. If the actual depth deviates from $d_{\mathbf{q}}$, then the colors in $s_j(\mathbf{p}, \mathbf{q})$ will be a shifted version of the colors in $s_i(\mathbf{p}, \mathbf{q})$. In both cases, we can use $\nabla s_{i,j}(\mathbf{p}, \mathbf{q})$ to compute the trajectory of the points between the two segments using the direction perpendicular to the gradient direction:

$$\gamma_{i,j}(\mathbf{p}, \mathbf{q}) = \tan^{-1}(-\nabla_x s_{i,j}(\mathbf{p}, \mathbf{q}) / \nabla_y s_{i,j}(\mathbf{p}, \mathbf{q})). \tag{2}$$

Using $\gamma_{i,j}(\mathbf{p}, \mathbf{q})$, we can find the mapping p_j^s of $\mathbf{p}$ in $s_j(\mathbf{p}, \mathbf{q})$:

$$p_j^s = 1 / \tan(\gamma_{i,j}(\mathbf{p}, \mathbf{q})), \tag{3}$$

Please refer to our supplemental material for the derivation of this formula. We then map $\mathbf{p}_j^s$ back to ℓ to compute the mapping $\mathbf{p}_j$, from which the actual depth $d_{\mathbf{p}}$ can be computed via triangulation (see Figure 3 for a visualization).

The question then becomes how to sample $\mathbf{q}$ in I_j. If $d_{\mathbf{q}}$ is close to the actual depth of the scene point at $\mathbf{p}$, then we can expect a reliable depth computation. However, if the difference between $\mathbf{q}$ and $\mathbf{p}_j$ is larger than a pixel, the gradient computation becomes unstable, leading to erroneous results. To that end, we sample ℓ multiple times between $\mathbf{q}_{min}$ and $\mathbf{q}_{max}$, which correspond to reference points for the minimum and maximum depths of the scene, and get a set of reference points $\mathbf{q}^k$, $k \in 1, \ldots, K$, where K is the

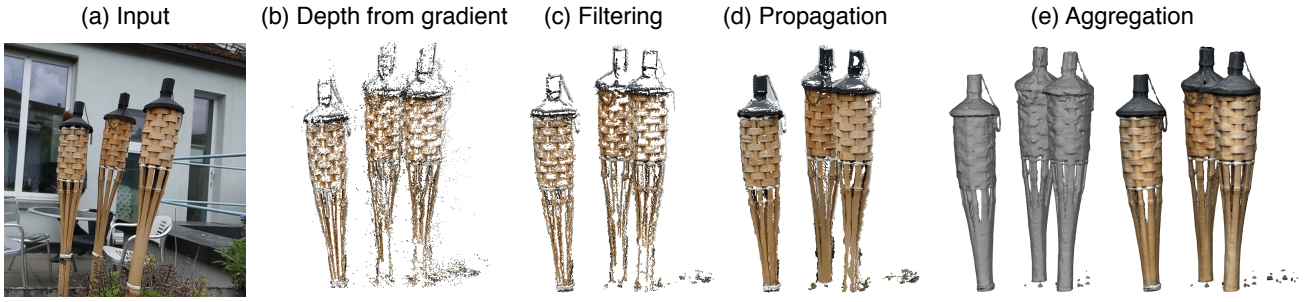

Figure 2: Steps of our technique. From left to right: One input frame, depth estimates using depth from gradient (Section 3.1), the filtered depth map (Section 3.2), the dense depth map after edge-aware depth propagation (Section 4), and the final meshes after depth aggregation (Section 5) with and without texture. The depth maps and meshes are rendered from a slightly different viewpoint for a better 3D visualization.

number of samples. We sample $\mathbf{q}^k$ one pixel apart from each other, compute a mapping $\mathbf{p}_j^k$ for each reference point $\mathbf{q}^k$ (see Figure 3), and choose the depth $d_{\mathbf{p}}^k$ that maximizes two confidence measures. First, we expect the colors of $\mathbf{p}$ and $\mathbf{p}_j^k$ to be similar due to color constancy:

$$C_i^c(\mathbf{p}, \mathbf{p}_j^k) = \exp\left(-\frac{1}{2\sigma_c^2}\left\|I_i(\mathbf{p}) - I_j(\mathbf{p}_j^k)\right\|_2^2\right). \quad (4)$$

For our experiments, we used $\sigma_c = 0.025$. The gradient computation results in more robust depth estimates, if $\mathbf{q}^k$ and $\mathbf{p}_j^k$ are close, *i.e.* the depth $d_{\mathbf{p}}^k$ of $\mathbf{p}$ is close to the depth of the plane $d_{\mathbf{q}}^k$ used for gradient computation:

$$C_i^d(\mathbf{p}, \mathbf{p}_j^k) = \exp\left(-\left|d_{\mathbf{q}}^k - d_{\mathbf{p}}^k\right|^2\right). \quad (5)$$

The final confidence measure is computed by multiplying the individual components in

$$C_i(\mathbf{p}, \mathbf{p}_j^k) = C_i^c(\mathbf{p}, \mathbf{p}_j^k) \cdot C_i^d(\mathbf{p}, \mathbf{p}_j^k). \quad (6)$$

For each $\mathbf{p}$, we choose $\mathbf{p}_j^k$ which maximizes this confidence measure as the mapping $\mathbf{p}_j$, and store the depth value $d_{\mathbf{p}}$ and the confidence value $c_{\mathbf{p}}$.

We start by computing the depth maps for I_i using the nearest neighbors I_{i-1} and I_{i+1}, and hierarchically move to next images in L to adjust the depth estimates further. After the initial step, we use the depth estimate $d_{\mathbf{p}}$ for $\mathbf{p}$ as the initial guess, and sample K reference points $\mathbf{q}^k$ around the new $\mathbf{q}$ in I_{i+2} corresponding to $d_{\mathbf{p}}$. We again sample the reference points one pixel apart from each other. Note that as we move further away from I_i in L, the relative motion of a point along the epipolar line with respect to the change in depth gets faster. However, since we again sample K reference points, we implicitly make the depth range smaller at each step, leading to more precise depth estimates. We compute depths over the views whose viewing directions are no more different from that of I_i than 5° for each viewpoint, and store the final depth maps D_i with their confidence maps C_i. Since the scene points' trajectories are only visible around high gradient regions, we compute the depths only around regions with enough gradient response, *i.e.* $\forall \mathbf{p}, \|\nabla I_i(\mathbf{p})\| > g$, where $g = 0.05$.

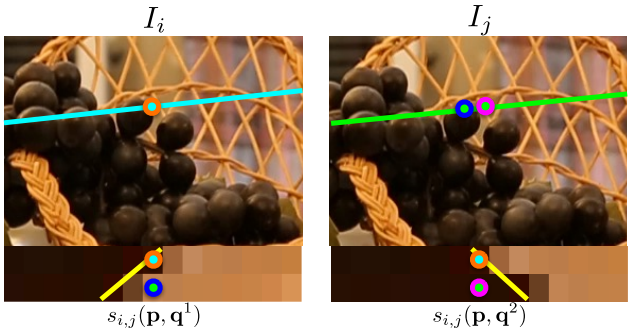

Figure 3: Top: Two close-up views of two images (I_i and I_j) from the BASKET dataset. The blue and green lines represent the lines that are sampled to generate $s_{i,j}(\mathbf{p}, \mathbf{q})$. The orange dot in I_i is $\mathbf{p}$, whose depth is computed. The blue and magenta dots in I_j are two different reference points $\mathbf{q}^1$ and $\mathbf{q}^2$, around which $s_j(\mathbf{p}, \mathbf{q})$ is sampled. Bottom: $s_{i,j}(\mathbf{p}, \mathbf{q})$ around the two different $\mathbf{q}$, on which the light field gradient $\nabla L_{i,j}(\mathbf{p}, \mathbf{q})$ is computed. The yellow lines show the trajectory directions $\gamma_{i,j}(\mathbf{p}, \mathbf{q})$. Note that different $\mathbf{q}^k$ result in different trajectory directions.

3.2. Depth filtering

Merging all per-view depth maps computed by our depth-from-gradient approach, one obtains a dense point set in the high-gradient regions of the object; see Figure 2(b). The depth computation assumes that the scene points are on fronto-parallel surfaces as seen from the central viewpoint. However, the actual surface shape may deviate from this assumption, and when using further-away views, this can lead to depth estimates that can differ from the actual surface.

We remove such noisy estimates of each depth map by examining their consistency with the estimates from other views with similar viewing directions. To this end, we discretize the 3D space where the foreground object resides using a fine, regular voxel grid $\mathcal{V}$. We denote the image regions that project inside this grid as foreground pixels, and

the rest as background pixels. In order to filter a depth map D_i of view I_i, we back-project to $\mathcal{V}$ the depth values and the confidences of all views whose viewing directions are different no larger than 15° from that of I_i. For each $\mathbf{v} \in \mathcal{V}$, we sum up the contributions of all back-projected 3D foreground points $\mathbf{x}$ using a voting scheme defined as follows:

$$H(\mathbf{v}) = \sum_{\mathbf{x}} c_{\mathbf{x}} \cdot \exp\left(-\frac{1}{2\sigma_v^2}\|\mathbf{v}-\mathbf{x}\|_2^2\right), \qquad (7)$$

where $c_{\mathbf{x}}$ is the confidence value associated to $\mathbf{x}$. We used σ_v equal to the length of a voxel in our experiments.

We reassign the depth $D_i(\mathbf{p})$ of each foreground pixel $\mathbf{p}$ to the depth value of the most voted voxel along the viewing ray from the camera center through $\mathbf{p}$; see Figure 2(c). Since we are interested in the shape of the foreground object, we filter only the foreground points, while background depths are kept as they are.

4. Depth propagation

The acquired point cloud, as described in Section 3, already reveals the accurate shape of the object, but only in highly textured regions and over the silhouettes. In order to generate a more complete 3D object reconstruction, we propagate the depth information towards low-gradient regions, for which we use the information about whether any high-gradient region corresponds to texture or object boundaries.

Thanks to the dense sampling of the data, we can differentiate between texture and silhouette edges by looking at the trajectory of 3D points corresponding the edges. For a texture edge, scene points on both sides of the edge will have similar trajectories, whereas for silhouette edges, only one side of the edge follows the same trajectory. To utilize this observation, we propose to use the *bidirectional* photoconsistency, which assesses how consistently both sides of an edge move in different views. The use of bidirectional photoconsistency has two advantages. First, we can differentiate texture edges from silhouette edges, since only texture edges will be consistent on both sides of the edge. Second, when propagating the depth values from edges to smoother regions, we can use it to decide on which side to propagate.

4.1. Bidirectional photoconsistency

The bidirectional photoconsistency measures the texture variation on both side of the image edge separately. For a pixel $\mathbf{p}$ in I_i, whose depth value is $d_{\mathbf{p}} = D_i(\mathbf{p})$, we first compute its image gradient direction:

$$\theta(\mathbf{p}) = \tan^{-1}(\nabla_y I_i(\mathbf{p}) / \nabla_x I_i(\mathbf{p})). \qquad (8)$$

Note that $\theta(\mathbf{p})$ is different than $\gamma_{i,j}(\mathbf{p},\mathbf{q})$; $\theta(\mathbf{p})$ is computed per image, whereas $\gamma_{i,j}(\mathbf{p},\mathbf{q})$ is computed between different images inside L. Then, we sample a thin rectangular patch on each side of $\mathbf{p}$ along $\theta(\mathbf{p})$; see Figure 4. We vectorize the sampled pixels within the two patches, and denote the one taken in the positive $\theta(\mathbf{p})$ direction by $\mathbf{f}_+$ and the other by $\mathbf{f}_-$. The two patches are then projected to the neighboring views in L through a fronto-parallel plane placed at $d_{\mathbf{p}}$. In a second view, say I_j, the pixels within the projected patches are sampled, forming $\mathbf{g}_+$ and $\mathbf{g}_-$, also vectorized. In our implementation, for each direction, we sample three pixels along $\theta(\mathbf{p})$ in I_i and three other pixels in I_j at the locations that are projected from the three pixels of I_i. One side of the photoconsistency for $\mathbf{p}$ between I_i and I_j is then defined as the patch difference between $\mathbf{f}_+$ and $\mathbf{g}_+$:

$$\rho(\mathbf{f}_+, \mathbf{g}_+) = \exp\left(-\frac{1}{2\sigma_p^2}\|\mathbf{f}_+ - \mathbf{g}_+\|_2^2\right). \qquad (9)$$

The other side of the photoconsistency is defined similarly for $\mathbf{f}_-$ and $\mathbf{g}_-$. We chose σ_p to be the same as σ_c in Eq. (4). The bidirectional photoconsistency values $C_+(\mathbf{p})$ and $C_-(\mathbf{p})$ are computed by averaging all pairwise photoconsistency values among the views in L whose viewing directions are no more different than 5° from that of I_i.

The bidirectional photoconsistency indicates the likelihood of both sides being on the same depth as $\mathbf{p}$: if $\mathbf{p}$ is on the silhouette, the background seen in one side will move at a different speed with respect to the camera, leading to a low consistency value for that side (Figure 4). The differentiation between texture and silhouette edges helps decide on the direction to which the depth is propagated.

4.2. Edge-aware depth propagation

The depth maps D_i are sparsely sampled, since the depths and the consistencies are computed only on high gradient regions. In this step, we propagate them to smooth regions using edge-aware filtering, thereby exploiting the computed photoconsistencies. However, each $\mathbf{p}$ on a high gradient region has two photoconsistency values, one for each direction along $\theta(\mathbf{p})$, which needs special care during filtering. Since we know that the direct neighbors in these directions should share the depth and confidence values with the edge regions, we propose a simple splatting strategy to avoid this special case: The neighboring pixel $\mathbf{p}'$ in the positive $\theta(\mathbf{p})$ direction from $\mathbf{p}$ is assigned $C_+(\mathbf{p})$, whereas the neighboring pixel in the negative $\theta(\mathbf{p})$ direction is assigned $C_-(\mathbf{p})$. The depth values $D_i(\mathbf{p}')$ are initialized with $D_i(\mathbf{p})$. If a pixel $\mathbf{p}'$ is affected by multiple pixels on high gradient regions, we choose the depth and confidence values from the neighbor with the highest confidence value. For the high gradient regions, we keep the higher value of $C_+(\mathbf{p})$ and $C_-(\mathbf{p})$ as $C_i(\mathbf{p})$.

Now that we have per-pixel depth and confidence maps for each view, we employ confidence-weighted joint-edge-aware filtering using the images I_i inside L as the joint-domains, as proposed by Lang *et al.* [25], which makes use of the geodesic filter by Gastal and Oliveira [16]. First, we multiply D_i and C_i element-wise and filter them using the

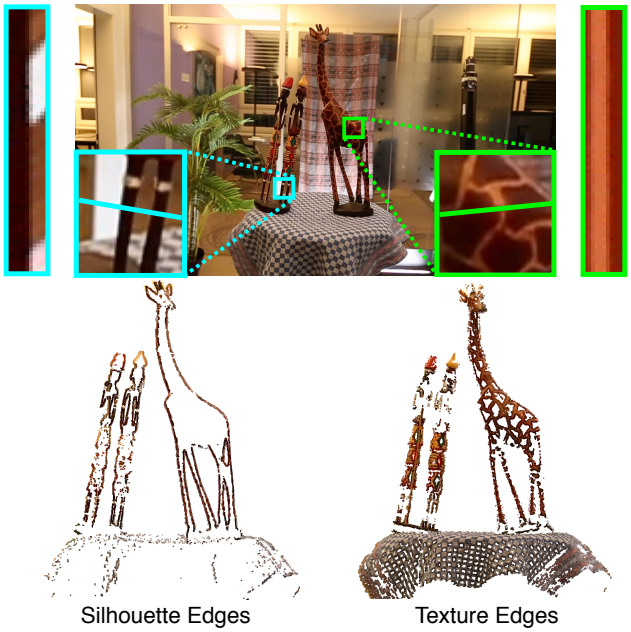

Figure 4: Top: One view of the AFRICA dataset in the center, with close-ups to two regions where bi-directional PC is computed. The leg of the person and the texture on the giraffe shown in cyan and green boxes, are sampled along the gradient directions, shown in matching colors. These patches and patches that are sampled from nearby views are stacked together for both edges and shown in boxes of matching colors. Note that only the texture edge is consistent on both sides of the image edge. Bottom: Reconstructed points marked as silhouette edges (left) and texture edges (right). For this visualization, we used the ratio $|c_+ - c_-|/(c_+ + c_-)$, and marked points with ratios higher than 0.3 as silhouette edges.

geodesic filter with I_i as the joint domain, which generates $(C_i \odot D_i)'$, where $\odot$ represents element-wise multiplication. This process gives higher emphasis to depth values with higher confidence. We then normalize the results by dividing $(C_i \odot D_i)'$ by C_i', the filtered version of the confidence map, again element-wise. The final depth map is computed as

$$D_i' = \frac{(C_i \odot D_i)'}{C_i'} \,. \tag{10}$$

We refer the readers to the original papers for a more in-depth discussion of this filtering operation.

In order to avoid depth values that are vaguely between the foreground object and the background clutter, we apply the filtering operation for the foreground and background depth maps seperately. If the confidence at $\mathbf{p}$ is larger in the foreground depth map, we keep this depth value for that pixel, and vice versa. The final confidence map is then the absolute difference between the confidence maps for the foreground and background depth maps. From this point on, we will refer to D_i' and C_i' as D_i and C_i, respectively.

5. Depth aggregation

The depth propagation step generates dense depth maps for each input image I_i independently, where smooth regions are assigned depth values by interpolating known depth values at image edges. These depth maps already describe the objects shape as a point cloud, but can have inconsistencies due to the inexact image-space filtering operation. We aggregate these depth maps in 3D space to reconstruct a globally consistent object representation in the form of a mesh.

Since the number of views is in the order of thousands, computing globally consistent depth maps is not a viable option due to the time complexity. On the other hand, having a very large number of depth maps has the advantage that their consensus in 3D space provides enough information to infer the actual surface. Noisy estimates from a small number of views can be tolerated by correct estimates from other views that see the same scene point. Our solution to compute the 3D surface relies on these observations and makes use of an efficient depth aggregation idea via a dense voxel grid.

We use the same voxel grid $\mathcal{V}$ as in Section 3.2, but this time, we utilize both foreground and background points. For each $\mathbf{v} \in \mathcal{V}$, we compute the probability $H(\mathbf{v})$ of that voxel being on the surface of the object. In order to compute these probabilities, we project every voxel $\mathbf{v}$ to the images, and interpolate D_i and C_i. Given a voxel $\mathbf{v}$ projects to a subpixel location $\mathbf{p}_i$ in I_i, with interpolated depth value d_i and confidence value c_i, we compute the per-view probability of having the surface at $\mathbf{v}$ by differentiating between two cases. If d_i falls inside $\mathcal{V}$, it is a foreground point. We then compute the confidence $c_{\mathbf{v},i}$ of having the surface at $\mathbf{v}$ using an exponential decay function, depending on the difference between d_i and $d_{\mathbf{v},i}$, the depth of $\mathbf{v}$ with respect to I_i:

$$c_{\mathbf{v},i} = c_i \cdot \exp\left(-|d_i - d_{\mathbf{v},i}|_2^2 / (2\sigma_v^2))\right). \tag{11}$$

If d_i is outside $\mathcal{V}$, *i.e.* is a background point, then we directly assign $c_{\mathbf{v},i} = -c_i$, since all voxels on this viewing ray should be in free space and affected in the same magnitude. Using these confidence values, we directly apply Bayes' rule to compute the per-view probability $P_i(\mathbf{v} \in \mathcal{S}|c_{\mathbf{v},i})$ of having the surface at $\mathbf{v}$, given the confidence value:

$$P_i(\mathbf{v} \in \mathcal{S}|c_{\mathbf{v},i}) = \frac{P(c_{\mathbf{v},i}|\mathbf{v} \in \mathcal{S}) \cdot P(\mathbf{v} \in \mathcal{S})}{P(c_{\mathbf{v},i})}, \tag{12}$$

where $\mathcal{S}$ stands for the set of voxels on the object surface. We model $P(c_{\mathbf{v},i}|\mathbf{v} \in \mathcal{S})$, *i.e.* the confidence value of a surface voxel $\mathbf{v}$, using $\mathcal{N}(1, \sigma_s)$, a normal distribution with mean of 1, to handle noise of per-view depth maps. The confidence value of a voxel in the free space, denoted by $P(c_{\mathbf{v},i}|\mathbf{v} \in \mathcal{F})$, is also modeled with a normal distribution $\mathcal{N}(-1, \sigma_s)$, but with mean of -1. The denominator in Eq. (12) can be computed as follows:

$$P(c_{\mathbf{v},i}) = P(c_{\mathbf{v},i}|\mathbf{v} \in \mathcal{S}) \cdot P(\mathbf{v} \in \mathcal{S}) + \\ + P(c_{\mathbf{v},i}|\mathbf{v} \in \mathcal{F}) \cdot P(\mathbf{v} \in \mathcal{F}). \quad (13)$$

We modeled $P(\mathbf{v} \in \mathcal{F})$ and $P(\mathbf{v} \in \mathcal{S})$ to be of equal probability, 0.5, due to no prior knowledge about the scene.

Our aggregation scheme is similar to that of Yücer *et al*. [40] in that we accumulate the per-image probabilities using a geometric mean. Given all $P_i(\mathbf{v} \in \mathcal{S}|c_{\mathbf{v},i})$, we compute the probability $H(\mathbf{v})$ using the following formula:

$$H(\mathbf{v}) = \left(\prod_{i=1}^{n} P_i(\mathbf{v} \in \mathcal{S}|c_{\mathbf{v},i}) \right)^{1/n}. \quad (14)$$

We generate the surface by thresholding $H(\mathbf{v})$ at 0.2 and applying marching cubes. We use a small value for thresholding, since our surface probabilities are generally of smaller magnitude compared to the free space probabilities.

Voxel carving The resulting mesh already captures most details of the object and is ready to be used as is. In order to pronounce the fine details further, we examine the photoconsistency of the voxels inside the surface.

A general solution for refining the mesh would be to apply volumetric graph-cuts inside the voxel grid. However, untextured thin features, like the legs and arms in the AFRICA dataset, or the straw details of the BASKET dataset, pose a problem for graph-cuts. Around such features, photoconsistency measures do not clearly point to the object boundary, and the graph-cut result can remove them from the final reconstruction altogether. Instead, we use an efficient voxel carving approach, which only carves out inconsistent voxels and keeps the thin features intact.

First, we compute a region of interest $\mathcal{R}$ inside the mesh, which is 3 voxels deep from the surface, and propagate mesh normals inside $\mathcal{R}$. The visibility of the voxels is computed using the current mesh as a prior and rendering the back-facing faces from each view point I_i. If a voxel's depth is smaller than the depth of the mesh seen from I_i, then it is counted as visible from I_i [22]. After all voxels $\mathbf{v} \in \mathcal{R}$ are projected to all images I_i, given a voxel $\mathbf{v}$, we gather the color values $\{\mathbf{c}_\mathbf{v}(i)\}$ and the weights $\{\mathbf{w}_\mathbf{v}(i)\}$ from all images I_i to which it projects. The weights of views that are not seeing the voxel are set to 0. For all other views, we compute the weight as the dot product of the voxel normal $\mathbf{n}_\mathbf{v}$ and the vieweing ray from I_i to $\mathbf{v}$, namely $\mathbf{r}_{\mathbf{v},i}$:

$$\mathbf{w}_\mathbf{v}(i) = \begin{cases} \mathbf{n}_\mathbf{v} \cdot \mathbf{r}_{\mathbf{v},i}, & \text{if } \mathbf{n}_\mathbf{v} \cdot \mathbf{r}_{\mathbf{v},i} > 0 \\ 0, & \text{otherwise} \end{cases}. \quad (15)$$

Given the colors and weights, we compute a weighted variance of the colors as the photoconsistency $PC(\mathbf{v})$:

$$PC(\mathbf{v}) = \sum_{i=1}^{n} \mathbf{w}_\mathbf{v}(i)(\mathbf{c}_\mathbf{v}(i) - \mu_\mathbf{v})^2 \Big/ \sum_{i=1}^{n} \mathbf{w}_\mathbf{v}(i), \quad (16)$$

where $\mu_\mathbf{v}$ is the weighted average of $\mathbf{c}_\mathbf{v}$. We carve out all voxels that have $PC(\mathbf{v})$ lower than a threshold, which we set to 0.95. The carving is repeated until no voxels are carved out. Our voxel carving approach is very efficient in removing unnecessary voxels from the surface, and converges very quickly. We performed a maximum of 3 iterations for all results in the paper. Finally, we supply all voxels $\mathbf{v}$ and their normals $\mathbf{n}_\mathbf{v}$ on the boundary of $\mathcal{R}$ to Poisson surface reconstruction [23] to generate our final results (See Figure 2).

6. Experiments and Results

In order to assess the quality and performance of our reconstruction method, we used the dense light field datasets of Yücer *et al*. [40]. The datasets feature various objects with fine details comprising many thin, intricate features, and complicated topology, which we aim at reconstructing. Since we intend to demonstrate our algorithm in a more realistic setup, we used the hand-held datasets. We compare our reconstructions with those from well-known reconstruction methods whose implementations are publicly available; namely PMVS [15] and MVE [12]. Additionally, we compared with the visual hull results accompanying the datasets.

Figure 5 shows the results of DECORATION, BASKET, and AFRICA obtained using PMVS, MVE, Yücer *et al*. (denoted by LFS), and our method. For the two multi-view stereo methods, we used as many views as possible until the results stop improving, which were about 200 views, and hand-picked the parameters to obtain the best results. The screened Poisson surface reconstruction (PSR) [23] was used to extract meshes from the PMVS point clouds, and the floating scale surface reconstruction (FSSR) [11] was used for MVE point clouds as it is bundled with MVE. For our method, we used the same set of parameters as reported in the paper regardless of the datasets.

While PMVS presents largely smooth results, it lacks detail around object features and shows irregular surfaces in the regions where the objects have complicate shapes. MVE results in clean surfaces in smooth regions, but in the regions with intricate detail, the reconstructed surfaces become noisy, which we believe is attributed to the FSSR's relative deficiency in tolerating outlier points. While LFS reveals a certain amount of details and topologically accurate surfaces, *e.g*. in the reconstructed BASKET, it is not able to handle concavities and the surfaces looking carved-out, *e.g*. see the insets of DECORATION. On the other hand, our method reveals the fine details of the object possessing many intricate features and topologically complicated structures; see the thin features and narrow openings in the insets.

In Figure 6 featuring vegetation, we compare our results also with the depth reconstruction method of Zhang *et al*. (ACTS) [41] which is tailored to dense video input. Since ACTS produces per-view depth maps, we merged them into point clouds for a better visualization. Compared to ACTS,

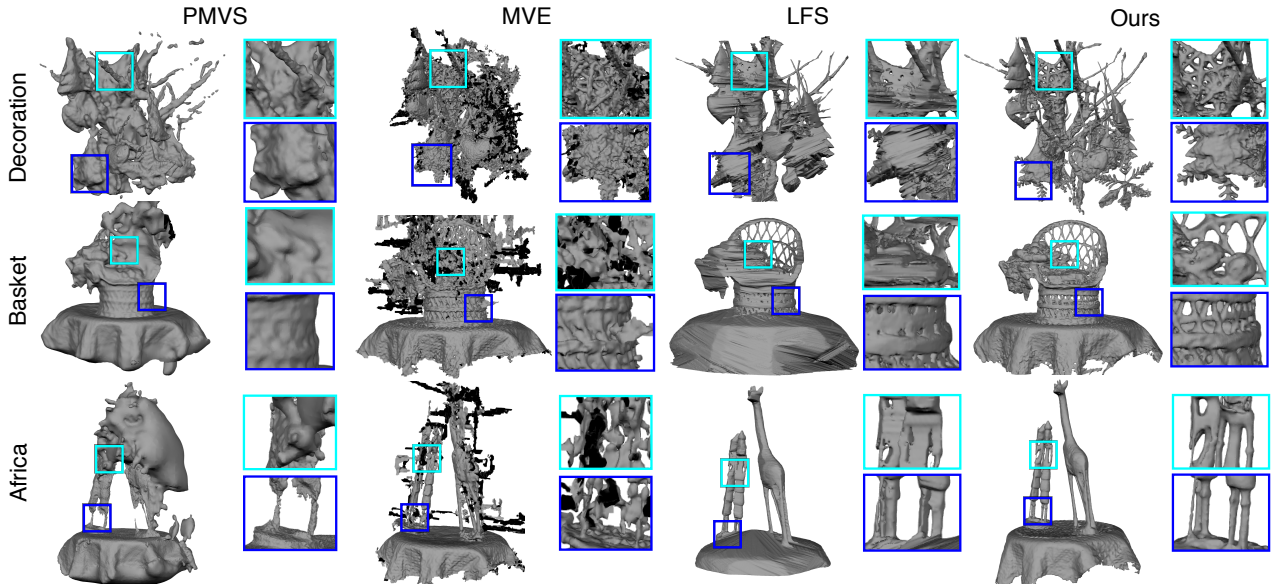

Figure 5: Comparison of our method to PMVS [15], MVE [18] and LFS [40] on 3 different datasets. The point clouds of PMVS are meshed with Poisson surface reconstruction [23] and the results of MVE are meshed using floating scale surface reconstruction [11] as part of their pipeline. For each result, we show two close-ups inside the cyan and blue boxes. Note the quality our method achieves around thin object features.

our method results in more faithful geometry of the plants with many thin elements, whereas ACTS results contain noise around the features and the method even consistently reconstructs points in free space as in the TRUNKS dataset. Please refer to the accompanying supplementary material for closer look at the results and input datasets.

Limitations. While most objects feature enough texture for our method to work, if the scene does not possess enough texture variations, our gradient-based depth calculation algorithm will leave large regions without depth estimates. In such cases, the propagation may not be effective enough to fill these areas and the subsequent aggregation step may leave holes. Our algorithm is both time and space efficient in most steps, but the final reconstruction resolution is tied to the voxel grid resolution. Its cubic complexity can be a downside when extremely high resolution results are required.

Performance. Many steps of our algorithm are suitable for parallelization, in particular depth computation and aggregation steps can be parallelized entirely. We ran an OpenMP-based parallel implementation on a desktop PC with 3.2 GHz Quad-core Intel CPU and 32 GB of RAM for the experiments. It takes about 120 minutes for our implementation to reconstruct the final oriented point cloud from a 720p video comprising 3000 frames. Depth computation and filtering are the most time consuming parts, taking about 55 minutes. PC computation and depth propagation require about 40 minutes, and the rest of time is spent for the aggregation step, which took around 25 minutes.

Figure 6: Comparison of our technique to the ACTS software [41]. For our algorithm, we show our resulting mesh with and without texture. Since ACTS produces point clouds, we show these from two different viewpoints.

7. Conclusions

We presented a method to reconstruct accurate and detailed 3D models from densely sampled light fields. A novel gradient-based method is proposed for efficient and accurate pixel-wise depth estimation, in addition to a depth propagation scheme assisted by bidirectional photoconsistency and an efficient and robust depth aggregation algorithm. Our method is designed to benefit and fully utilize extremely dense spatio-angular information, and reveals the highly detailed 3D models of objects with intricate and complex shape. We believe our contributions complement the existing multi-view stereo methods and will inspire further research.

References

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building Rome in a day. *Com-*

munications of the ACM, pages 105–112, 2011. 1, 2

[2] T. Basha, S. Avidan, A. Hornung, and W. Matusik. Structure and motion from scene registration. In *CVPR*, 2012. 2

[3] T. E. Bishop and P. Favaro. Full-resolution depth map estimation from an aliased plenoptic light field. In *ICCV*, 2010. 2

[4] R. C. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *IJCV*, 1(1):7–55, 1987. 2, 3

[5] D. Bradley, T. Boubekeur, and W. Heidrich. Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *CVPR*, 2008. 3

[6] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, 2001. 2

[7] C. Chen, H. Lin, Z. Yu, S. Bing Kang, and J. Yu. Light field stereo matching using bilateral statistics of surface cameras. In *CVPR*, 2014. 2

[8] A. Criminisi, S. B. Kang, R. Swaminathan, R. Szeliski, and P. Anandan. Extracting layers and analyzing their specular properties using epipolar-plane-image analysis. *CVIU*, 97(1):51–85, 2005. 2

[9] A. Davis, M. Levoy, and F. Durand. Unstructured light fields. *Comput. Graph. Forum*, 31(2):305–314, 2012. 2, 3

[10] I. Feldmann, P. Kauff, and P. Eisert. Extension of epipolar image analysis to circular camera movements. In *ICIP*, 2003. 2, 3

[11] S. Fuhrmann and M. Goesele. Floating scale surface reconstruction. *ACM Trans. Graph.*, 33(4):46, 2014. 7, 8

[12] S. Fuhrmann, F. Langguth, and M. Goesele. MVE – a multi-view reconstruction environment. In *Eurographics Workshop on Graphics and Cultural Heritage*, 2014. 7

[13] Y. Furukawa and C. Hernández. Multi-view stereo: A tutorial. *Foundations and Trends in Computer Graphics and Vision*, 9(1-2):1–148, 2015. 2

[14] Y. Furukawa and J. Ponce. Carved visual hulls for image-based modeling. *IJCV*, 81(1):53–67, 2009. 2

[15] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. *PAMI*, 32(8):1362–1376, 2010. 1, 2, 7, 8

[16] E. S. L. Gastal and M. M. Oliveira. Domain transform for edge-aware image and video processing. *ACM Trans. Graph.*, 30(4):69:1–69:12, 2011. 5

[17] M. Goesele, B. Curless, and S. M. Seitz. Multi-view stereo revisited. In *CVPR*, 2006. 3

[18] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *ICCV*, 2007. 2, 8

[19] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The Lumigraph. In *SIGGRAPH*, 1996. 2

[20] S. Heber and T. Pock. Convolutional networks for shape from light field. In *CVPR*, 2016. 2

[21] A. Hornung and L. Kobbelt. Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In *CVPR*, 2006. 1, 2

[22] A. Hornung and L. Kobbelt. Robust and efficient photo-consistency estimation for volumetric 3d reconstruction. In *ECCV*, 2006. 7

[23] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3), 2013. 2, 7, 8

[24] C. Kim, H. Zimmer, Y. Pritch, A. Sorkine-Hornung, and M. Gross. Scene reconstruction from high spatio-angular resolution light fields. *ACM Trans. Graph.*, 32(4), 2013. 1, 2

[25] M. Lang, O. Wang, T. Aydin, A. Smolic, and M. Gross. Practical temporal consistency for image-based graphics applications. *ACM Trans. Graph.*, 31(4):34:1–34:8, 2012. 5

[26] M. Levoy and P. Hanrahan. Light field rendering. In *ACM SIGGRAPH*, pages 31–42, 1996. 2

[27] C.-K. Liang and R. Ramamoorthi. A light transport framework for lenslet light field cameras. *ACM Trans. Graph.*, pages 16:1–16:19, 2015. 2

[28] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nistér, and M. Pollefeys. Real-time visibility-based fusion of depth maps. In *ICCV*, 2007. 3

[29] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, pages 519–528, 2006. 2

[30] Q. Shan, B. Curless, Y. Furukawa, C. Hernandez, and S. M. Seitz. Occluding contours for multi-view stereo. In *CVPR*, pages 4002–4009, 2014. 3

[31] S. N. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: A maximum-flow formulation. In *ICCV*, 2005. 2

[32] A. Tabb. Shape from silhouette probability maps: Reconstruction of thin objects in the presence of silhouette extraction and calibration error. In *CVPR*, pages 161–168, 2013. 2

[33] M. Tao, P. Srinivasa, J. Malik, S. Rusinkiewicz, and R. Ramamoorthi. Depth from shading, defocus, and correspondence using light-field angular coherence. In *CVPR*, 2015. 2

[34] K. Venkataraman, D. Lelescu, J. Duparré, A. McMahon, G. Molina, P. Chatterjee, R. Mullis, and S. Nayar. PiCam: An ultra-thin high performance monolithic camera array. *ACM Trans. Graph*, 32(6):166:1–166:13, 2013. 2

[35] G. Vogiatzis, C. Hernández Esteban, P. H. S. Torr, and R. Cipolla. Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *PAMI*, 29(12), 2007. 1, 2

[36] T.-C. Wang, A. A. Efros, and R. Ramamoorthi. Occlusion-aware depth estimation using light-field cameras. In *ICCV*, 2015. 2

[37] S. Wanner and B. Goldluecke. Globally consistent depth labeling of 4D lightfields. In *CVPR*, 2012. 1, 2

[38] S. Wanner, C. N. Straehle, and B. Goldluecke. Globally consistent multi-label assignment on the ray space of 4D light fields. In *CVPR*, pages 1011–1018, 2013. 2

[39] Z. Yu, X. Guo, H. Ling, A. Lumsdaine, and J. Yu. Line assisted light field triangulation and stereo matching. In *ICCV*, pages 2792–2799, 2013. 2

[40] K. Yücer, A. Sorkine-Hornung, O. Wang, and O. Sorkine-Hornung. Efficient 3D object segmentation from densely sampled light fields with applications to 3D reconstruction. *ACM Trans. Graph.*, 35(3), 2016. 2, 7, 8

[41] G. Zhang, J. Jia, T. Wong, and H. Bao. Consistent depth maps recovery from a video sequence. *PAMI*, 31(6), 2009. 7, 8