Abstract of “Are Multi-view Edges Incomplete?”
by Numair Khan, Ph.D., Brown University, February 2022.

Depth reconstruction tries to obtain 3D scene geometry from incomplete or low-dimensional data — and it is usually a vital first step in many computational photography tasks. Most image-space geometric representations, however, fail to be general-purpose as they focus on a narrow set of metrics, and do not preserve all information of potential relevance. This dissertation shows that multi-view edges encode all relevant information for supporting higher level computational photography tasks that rely on depth reconstruction. We do this by presenting a novel encoding of multi-view scene geometry from structured light fields, and a reconstruction method for inverting this code. Our model is based on edges in the Epipolar Plane Images (EPIs) of a light field. These edges provide a small number of high-gradient key points and depth labels that can be used to accurately identify occlusion boundaries, and also to anchor the reconstruction in the angular domain for view-consistency. We present a differentiable representation of our model which allows the reconstruction to be optimized via gradient descent on a multi-view reconstruction loss. We evaluate our reconstruction for accuracy, view consistency, and occlusion handling to show that it retains all the geometric information required for higher level computational photography tasks.

Are Multi-view Edges Incomplete?

by

Numair Khan

B. S., Lahore University of Management Sciences, 2009

Sc. M., New York University, 2014

Sc. M., Brown University, 2018

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island
February 2022

This dissertation by Numair Khan is accepted in its present form by
the Department of Computer Science as satisfying the dissertation requirement
for the degree of Doctor of Philosophy.

Date ______________ ________________________________

James Tompkin, Director

Recommended to the Graduate Council

Date ______________ ________________________________

John F. Hughes, Reader

Date ______________ ________________________________

Min H. Kim, Reader
(KAIST, South Korea)

Approved by the Graduate Council

Date ______________ ________________________________

Andrew G. Campbell
Dean of the Graduate School

To all my teachers over the years, especially my parents.

And to the memory of the Spring of 2020.

Acknowledgements

I would like to thank my advisor James Tompkin for his mentorship and advice; my committee members, Min H. Kim and John F. Hughes, for their guidance; Ghulam Murtaza, Usama Naseer, Abdul Manan and Suhaib Imtiaz, my Pakistani compatriots at Brown, for their friendship and support. I would also like to thank all my colleagues at the Visual Computing Lab, especially Qian Zhang, Eleanor Tursman, Yiqing Liang and Mikhail Okunev.

I am, of course, forever indebted to Mumtaz Ali, Kashif Qureshi, Ghulam Muhammad, Shahid Sultan, my siblings, and my parents.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

> ”[E]xperience building vision systems suggests that interesting things often happen at an edge in an image and it is worth knowing where edges are.” [27]

This dissertation shows that multi-view edges encode all relevant information for supporting higher level computational photography tasks that rely on depth reconstruction. Depth reconstruction is a vital first step in many computational photography tasks such as novel view synthesis [29, 19, 84, 85], scene editing [42, 68, 63], lighting and material estimation [30], and augmented reality [34]. The problem has a long history in the field of computer vision, with a wide variety of solutions having been proposed over the years. These include photometric stereo [118], shape from shading [124], depth from defocus [99, 37], active light methods [9, 23, 66, 71, 118], and increasingly in recent years, learning-based monocular methods [15, 61, 78, 77].

But the most popular and widely studied approach remains that of binocular and multi-view stereo depth estimation [91, 90]. In its basic form, this involves using epipolar constraints to perform a correspondence search at each pixel in neighboring images. The popularity of stereo depth estimation can be attributed to its ability to work in all environments and lighting conditions, its immunity to interference from competing signals, and the well-understood constraints of epipolar geometry that help solve the problem unambiguously.

However, stereo depth estimation is not without its drawbacks. Not only is the correspondence step computationally expensive, it is susceptible to failure in textureless, specular and disoccluded regions. Moreover, the baseline between neighboring cameras can have a significant impact on the quality of results. Small baselines reduce accuracy as the change in disparity relative to depth is low. Large baselines make it difficult to find corresponding points in neighboring images [48].

Nonetheless, stereo depth estimation remains a preferred solution — especially with the proliferation of camera sensors in recent years. Most smartphones these days have at least two front-facing cameras, with the Nokia 9 PureView featuring five. The Oculus Quest 2 headset has four built-in cameras. The Light L16 had sixteen! This explosion in the number of sensors has enabled new and interesting applications in computational photography. The five cameras of the Nokia 9 PureView allow it to synthetically refocus a photo. The Quest 2 uses its cameras to provide mixed reality experiences. And the L16 captured DSLR-quality photographs, even in low light conditions. Other applications include novel view synthesis or 3D photography [29, 75, 69, 94, 54, 70], high dynamic range (HDR) imaging [116], and extremely high speed video [116]. On the other hand, the large number of images leads to increased — often prohibitive — data and computational costs.

While depth estimation is a vital *first step* in many computational photography tasks, it is usually not the final goal. The quality of depth reconstruction often correlates with the quality of a higher level computational photography task to varying degrees and along different dimensions. On tasks such as novel view synthesis, for instance, even crude 3D reconstructions suffice for achieving high levels of performance [29, 126, 69, 54]. Mildenhall et al. [69] show that the quality of novel view synthesis plateaus after a certain level of depth complexity beyond which factors like disocclusions start playing the determining role. Moreover, the incorrect metric depth that correspondence-based depth estimation methods generate in specular regions, in fact, allows view-dependent effects to be accurately simulated in novel view synthesis. For certain 4D light field editing tasks even piece-wise constant depth — implicitly encoded in 4D segmentation masks — proves sufficient for achieving high quality results [68]. However, on tasks such as 3D painting [43, 42], scene relighting [118] or material estimation [30] that rely on correct surface normals, the geometric accuracy of the estimated depth is an important factor.

Given this, one may naturally ask about the *completeness* of a given depth reconstruction. Elder [25] defines completeness as the ability of a representation to capture all information of potential relevance to any higher level visual task. This dissertation shows the multi-view edges are indeed complete: they encode all relevant information for supporting higher level computational photography tasks that rely on depth reconstruction.

1.1 A Representation for Multi-View Depth

In addition to color data, multi-view images implicitly store the geometric structure of a scene. A 2D depth map provides a more explicit representation of this data. For computational photography tasks that rely on knowledge of 3D scene structure, clearly a depth map is more useful as a representation than a collection of RGB images. The depth map representation has lower entropy and, thus, requires less processing to extract the same information. But is it the most efficient such representation in terms of information? Based on the work of Elder [25] and the specific constraints of multi-view scene reconstruction, we propose the following criteria for evaluating a general-purpose image space geometric representation:

1. *Concision:* Any redundant information in the input should be discarded. This property — based on Barlow's efficient coding hypothesis [4], and proposed by Elder [25] for the representation of a single image — becomes especially useful for multi-view input. Multi-view images implicitly encode depth information in their angular dimension with high redundancy. Many methods [111, 107, 108, 125, 46, 90, 36, 119, 120] successfully exploit this redundancy for higher quality depth reconstruction than traditional stereo. However, storing the reconstructed result as a depth map per input view retains the redundancy even though it is no longer useful. Per-view depth maps, therefore, are not a concise representation and, thereby, have high data and computational costs.

2. *Explicitness:* Important structural information should be explicitly represented. Elder [25], quoting a distinction originally made by Adelson [2], describes this as the encoding representing "things" not "stuff." Depth maps, as described above, explicitly store geometric information. In addition, through their gradients, they describe occlusion boundaries. Spatio-angular segmentation masks [31, 68] define object surfaces. Both representations are more explicit than multi-view RGB images, which only implicitly encode this information.

3. *Completeness:* The representation should encode all relevant information to be able to support a variety of higher level computational photography tasks.

Elder also lists *generality*, *reliability* and *precision* as the evaluative measures for a visual representation. But we believe these are subsumed in the definition of completeness we provide next.

1.1.1 Completeness for Computational Photography Tasks

As discussed above, different applications are variously correlated with the quality dimensions of depth reconstruction. Approximate reconstructions suffice for novel view synthesis tasks [126, 69], but not for BRDF estimation [106, 74]. A complete representation should support the requirements of all these tasks. To quantify this criterion, we identify the following three metrics that the reconstructed depth from the representation is measured on:

Accuracy: Accuracy refers to the *correctness* of the estimated depth maps in metric terms. It quantifies the difference between the estimated depth and a known ground truth measure. Common quantitative metrics include the Mean Absolute Error (MAE), the Mean Squared Error (MSE), Q25, and a *bad pixel* measure $\mathrm{BP}(\cdot)$. The Q25 metric represents the 25th percentile of the absolute error, and $\mathrm{BP}(t)$ is the percentage of pixels falling above threshold t in absolute error.

As the goal of 3D reconstruction is the recovery of a faithful, or *correct*, representation of a scene, it is obvious that accuracy is an important metric and explains its primacy in reconstruction benchmarks [87, 105, 28, 67]. It is certainly important for computational photography applications — such as 3D painting [42], scene relighting [118], or material estimation [30] — that rely on correct surface normals. For other tasks, however, the mean reconstruction error over *all* pixels is less relevant than the error specifically for pixels lying on depth boundaries; in other words, the accuracy of occlusion edge reconstruction.

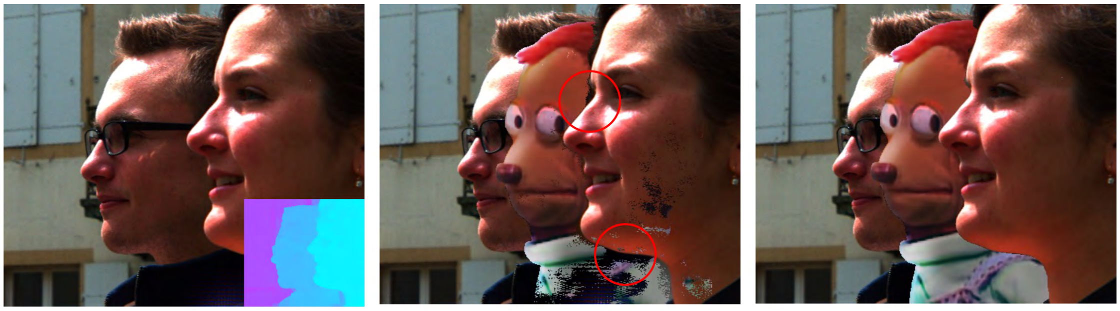


Figure 1.1: Scene editing requires high-accuracy occlusion edges. *Middle:* The dense depth estimated by Zhang et al. [125] illustrates the difficulty of extracting correct edges. *Right*: Using the approach we describe in Chapter 6 allows effective occlusion handling when editing light fields. The inset shows our disparity map.

Occlusion Edge Accuracy: Occlusion edge reconstruction accuracy is measured by restricting MSE, MAE, Q25 and $\mathrm{BP}(\cdot)$ to the vicinity of depth edges defined by the gradients of a ground truth measure or, more commonly, through precision-recall curves of the same edges.

As Figure 1.1 shows, high-accuracy depth edges are vital for image editing tasks. Correctly localizing depth edges, however, proves to be a difficult task. CNNs trained on a mean loss over all pixels fail to capture high frequencies (this is also due in part to spectral bias [76, 6], and the averaging effect implicit in the convolution operator). Methods such as Neural Radiance Fields [70] also inherently have a smoothness bias which lets them avoid degenerate solutions that may result from the shape-radiance ambiguity [123] but also prevents them from representing high frequency details effectively [101, 95]. Similar smoothing artifacts can be observed in depth fusion approaches [114, 18], especially those based on averaging signed distance functions [21, 41]. Many depth estimation methods, including the top-performers on the Middlebury Stereo Dataset [105], mitigate the blurriness of depth edges by using a discrete range of depth values. Others [54, 94] enforce strong occlusion edges by using a weighted median filter [64] on the estimated depth.

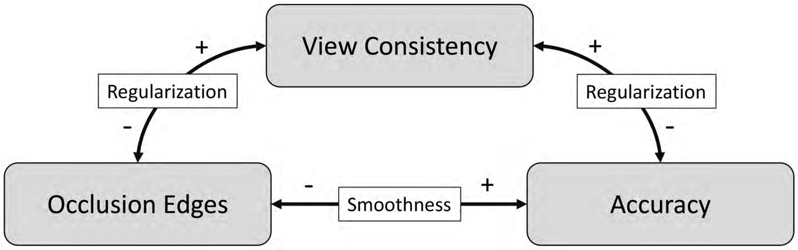


Figure 1.2: A fundamental trade-off exists between view-consistency, accuracy, and occlusion edge correctness. A complete depth representation should be general enough to optimize each metric separately.

View-Consistency: View consistency in multi-view depth estimation requires that the reconstruction of each view I_i as represented by a depth map $D_i : \mathbb{R}^2 \to \mathbb{R}$ in camera space coordinates, is globally consistent. View-consistency is vital for avoiding flickering and *swimming* artifacts in applications that involve interaction with all input views simultaneously or in quick succession such as when editing a light field when every output view will be seen on a light field display [42, 103].

Given the above description, we observe that a fundamental trade-off exists between the three metrics described above (Figure 1.2). Increased regularization makes the results more consistent, but at the cost of occlusion edge and general accuracy; in the extreme case, a single depth value for all pixels would provide the highest consistency with very low accuracy and no occlusion edges at all. A continuous or smooth range of depth values allows greater precision and, thereby, greater accuracy, but leads to lower gradient edges. A complete depth representation should, therefore, be general enough to optimize each metric separately.

1.2 Multi-view Edge Model

We propose a representation for multi-view depth that satisfies the criteria listed in Section 1.1. Inspired by Elder's [25] encoding of intensity images, we formulate our representation of geometry as a sparse model based on *multi-view edges* in image space. A multi-view edge refers to the pairing of a 2D edge location with a depth label which allows the edge point to be uniquely identified and localized in multiple views. Our model, described in detail in Chapter 4, is composed of multi-view edges, a parameter representing the occlusion direction at each edge location, and a confidence estimate of the occlusion boundaries in the input pixel array.

Our representation makes structural features of the input explicit. This not only includes edges, which provide important information about changes in the visual composition of a scene, but occlusion surfaces and depth boundaries too. The latter together provide important information about the 3D structure of a scene. Moreover, this information is encoded in a concise representation. Not only are redundant measurements avoided, but noisy estimates in smooth and texture-less regions — a banana-skin for most correspondence based dense depth estimation methods — are also discarded.

1.3 This Dissertation

This dissertation shows that multi-view edges encode all relevant information for supporting higher level computational photography tasks that rely on depth reconstruction. That is, we show that our proposed multi-view edge model is complete.

Chapter 4 shows how our model parameters are estimated from structured light field images. We present a method for detecting multi-view edges, and solve the ill-posed problem of extracting depth gradients from sparse depth labels by a bi-directional diffusion process. A confidence estimate of the occlusion boundaries is obtained by exploiting the fact that depth edges are more sensitive than texture edges to local constraints.

The completeness of our model is demonstrated in Chapters 5–7. This is done via two application-specific reconstruction methods that convert the multi-view edge model into a 2D depth map for evaluation. Chapter 5 poses reconstruction as a 4D superpixel segmentation problem and generates view-consistent piece-wise constant depth maps. Chapters 6 and 7 respectively present a smooth reconstruction method, and a differentiable representation of our multi-view edge model which can be optimized with respect to the completeness metrics described in Section 1.1.1 through gradient descent. Also in Chapter 7 we remove the structuring constraints of light fields and show the applicability of our model to general multi-view images.

Limitations of Scope Multi-view images encode both photometric and geometric information. Our work focuses only on the latter aspect by evaluating the criteria of Section 1.1.1 on a geometric reconstruction from our proposed model. In addition, we are interested only in showing the completeness property from Section 1.1, and do not study the concision or explicitness of our model. Moreover, we are only interested in answering whether our model is *a* complete representation of multi-view geometry, and not if it is *the best* such representation. Our work primarily uses structured light fields as a form of multi-view images, and we assume the scene is composed of Lambertian surfaces. We discuss the implications of these choices in Chapter 8.

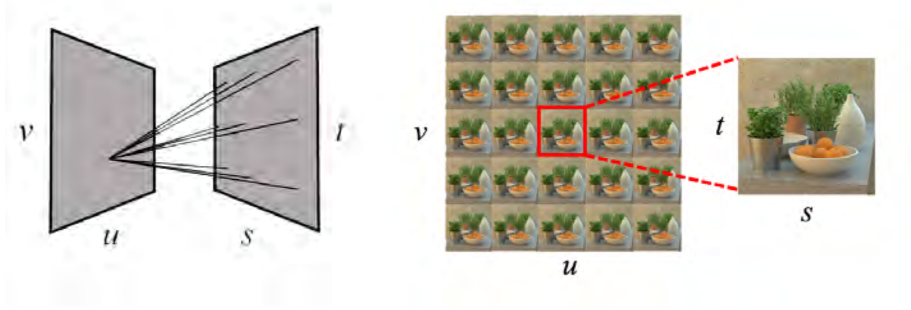
Chapter 2

Technical Background

2.1 The Plenoptic Function

The plenoptic function describes the distribution of light in a scene. It is a seven-dimensional function $P(x, y, z, \theta, \lambda, \phi, t)$ of spatial position x, y, z, the viewing direction given in spherical coordinates by θ, and ϕ, wavelength λ, and time t. Intuitively, the plenoptic function defines the amount of light traveling in any direction, though any point, at a particular instance of time. When dealing with static scenes the dependence on time t can be dropped. Moreover, the wavelengths of interest for most visual computing tasks are fixed at red, green and blue. This simplifies P to a five dimensional function $P(x, y, z, \theta, \phi)$.

Except in the simplest of cases, the plenoptic function does not have an analytic solution so that it can only be estimated by interpolating from discrete samples. These samples are commonly provided by multi-view images of a scene. Given the large dimensionality of P, the number of images required to reconstruct it can be quite high. Chai et al. [13] study the sampling requirements of the plenoptic function and show that the maximum baseline between a pair of neighboring images must be less than one pixel for alias-free interpolation of P. This limit is, however, inversely related to the amount of geometric information [13, 69]: the greater our knowledge of scene geometry, the lower the sampling requirement. In our experiments, we use the term *light field* to refer to a set of multi-view images that satisfies Chai et al.'s sampling requirements. Note that some work, especially in the computer graphics community, uses the term *light field* for the plenoptic function itself. We do not follow this convention.



(a) (b)

Figure 2.1: 4D parameterization of light fields. **(a)** The camera capturing the light field moves on the uv plane; st is the image plane. **(b)** Intuitively, the uv coordinates reference a light field view, and the st coordinates reference a pixel in that view.

2.2 Light Fields

A light field is a set of images that represents samples of the plenoptic function. If we assume a space free of occluders with views captured on a single plane from outside the convex hull of the scene [57], then a light field provides a four-dimensional representation $L(u, v, s, t)$ of the 5D plenoptic function P. This is achieved via a two-plane parameterization [57, 29, 40] (Figure 2.1). Since the camera capturing the light field is moving on a 2D plane – the uv plane – then any pixel in the light field can be referenced by four coordinates: the 2D position of the camera (u, v) on the plane, and the 2D pixel coordinates (s, t) of the pixel in the image captured from that position. The uv coordinates form the *angular* dimensions, whereas the st coordinates form the *spatial* dimensions of the light field. Hence, in all subsequent discussions, we use the descriptor “angular” to describe properties across different views, and “spatial” to refer to properties of pixels within a single image. A light field is *structured* if the uv samples lie on a rectilinear grid. All our experiments assume structured light fields, unless otherwise stated.

2.3 Epipolar Plane Images

An Epipolar Plane Image, or an EPI, represents an angular slice of a light field [10]. Specifically, an EPI $E(v, t)$ is the function obtained by fixing one angular and one spatial dimension of the light field L: $E(v, t) \equiv L(u^*, v, s^*, t)$ for constant u^* and v^*. In practice, this corresponds to the image created by stacking the s^*th row of pixels from the u^*th row of images. For Lambertian surfaces, EPIs provide a highly regular and structured representation of the scene: a point in world space becomes a line in an EPI, and the slope of this line corresponds, via disparity, to the point’s depth.

Chapter 3

Related Work

3.1 Light Field Depth Estimation

Many light field depth estimation methods seek to exploit the regular structure of an EPI [73]. Wanner and Goldluecke's [111] work was among the earliest widely-applicable method to use EPI lines for local depth estimates. They use a structure tensor to estimate depth as the gradient in EPI space. The estimate is refined in a variational framework — first in the angular dimension to enforce visibility constraints, and then in the spatial dimension. Many subsequent methods have adopted a similar approach by posing depth estimation as an energy-minimization problem in EPI space. However, the latter optimization does not include any cross-view consistency constraints. Thus, while capable of generating depth maps for off-center views, their results are not consistent. Moreover, the variational approach turns out to be computationally untenable when generating results for each view. Due to the local nature of the structure tensors, Wanner and Goldluecke's method can only reliably detect pixels with a disparity no larger than two pixels. This limits its application to light fields with larger baselines. Diebold and Goldluecke [24] address this by refocusing each EPI to several virtual depth layers before local disparity estimation. In addition, they present a method for handling incoherent depth estimates around occlusion boundaries. While this generates sharper edges in the resultant depth map, it requires the integration of estimates over all views and so restricts the output to a disparity map for the central view only. Zhang et al.'s [125] spinning parallelogram operator works in a similar fashion on EPIs, but has a larger support than the 3 × 3 Scharr filters used by Wanner and Goldluecke [111] and Diebold and Goldluecke [24], and provides more accurate estimates. This approach is similar to Tošić and Berkner's [104] convolution with a set of specially adapted kernels to create light field scale-depth spaces. Wang et al. [107] [108]

build on this by proposing a photo-consistency measure to address occlusion. Their method computes depth maps with sharp transitions at occlusion edges, but only produces depth for the central light field view. Tao et al.'s [102] work considers higher dimensional representations of EPIs which allows them to use both correspondence and defocus for depth estimation. These latter two works use the graph-cuts of Kolmogorov and Zabih [53] to minimize an NP-hard energy function.

The relation between defocus and depth is also exploited by the sub-pixel cost volume of Jeon et al.[44] who also present a method for dealing with the distortion induced by micro-lens arrays. An efficient and accurate method for wide-baseline light fields was proposed by Chuchwara et al. [20]. They use an oversegmentation of each view to get initial depth proposals, which are iteratively improved using PatchMatch [5]. Their work demonstrates the use of superpixels for higher level vision tasks. Closely related to our method of edge-aware bidirectional diffusion (Chapter 4.2) is the work of Holynski and Kopf [34] who present an efficient method for depth densification from a sparse set of points for augmented reality applications. Similar to our smooth reconstruction method (Chapter 6), Yucer et al. [122] present a diffusion-based method that uses image gradients to estimate a sparse label set. However, their method is designed to work only for light fields with thousands of views. Chen et al. [14] estimate accurate occlusion boundaries by using superpixels in the central view to regularize the depth estimation process. In general, densification methods [118, 109, 17] largely seek to recover accurate metric depth without considering occlusion boundaries.

In recent years, many methods have sought to use data-driven methods to learn priors to avoid the cost of dealing with a large number of images, and to overcome the loss of spatial information induced by the spatio-angular tradeoff in lenslet images. Alperovich et al. [3] use an encoder-decoder architecture to perform an intrinsic decomposition of a light field, and also recover disparity for the central cross-hair of views. Huang et al. [36] provide a network-based solution that is able to handle an arbitrary number of uncalibrated views. Jiang et al. [46, 47] fuse the disparity estimates at four corner views estimated using a deep learning optical-flow method. Shi et al. [93] build on this by adding a refinement network to the fusion pipeline.

3.2 Sparse Depth Estimation and Densification

Our work on depth reconstruction by diffusion (Chapters 6) begins with sparse depth estimates from our multi-view edge model. This relates to work on depth densification and completion. Early work in this field used cross-bilateral filters to complete missing depth samples [82]. Chen et al. learn to upsample low-resolution depth camera input and regularize it from paired RGB data [16]. Imran et al. consider the problem of depth pixels being interpolated across discontinuities, and compensate by learning inter-depth object mixing [39]. Efficient computation is also addressed by Holynski and

Kopf [34], who estimate disparity maps for augmented reality. With accurate depth samples, such as from LIDAR, simple image processing-based methods are competitive with more complex learning-based methods [55]. Our method considers the problem of when depth samples themselves may not be accurate, and any resulting densification without correcting the samples will lead to error.

3.3 Differentiable Rendering

The densification process described in Chapter 7 uses differentiable rendering to optimize sparse constraints through a differentiable diffusion process. Xu et al. [118] use differentiable diffusion based on convolutions for coarse depth refinement. We build upon radiative energy transport models that approximate transmittance through a continuously-differentiable isotropic Gaussian representation [81]. In this area, and related to layered depth images [92], recent work in differentiable rendering has addressed multi-plane transmittance for view synthesis [126, 59]. Other works consider transmittance in voxel-based representations [62] and for differentiable point cloud rendering [121, 22]. A known challenge with differentiable point clouds is backpropagating the 3D point locations through a differentiable renderer via a splatting algorithm [83]. Wiles et al. [117] present a neural point cloud renderer that allows gradients to be back propagated to a 3D point cloud for view synthesis. We propose a method to meet this challenge by directly rendering depth, and use it to optimize sparse depth samples to correctly reproject RGB samples across multi-view data.

3.4 Light Field Segmentation

Some computational photography applications requires the user to provide label annotations marking objects to be segmented. Wanner et al. [113] used a Markov random field (MRF) to assign per-pixel labels to the central view of a light field. Mihara et al. [68] extended Wanner's work by segmenting using an MRF-based graph-cut algorithm which produces labels for all views of the light field. Hog et al.'s work [31] improves the running time of a naive MRF graph-cut by bundling rays according to depth. Campbell et al. [11] presented a method without user input for automatic foreground-background segmentation in multi-view images. This uses color-based appearance models and silhouette coherence; as such, their method is more effective for larger baselines with larger change in object silhouettes. All these methods seek to calculate object-level labels; however, our goal requires automatically producing superpixel segmentations of light fields as piece-wise constant depth reconstructions (Chapter 5).

3.5 Light Field Superpixel Segmentation

Given the familiar 2D simple linear iterative clustering algorithm (SLIC) for superpixel segmentation [1], Hog et al. [32] propose an approach for light fields which is focused on speed, computing in 80s on CPU and 4s on GPU on the HCI dataset [112]. Then, the authors extend this work to handle video processing [33]. However, with their focus on fast processing, the results are not view consistent. Given a disparity map for the central view, Zhu et al. [127] pose the oversegmentation problem in a variational framework, and solve it efficiently using the Block Coordinate Descent algorithm. While their method generates compact superpixels, these sometimes flicker as shape changes across views. Our approach specifically enforces view consistency, which is desirable for many light field applications. Li et al. [60] generate light field superpixels by optimizing an energy on a 4D graph using a greedy approach. While their energy term explicitly enforces view consistency, it isn't a hard constraint. As a result, their output also lacks view consistency in some cases, with superpixels flickering or changing shape.

Chapter 4

A Multi-view Edge Model

This chapter presents our proposed multi-view edge model. Our model consists of 1) the 2D pixel location and depth values of edges across light field views — a pair of parameters we will refer to as a *multi-view edge* and differentiate from the full *multi-view edge model*, 2) a *surface vector* that indicates the direction of occlusion, and 3) a confidence value of the edge being a depth, rather than a texture, edge. It will be noticed that our model reflects Elder's single-view edge model [25] in all parameters but the information being encoded: in Elder's case this is photometric information, whereas in our case it is geometric information.

4.1 Multi-view Edges

A multi-view edge refers to the pairing of a 2D edge location with a depth label which allows the edge point to be uniquely identified and localized in multiple views. These two parameters correspond exactly to the edges in an Epipolar Plane Image (EPI). Thus, we propose an algorithm to jointly estimate them via line-detection in the EPI domain.

For robust occlusion handing, we must accurately detect the intersections of lines in EPIs (Fig. 4.1). However, classical edge detectors like Canny [12] and Compass [86] often generate curved or noisy responses at line intersections, which makes later line fitting and occlusion localization difficult. Instead, we propose an EPI-specific method. *Note:* We describe line detection for the central horizontal views; central vertical views follow similarly.

4.1.1 EPI Edge Detection

We take all EPIs $E_i(x, u)$ (size $w \times h$) from the horizontal central view images $I \in \mathcal{H}$. We convolve them with a set of 60 oriented Prewitt edge filters with each representing a particular disparity. We

Algorithm 1: EPI edge detection

```
FindEdgesEPI $(E, F)$
    Input: $E$: A $w \times h$ EPI
           $F$: A set of 60 $2h \times 2h$ directional filters.
    Output: An edge slope map $Z$ with confidences $C$.

    foreach $f_i \in F$ do
        $r_i \leftarrow E \circledast f_i$;
    end
    foreach pixel location$(u, v) \in I$ do
        $Z(u, v) \leftarrow \arg\max_i r_i(u, v)$ ;
        $C(u, v) \leftarrow \max_i r_i(u, v)$ ;
        $V(u, v) \leftarrow \mathrm{StdDev}(I(\mathcal{N}_{(u,v)}))$ for neighborhood $\mathcal{N}_{(u,v)}$ around $(u, v)$ ;
    end
    $C \leftarrow \mathrm{NonMaxSuppress}(C) \odot V$ ;
    return $Z$, $C$
end
```

filter only the central views for efficiency, and later on will propagate their edges across all light field views. To detect small occluded lines, we use $2h \times 2h$ filters and convolve the entire (x, u) space. This effectively extends occluded edge response to span the height of the EPI.

From this, we pick the filter with maximal response per pixel, which is a disparity map Z at edges, and we take the value of the filter response as an edge confidence map C. Then, we perform non-maximal suppression per EPI. To suppress false response in regions of uniform color, we modulate edge response by the standard deviation of a 3×3 window around each pixel in the original EPI [50]. As seen in Figure 4.1, our final C map has clean intersections and straight edges. Our approach is summarized in Algorithm 1.

4.1.2 Line Fitting

To create a parametric line set $\mathcal{L}$, we form lines l_i from each pixel in C in confidence order, with line slopes from Z. As we add lines, any pixels in C which lie within an λ-pixel perpendicular distance of the line l_i are discarded. λ determines the minimum feature size that our algorithm can detect. In all our experiments, we set $\lambda = 0.2h$. We proceed until we have considered all pixels in C. For efficiency, we detect edges and form line sets in a parallel computation per EPI.

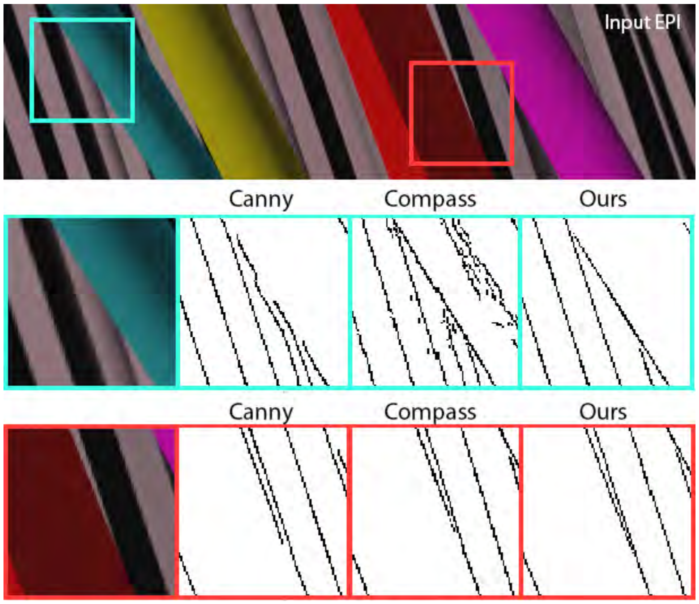


Figure 4.1: Our method can detect edge intersections more accurately than the Canny or Compass methods. These intersections provide valuable occlusion information.

4.1.3 Outlier Rejection

We wish to exploit information from across the spatio-angular light field. As such, we defer outlier rejection until *after* we have discovered $\mathcal{L}$ for each EPI in each horizontal view, and then project all discovered lines into the central view. Given this, we wish to keep both (a) high confidence lines, and (b) low confidence lines which have similar spatio-angular neighbors, and reject faint lines caused by noise.

Given a line $l_i \in \mathcal{L}$ with confidence c_i and disparity z_i, we count the number of lines within a $p \times q$ pixel spatial neighborhood $\mathcal{N}(l_i)$, and weight this number by the confidence c_i:

$$\mathcal{A}(l_i) = \{l_k \in \mathcal{N}(l_i) \mid z_k = z_i\}. \tag{4.1}$$

Then, we discard a line $l_i \in L$ if:

$$\frac{c_i|\mathcal{A}(l_i)|}{pq} < \tau, \tag{4.2}$$

where p and q are $1/15$th of the width and height of the light field, and $\tau = 8 \times 10^{-5}$. This is similar to Canny's use of a double threshold to robustly estimate strong edges: strong lines must have a confidence greater than τpq, and weak lines must have $\tau pq/c_i$ neighbors at the same disparity.

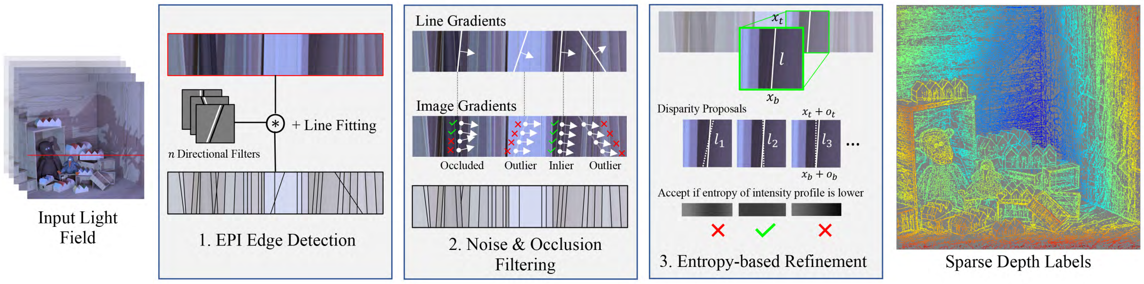


Figure 4.2: EPI edges provide both the location and disparity labels of a sparse point set $\mathcal{P}$. Thus, the first stage of multi-view edge estimation consists of EPI edge detection and line fitting. In the second stage, we compare the direction of each EPI line with underlying image gradients to remove noisy labels and points that are occluded in the central view. Finally, we improve the disparity estimates of the sparse set through an entropy-based random search.

4.1.4 Spatial Multi-scale Processing

To detect broader lines and improve consistency between neighboring EPIs, we compute coarse-to-fine edge confidence across a multi-scale pyramid with 2× scaling *in the spatial dimensions only*. At each scale and after the outlier removal processes, we double the x location of detected lines intercepts, and repeat each line twice along u. We replace any lines in a coarser scale which are close to lines in a finer scale. That is, we replace a coarse line only if both its end points are within λ pixels of the fine line. Thus, broader spatial lines not detected at a finer scale are still kept. With this we have a line set $\mathcal{L}$ for each EPI of the central horizontal and vertical views of our light field.

4.1.5 Line Refinement

We propose several additional refinement steps to the core edge detector described above. These provide greater accuracy, and a continuous depth range for the detected edge labels.

Noise & Occlusion Filtering

The above process, while fast, may fail to remove all false-positives. To filter these out we use a gradient-based alignment scheme: each line $l \in \mathcal{L}$ is sampled at n locations to generate the set of samples $S_l = \{(x_i, y_i)\}$. The line l is considered a false-positive if the local image gradient of I does not align with the line direction at a minimum k number of samples:

$$\sum_{\mathbf{s}\in S_l} \mathbb{1}\left(\frac{\nabla I(\mathbf{s})(\nabla l)^T}{\|\nabla I(\mathbf{s})\|\|\nabla l\|} > cos(\tau_f)\right) < k, \tag{4.3}$$

where $\mathbb{1}(\cdot)$ is the indicator function that counts the set of aligned samples, ∇I is the first-order image gradient approximated using a 3×3 Sobel filter, and ∇l is perpendicular to the line. The parameters

τ_f and k are constants with $\tau_f = \pi/13$ and $k = (\text{EPI height})/c$, with $1 \leq c \leq \text{EPI height}$. To determine the constant value c, we consider two factors: 1) the accuracy of EPI line fitting, and 2) the expected minimum number of views a point is visible in. In the case of perfect alignment between the line and EPI gradients, $c = 1$. This means that a line with even a single misaligned sample is rejected. However, if a point is occluded in some views, the corresponding EPI line will be hidden and misalignment of samples in those views is inevitable. If we set $c = 1$ we risk discarding such lines. We determine empirically that $c = 4$ provides good results across the synthetic and real world scenes, and across the narrow and wider baseline light fields that we evaluate on.

The parametric definition of EPI lines does not carry any visibility information for a point across light field views. We determine visibility $v(l)$ of a point $l \in \mathcal{L}$ in the central view as:

$$v(l) = \mathbb{1}\left(\frac{\nabla I(\mathbf{s}_c)(\nabla l)^T}{\|\nabla I(\mathbf{s}_c)\|\|\nabla l\|} > cos(\tau_v)\right), \tag{4.4}$$

where $\mathbf{s}_c$ is the EPI sample corresponding to the central view and $\tau_v = \pi/10$.

Entropy-based Disparity Refinement.

Notice that the number of discrete disparity values of points in $\mathcal{L}$ is bounded by the number of large Prewitt filters used for EPI line fitting. Computational efficiency considerations prevent this number from becoming too large. Moreover, numerical precision and sampling errors result in the granularity of depth estimates plateauing beyond a certain number of filters. Thus, to enable the calculation of sub-pixel disparity values we fine-tune the initial estimates through random search and filtering. Let $\mathcal{L}_c = \{l \in \mathcal{L} \mid v(l) = 1\}$. Then for each $l \in \mathcal{L}_c$ and image samples $S_l = \{(x_i, y_i)\}$ on the line we minimize the energy function defined by the entropy of normalized intensity values:

$$E(l) = \sum_{\mathbf{s} \in S_l} -P(I(\mathbf{s})) \log_2(P(I(\mathbf{s}))), \tag{4.5}$$

where $I(\mathbf{s})$ is the intensity value at $\mathbf{s}$ and $P(\mathbf{s})$ is estimated from a histogram.

We minimize $E(l)$ by performing a random search in the 2D parameter space defined by the x-intercepts of l on the top and bottom edge of the EPI, $l = (x_t, x_b)$: at the jth iteration of the search we generate uniform random numbers $(o_t, o_b) \sim U(-1,1)(\alpha t^j)$, to generate a proposal $l_j = (x_t + o_t, x_b + o_b)$ (Fig. 4.2). This is accepted with probability one if $E(l_j) < E(l_{j-1})$. We use $t = 0.88$, $\alpha = 0.15$ and run the search for a maximum of 10 iterations.

The resulting disparity estimates are then refined by joint filtering in the spatial, disparity, and LAB color space. Let $\mathcal{P}$ represent the spatial projection of $\mathcal{L}_c$ into the central view, and let p_s, p_d, and p_c be the spatial position, disparity, and color of a point $p \in \mathcal{P}$. The filtered disparity estimate

$f(p_d)$ is calculated via a spatial neighborhood $\mathcal{S}$ around p:

$$f(p_d) = \frac{1}{W} \sum_{q \in \mathcal{S}} \mathcal{N}_{\sigma_s}(\|p_s - q_s\|) \mathcal{N}_{\sigma_d}(p_d - q_d) \mathcal{N}_{\sigma_c}(\|p_c - q_c\|) p_d,$$

where the normalization factor W is given by

$$W = \sum_{q \in \mathcal{S}} \mathcal{N}_{\sigma_s}(\|p_s - q_s\|) \mathcal{N}_{\sigma_d}(p_d - q_d) \mathcal{N}_{\sigma_c}(\|p_c - q_c\|). \tag{4.6}$$

In theory, the parameters $\sigma_c =$ and $\sigma_d =$ depend on the scene content and the maximum disparity. In practice, however, the maximum disparity is usually bounded and the color gradient characteristics of most real-world scenes are fairly uniform. Thus, we found that the combination $\sigma_s = 10$, $\sigma_d = 0.1$ and $\sigma_c = 0.5$ works for all scenes in our experiments.

4.2 Surface Vectors

At each depth edge, a *surface vector* indicates the direction of the occluding surface. This information is required since multi-view edges by themselves do not encode sufficient information to uniquely reconstruct a scene. As shown in Figure 4.3(a), two different scene configurations may generate a similar EPI and, thus, similar multi-view edge parameters. The problem is exacerbated by the fact that in practice we don't even know the location of the discontinuity — that is, the edge — precisely in pixels when using Prewitt filters: Figure 4.3(b) shows that a Prewitt filter convolved with a set of pixels representing an edge will generate a non-zero activation along two columns of pixels. Using a model without surface vector parameters results in significant errors around edges when using the smooth reconstruction method of Chapter 6 (supplemental Fig. B.1).

However, retrieving the direction of occlusion as the surface vector is a difficult task: determining it requires us to know the depth at pixels around each label, but we only have a sparse set of labeled points at edges. Holynski and Kopf [34] deal with this by assuming that sparse labels do not lie on depth edges so that neighboring pixels have a similar label. Yucer et al. [122] handle labels on depth edges, but their method is designed for light fields with a large number (≈3000+) of views. Our novel contribution is that we determine surface direction from other sparse labels within context. We present two methods for doing this. The first poses the problem as an EPI segmentation problem and generates a surface vector upto a 90 degree accuracy (that is, left/right or up/down). The second method uses a bidirectional 'backward-forward' diffusion process to generate a surface vector parallel to the image gradient.

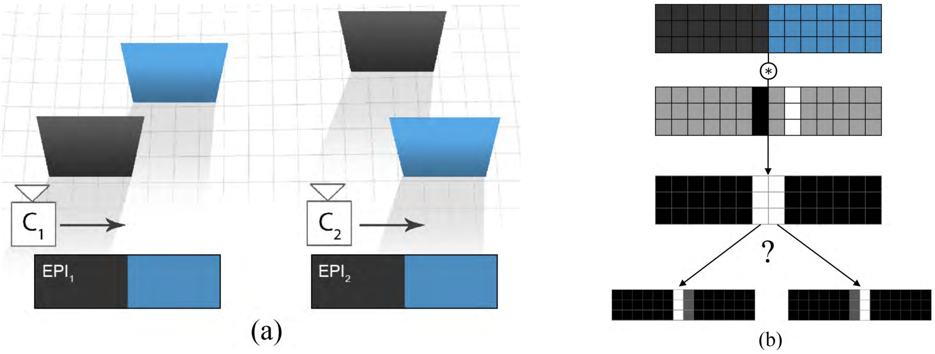
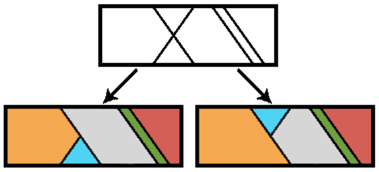


Figure 4.3: Sparse labels at edges are difficult to propagate because the edge is weakly localized at the boundary of two projected surfaces. As a result, labels may be assigned to the incorrect side of a depth boundary. **(a)** Two different scene configurations captured with cameras C_1 and C_2 may generate similar EPIs. The EPI edge represents the boundary of the occluding surface. For C_1 this is the surface on the left (black); for C_2 it is on the right (blue). **(b)** The direction from which occlusion happens cannot be disambiguated from edge activations alone, leading to incorrect label placement.

4.2.1 Occlusion-aware EPI Segmentation



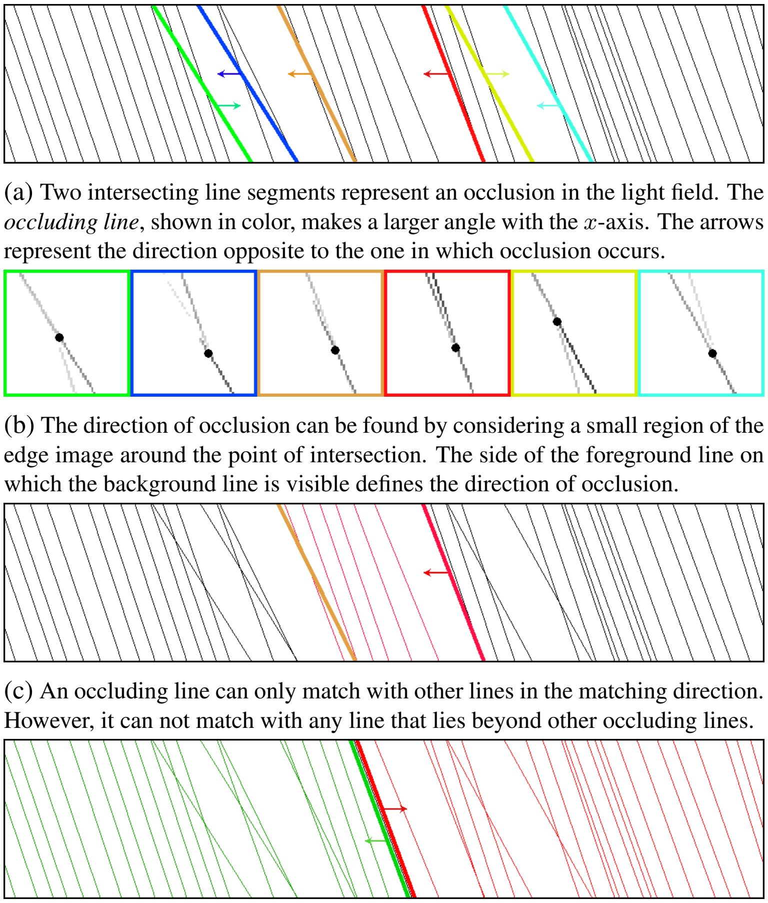
Given a set of lines $\mathcal{L}$, we wish to match lines into pairs to define an EPI segmentation. One simple approach is to match every line twice: once each to its left and right neighbors. However, as per the inset diagram, this fails when lines intersect at occlusions as it produces an under-constrained problem in which segment order cannot be uniquely determined.

We solve it by considering a small region around the point of intersection in the edge image E, which allows us to constrain the occlusion direction and determine the correct matching (Fig. 4.4b). The occlusion direction is given by the side of the foreground line in which the background line is visible. The foreground line is determined by the relative slope of the two lines.

The sequence of steps to narrow down the potential matches for each line is shown in Figure 4.4. Once we have omitted line pairings which violate the occlusion order, we pose line matching as a two-step maximum value bipartite matching problem on a complete bipartite graph $G(\mathcal{L}, \mathcal{L}, E)$ and solve it using Dulmage-Mendelsohn decomposition. First, we match only intersecting lines to resolve occlusions. Then all remaining lines are matched. We compute line distance as:

$$\text{Distance}(l_i, l_k) = (\omega_d|t_i - t_k - b_i + b_k| + (1 - \omega_d)|t_i + b_i - t_k - b_k|)^{-1}, \tag{4.7}$$

where t_i and b_i are the line intercepts l_i at the top and bottom of the EPI image. ω_d is a constant which determines the relative importance of disparity similarity over spatial proximity of lines.



(a) Two intersecting line segments represent an occlusion in the light field. The *occluding line*, shown in color, makes a larger angle with the x-axis. The arrows represent the direction opposite to the one in which occlusion occurs.

(b) The direction of occlusion can be found by considering a small region of the edge image around the point of intersection. The side of the foreground line on which the background line is visible defines the direction of occlusion.

(c) An occluding line can only match with other lines in the matching direction. However, it can not match with any line that lies beyond other occluding lines.

(d) A background line can match twice: once each to its left and right.

Figure 4.4: Illustrating the rules which govern line coupling.

Finally, to prevent forming large superpixels in uniform regions, we recursively split any segment that has a width larger than 15 pixels by adding new lines. To regularize segments across the vertical and horizontal EPI directions—especially in textureless regions—the slope of new lines is always set to match the disparity of the vertical segment covering that spatial region.

The full EPI segmentation procedure is described in Algorithm 2. Figure 4.5 shows an example EPI result after the computations of Sections 4.1 and 4.2.1.

4.2.2 Bidirectional Diffusion

As all potential occlusion edges are also depth edges, one way to determine occluding surface, or diffusion, direction is by distinguishing depth and texture edges. Yucer et al. [122] do this by comparing the variation in texture on both sides of an edge as the view changes: the background

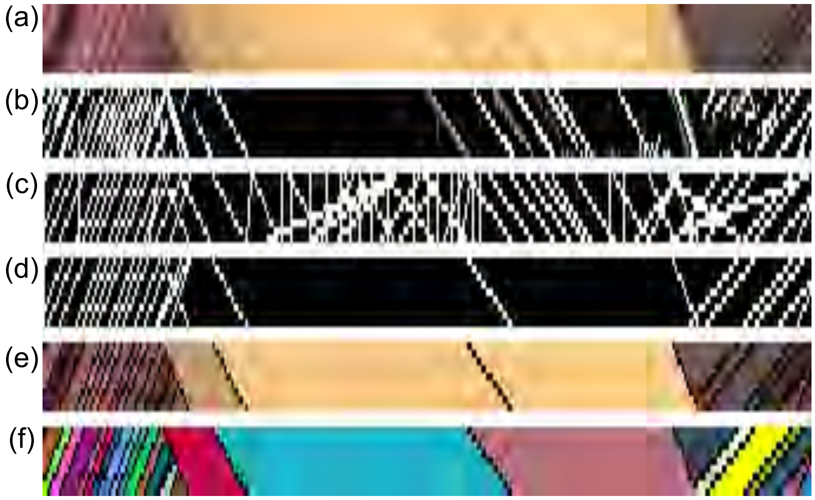


Figure 4.5: From top to bottom: (a) Input EPI. (b) Edge confidence map C from Sec. 4.1. (c) Parametric lines $\mathcal{L}$ fit to C. As we do not threshold, faint edge lines are visible along with some outliers. (d) Outliers are robustly removed via spatial neighbor statistics, depth, and edge confidence weights. *Note:* remaining overlapping lines represent occlusions. (e) & (f) Line pairs are matched in an occlusion-aware manner to form angular segments. *Note:* The EPIs have been stretched vertically for viewing; input is 9 views.

seen around a depth edge will change more rapidly than the foreground, leading to a larger variation in texture along one side of the edge. The correct diffusion direction is to the side with lower variation. This method works for light fields with thousands of views (3000+ images), but proves ineffective on datasets that are captured using a lenslet array or camera rig (Fig. 6.3). This is because the assumption fails to hold in cases where 1) the background lacks texture, and 2) the light field has a small baseline with relatively few views, as is common for handheld cameras. Here, occlusion is minimal and intensity variation is caused more by sensor noise than by background texture variation.

Our proposed solution to the depth edge identification problem works for light fields with few views (e.g., 7×7 from a Lytro). We use $S[\mathcal{A}]$ to represent the image created by splatting sparse points in a set $\mathcal{A}$ onto a $w \times h$ raster grid, and D to be a dense $w \times h$ disparity map. Diffusion is formulated as a constrained quadratic optimization problem:

$$\hat{D}[\mathcal{A}] = \underset{D}{\operatorname{argmin}} \sum_{p \in \mathcal{A}} E_d(p) + \sum_{(p,q) \in \mathcal{S}} E_s(p,q), \tag{4.8}$$

where $\hat{D}[\mathcal{A}]$ is the optimal disparity map given the sparsely labeled image $S[\mathcal{A}]$ and $\mathcal{S}$ is the set of all four-connected neighbors in D. The data term $E_d(p)$ and smoothness term $E_s(p,q)$ are:

$$E_d(p) = \lambda_d(p) \big\| S[\mathcal{A}](p) - D(p) \big\|, \quad \text{and} \quad E_s(p,q) = \lambda_s(p) \big\| D(p) - D(q) \big\|, \tag{4.9}$$

with $\lambda_d(\cdot)$ and $\lambda_s(\cdot)$ being the spatially-varying data and smoothness weights.

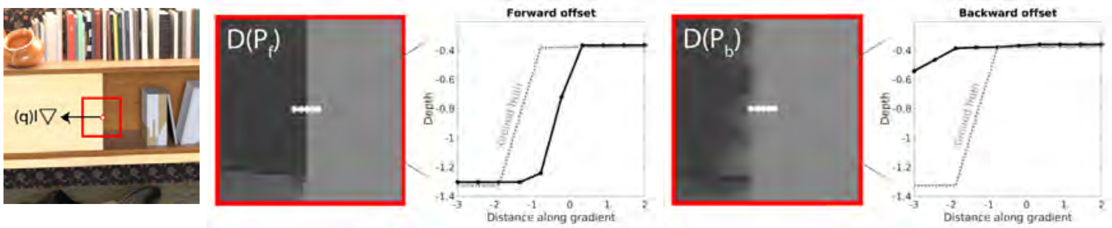


Figure 4.6: **(a)** Given an edge point p with image gradient $\nabla I(p)$ and depth label p_d we would like to determine which side of the edge to propagate p_d. We generate images $\hat{D}[\mathcal{P}_f]$ (**b**), and $\hat{D}[\mathcal{P}_b]$ (**c**) by solving a Poisson optimization problem with diffusion direction $p + \nabla I(p)$ and $p - \nabla I(p)$ respectively. The correct diffusion direction (**b**) generates an intensity profile resembling a step function. In the example shown, p_d corresponds to the surface on the right of the edge as $p + \nabla I(p)$ generates a profile more closely resembling a step function.

Equation (4.8) represents a standard Poisson problem, and we solve it using an implementation of the LAHBPCG solver [100] by posing the constraints in the gradient domain [8]. We begin by defining two sets formed from opposite offset directions $\nabla I(p)$ and $-\nabla I(p)$:

$$\mathcal{P}_f = \{p + \nabla I(p)\ \forall\, p \in \mathcal{P}\}, \quad \text{and} \quad \mathcal{P}_b = \{p - \nabla I(p)\ \forall\, p \in \mathcal{P}\}, \tag{4.10}$$

where $\nabla I(p)$ is the gradient of the central light field view at point p. Then, we solve Equation (4.8) for both offset directions $\hat{D}[\mathcal{P}_f]$ and $\hat{D}[\mathcal{P}_b]$ using data and smoothness weights:

$$\lambda_d(p) = \begin{cases} 10^6 & \text{if } p \in \mathcal{A}, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and} \quad \lambda_s(p) = \frac{1}{\|\nabla I(p)\| + \epsilon}. \tag{4.11}$$

Given both solutions, we compare the normalized depth profile around each point $p \in P$ along $\nabla I(p)$ in $\hat{D}[\mathcal{P}_f]$ and $\hat{D}[\mathcal{P}_b]$. Figure 4.6 shows that the profile for the correct offset direction ($\nabla I(p)$ or $-\nabla I(p)$) more closely resembles a step function around p due to a strong depth gradient. This is because neighboring points in the correct offset direction will have a disparity value similar to p. The high data term together with the global smoothness constraint results in a small gradient around p when the incorrect offset pushes it to the wrong side of the edge. We estimate the profile around p in $\hat{D}[\mathcal{P}_f]$ and $\hat{D}[\mathcal{P}_b]$ by convolving the normalized value of a set N_p of pixels around p with the step filter $\mathbf{F} = [-1\ -1\ 1\ 1]$.

4.3 Depth Edge Confidence

The bi-directional diffusion process descried above also allows us to identify the final parameter of our multi-view edge model, depth edge confidence. This is given by the mean gradient at each pixel across the backward-forward pass (Figure 4.7). Texture edge gradient remains low in both passes.

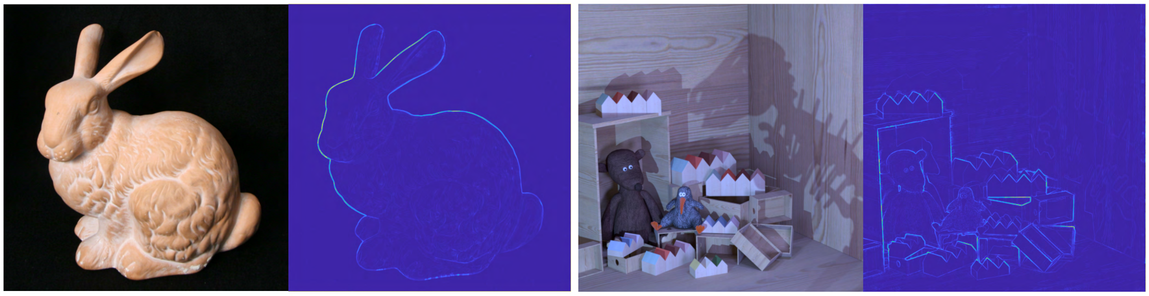


Figure 4.7: Estimated depth edge confidence λ_s.

For depth edges, the gradient is higher in one pass. For depth edges that are not meant to be sharp, the change in depth around that region from the bi-directional solve is small, and picking either offset leads to low error.

Algorithm 2: EPI line segment matching.

```
SegmentEPI ($\mathcal{L}$)
    Input:  $\mathcal{L}$: An ordered list of line segments bounded
            by the top and bottom edges of EPI $I$.
    Output: A set $M \in \mathcal{L} \times \mathcal{L}$ of line couplings.

    Create the complete bipartite graph $G = (\mathcal{L}, \mathcal{L}, E_f)$ for matching all occluding lines ;
    $S \leftarrow \mathrm{OccludingLines}(\mathcal{L})$ ;
    foreach $e = (l_i, l_j) \in E_f$ do
        if $l_i \notin S$ and $l_j \notin S$ then
            $w(e) \leftarrow -\infty$ ;
        else if $l_j$ does not lie to the left of $l_i$ then
            $w(e) \leftarrow -\infty$ ;
        else if $\exists k \in S$ to the left of $l_i$ |
        $\mathrm{Distance}(l_i, k) < \mathrm{Distance}(l_i, l_j)$ then
            $w(e) \leftarrow -\infty$ ;
        else
            $w(e) \leftarrow \mathrm{Distance}(l_i, l_j)$ ;
        end
    end
    $A \leftarrow \mathrm{MaxBipartiteMatching}(G)$ ;
    $U \leftarrow \{l \in \mathcal{L} \mid (\exists k)[k \in \mathcal{L} \wedge (l, k) \in A]\}$ ;
    $V \leftarrow \{k \in \mathcal{L} \mid (\exists l)[l \in \mathcal{L} \wedge (l, k) \in A]\}$ ;
    Create the complete bipartite graph $H = (\mathcal{L} \setminus U, \mathcal{L} \setminus V, E)$ for matching all other lines;
    foreach $e = (l_j, l_k) \in E$ do
        if $l_k$ does not lie to the left of $l_j$ then
            $w(e) \leftarrow -\infty$ ;
        else
            $w(e) \leftarrow \mathrm{Distance}(l_j, l_k)$
        end
    end
    $B \leftarrow \mathrm{MaxBipartiteMatching}(H)$ ;
    return $A \cup B$
end
```

Chapter 5

Piece-wise Constant Reconstruction via View-consistent 4D Superpixels

This chapter presents the first of our two reconstruction methods that allow us to invert our multi-view edge encoding. This method shows that the view-consistency and occlusion edge accuracy criteria for completeness (Section 1.3) can be satisfied by using only a subset of our model parameters, and so the reconstruction method does not utilize occlusion edge confidence. We observe that a piece-wise constant depth reconstruction suffices to satisfy the two criteria. Thus, we pose the reconstruction task as a view-consistent light field superpixel segmentation problem.

Superpixel segmentation attempts to simplify a 2D image into small regions. Desirable superpixel qualities vary between applications [98], but generally we wish for them to be *accurate*, *i.e.*., to adhere to image edges; to otherwise be *compact* in shape, and to be *efficient* to compute [97].

In addition to verifying the completeness of our model with respect to view consistency and occlusion accuracy, light field superpixels have broader utility for computer vision tasks. Processing light fields is computationally hard due to the large number of pixels, but many of these pixels are similar because the view change is small. As such, we have much to gain from simplifying light field images into superpixels. We wish the superpixels to be view consistent, *e.g.*, they do not drift, swim, or flicker as the view changes, and we wish superpixels to include all similar pixels across views such that they respect occlusions. This is particularly important for applications which use every light field view, such as editing a light field photograph for output to a light field display [103].

It is difficult to achieve the four properties of accuracy, compactness, efficiency, and view consistency. Existing approaches often propagate superpixel labels into other views via a central-view disparity map. However, this can cause inconsistency for regions occluded in the central view, *e.g.*, the light field superpixel (LFSP) method [127] does not always maintain view consistency. Per-view

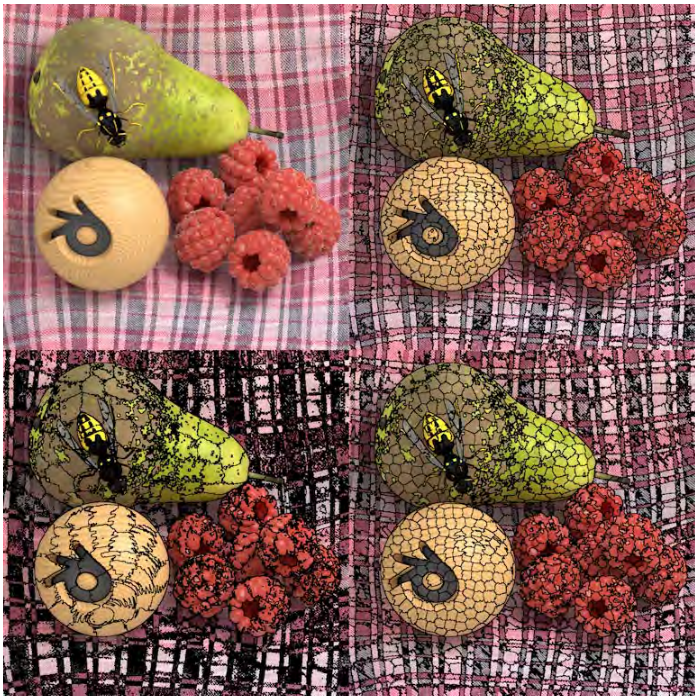


Figure 5.1: Light field superpixel comparison for the central view. *Top left:* Input scene. *Top right:* Our method. *Bottom left:* k-means on (x, y, d, L^*, a^*, b^*) with disparity maps computed by Wang et al. [107, 108]. *Bottom right:* LFSP [127] computed with Wang et al. disparity map.

disparity maps may help, but this can be difficult for small occluded regions in non-central views.

We propose a method for accurate and view-consistent superpixel segmentation on 4D light fields which implicitly computes disparity per view and explicitly handles occlusion (Fig. 5.2). First, we robustly segment horizontal and vertical epipolar plane images (EPIs) of the 4D light field. This provides view consistency in an occlusion-aware way by explicit line estimation, depth ordering, and bipartite graph matching. Then, we combine the angular segmentations in horizontal and vertical EPIs via a view-consistent clustering step. Qualitative results (Fig. 5.1) show that this reduces flickering from inconsistent boundary shapes when compared to the state-of-the-art LFSP approach [127], and quantitative metrics reflect these findings.

5.1 View-consistent Superpixel Segmentation

Given a 4D light field $LF(x, y, u, v)$, we define the central horizontal row of views $\mathcal{H} = LF(x, y, u, v_c)$ and central vertical column of views $\mathcal{V} = LF(x, y, u_c, v)$. Each view $I \in \mathcal{H}$ contains a set of EPIs $E_i(x, u) = I(x, y_i, u)$, with corresponding $I \in \mathcal{V}$ containing $E_j(y, v) = I(x_j, y, v)$.

With a Lambertian reflectance assumption, a 3D scene point corresponds to a straight line l in

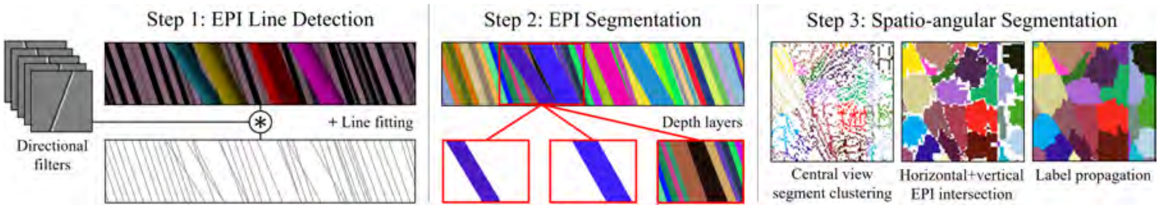

Step 1: EPI Line Detection
Directional filters
+ Line fitting
Step 2: EPI Segmentation
Depth layers
Step 3: Spatio-angular Segmentation
Central view segment clustering
Horizontal+vertical EPI intersection
Label propagation

Figure 5.2: Overview of our algorithm. *Step 1:* We find lines within EPIs extracted from the central horizontal and vertical views of a 4D light field. *Step 2:* Then, we use an occlusion-aware bipartite patching to pair these lines into regions with explicit depth ordering. *Step 3:* We cluster these segments and propagate labels into a view-consistent superpixel segmentation.

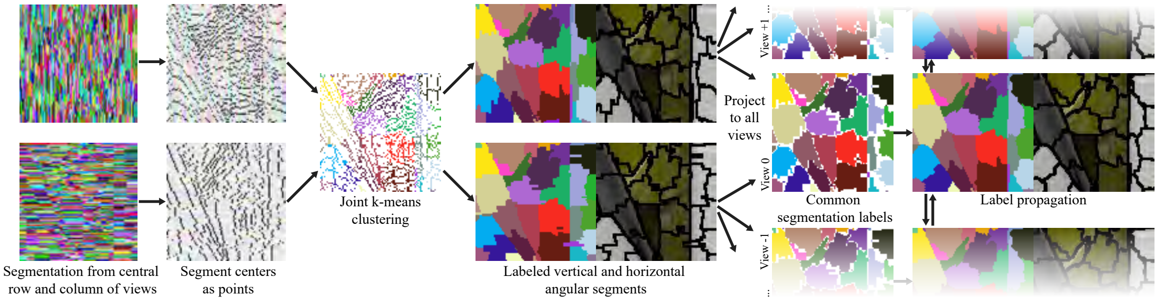

Segmentation from central row and column of views
Segment centers as points
Joint k-means clustering
Labeled vertical and horizontal angular segments
View +1 ...
Project to all views
View 0
View -1
...
Common segmentation labels
Label propagation

Figure 5.3: Vertical and horizontal view-consistent segments are clustered in the central light field view to obtain spatio-angularly consistent labels. Pixel labels which are not consistent across the vertical and horizontal segmentations are recalculated in the label propagation step.

an EPI, where the depth of the point determines the slope of the line. By extension, a region of neighboring 3D surface points with similar depth and visual appearance is topologically bound in each EPI by a set $\hat{\mathcal{L}}$ of two lines (l^1, l^2) on the boundary of $\mathcal{R}$. Either one or both of l^1 and l^2 may be occluded in any particular E_i. Our goal is to identify the boundaries $\hat{\mathcal{L}} = \{(l^1, l^2)\}$ for all visible regions $\{\mathcal{R}\}$ across all EPIs in an accurate, occlusion-aware, and spatio-angularly-consistent way and as efficiently as possible.

Our superpixel algorithm merges the multi-view edges and the surface vectors represented as occlusion-aware EPI regions (Section 4.2.1) into a consistent segmentation via a segment clustering, which uses the estimated disparity of the edges to regularize the process. Remaining unlabeled off-central-directions occluded pixels are labeled via a simple propagation step.

5.1.1 Spatio-angular Segmentation via Clustering

We combine the occlusion-aware segmentation per EPI of Section 4.2.1 across different EPIs to get correspondence between the horizontal and vertical EPI segments. We achieve this by clustering the segments in the central view of the light field using k-means in (x, y, d, L^*, a^*, b^*) space (Fig. 5.3).

This clustering approach with disparity d might seems similar to methods which exploit a central depth map for propagation, like LFSP [127]. However, our method is view consistent: our EPI segment-based computation allows us to estimate d for every light field view, including those segments occluded from the central view. These are all considered within the clustering.

For each segment, we compute the average pixel value in the CIELAB color space: L^*, a^*, b^*. We define the disparity d from the larger (deeper) slope of the two segment lines. For segments in horizontal EPIs, y equals the EPI index and we determine x to be the midpoint of the segment lines in the central view. For vertical EPIs, we reverse this relation. The number of clusters is user specified and determines superpixel size. We seed clusters at uniformly-distributed spatial locations [1], and assign x, y, d, L^*, a^*, b^* from the segment center closest in image space.

Within the feature vector, x, y have weight 1 and L^*, a^*, b^* have weight 3. We normalize d given our current scene estimates then weight it by 120. This larger weight helps the method not to cluster across occlusions, which usually have different disparities.

Clustering within the central view allows us to correspond and jointly label the horizontal and vertical EPI segments, and to provide spatial coherence. However, the boundaries from these two EPI segmentations do not always align. Thus, after projecting these segments into *all* light field views, we discard labels for pixels where the two segmentations disagree.

Label Propagation

At this point, our only unlabeled pixels are those either occluded from or in disagreement between both central sets of views in the vertical and horizontal directions. We note that 1) the set U of unlabeled pixels is sparse even within a local neighborhood; and that 2) at this stage, we know the disparity of each labeled pixel in the light field. As such, we minimize a cost with color, spatial, and disparity terms to label the remaining pixels.

Given an unlabeled pixel $(x, y) \in U$ in light field view $I_{u,v}$, let $\hat{\mathcal{L}}(x, y)$ define the set of labeled pixels in a spatial neighborhood around (x, y). For every pixel $(p, q) \in \hat{\mathcal{L}}(x, y)$, let $\ell(p, q)$ denote its label, and $d(p, q)$ its disparity. Moreover, let $I_{s,r}(\cdot, \cdot)$ represent the color of any pixel, labeled or unlabeled, in light field view $I_{s,r}$. We define the cost of assigning (x, y) label $\ell(p, q)$ as:

$$\begin{aligned} E_{(x,y)}(\ell(p,q)) &= \omega_c \left(I_{u,v}(x,y) - I_{u,v}(p,q)\right)^2 + \omega_s \left((x-p)^2 + (y-q)^2\right) \\ &+ \omega_d \left(\sum_s \sum_r I_{u,v}(x,y) - I_{s,r}(x + d(p,q), y + d(p,q)) \right). \end{aligned} \tag{5.1}$$

We set weights empirically: $w_c = 1$, $w_s = 1$, $w_d = 1e^{-5}$.

Label assignment total cost is $E = \sum_{(x,y) \in U} E_{(x,y)}$. We efficiently compute this by minimizing $E_{(x,y)}$ per pixel. Along with finding $\ell(x, y)$, we set $d(x, y)$ equal to $d(\text{argmin} E_{(x,y)})$, which allows us to project newly-assigned labels to any unlabeled pixels in other views. In practice, this strategy only requires minimization over the central row and column of light field views, with the few remaining pixels in off-center views after projection labeled by nearest neighbor assignment.

5.2 Experiments

5.2.1 Datasets

We use synthetic light fields with both ground truth disparity maps and semantic segmentation maps. From the HCI Light Field Benchmark Dataset [112], we use the four scenes with ground truth: *papillon*, *buddha*, *horses*, and *still life*. Each light field image has 9×9 views of 768×768 pixels, except *horses* with 1024×576 pixels. For real-world scenes, we use the EPFL MMSPG Light-Field Image Dataset [80]. These images were captured with a Lytro Illum camera (15×15 at 434×625). Please refer to our supplementary materials for more results.

5.2.2 Baselines

We compare to the state-of-the-art LFSP (light field superpixel segmentation) approach of Zhu et al. [127]. This method takes as input a disparity map for the central light field view. We apply their method on the disparity estimates from Wang et al. [107, 108] as originally used in the Zhu et al. paper, and on ground truth disparity. Comparing these two results shows the errors which are introduced from inaccurate disparity estimation.

We also compute a k-means clustering baseline, which is similar in spirit to RGBD superpixel methods like DASP [115] methods. Given a disparity map for the central light field view, we convert the input images to CIELAB color space and form a vector $f = (x, y, d, L^*, a^*, b^*)$ for each pixel in the central view of the light field. Then, from uniformly-distributed seed locations, we cluster using the desired number of output superpixels, and project these labels into other views. For any pixels in non-central views which remain unlabelled, we assign the label of the nearest neighbor based on $f = (x, y, L^*, a^*, b^*)$. We use the same weight for each feature as in our method. For LFSP, we compute results with ground truth disparity, and also with the method of Wang et al. [107, 108].

5.2.3 Metrics

We use two view-consistency-specific metrics: self similarity error [127] and number of labels per pixel; explained below. We also use three familiar 2D boundary metrics: achievable accuracy, boundary recall, and undersegmentation error. Achievable accuracy, self similarity, and number of labels per pixel describe overall accuracy and consistency across views. Boundary recall and undersegmentation error describe characteristics of over segmentation [72]. As a measure of superpixel shape, we use the compactness metric from Schick et al. [88]. We compute each metric across average superpixel sizes of 15–40 square (225–1600 pixels each).

Achievable Segmentation Accuracy Given a ground-truth object-level segmentation map, this metric measures the achievable accuracy possible by the oversegmentation, *e.g.*, in a later interactive object selection stage. To compute the metric, we assign labels to superpixels according to the ground truth object labels from the synthetic HCI dataset [112]. The label which maximizes overlap with a superpixel across views is the assigned label.

Boundary Recall Given a ground truth boundary image G and an algorithm's boundary image B, we compute the fraction R of ground truth edges that fall within a certain distance d of at least one superpixel boundary [72]. We use $d = 2$ chessboard distance. True Positives (TP) is the number of boundary pixels in G for whose exist a boundary pixel in B in range d; False Negative (FN) is the number of boundary pixels which do not fall within this range. Boundary recall is:

$$R = \frac{TP}{TP + FN}. \tag{5.2}$$

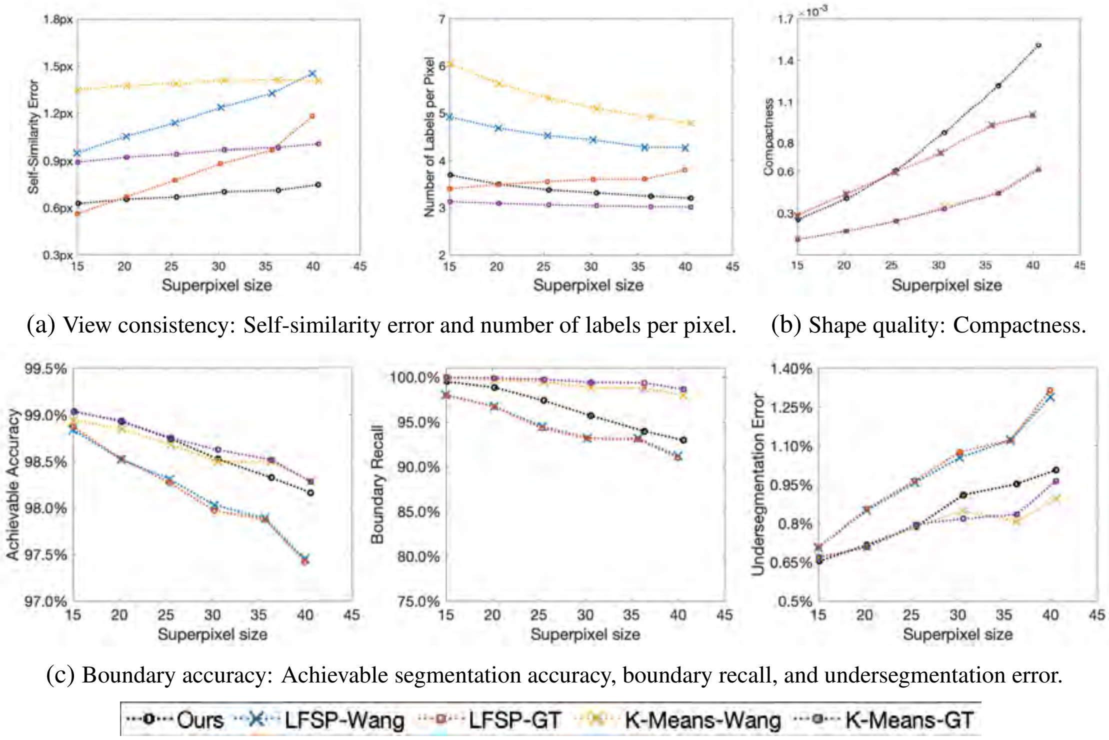
Intuitively, the higher the boundary recall, the better the superpixels adhere to object boundaries. However, using this alone favors long segment boundaries. Thus, we plot the metric with different superpixels sizes and use it with the undersegmentation error for more considered evaluation.

Undersegmentation Error A segment S in the ground truth segmentation image G divides a superpixel P into an *in* and an *out* part. The undersegmentation error compares segment areas and provides the percentage of superpixels which overlap ground-truth segment borders. Various implementations of the undersegmentation error metric exist; we adopt the formulation of Neubert et al. [72] which does not penalize large superpixels with a small overlap with a ground truth segment:

$$UE = \frac{1}{N_S} \sum_{S \in G} \frac{\sum_{P:P \cap S \neq \varnothing} min(|P_{in}|, |P_{out}|)}{|S|}. \tag{5.3}$$

The inner sum is the error introduced by this specific combination of ground truth segment and superpixel, depending on their overlap.

Compactness The compactness metric provides a measure of superpixel boundary curvature. We use Schick et al.'s [88] definition of compactness as:



(a) View consistency: Self-similarity error and number of labels per pixel.

(b) Shape quality: Compactness.

(c) Boundary accuracy: Achievable segmentation accuracy, boundary recall, and undersegmentation error.

Figure 5.4: Quantitative evaluation metrics for light field oversegmentation.

$$C(\mathcal{S}) = \sum_{S \in \mathcal{S}} \frac{4\pi A_S |S|}{|I| L_S^2}, \tag{5.4}$$

where $\mathcal{S}$ is the set of superpixels, $|I|$ is the size of a single light field view, and A_S and L_S are the area and perimeter of superpixel S, respectively. We compute the median superpixel compactness per light field for a fixed superpixel size of 20, then average across the HCI dataset.

Self Similarity Error As defined in Zhu et al. [127], we project the center of superpixels from each view into the center view, and compute the average deviation versus ground truth disparity. Smaller errors indicate better consistency across views.

Number of Labels Per View-dependent Pixel We compute the mean number of labels per pixel in the central view as projected into all other views via the ground truth disparity map. This gives a sense of the number of inconsistent views on average (cf. HCI dataset with 81 input views). For ease of computation, we discard pixels which are occluded in the central view.

5.2.4 Results

Figure 5.4 shows all metrics averaged over all four scenes; our supplementary material includes per-scene metrics. For qualitative results, please see our supplemental video.

View Consistency Our method outperforms both LFSP and the k-means baselines using estimated disparity maps (Fig. 5.4a). These findings are reflected in qualitative evaluation where we reduce view inconsistencies such as flickering from superpixel shape change over views (Fig. 5.5). Using ground truth disparity maps, our method outperforms LFPS on both metrics, but only outperforms k-means on *self similarity error*: k-means with ground truth disparity produces fewer *numbers of labels per pixel* than our method. As a reference for interpretation, the small baselines cause occlusion in $\sim$3–5% of light field pixels.

Figure 5.9 shows the average number of labels per pixel metric as a heatmap for the central view, where blue is low (good view consistency) and red is high (bad view consistency). We see that most errors occur at edges; that LFSP has more inconsistency around edges; and that k-means is sometimes sensitive to high-frequency pattern textures.

While this could be alleviated with clustering feature weight parameter tuning, we note that our method does not suffer this issue even though it uses the same component weight parameters as our k-means baseline. This is because we cluster on the EPI segments of Section 4.2.1, rather than individual pixels. Our method, effectively, performs a per-scanline segmentation before clustering.

Achievable Accuracy, Boundary Recall, and Undersegmentation Error Our method outperforms LFSP for all three metrics on both estimated and ground truth disparity for all superpixel sizes (Fig. 5.4c). For smaller superpixel sizes (15–25), we are competitive in accuracy and undersegmentation error with k-means using ground truth disparity; at larger sizes k-means is better. Our method recalls fewer boundaries than k-means: we occasionally miss an edge section during step 1, which defers these regions to our less robust final propagation step for unlabeled pixels instead. However, k-means can create very small regions (Fig. 5.5) which are broadly undesirable.

Figure 5.8 shows a visualization of achievable accuracy via the ground truth semantic-level segmentation maps provided in the HCI dataset. Red is where a superpixel crosses a ground-truth boundary. Generally, our approach performs better than LFSP and comparably to k-means.

Compactness Our method is competitive with LFSP at smaller superpixel sizes (15–25), and better at larger sizes (Fig. 5.4b). The k-means baseline generates the least compact superpixels, even with ground truth disparity. This shape freedom, however, helps it recall more boundaries.

Computation Time We use an Intel i7-5930 6-core CPU and MATLAB for our implementation. We report times on the 9 × 9 view light fields with images of 768 × 768 pixels. Disparity map computation for Wang et al. takes $\sim$8 minutes, which is a pre-process to both the k-means baseline and LFSP. LFSP itself takes $\sim$2 minutes, with k-means taking $\sim$2.5 minutes. Our approach implicitly computes a disparity map and takes $\sim$3.3 minutes total.

5.3 Discussion and Limitations

Our approach attempts to compute a view-consistent superpixel segmentation and produces competitive results; however, some issues still remain as not every pixel in the light field is view consistent. First, our occlusion-aware EPI segmentation is explicitly enforced by matching rules; however, the clustering step in Section 5.1.1 does not explicitly handle occlusion—this is only softly considered within the clustering by a high disparity weight. Further, for efficiency, we rely on only the central horizontal and vertical views. When segment boundary estimates do not align between these two sources, or when pixels are occluded from both of these sets of views, we rely on our less robust label propagation (Section 5.1.1) which is not occlusion aware and uses no explicit spatial smoothing, e.g., via a more expensive pairwise optimization scheme. Both these issues can cause minor label 'speckling' at superpixel boundaries.

While a valued resource for its labels, the HCI dataset [112] has minor artifacts in its ground truth disparity, such as jagged artifacts on the wooden plank in the 'buddha' scene. It is no longer supported and a replacement exists [35]; however, this does not include object segmentation labels for non-central views, which makes evaluating view consistency with it difficult.

Our Lambertian assumption makes it difficult to handle specular objects: view-dependent effects break the assumption that a 3D scene point maps to a line in EPI space, *e.g.*, in the HCI dataset 'horses' scene where all methods have trouble. Further, as the normalized ratio of area to perimeter, compactness is only a measure of average shape across the superpixel, and sometimes our superpixel boundaries have higher curvature than LFSP.

5.4 Summary

We presented the first of our reconstruction methods for decoding our multi-view edge model. Our method generates piece-wise constant depth by posing reconstruction as 4D light field superpixel segmentation. It outperforms LFSP [127] on view consistency and boundary accuracy metrics, and achieves similar superpixel compactness. It also outperforms a depth-based k-means clustering baseline on view consistency and compactness, and is competitive in boundary accuracy measures.

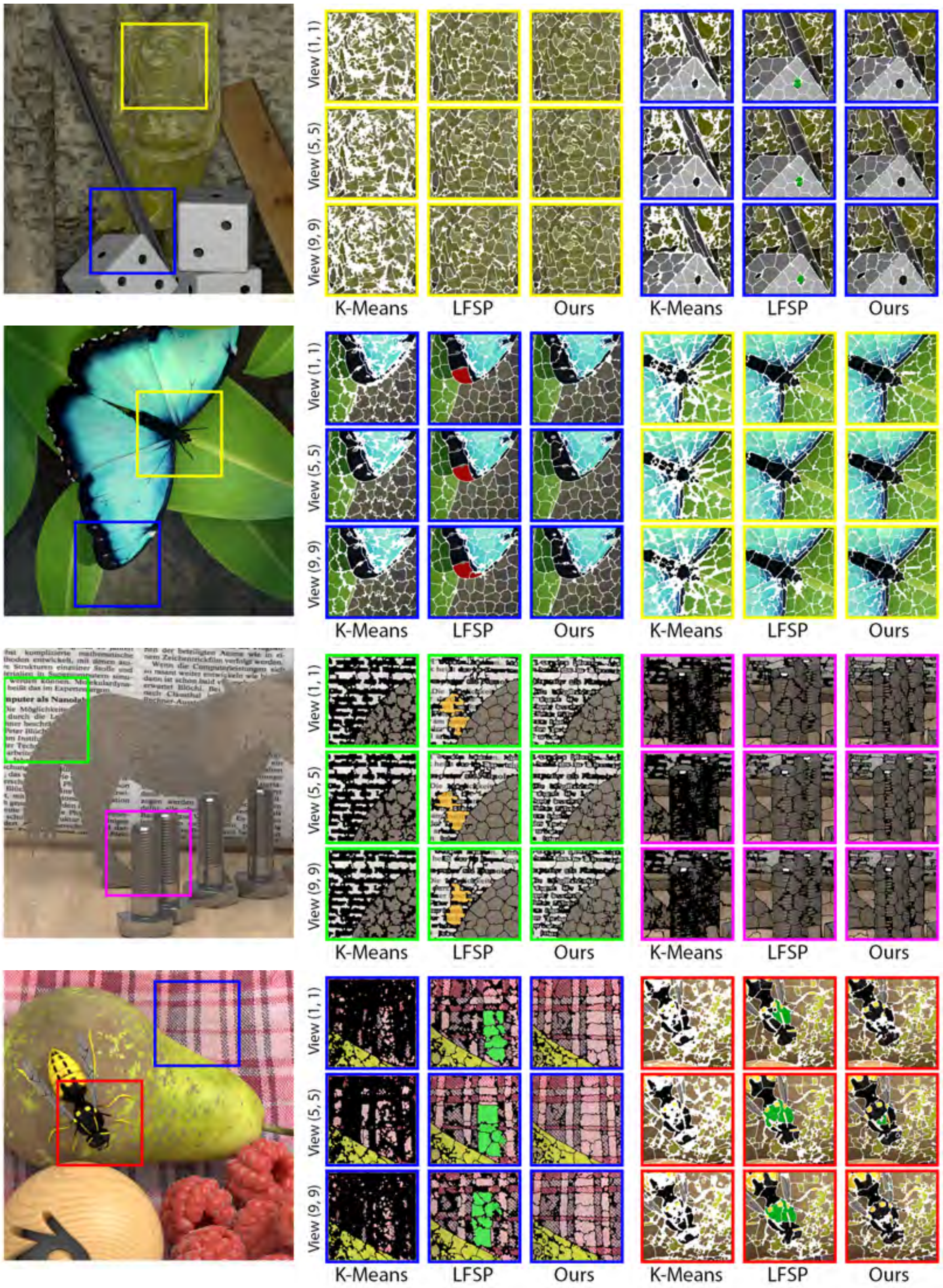


Figure 5.5: Superpixel segmentation boundaries and view consistency for the k-means baseline, LFSP [127], and our method. Disparity maps for LFSP and k-means were calculated using the algorithm of Wang et al. [107, 108]. We highlight superpixels which either change shape or vanish completely across views. Our superpixels tend to remain more consistent over view space, which can be easily seen as reduced flickering in our supplementary video. *Note:* Small solid white/black regions appear when superpixels are enveloped by the boundary rendering width. k-means tends to have more of these regions which helps it increase boundary recall, but this behavior is not useful for a superpixel segmentation method.

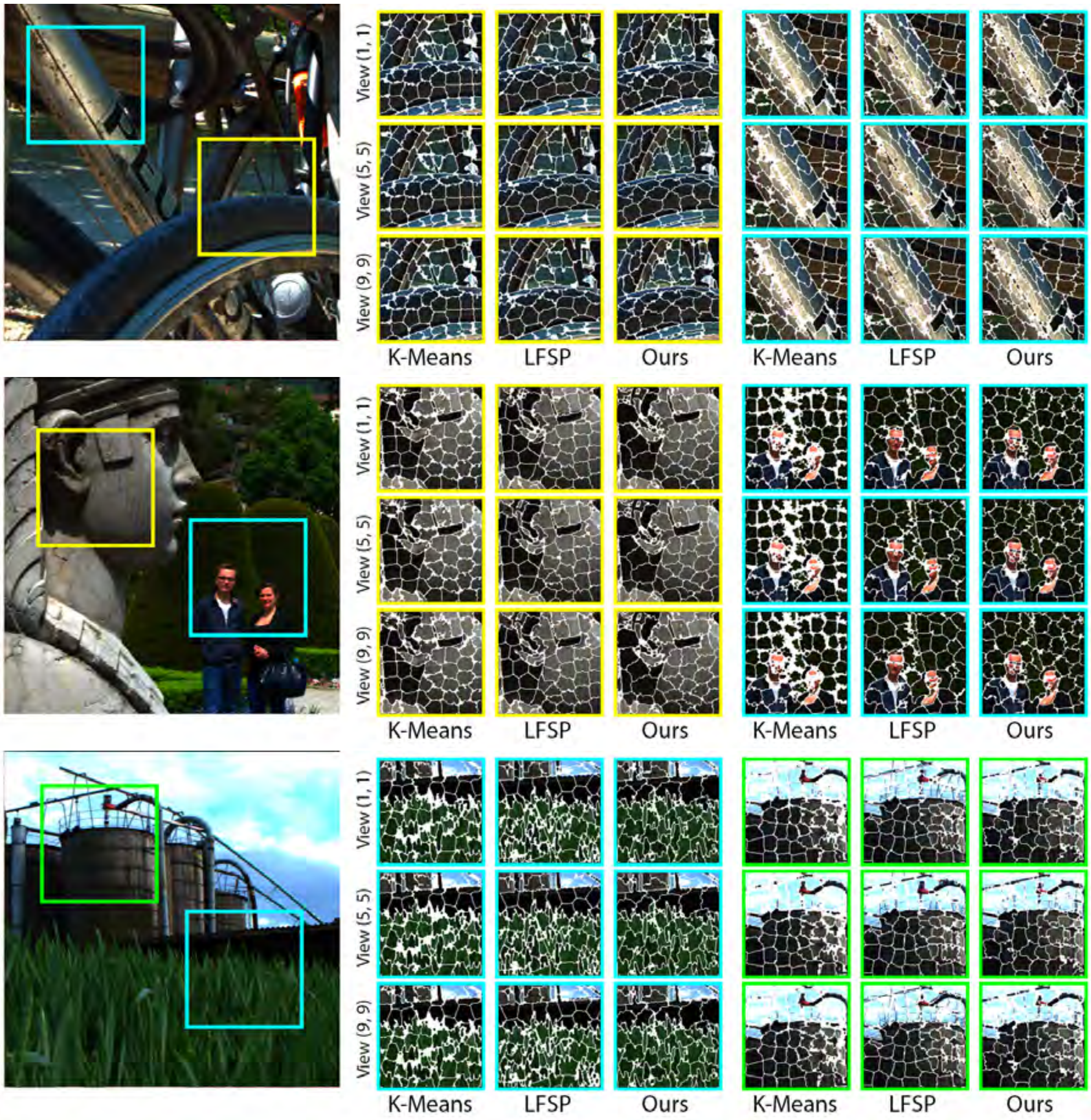


Figure 5.6: Superpixel segmentation boundaries and view consistency for the k-means baseline, LFSP [127], and our method. Disparity maps for LFSP and k-means were calculated using the algorithm of Wang et al. [107, 108]. We include six light fields from the EPFL Lytro light field dataset [80], with three more in Figure 5.7. Our superpixels tend to remain more consistent over view space, which can be easily seen as reduced flickering in our supplementary video. *Note:* Small solid white regions appear when superpixels are enveloped by the boundary rendering width. k-means tends to have more of these regions which helps it increase boundary recall, but this behavior is not useful for a superpixel segmentation method.

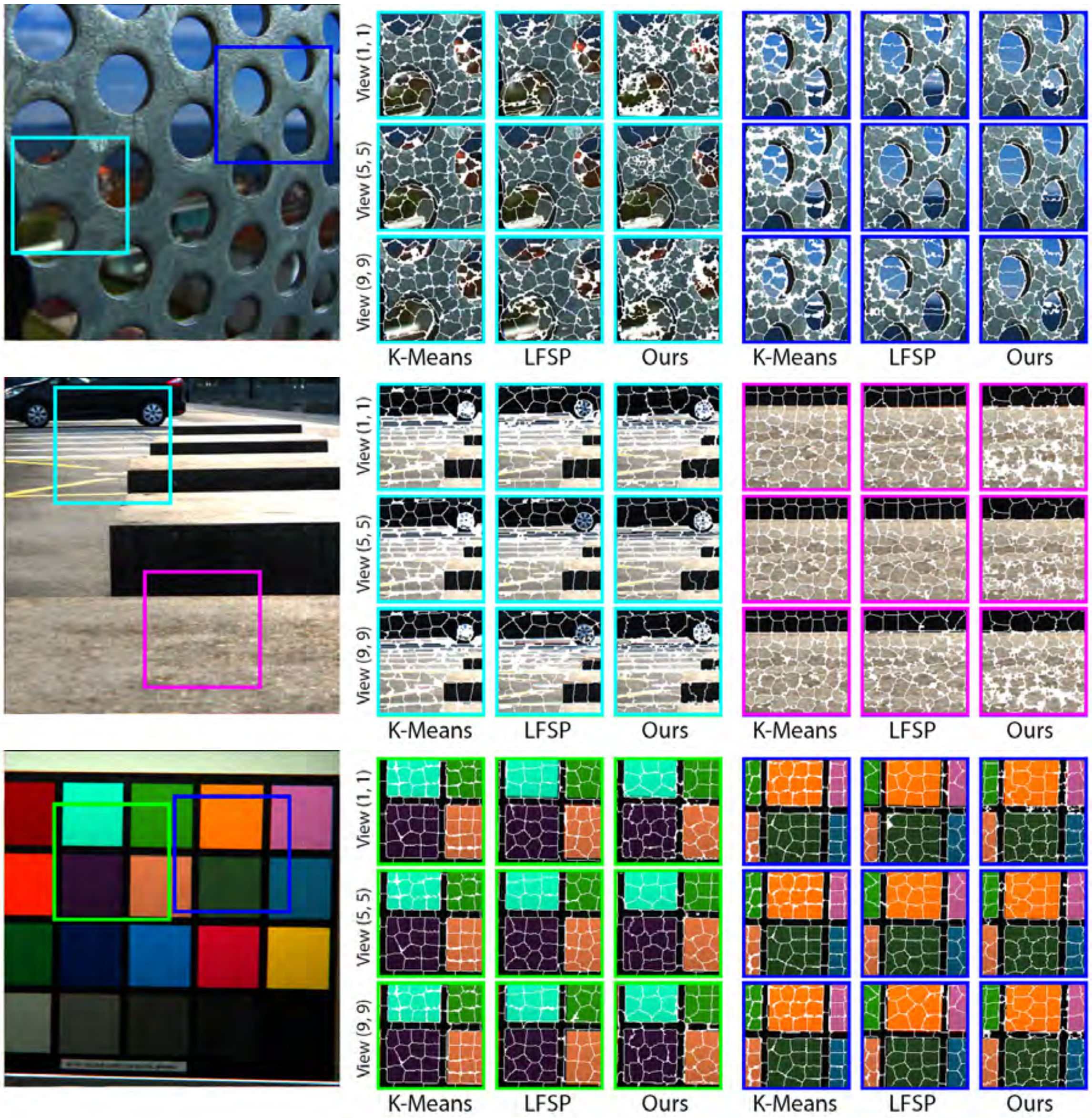


Figure 5.7: Superpixel segmentation boundaries and view consistency for the k-means baseline, LFSP [127], and our method. Disparity maps for LFSP and k-means were calculated using the algorithm of Wang et al. [107, 108]. We include six light fields from the EPFL Lytro light field dataset [80], with three more in Figure 5.6. Our superpixels tend to remain more consistent over view space, which can be easily seen as reduced flickering in our supplementary video. *Note:* Small solid white regions appear when superpixels are enveloped by the boundary rendering width. k-means tends to have more of these regions which helps it increase boundary recall, but this behavior is not useful for a superpixel segmentation method.

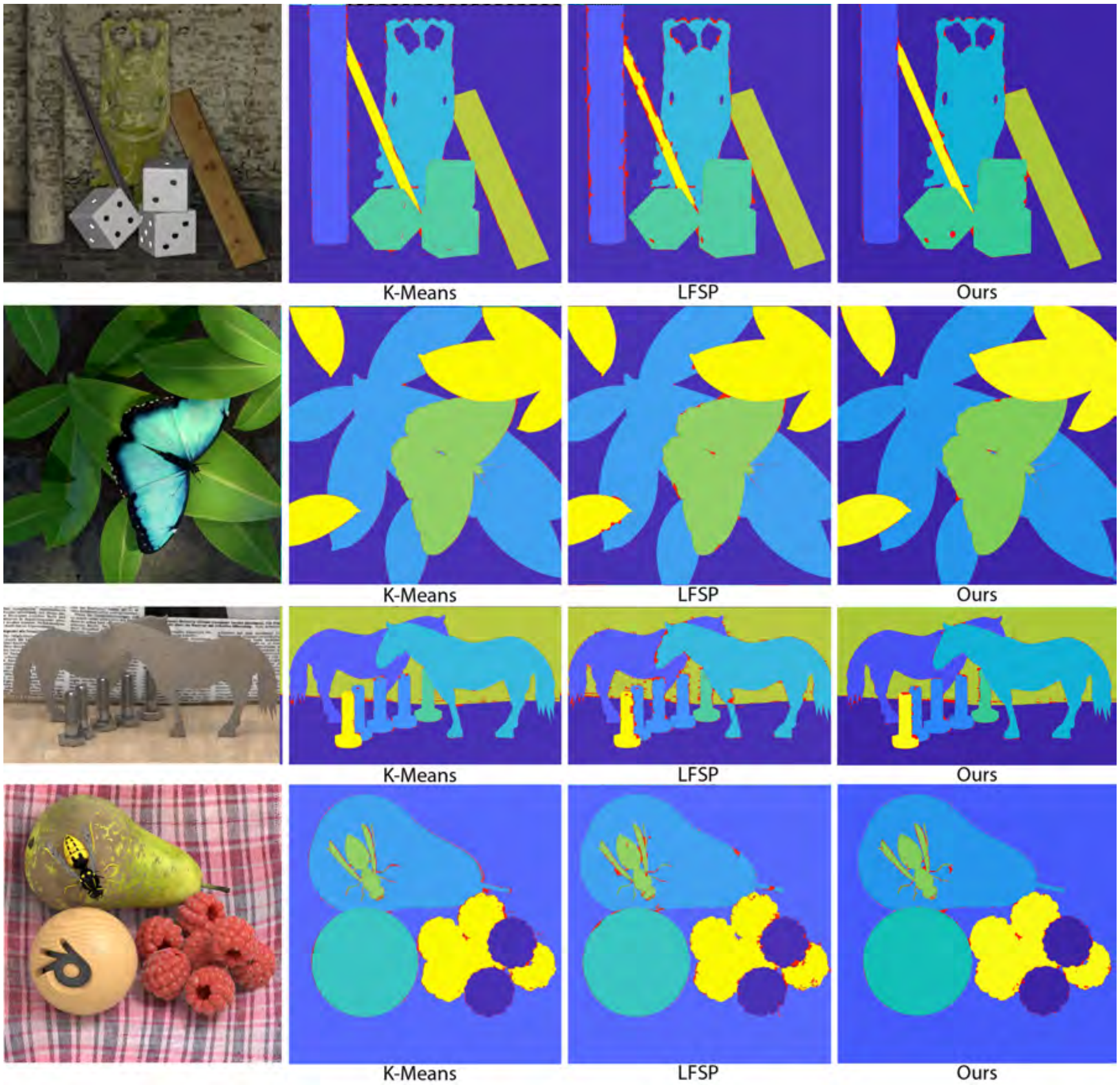


Figure 5.8: A visualization of the achievable segmentation accuracy in the central light field view. Red regions are superpixel sections which cross ground truth object segmentation boundaries. All other colors denote object labels.

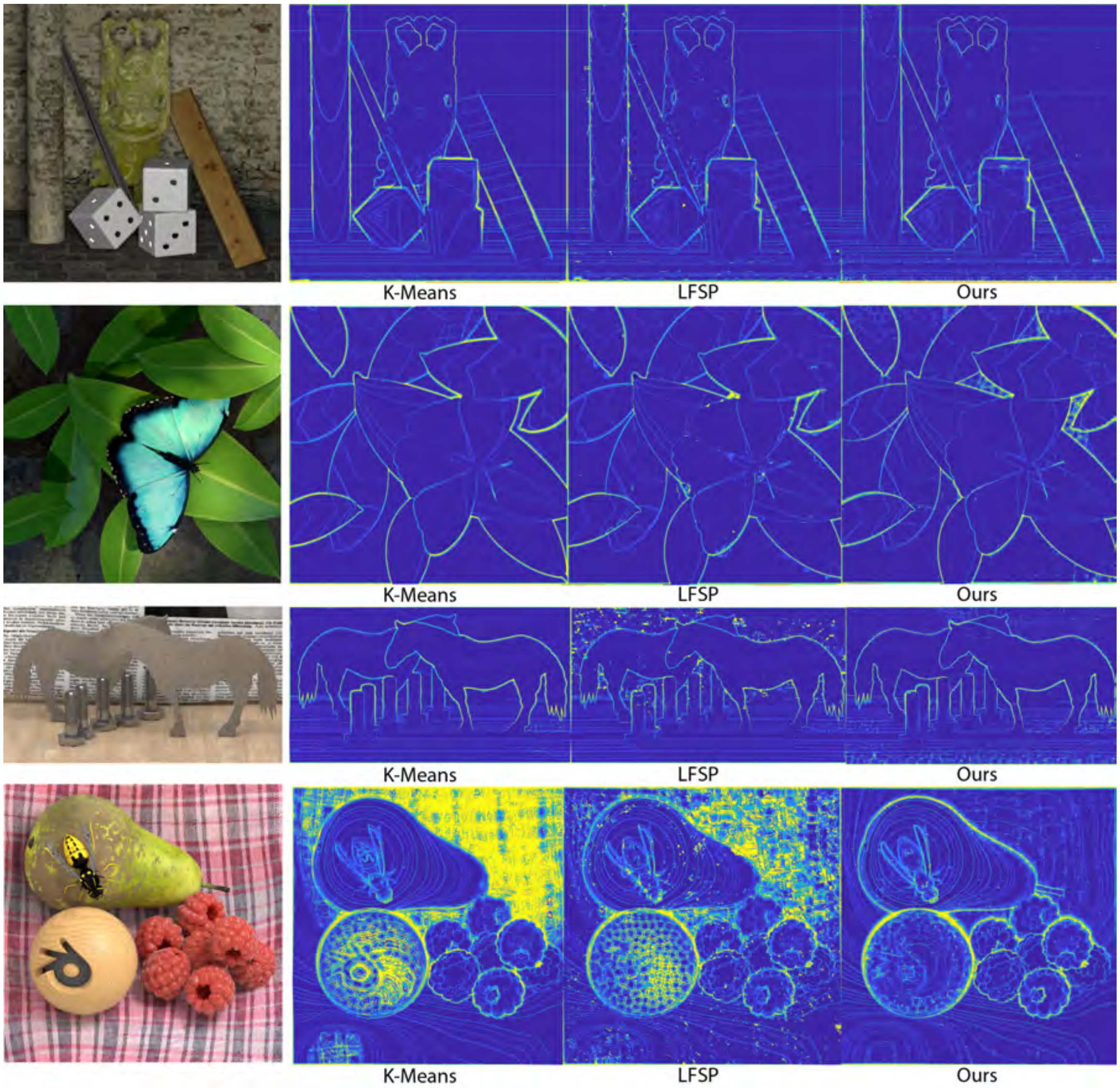


Figure 5.9: Average number of labels per pixel metric, shown as a heatmap for the central view where blue is low (good view consistency) and red is high (bad view consistency). From left to right, the images are: input, k-means, LFSP, and our result. Both k-means and LFSP use the Wang et al. generated depth map. We can see that most errors occur at edges; that LFSP has more edge inconsistency; and that k-means is sometimes sensitive to high-frequency pattern textures.

Chapter 6

Smooth 4D Reconstruction via Angular Diffusion

This chapter presents the second of our reconstruction methods which utilizes the full model described in Chapter 4. We demonstrate the advantage of the surface vector for smooth and, thus, more accurate reconstructions. By showing that our edge model encodes all relevant information required for generating accurate and view-consistent depth reconstructions with strong occlusion edges, we establish the completeness of our model.

In addition to validating the completeness of our multi-view edge model, the reconstruction method, when paired with the encoding scheme, may be used for light field depth estimation. Light field depth estimation is a difficult problem. Oftentimes, state-of-the-art methods strive for geometric accuracy without always considering occlusion edges, which are especially important for handling visibility in computational photography applications. Moreover, light fields allows high-quality depth estimation of fine detail by aggregating disparity information across many sub-aperture views. This typically results in a depth map for the central view of a light field only. However, in principle we can estimate depth for every pixel in the light field. Some applications require this ability, such as when editing a light field photograph when every output view will be seen on a light field display. Estimating depth reliably for disoccluded regions is difficult because it requires aggregating information from fewer samples. Few existing methods estimate depth for every light field pixel. These are typically computationally expensive [111, 125] or not strictly occlusion aware. Further, while the many views allow dense and accurate depth to be derived, the extra angular dimension carries large data costs that makes most depth estimation algorithms computationally inefficient [44, 125]. Recent methods have sought to overcome this barrier by learning data-driven priors with deep learning. Jiang et al. [47, 46] presented the first practical view consistent method based on deep learning. While this approach can be effective, it requires additional training data, and may overfit to scenes or capture scenarios [58].

Our multi-view edge encoding and smooth reconstruction method may be considered as a counterpart first principles light field depth estimation method with no learned priors, which produces comparable or better accuracy and view consistency than the current state of the art while being 2–4× faster. Our method is based around estimating accurate view-consistent disparity at edges, and then completing an occlusion-aware diffusion process to fill in missing regions. Depth diffusion is a long-standing problem in which it is difficult to ensure both consistency and correctness in disoccluded regions. Our key contribution is an angular inpainting method that ensures depth consistency by design, while accounting for the visibility of points in disoccluded regions. In this way, we avoid the problem of trying to constrain or regularize view consistency after estimating depth spatially, and so can maintain efficiency.

6.1 Occlusion-aware Depth Diffusion

A naive solution to reconstruct per-view depth from our multi-view edge encoding would be to compute a disparity map for each sub-aperture view separately. However, this is typically challenging for edge views and is highly inefficient, not only in terms of redundant computation but also due to the spatial domain constraints or regularization that must be added to ensure that depth maps are mutually consistent across views. Another simple approach might be to calculate a disparity map for a single view, and then reproject it into all other views. However, this approach fails to handle scene points that are not visible in the single source view. Such points cause holes in the case of disocclusions, or lead to inaccurate disparity estimates when the points lie on an occluding surface. While most methods try to deal with the former case through inpainting, for instance via diffusion, the latter scenario is more difficult to deal with as the occluding surface may have a depth label not seen in the original view. Thus, techniques like diffusion are insufficient on their own to prove or disprove the completeness of our edge model.

Our proposed method deals with this issue of depth consistency in subviews of light fields via an occlusion-aware diffusion process. It uses our occlusion-aware multi-view edges which persist across views as reliable guides for filling any holes in reprojected views. Since the edge depth labels are not restricted to the source view, we capture any occluding surfaces not visible in the source view. This avoids the aforementioned problem of unseen depth labels. In addition, by performing our inpainting step in the angular rather than the spatial domain of the light field, we improve cross view consistency and occlusion awareness.

Central View Depth Estimation We begin with our multi-view edge model from Chapter 4. This includes a sparse set of multi-view edge points $\mathcal{P}$, surface vectors parallel to the image gradient $\nabla I(\cdot)$, and an occlusion edge confidence. Let $\mathcal{Q} = \{p \pm \nabla I(p) \; \forall \, p \in \mathcal{P}\}$ be the sparse set of points $\mathcal{P}$ offset by the surface vector. We use $S[\mathcal{Q}]$ to represent the image created by splatting sparse points in a set $\mathcal{Q}$ onto a $w \times h$ raster grid, and D to be a dense $w \times h$ disparity map. Diffusion is formulated as a constrained quadratic optimization problem:

$$\hat{D}[\mathcal{Q}] = \underset{D}{\operatorname{argmin}} \sum_{p \in \mathcal{Q}} E_d(p) + \sum_{(p,q) \in \mathcal{S}} E_s(p,q), \tag{6.1}$$

where $\hat{D}[\mathcal{Q}]$ is the optimal disparity map given the sparsely labeled image $S[\mathcal{Q}]$ and $\mathcal{S}$ is the set of all four-connected neighbors in D. The data and smoothness terms are defined as

$$E_d(p) = \lambda_d(p) \big\| S[\mathcal{Q}](p) - D(p) \big\|, \quad \text{and} \quad E_s(p,q) = \lambda_s(p) \big\| D(p) - D(q) \big\|, \tag{6.2}$$

The data weight $\lambda_d(p)$ is defined in terms of the sets $\mathcal{P}_f = \{p + \nabla I(p) \; \forall \; p \in \mathcal{P} \;\}$ and $\mathcal{P}_b = \{p - \nabla I(p) \; \forall \, p \in \mathcal{P} \;\}$ as

$$\lambda_d(p) = \omega \exp(a \lambda_e(p)) \quad \text{with} \quad \lambda_e(p) = \max_{\{\hat{D}[\mathcal{P}_f], \hat{D}[\mathcal{P}_b]\}} \| N_p \circledast \mathbf{F} \|. \tag{6.3}$$

where N_p is a set of pixels around p, and $\mathbf{F}$ is the step filter $[-1 \; -1 \; 1 \; 1]$. The parameters ω and a are set to 1.5×10^2 and 3, respectively. These values work for all scenes. The smoothness term $\lambda_s(p)$ is provided by the depth edge confidence at every pixel estimated in Section 4.3.

Equation (6.1) represents a standard Poisson problem, and we solve it using an implementation of the LAHBPCG solver [100] by posing the constraints in the gradient domain as proposed by Bhat et al. [8]. The result $\hat{D}[\mathcal{Q}]$ provides a dense depth estimate for the central view.

Cross-hair View Projection The EPI line-fitting algorithm works on EPIs in the central cross-hair views—that is, the central row and column of light field images. While it is possible to run it on other rows and columns, this can become expensive, and the central set is usually sufficient to detect visible surfaces in the light field [113]. Hence, we project the estimated disparity map from the center view into all views along the cross-hair. Since gradients at depth edges in the estimated disparity map are not completely sharp, this leads to some edges being projected onto multiple pixels in the target view. We deal with this by sharpening the edges of the disparity map before projection, as in Shih et al. [94], using a weighted median filter [64] with parameters $r = 7$ and $\epsilon = 10^{-6}$. Omitting this step causes inaccurate estimates around strong depth edges. The result is not very sensitive to parameters r and ϵ since most settings will target the error-prone strong edges.

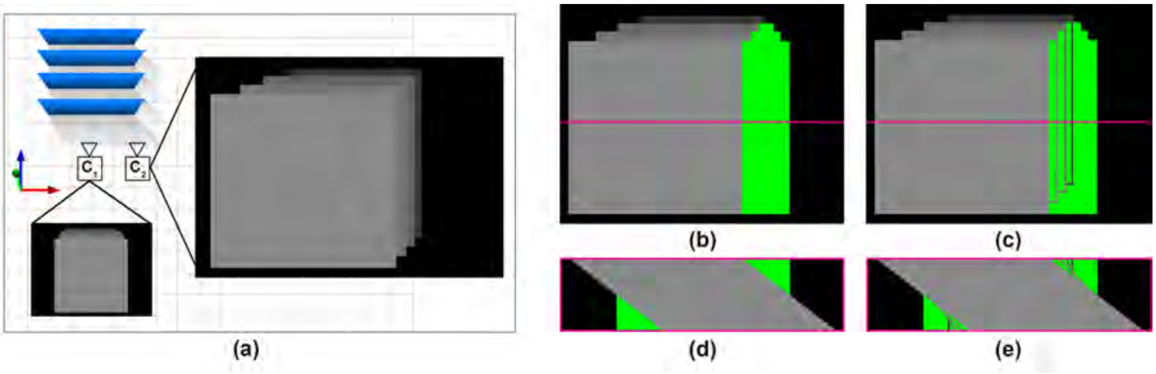


Figure 6.1: **(a)** Ground-truth disparity maps for a scene from two different camera positions C_1 and C_2. **(b)** Naively attempting to generate the output of C_2 by reprojecting C_1 results in large holes, shown here in green. **(c)** Our method uses depth edges to guide disparity propagation in such disoccluded regions. The EPIs corresponding to the highlighted row are shown in **(d)** and **(e)**. The EPI in **(e)** constitutes our depth EPI D_o.

Angular Inpainting After depth reprojection, we must deal with the two problems highlighted in the overview: inpainting holes, and accounting for occluding surfaces in off-center sub-aperture views. We tackle this by using our multi-view edges to guide a dense diffusion process. Moreover, we ensure view consistency by performing diffusion in EPI space.

The multi-view edges constitute a set $\mathcal{L}$ of cross-view edge features (Chapter 4.1) which is robust to occlusions in a single view as it exists in EPI space. As such, $\mathcal{L}$ provides occlusion-aware sparse depth labels to guide dense diffusion in EPI space. Diffusion in EPI space has the added advantage of ensuring view consistency.

Let D_o represent an angular slice of the disparity maps with values reprojected from the center view and with propagation guides (Figure 6.1). Again, we formulate diffusion as a constrained quadratic optimization problem:

$$\hat{D} = \underset{D}{\operatorname{argmin}} \sum_{p \in D} E_d(p) + \sum_{(p,q) \in \mathcal{S}} E_s(p,q), \tag{6.4}$$

where $\hat{D}$ is the optimal depth labeling of the EPI, and $\mathcal{S}$ is the set of four-connected neighboring pixels. The data $E_d(p)$ and smoothness terms $E_s(p,q)$ are defined as:

$$E_d(p) = \lambda_d(p) \left\| D(p) - D_o(p) \right\|_2^2, \tag{6.5}$$

$$E_s(p,q) = \lambda_s(p,q) \left\| D(p) - D(q) \right\|_2^2, \tag{6.6}$$

We take the weight for the smoothness term from the EPI intensity image I:

$$\lambda_s(p,q) = \frac{c}{\|\nabla I(p)\| + \epsilon}, \tag{6.7}$$

where $c = 0.1$. We define the weight for the data term as:

$$\lambda_d(p) = \begin{cases} 15 & \text{if } p \in \mathcal{C}, \\ \lambda_e(p) & \text{if } p \in \mathcal{L}, \\ 0 & \text{otherwise}, \end{cases} \tag{6.8}$$

where $\lambda_e(p)$ is the edge-importance weight from Equation 6.3, and $\mathcal{C}$ and $\mathcal{L}$ are the set of pixels coming from the reprojected center view disparity map and EPI line guides, respectively.

Equation (6.4) defines the optimal disparity map $\hat{D}$ as one that minimizes divergence from the labeled data (Equation. (6.5)) while being as smooth as possible. Equation (6.6) measures smoothness as the similarity between disparities of neighboring pixels. We wish to relax the smoothness constraint for edges, so smoothness weight is chosen as the inverse of the image gradient (Equation (6.7)). This allows pixels across edges to have a disparity difference without being penalized. The data weight (Equation (6.8)) is determined empirically and works for all datasets.

Equation (6.4) is again a standard Poisson optimization problem. We solve this using the Locally Adaptive Hierarchical Basis Preconditioning Conjugate Gradient (LAHBPCG) solver [100] by posing the data and smoothness constraints in the gradient domain [8].

Non-cross-hair View Reprojection We now have view-consistent disparity estimates for every pixel in the central cross-hair of light field views: $(u_c, \cdot)$, and $(\cdot, v_c)$. As noted, this set is usually large enough to cover every visible surface in the scene. Hence, all target views (u_i, v_i) outside the cross-hair can be simply computed as the mean of the reprojection of the closest horizontal and vertical cross-hair view ((u_c, v_i) and (u_i, v_c), respectively). The result is a view-consistent, dense depth reconstruction for each image in the light field.

6.2 Experiments

6.2.1 Datasets

For our evaluation, we used both synthetic and real world light fields with a variety of disparity ranges. For the synthetic light fields, we used the HCI Light Field Benchmark Dataset [35]. This dataset consists of a set of four 9 × 9, 512 × 512 pixels light fields: *Dino*, *Sideboard*, *Cotton*, and *Boxes*. Each has a high-resolution ground-truth disparity map for the central view only.

For real-world light field data, we use the EPFL MMSPG Light-Field Image Dataset [80] and the New Stanford Light Field Archive [56]. The EPFL light fields are captured with a Lytro Illum and consist of 15×15 views of 434×625 pixels each. However, as the edge views tend to be noisy, we only use the central 7×7 views in our experiments. We show results for the *Bikes* and

Sphynx scenes. The light fields in the Stanford Archive are captured with a moving camera and have a larger baseline than the Lytro and synthetic scenes. Each scene consists of 17×17 views with varying spatial resolution. We use all views from the *Lego* and *Bunny* scenes, scaled down to a spatial resolution of 512×512 pixels.

6.2.2 Baselines

We compare our reconstruction results to three non-learning-based methods: the defocus and correspondence cues methods by Jeon et al. [44] and Wang et al. [108], and the spinning parallelogram operator of Zhang et al. [125]. We also compare with the learning-based methods of Jiang et al.. [46], Shi et al.. [93], and Li et al.. [58]. Both Shi et al.and Jiang et al.use the deep-learning-based Flownet 2.0 [38] network to estimate optical flow between the four corner views of a light field, then use the result to warp a set of anchor views. In addition, Shi et al. further refine the edges of their depth maps using a second neural network trained on synthetic light fields. While Shi et al.'s method generates high-quality depth maps for each sub-aperture view, they do not have any explicit cross-view consistency constraint. We do not compare to Holynski and Kopf [34]: this uses COLMAP, which fails on typical skew-projected light field data.

6.2.3 Metrics

We evaluate both the accuracy and consistency of the depth maps. For accuracy, we use the mean-squared error (MSE) multiplied by a hundred, and the 25th percentile of absolute error (Q25). The unavailability of ground truth depth maps for the EPFL and Stanford light fields prevents us from presenting accuracy metrics for those light fields.

To evaluate view consistency, we reproject the depth maps onto a reference view and compute the variance. Let $\rho_0, \rho_1, \ldots, \rho_n$ represent the depth maps for the n light field views warped onto a target view (u, v). The view consistency at pixel s in view (u, v) is given by:

$$\mathcal{C}_{(u,v)}(s) = \frac{1}{n}\sum_{i=0}^{n}(\rho_i(s) - \mu_s)^2, \qquad \text{where } \mu_s = \frac{1}{n}\sum_{i=0}^{n}\rho_i(s), \tag{6.9}$$

and overall light field consistency is given as the mean over all pixels s in the target view S:

$$\mathcal{C}_{(u,v)} = \frac{1}{S}\sum_{s=0}^{S}\mathcal{C}_{(u,v)}(s). \tag{6.10}$$

This formulation allows for the consistency to be evaluated quantitatively for both the synthetic and real world light fields.

We also compare running time. Our method (excepting the C++ Poisson solver) was implemented in MATLAB, as were the three traditional algorithms and parts of Jiang et al. All networks were implemented in Tensorflow. All CPU code was run on an AMD Ryzen Threadripper 2950X 16-Core Processor, and GPU code on an NVIDIA GeForce RTX 2080Ti.

6.2.4 Results

Occlusion Edge Accuracy. Qualitatively, our method produces sharper and more accurate occlusion edges than state-of-the-art light field depth estimation methods. In Figure 6.3, we visualize occlusion boundaries as depth gradients. While the learning-based methods of Shi et al. and Li et al. generating spurious boundaries in textureless regions, the approach of Yucer et al. [122] fails entirely in the absence of thousands of views. We also evaluate our edges quantitatively on four scenes from the synthetic HCI Dataset [35] via ground truth disparity maps for the central view (Fig. 6.4 and Tab. 6.1). Although our Q25 error is higher, our method has high boundary-recall precision, and a lower average mean-squared error than all baselines.

Our method works on 2D slices of a 4D light field. While jointly considering the 4D structure may improve accuracy, edge detection and diffusion become computationally expensive. In principle, the accuracy of our edge detection can be improved by entropy-based refinement of labels (Sec. 4.1.5) in both vertical and horizontal EPIs. In practice, we found no advantage of doing so.

Diffusion Gradients as Self-supervised Loss. One way to think about bidirectional diffusion gradients is as a self-supervised loss function for depth edge localization. With this view, we compare its performance to *multi-view reprojection error*—a commonly used self-supervised loss in disparity optimization. We use the dense disparity maps $\hat{D}[\mathcal{P}_f]$ and $\hat{D}[\mathcal{P}_b]$ to warp all light field views onto the central view through an occlusion-aware inverse projection. A reprojection error map is calculated as the mean per-pixel L1 intensity error between the warped views and the central view. The offset direction at each point $p \in \mathcal{P}$ is then determined based on the disparity map that minimizes the reprojection error at the pixel location of p. Table 6.2 evaluates the result of calculating $\mathcal{Q} = \{p \pm \nabla I(p) \, \forall \, p \in \mathcal{P}\}$ based on the reprojection error maps instead of our bidirectional diffusion gradients. Our method has consistently lower MSE, indicating better edge performance. This intuition is qualitatively confirmed by supplemental Figure B.7.

Accuracy Table 6.1 presents quantitative results for the central view of all light fields in accuracy comparisons against ground truth depth. Our method is competitive or better on the MSE metric against the baseline methods, reducing error on average by 20% across the four light fields. However, our method produces more erroneous pixels than the baseline methods as given by the Q25

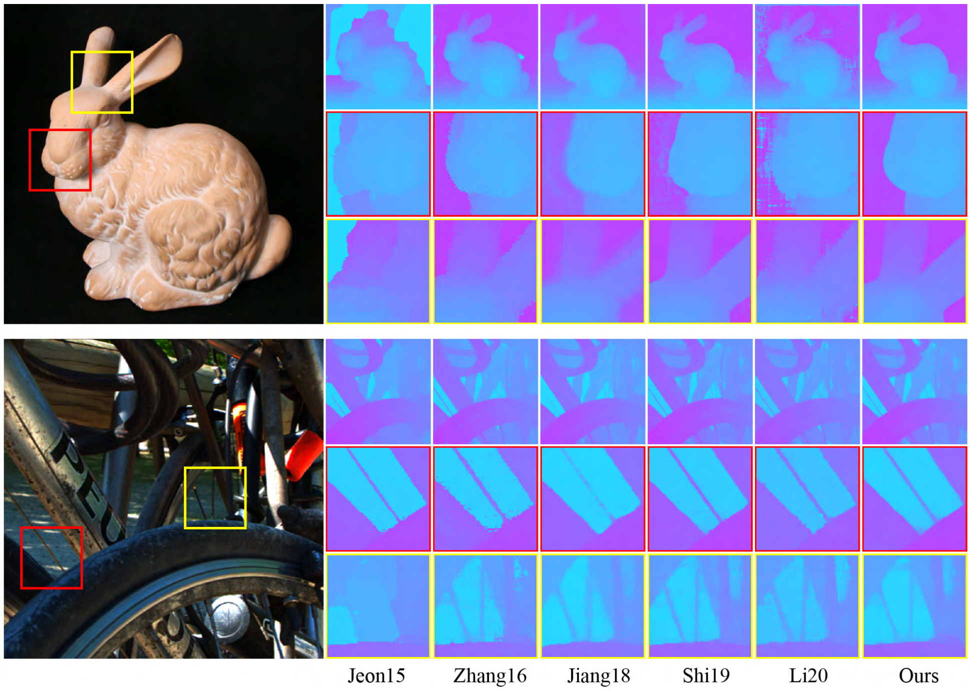


Figure 6.2: Occlusion edges in disparity maps. *Top:* Stanford dataset light field captured with a camera rig. *Bottom:* EPFL light field from a Lytro Illum. *Left to right:* Jeon et al. [44], Zhang et al [125], Jiang et al. [46], Shi et al. [93], and ours.

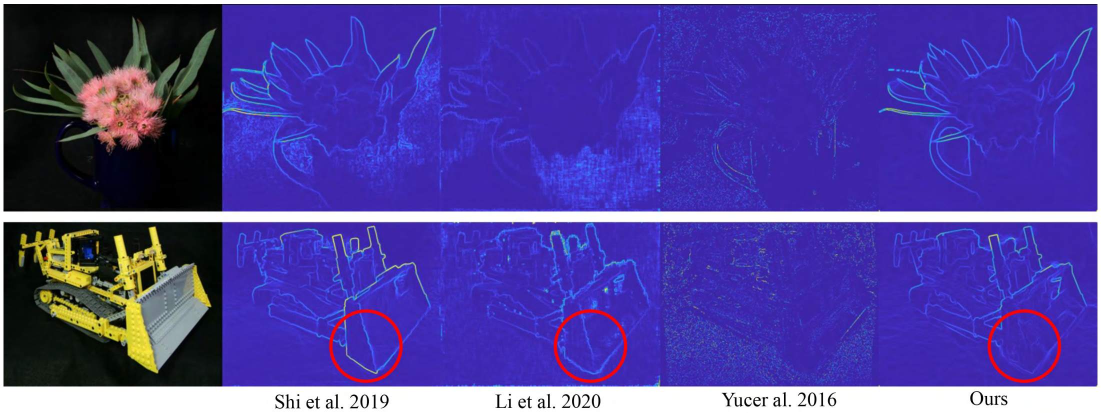


Figure 6.3: Visualizing occlusion edges as gradients of disparity maps. *Left to right:* Shi et al. [93], Li et al [58], Yucer et al. [122], and ours. *Bottom row, red circle:* the learning-based methods hallucinate a strong depth edge on the plow even though it is in contact with the black ground cloth at the same depth (supplemental Sec. B.5). Yucer et al.'s method fails in the absence of many views.

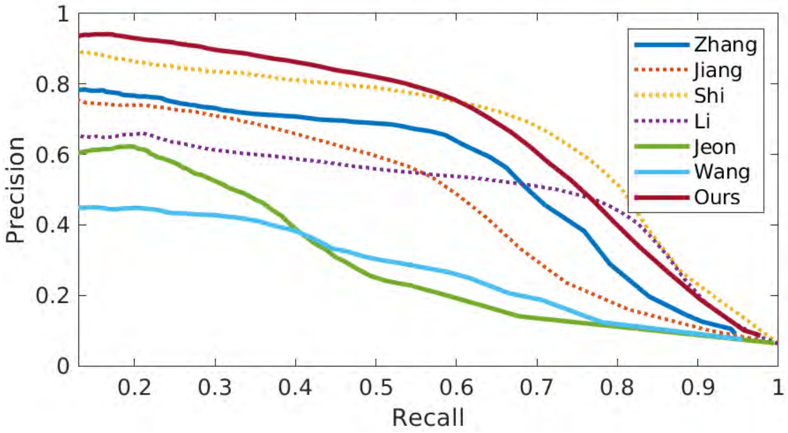


Figure 6.4: Average precision-recall curves of depth boundaries for all baseline algorithms (HCI dataset). Learning-based methods are shown as dotted lines. Our approach consistently outperforms traditional algorithms [44, 108, 125] and the learning-based method of Jiang et al. [46], while outperforming Shi et al. [93] and Li et al. [58] at medium-to-low recall rates.

error. For baseline techniques to have higher MSE but fewer bad pixels means that they must have larger outliers. This can be confirmed by looking at the error plots in Figure 6.8.

View Consistency Figure 6.5 presents results for view consistency across all three datasets. The box plots at the top show that our method has competitive or better view consistency than the baseline methods. As expected, Shi et al.'s method without an explicit view consistency term has significantly larger consistency error. At the bottom of the figure, we visualize how this error is distributed spatially across the views in the light field. Both our method and Jiang et al.'s method produce relatively even distributions of error across views.

Computational Resources Figure 6.6 presents a scatter plot of runtime versus view consistency across our three datasets. Our method produces comparable or better consistency at a faster runtime, being 2–4× faster than Jiang et al.'s methods per view for equivalent error.

Qualitative Figures 6.8 and 6.9 present qualitative single-view depth map results. Overall, all methods produce broadly comparable results, though each method has different characteristics. The learning-based methods tend to produce smoother depths across flat regions. All methods struggle with thin features. On the *Bunny* scene, our approach introduces fewer background errors and shows fewer 'edging' artifacts than Jiang et al. Shi et al. produces cleaner depth map appearance for *Lego*, but is view inconsistent. Jiang et al. is view consistent, but introduces artifacts on *Lego*. On *Sphynx*, a distant scene and narrow baseline cause noise in our EPI line reconstruction.

Table 6.1: Quantitative comparison of our method and the baselines on the synthetic HCI light fields. The top three results are highlighted in gold, silver and bronze. MSE is the mean squared error; Q25 is the 25th percentile of the absolute error.

Light Field	MSE × 100							Q25							Run time (s)						
	[44]	[125]	[46]	[93]	[58]	[108]	Ours	[44]	[125]	[46]	[93]	[58]	[108]	Ours	[44]	[125]	[46]	[93]	[58]	[108]	Ours
Sideboard	3.21	1.02	1.96	1.12	1.89	13.3	1.03	0.61	1.15	0.37	0.48	0.66	2.46	1.22	754	537	507	72.3	77.1	635	35.5
Dino	1.73	0.36	0.47	0.43	3.28	4.19	0.45	1.07	1.40	0.25	0.31	0.50	2.02	0.85	805	531	500	59.3	76.8	609	37.7
Cotton	12.5	1.81	0.97	0.88	1.95	9.56	0.70	0.50	1.01	0.21	0.36	0.59	2.30	0.74	748	530	500	79.8	76.9	612	34.0
Boxes	16.0	7.90	11.6	8.48	4.67	12.5	7.52	0.75	1.64	0.42	0.69	0.78	2.21	1.41	736	541	491	56.2	78.0	667	34.3
Average	8.37	2.77	3.75	2.72	2.94	9.91	2.43	0.73	1.3	0.31	0.46	0.63	2.25	1.05	761	535	500	66.9	77.2	631	35.4

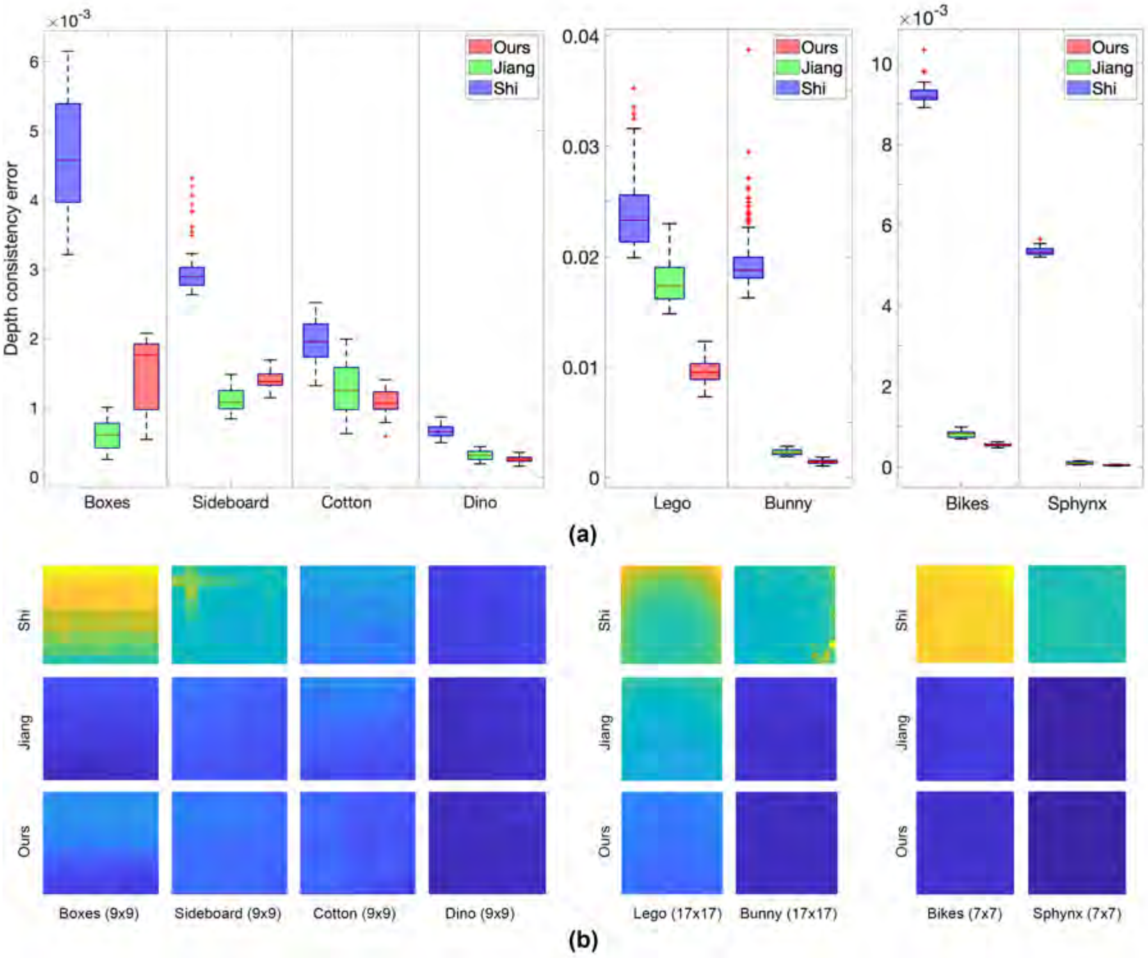


Figure 6.5: Quantitative view consistency comparison of our method and Jiang et al. [46] and Shi et al. [93]. While the method of Jiang et al. enforces cross view consistency, Shi et al. operates on each view individually and has no explicit consistency constraint. **(a)** For each light field, we plot summary statistics over $\mathcal{C}_{(u,v)}$ for all views (u, v) in the light field (Equation (6.10)). **(b)** The angular distribution of the error over all views.

Table 6.2: Evaluating disparity maps with depth edges identified via reprojection error and via our approach of diffusion gradients on the synthetic HCI dataset. MSE is the mean squared error; Q25 is the 25th percentile of absolute error.

Light	MSE × 100		Q25	
Field	Reproj	Ours	Reproj	Ours
Sideboard	1.39	**1.03**	**1.20**	1.22
Dino	0.64	**0.45**	**0.81**	0.85
Cotton	1.04	**0.70**	**0.68**	0.74
Boxes	9.32	**7.52**	1.65	**1.41**
Average	3.10	**2.43**	1.08	**1.05**

Table 6.3: Mean and Peak F1 across all thresholds, and the area under the precision-recall Curve (AUC) for the HCI dataset. Our method has the second-highest F1 score and, along with the learning-based method of Shi et al., the highest AUC.

Light Field	Mean F1†	Peak F1†	AUC†
Zhang [125]	3.41	6.07	5.07
Jiang [46] *	2.64	5.23	4.52
Shi [93] *	3.29	6.85	6.30
Li [58] *	3.72	5.78	4.60
Jeon [44]	2.14	3.67	3.06
Wang [108]	2.02	3.56	2.70
Ours	3.42	6.52	6.30

$^\dagger \times 10^{-1}$ * Learning-based

Applications Consistent per-view disparity estimates are vital for practical applications such as light field editing: knowing the disparity of regions occluded in the central view allows view-consistent decals or objects to be inserted into the 3D scene (Figure 6.7). Moreover, without per-view disparity, users can only edit the central view as changes in other views cannot be propagated across views. This limits editing flexibility.

6.3 Discussion and Limitations

Light Field Editing. As our method generates accurate depth edges that allow visibility to be handled correctly, our depth allows simple object insertion with few artifacts (Figures. 6.11 and B.10).

Errors. Our method has consistently lower mean squared error (MSE), but suffers a higher number of erroneous pixels (Q25). As Q25 measures the first quantile of absolute error, this indicates that baseline methods must have more outliers: the errors that they do have must be considerably large. This intuition is confirmed by visualizing the absolute error (Figure. B.8) which shows regions of large error around occlusion boundaries for the baseline methods.

Our method outperforms deep learning methods when they suffer from under-specification and over-fitting. Our three learning-based baselines are trained by supervision on the synthetic HCI dataset. When tested on real-world light fields, they produce artifacts along depth edges (Figure 6.3). This failure to generalize is especially evident with Li et al. [58] which suffers severe artifacts on real world data (compare Figure B.6 and B.9). Our method is not susceptible to these problems, producing comparable output on both synthetic and real-world light fields. Moreover, our method is robust to noise in the label set and low-gradient edges (Figures B.3 and B.4).

In addition, our method explicitly optimizes depth-edge localization whereas the learning-based methods are trained to minimize mean depth error over all pixels—edge and non-edge. Incorporating hard edge information in CNNs is generally not straightforward due to its sparsity and non-differentiability. Shi et al. [93] include a Canny edge-based loss term in their training routine, and consequently their performance on edges is relatively higher. Nonetheless, they are unable to effectively differentiate depth and texture edges (Figure 6.3).

Hyperparameters Figure 6.10 demonstrates the variation in error as hyperparameter values change. Across all parameters, our approach is stable around our declared values.

Limitations. As we estimate depth explicitly only around potential occlusion boundaries our method has lower accuracy in non-edge regions, reflected by the Q25 error (Table 6.1). Our lack of explicit (u,v) regularization can sometimes create inconsistent spatial edges, e.g., for diagonal angles in non-cross-hair views. Adding additional regularization begins another trade-off between smoothness, accuracy, and computational cost. Further, our method is limited when an EPI contains an area enclosed by high gradient boundaries with no data term depth value in it (this is the “island problem” for a discussion of which see Section 8.1.2).

6.4 Summary

We presented the second of our reconstruction methods for decoding our multi-view edge model. We show that careful handling of depth edge estimation, occlusions, and view consistency can produce per-view disparity maps with comparable performance to state of the art learning-based methods in terms of average accuracy, occlusion-accuracy, and view consistency. These results strongly indicate that our edge model encodes all relevant information required for generating accurate and view-consistent depth reconstructions with strong occlusion edges.

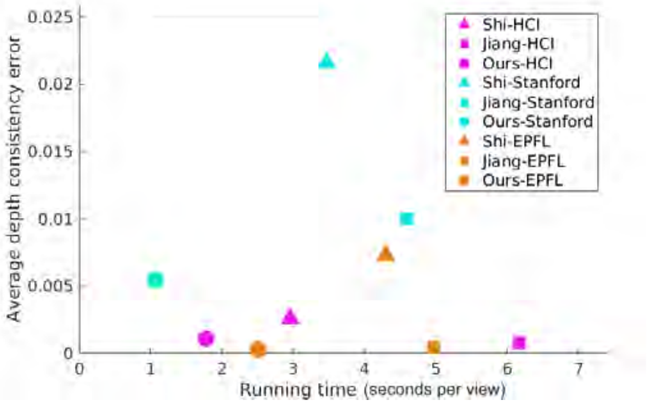
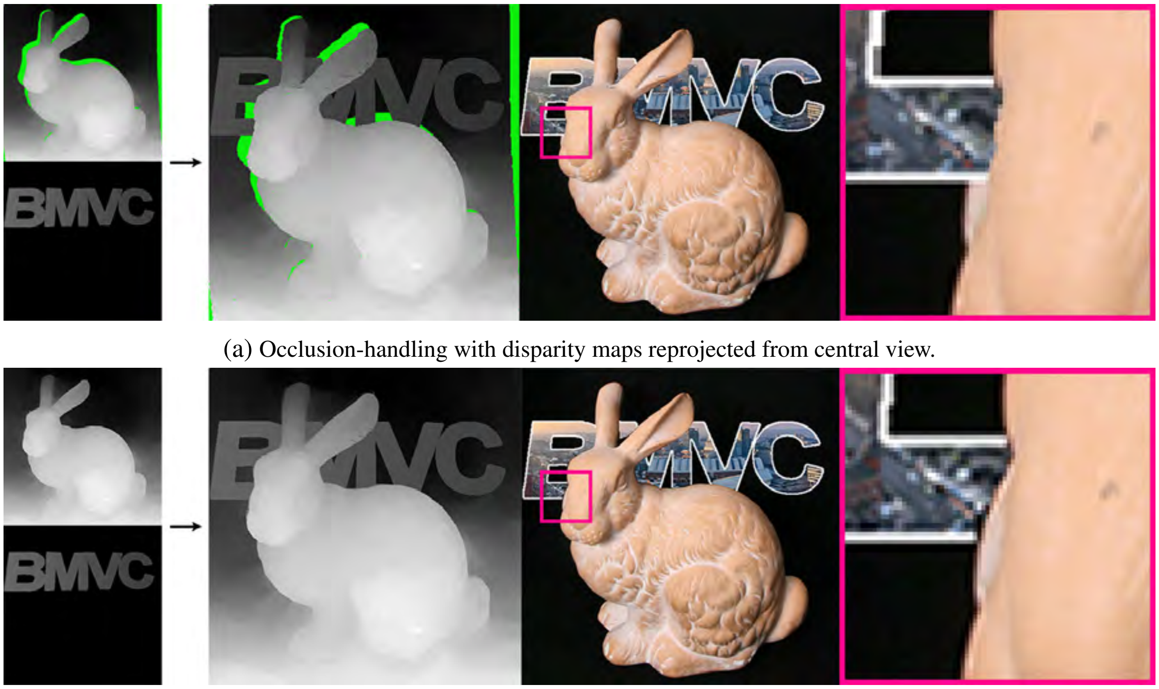


Figure 6.6: Average depth consistency error and runtimes for the three assessed datasets. Our method runs consistently faster than the baselines, while having comparative or better depth consistency. Note that errors across datasets are shown in absolute terms.



(a) Occlusion-handling with disparity maps reprojected from central view.

(b) Occlusion-handling with per-view disparity estimates.

Figure 6.7: With consistent per-view disparity estimates, objects can be inserted into the 3D scene with accurate occlusions. **(a)** Since the right cheek of the bunny is not visible in the central view, a simple reprojection of the disparity from the central view fails to handle occlusion correctly for the inserted object. The green regions denote holes in the reprojection. **(b)** With per-view disparity, the object can be placed correctly in all views.

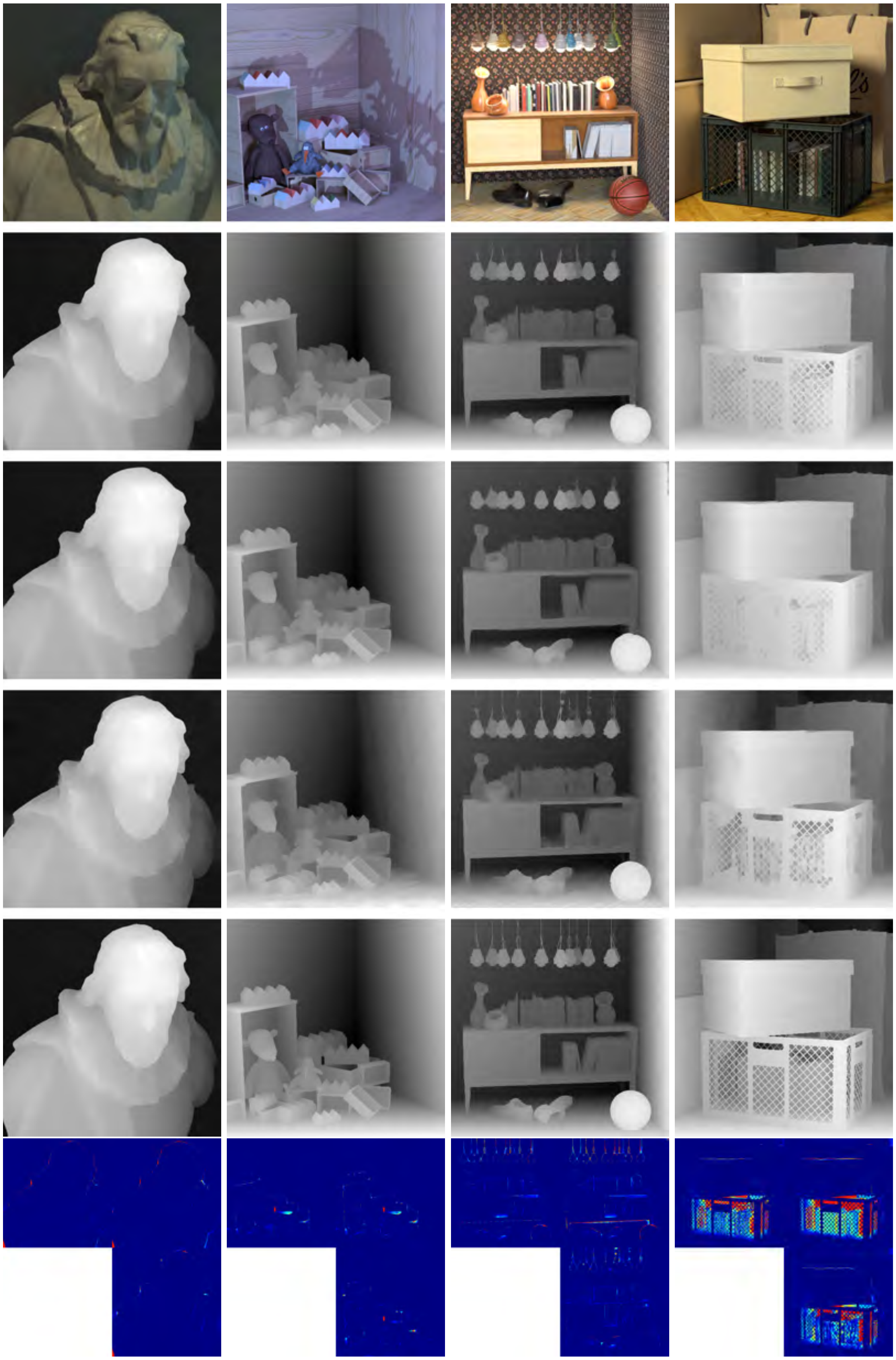


Figure 6.8: HCI dataset. Top to bottom: light field central view, Shi et al. [93], Jiang et al. [46], our method, ground truth depth, error maps in clockwise order (Shi, Jiang, Ours). In general, our method has a lower mean squared error (MSE) with fewer large outliers (please zoom into error maps), captures thin features better, and generates more view-consistent depth maps. However, their depth maps are more geometrically accurate more often (lower bad pixel percentages) and less sensitive to variations in image texture.

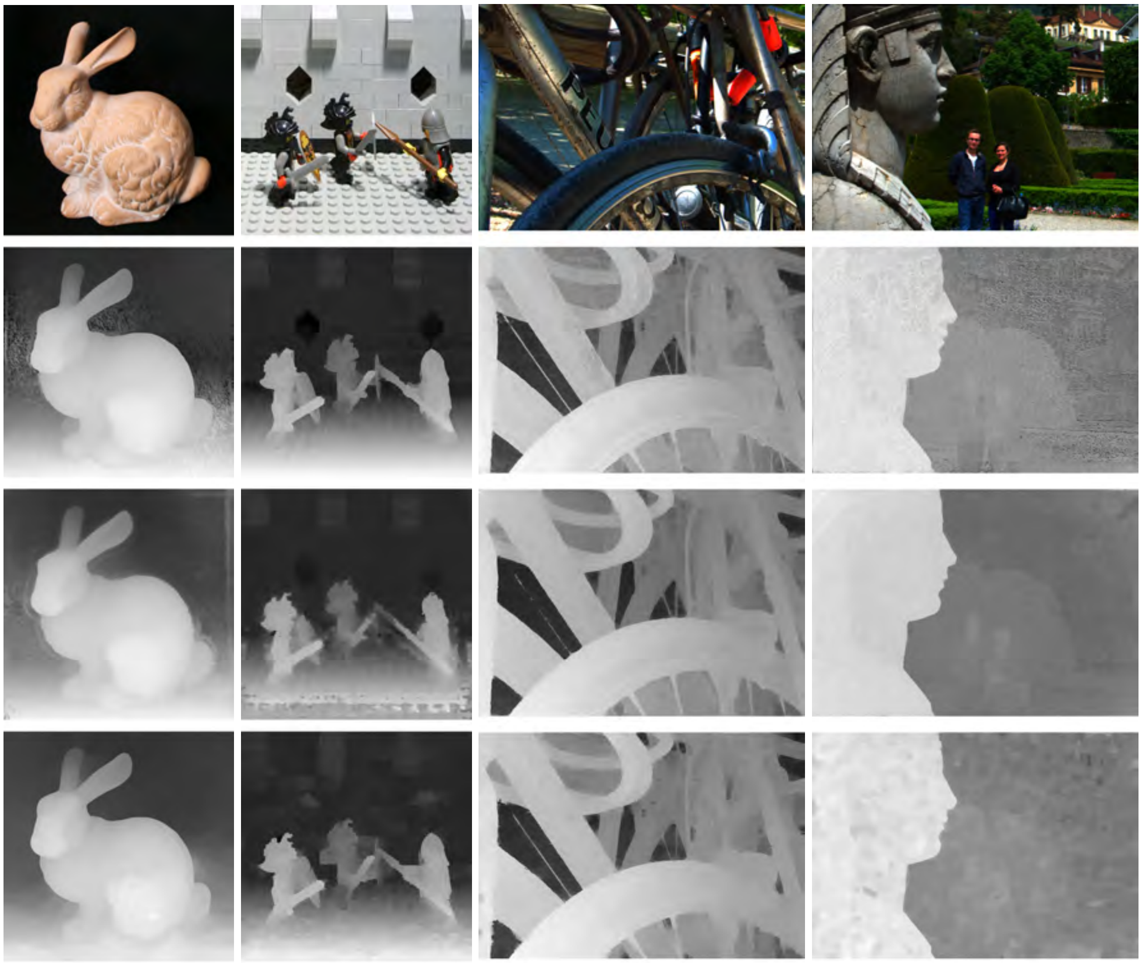


Figure 6.9: Real-world light fields from the Stanford (*left pair*) and EPFL (*right pair*) datasets. *Top to bottom:* central RGB view, Shi et al. [93], Jiang et al. [46], and our method. While our method has more bad pixels and can be sensitive in narrow baseline cases (*far right:* limitation Sphynx case), in general our method has equivalent or lower view consistency error, runs faster, and has no training data or pre-trained network dependency.

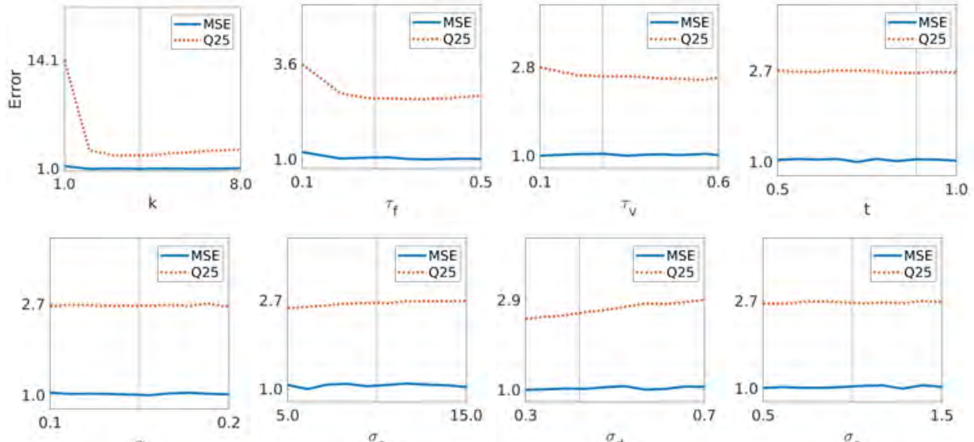


Figure 6.10: Effect of hyperparameter values on the MSE and Q25, averaged across the HCI dataset: k and τ_f (Equation 4.3), τ_v (Equation 4.4), t and α (entropy-based refinement), and σ_s, σ_d and σ_c (Equation 4.6). The vertical lines indicate our chosen values. The stochasticity of our algorithm means the chosen values may not be optimal in all cases. However, the method is stable to variation around these values.

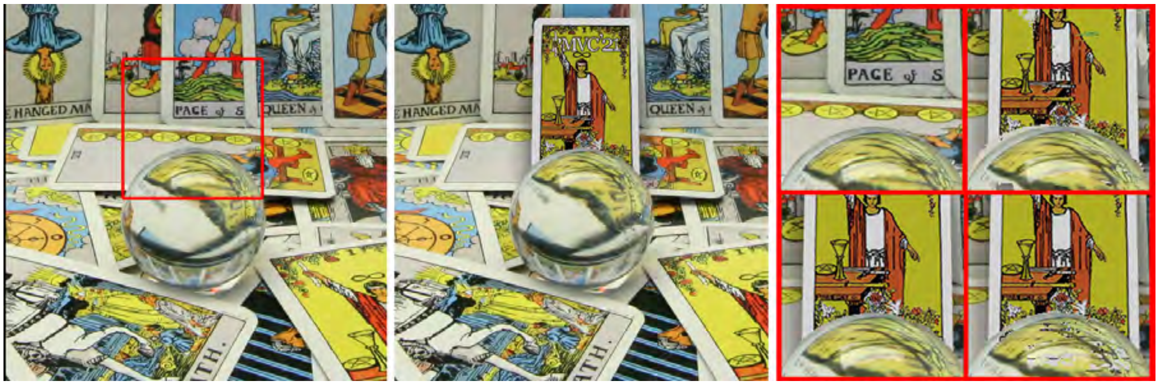


Figure 6.11: Adding an additional tarot card to the scene. *Left:* input scene. *Center:* Our editing results. *Right, clockwise from top-left*: Detail of the unmodified light field image, Zhang et al. [125]'s editing result, Shi et al. [93]'s editing result, and our result with fewer artifacts.

Chapter 7

A Differentiable Multi-view Edge Model

In the last chapter we saw that even though the MSE of our reconstruction is low it suffers from a high number of erroneous pixels. Thus, in this chapter we present a differentiable variant of our edge model which can be optimized via gradient descent. Furthermore, we validate the completeness of our depth encoding by using the differentiable variant to encode and then reconstruct a pre-computed depth map. Our differentiable representation allows us to optimize the output of the gradient-based diffusion process proposed in Chapter 6.

Our smooth reconstruction performs gradient-based diffusion using only a sparse set of depth labels in image space provided by multi-view edges. But, diffusion from noisy point samples may produce results with lower accuracy, and it can be difficult to identify and filter out noisy or erroneous points from a sparse set. However, given the correct noise-free constraints, diffusion can be shown to produce comparable or better results than state-of-the-art dense processing methods.

So, how can we handle noisy points? We present a method to optimize point constraints for a set of linear equations representing the solution to the standard Poisson problem of depth diffusion. For this, we develop a differentiable and occlusion-aware image-space representation of sparse multi-view edges that allows us to solve the inverse problem efficiently using gradient descent. We treat each edge point as a Gaussian to be splatted into the camera, and use the setting of radiative energy transfer through participating media to model the occlusion interaction between Gaussians. This method allows us to optimize over position, depth, and weight parameters per point, and to optimize the point set via reprojection error from multiple RGB images.

On synthetic and real-world data across narrow-baseline light field multi-view data, and with initial results on wider-baseline unstructured data, we show that our method reduces significant diffusion errors caused by noisy or spurious points. Further, we discuss why edges are difficult to optimize via reprojection from depth maps. Finally, in comparisons to both image processing

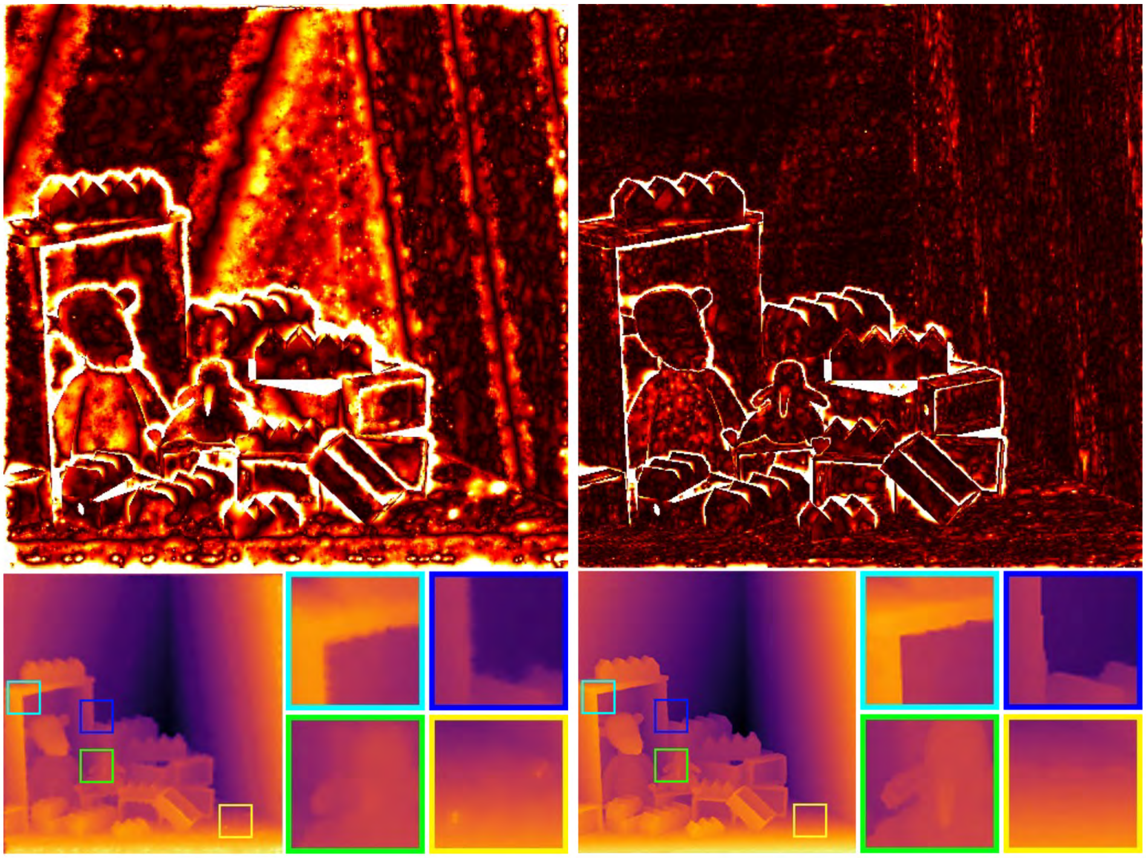


Figure 7.1: **Left:** Diffusion from an noisy point set produces significant errors from points found along RGB edges instead of depth edges (*top*, log absolute error). In the background, smooth depth regions have banding; in the foreground, RGB texture details are pulled into the depth causing outliers on the floor. **Right:** Our differentiable point optimization reduces error across the image, removing banding errors and minimizing texture pull.

and deep learning baselines, our method shows competitive performance especially in reducing bad pixels. Thus, we show the promise of direct optimization for diffusion-based dense depth estimation.

7.1 Depth via Differentiable Diffusion

Given a set of n multi-view images $\mathcal{I} = \{\mathrm{I}_0, \mathrm{I}_1, ..., \mathrm{I}_n\}$, and a sparse set of noisy scene points $\mathcal{P} \in \mathbb{R}^3$, our goal is to generate a dense depth map for central view I_c. To achieve this, we optimize the set of scene points so that their diffused depth minimizes a reprojection error across $\mathcal{I}$.

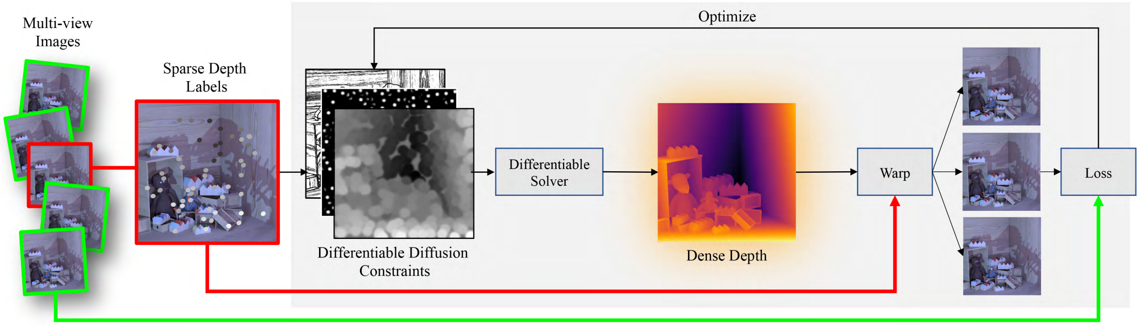


Figure 7.2: From a set of noisy sparse depth samples, our method uses differentiable splatting and diffusion to produce a dense depth map. Then, we optimize point position, disparity, and weight against an RGB reprojection loss. This reduces errors in the initial set of points.

7.1.1 Depth Diffusion

Let $\mathrm{S} \in \mathbb{R}^2$ denote the sparse depth labels obtained by projecting $\mathcal{P}$ onto the image plane of some $\mathrm{I} \in \mathcal{I}$. That is, for a given scene point $\mathbf{x} = (X_\mathbf{x}, Y_\mathbf{x}, Z_\mathbf{x}) \in \mathcal{P}$ and camera projection matrix K, $\mathrm{S}(\mathrm{K}\mathbf{x}) = Z_\mathbf{x}$. We wish to obtain a dense depth map D_o by penalizing the difference from the sparse labels S while also promoting smoothness by minimizing the gradient $\nabla\mathrm{D}$:

$$\mathrm{D}_o = \underset{\mathrm{D}}{\operatorname{argmin}} \iint_\Omega \lambda(x,y)\left(\mathrm{D}(x,y) - \mathrm{S}(x,y)\right)^2 + \vartheta(x,y)\|\nabla\mathrm{D}(x,y)\|\,dx\,dy, \tag{7.1}$$

where $\lambda(x,y) = \sum_{\mathbf{x}\in\mathcal{P}} \delta((x,y) - \mathrm{K}\mathbf{x})$ is a sum of point masses centered at the projection of $\mathcal{P}$—the *splatting* function. The second term enforces smoothness; ϑ is low around depth edges where it is desirable to have high gradients. Solving Equation (7.1) in 3D is expensive and complex, needing, e.g., voxels or a mesh. More practically, the energy in Equation (7.1) is minimized over a discrete pixel grid with indices x, y:

$$\begin{aligned}\mathrm{D}_o = \underset{\mathrm{D}}{\operatorname{argmin}} \sum_{(x,y)} \Bigg(& \lambda^{\mathbb{Z}}(x,y)\left(\mathrm{D}(x,y) - \mathrm{S}^{\mathbb{Z}}(x,y)\right)^2 \\ & + \sum_{(u,v)\in\mathcal{N}(x,y)} \vartheta^{\mathbb{Z}}(x,y)\|\mathrm{D}(u,v) - \mathrm{D}(x,y)\|\Bigg),\end{aligned} \tag{7.2}$$

where $\mathcal{N}(x,y)$ is a four-pixel neighborhood around (x,y), and $\lambda^{\mathbb{Z}}$, $\vartheta^{\mathbb{Z}}$ and $\mathrm{S}^{\mathbb{Z}}$ are the discrete counterparts of the splatting function λ, the local smoothness weight ϑ, and the depth label $\mathrm{S} \in \mathbb{R}^2$.

Deciding how to perform this discretization has important consequences for the quality of results and is not easy. For instance, λ and S are defined as point masses and hence are impossible to sample. The simplest solution is to round our projected point $\mathrm{K}\mathbf{x}$ to the nearest pixel. However, quite apart from the aliasing that this is liable to cause, it is unsuitable for optimization as the underlying representation of $\lambda^{\mathbb{Z}}$ and $\mathrm{S}^{\mathbb{Z}}$ remains non-differentiable. As Figure 7.3 shows, we require

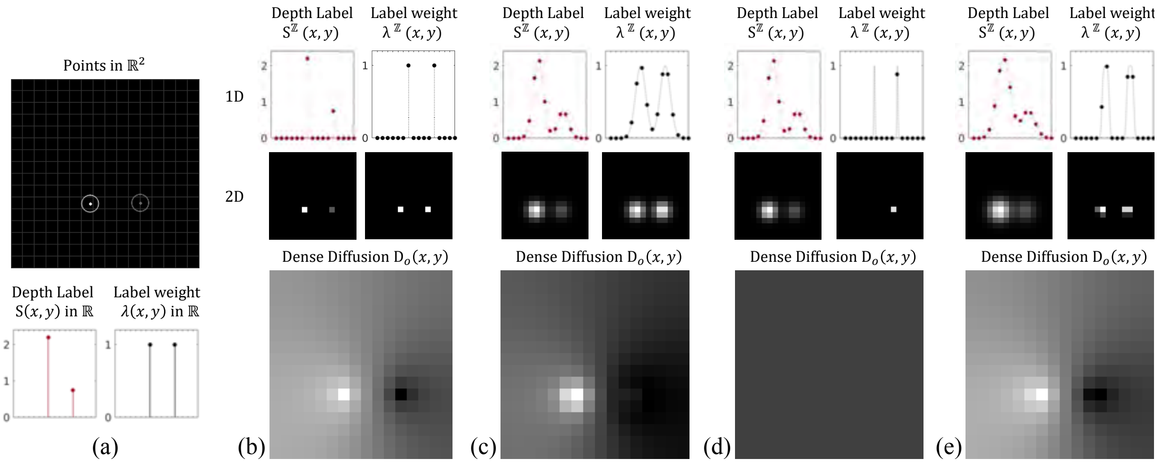


Figure 7.3: Depth diffusion happens in image space, so how we splat a set of scene points in $\mathbb{R}^3$ onto a pixel grid in $\mathbb{Z}^2$ has a significant impact on the results. **(a)** The image-space projection of scene points are Dirac delta functions which cannot be represented in discrete pixels. **(b)** Rounding the projected position to the closest pixel provides the most accurate splatting of depth labels for diffusion, even if it introduces position error. Unfortunately, the functional representation of the splatted point remains a non-differentiable Dirac delta. **(c)** Image-space Gaussians provide a differentiable representation but the depth labels are not accurate. Since the label weights $\lambda^{\mathbb{Z}}$ are no longer point masses, non-zero weight is assigned to off-center pixels. **(d)** Attempting to make $\lambda^{\mathbb{Z}}$ more similar to a point mass by reducing the Gaussian σ results in sub-pixel points vanishing: the Gaussian on the left no longer has extent over any of the sampled grid locations. **(e)** Our higher-order Gaussian representation provides dense diffusion results closest to **(a)** while being differentiable.

a representation that is differentiable and has the appropriate compactness for correctly representing the weight and depth value of each point on the raster grid: points projected to the raster grid should 'spread' their influence only where necessary for differentiability.

7.1.2 Differentiable Image-space Representation

A common smooth representation is to model the density $\mathbf{x}$ at a three-dimensional scene point as a sum of scaled isotropic Gaussians [81, 96]. The problem with this approach is that rendering all such points $\mathbf{x} \in \mathcal{P}$ requires either ray-marching through the scene, or representing the viewing-frustum as a voxel grid. The former is computationally expensive and the latter limits rendering resolution. Moreover, with points defined in scene space, it becomes difficult to ensure depth values are accurately splatted onto discrete pixels. This is demonstrated in Figure 7.3(e) where the scene point projecting onto a sub-pixel location ends up with zero pixel weight—effectively vanishing.

Our proposed representation overcomes these problems by modeling depth labels as scaled Gaussians centered at the 2D projection $\mathrm{K}\mathbf{x}$ of points $\mathbf{x} \in \mathcal{P}$, and using a higher-order Gaussian (or *super-Gaussian*) for the label weight to ensure non-zero pixel contribution from all points. A higher-order Gaussian is useful for representing weight as it has a flatter top, and falls off rapidly.

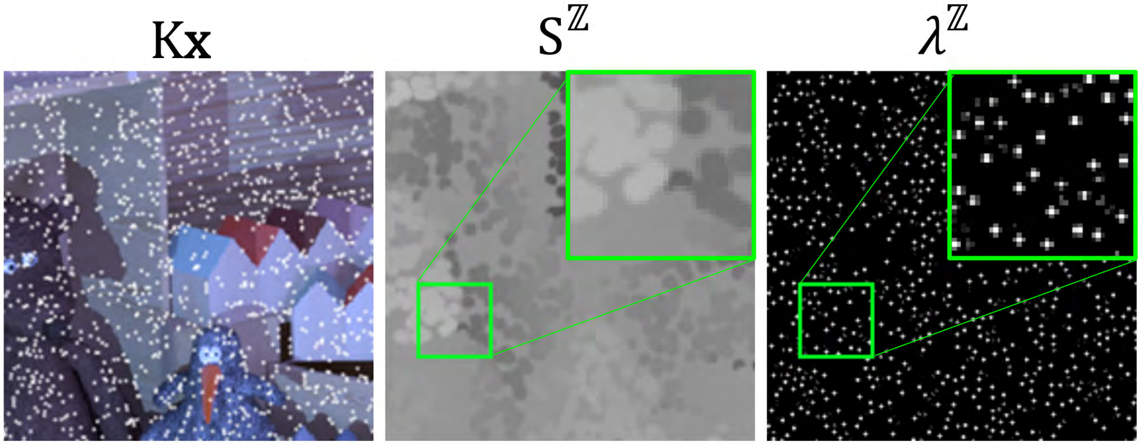


Figure 7.4: *Left:* The image-space projection $\mathrm{K}\mathbf{x}$ of scene points $\mathbf{x} \in \mathcal{P}$ plotted in white. *Middle:* Our differentiable labeling function $\mathrm{S}^{\mathbb{Z}}$ accurately splats depth labels while handling occlusion. *Right:* A higher-order Gaussian representation of $\lambda^{\mathbb{Z}}$ is differentiable, and provides weights that are close to point masses without any points vanishing during discretization.

Thus, its behavior is closer to that of a delta function and it helps minimize the "leakage" of weight onto neighboring pixels (Figure 7.3c). But unlike a delta, it is differentiable and can be sized so that points do not vanish (Figure 7.4d & 7.4e). We define the discrete functions:

$$\mathrm{S}^{\mathbb{Z}}(x,y) = \sum_{\mathbf{x}\in\mathcal{P}} \alpha_{\mathbf{x}}(x,y)\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y), \tag{7.3}$$

where $\alpha_{\mathbf{x}}(x,y)$ is a function that will merge projected labels in screen space (we will define $\alpha_{\mathbf{x}}$ in Sec. 7.1.3), and $\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}$ declares the label contribution at pixel (x,y) from a *single* scene point $\mathbf{x} = (X_{\mathbf{x}}, Y_{\mathbf{x}}, Z_{\mathbf{x}})$ with projection $\mathrm{K}\mathbf{x} = (x_{\mathbf{x}}, y_{\mathbf{x}})$. We define $\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}$ as:

$$\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y) = Z_{\mathbf{x}} \exp\bigg(-\frac{(x-x_{\mathbf{x}})^2 + (y-y_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2}\bigg). \tag{7.4}$$

Similarly, the discrete label weights are defined as:

$$\lambda^{\mathbb{Z}}(x,y) = \sum_{\mathbf{x}\in\mathcal{P}} \alpha_{\mathbf{x}}(x,y)\lambda^{\mathbb{Z}}_{\mathbf{x}}(x,y), \tag{7.5}$$

with $\lambda^{\mathbb{Z}}_{\mathbf{x}}$ taking the higher-order Gaussian form:

$$\lambda^{\mathbb{Z}}_{\mathbf{x}}(x,y) = w_{\mathbf{x}}\mathrm{exp}\bigg(-\frac{(x-x_{\mathbf{x}})^2 + (y-y_{\mathbf{x}})^2}{2\sigma_{\lambda}^2}\bigg)^p, \tag{7.6}$$

for some scaling factor $w_{\mathbf{x}}$.

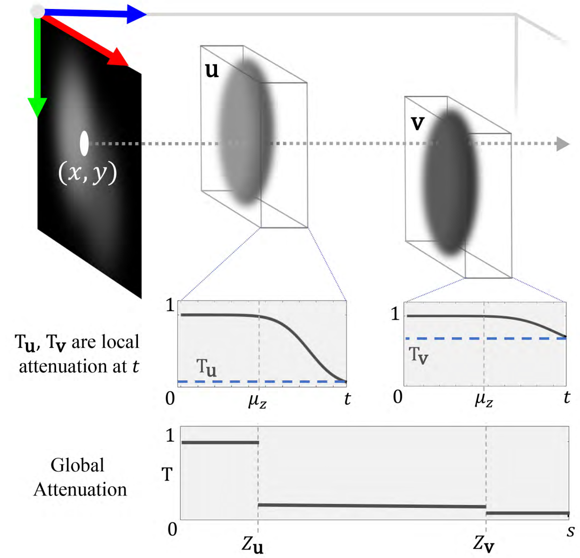


Figure 7.5: Depth labels at points overlapping in xy are estimated using a radiative transfer formulation with Gaussians in orthographic space. If σ_Z is small, the influence of the points $\mathbf{u}$ and $\mathbf{v}$ in scene space is restricted to small windows around $Z_\mathbf{u}$ and $Z_\mathbf{v}$. As $\sigma_Z \rightarrow 0$, we assume the density contribution at any point s along a ray comes from a single Gaussian. This allows the attenuation due to each Gaussian to be calculated independently. The global attenuation at s is the product of local attenuation for all points with $z < s$.

Discussion One might ask why we do not use higher-order Gaussians for the depth label, too. Depth labels require handling occlusion (unlike their weights), and we model this using radiance attenuation in the next section (Sec. 7.1.3). Using higher-order Gaussians for depth requires differentiating a transmission integral (upcoming Eq. (7.7)), yet no analytic form exists for higher-order Gaussians (with an isotropic Gaussian, a representation in terms of the *lower* incomplete gamma function γ is possible, but the derivative is still notoriously difficult to estimate).

7.1.3 Rendering and Occlusion Handling

While a Gaussian has infinite extent, the value of the depth label function $\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}$ and the label weight function $\lambda^{\mathbb{Z}}_{\mathbf{x}}$ at non-local pixels will be small and can be safely ignored. However, we need the operator $\alpha_\mathbf{x}$ from Equations (7.3) and (7.5) to accumulate values at any local pixel (x, y) that receives significant density contribution from multiple $S^{\mathbb{Z}}_{\mathbf{x}}$. This accumulation must maintain the differentiability of $\mathrm{S}^{\mathbb{Z}}$ and must ensure correct occlusion ordering so that an accurate depth label is splatted at (x, y). Using a Z-buffer to handle occlusion by overwriting depth labels and weights from back to front makes $\mathrm{S}^{\mathbb{Z}}$ non-differentiable.

We diffuse projected points in 2D; however, to motivate and illustrate the derivation of $\alpha_\mathbf{x}$, we will temporarily elevate our differentiable screen-space representation to $\mathbb{R}^3$ and use an orthographic projection—this provides the simplest 3D representation of our '2.5D' data labels and allows us to formulate $\alpha_{\mathbf{x}}$ using the tools of radiative energy transfer through participating media [81].

Thus, we model the density at every 3D scene point as a sum of scaled Gaussians of magnitude ρ centered at the orthographic reprojection $\mathbf{u} = (x_{\mathbf{x}}, y_{\mathbf{x}}, Z_{\mathbf{x}})$ of each $\mathbf{x} \in \mathcal{P}$. Then, for a ray originating at pixel (x, y) and traveling along z, the attenuation factor T at distance s from the image plane is defined as:

$$\mathrm{T}(x,y,s) = \exp\Bigg(-\int_0^s \rho \sum_{\mathbf{x}\in\mathcal{P}} \exp\Bigg(-\Big(\frac{(x-x_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} + \frac{(y-y_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} + \frac{(z-Z_{\mathbf{x}})^2}{2\sigma_Z^2}\Big)\Bigg)\, dz\Bigg). \tag{7.7}$$

As $\sigma_Z \to 0$, the density contribution at any point s along the ray will come from only a single Gaussian. Furthermore, as the contribution of each Gaussian is extremely small beyond a certain distance, and as the attenuation along a ray in empty space does not change, we can redefine the bounds of the integral in a local frame of reference. Thus, we consider each Gaussian as centered at μ_z in its local coordinate frame with non-zero density only on $[0, t]$ (Figure 7.5). The independence of Gaussians lets us split the integral over $[0, s]$ into a sum of integrals, each over $[0, t]$ (please see supplemental document for detailed derivation). Using the product rule of exponents, we can rewrite Equation (7.7) as:

$$\begin{aligned}\mathrm{T}(x,y,s) &= \prod_{\mathbf{x}} \exp\Bigg(-\int_0^t \rho \frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)}{Z_{\mathbf{x}}} \exp\Big(-\frac{(z-\mu_z)^2}{2\sigma_Z^2}\Big)\, dz\Bigg) \qquad (7.8)\\ &= \prod_{\mathbf{x}} \mathrm{T}_{\mathbf{x}}(x,y),\end{aligned}$$

where the product is over all $\mathbf{x} \in \mathcal{P} \mid Z_{\mathbf{x}} < s$. By looking again at Equation (7.4), we can see that $\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)/Z_{\mathbf{x}}$ is simply the normalized Gaussian density in xy. Each $\mathrm{T}_{\mathbf{x}}$ is independent, allowing parallel calculation:

$$\mathrm{T}_{\mathbf{x}}(x,y) = \exp\Bigg(\sqrt{\frac{\pi}{2}} \frac{\sigma_Z\, \rho\, \mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)}{Z_{\mathbf{x}}}\Bigg(-\operatorname{erf}\Big(\frac{\mu_z}{\sigma_Z\sqrt{2}}\Big) - \operatorname{erf}\Big(\frac{t-\mu_z}{\sigma_Z\sqrt{2}}\Big)\Bigg)\Bigg) \tag{7.9}$$

$$= \exp\Bigg(c\frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)}{Z_{\mathbf{x}}}\Bigg), \tag{7.10}$$

where erf is the error function. We can now define the label contribution of each $\mathbf{x}$ at pixel (x, y). For this, we use the radiative transfer equation which describes the behavior of light passing through a participating medium [81]:

$$\mathrm{S}^{\mathbb{Z}}(x,y) = \int_0^\infty \mathrm{T}(s,x,y)\mathrm{a}(s,x,y)\mathrm{P}(s,x,y)\, ds, \tag{7.11}$$

where T, a, and P are the transmittance, albedo, and density, respectively, at a distance s along a ray originating at (x, y). Albedo represents the proportion of light reflected towards (x, y), and intuitively, we may think of it as the color of the point seen on the image plane in the absence of any occlusion or shadows. In our case, we want the pixel value to be the depth label $Z_{\mathbf{x}}$. Making this substitution, and plugging in our transmittance and Gaussian density function, we obtain:

$$\mathrm{S}^{\mathbb{Z}}(x,y) = \int_0^{\infty} \mathrm{T}(x,y,s) \sum_{\mathbf{x}\in\mathcal{P}} Z_{\mathbf{x}}\, \rho \exp\left(-\frac{(x-x_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} + \frac{(y-y_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} + \frac{(s-Z_{\mathbf{x}})^2}{2\sigma_Z^2} \right) ds. \quad (7.12)$$

Again, with $\sigma_Z \to 0$, the density contribution at a given s may be assumed to come from only a single Gaussian. This lets us remove the summation over **x**, and estimate the integral by sampling s at step length $\mathrm{d}s$ over a small interval $\mathcal{N}_{\mathbf{x}}$ around each $Z_{\mathbf{x}}$:

$$\begin{aligned}
\mathrm{S}^{\mathbb{Z}}(x,y) &= \sum_{\mathbf{x}\in\mathcal{P}} \sum_{s\in\mathcal{N}_{\mathbf{x}}} \mathrm{d}s\, \mathrm{T}(x,y,s)\rho\, \mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y) \exp\left(-\frac{(s-Z_{\mathbf{x}})^2}{2\sigma_Z^2} \right) \\
&= \sum_{\mathbf{x}\in\mathcal{P}} \mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y) \sum_{s\in\mathcal{N}_{\mathbf{x}}} \mathrm{d}s\, \mathrm{T}(x,y,s)\; \rho \exp\left(-\frac{(s-Z_{\mathbf{x}})^2}{2\sigma_Z^2} \right) \\
&= \sum_{\mathbf{x}\in\mathcal{P}} \alpha_{\mathbf{x}}(x,y) \mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y). \qquad (7.13)
\end{aligned}$$

This allows us to arrive at a differentiable form of our screen-space aggregation function $\alpha_{\mathbf{x}}$:

$$\alpha_{\mathbf{x}}(x,y) = \frac{\rho\, \mathrm{d}s}{Z_{\mathbf{x}}} \sum_{s\in\mathcal{N}_{\mathbf{x}}} \mathrm{T}(s,x,y)\; \rho \exp\left(-\frac{(s-Z_{\mathbf{x}})^2}{2\sigma_Z^2} \right). \quad (7.14)$$

7.1.4 Optimization by Gradient Descent

To restate our goal, we want to optimize the parameters $\Theta = \{\mathrm{S}^{\mathbb{Z}}, \lambda^{\mathbb{Z}}, \vartheta^{\mathbb{Z}}\}$ for dense depth diffusion (Eq. (7.2)). The function $\mathrm{S}^{\mathbb{Z}}(x,y)$ proposes a depth label at pixel (x,y), $\lambda^{\mathbb{Z}}(x,y)$ determines how strictly this label is applied to the pixel, and $\vartheta^{\mathbb{Z}}(x,y)$ controls the smoothness of the output depth map at (x,y). We find Θ by using gradient descent to minimize a loss function $\mathrm{L}(\Theta)$. Using our differentiable representation, we can express $\mathrm{S}^{\mathbb{Z}}$ and $\lambda^{\mathbb{Z}}$ in terms of the image-space projection of the sparse point set $\mathcal{P}$. Doing so provides strong constraints on the initial value of these two functions and on how they are updated at each step, leading to faster convergence.

Supervised Loss To validate our image-space representation and optimization, we first use ground truth depth to supervise the optimization of the different parameters in Θ. This is effective and generates high-quality depth maps (Section 7.2.3). This shows the potential of our differentiable sparse point optimization and diffusion method, and inform us of the contribution on the final result of the follow self-supervised loss for captured images.

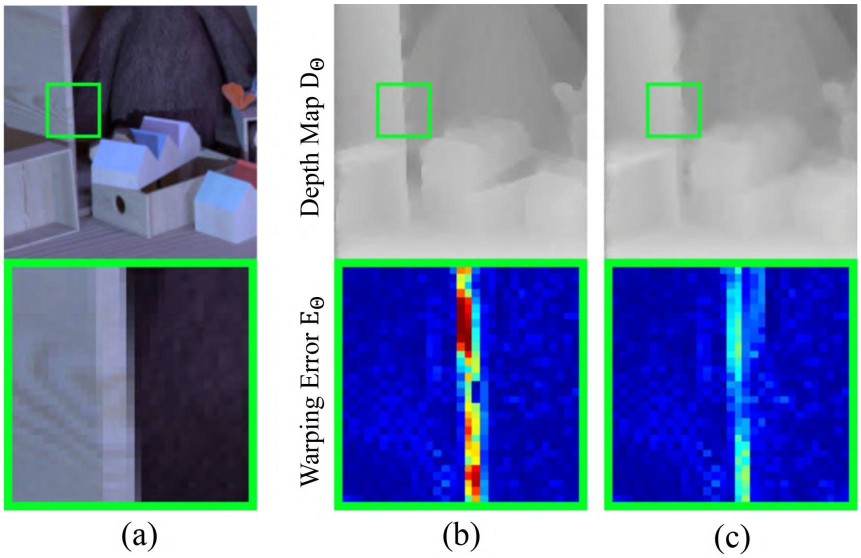


Figure 7.6: Our everyday intuition says that depth edges should be sharp, but a limited sampling rate blurs them in the RGB input **(a)**. This can cause unintended high error during optimization via RGB reprojection loss. In **(b)**, the depth edge is sharp, but reprojecting it into other views causes high error as the edge in the RGB image is blurred. Counterintuitively, in **(c)**, the depth edge is soft and less accurate, but leads to a lower reprojection error. If sharp edges are desired, we can reward high gradient edges in the error (Eq. (7.16)).

Self-supervised Loss Working with a set of multi-view images $\mathcal{I} = \{\mathrm{I}_0, \mathrm{I}_1, ..., \mathrm{I}_n\}$ allows us to define a self-supervised loss function for the optimization. Given a dense depth map D_Θ generated by diffusion with parameters Θ, we define the warping operator $\mathcal{W}_\Theta$ to reproject each view I^i onto I_c; where I_c is the view we want to compute dense depth for. The warping error is then given by:

$$\mathrm{E}_\Theta(x,y) = \frac{1}{\sum_i \mathrm{M}^i_\Theta(x,y) + \epsilon} \sum_i \Big(|\mathrm{I}(x,y) - \mathcal{W}_\Theta[\mathrm{I}^i](x,y)| \, \mathrm{M}^i_\Theta(x,y) \Big), \tag{7.15}$$

where $\mathrm{M}^i_\Theta(x,y)$ is the binary occlusion mask for view i, computed dynamically at each iteration.

We observe that E_Θ is non-zero even if we use the ground truth depth map, because small pixel errors are inevitable during the sub-pixel interpolation for warping. However, the more significant errors come from an unexpected source: the sharpness of depth edges. Depth labels are ambiguous at pixels lying on RGB edges, and limited sampling frequency blurs these edges within pixels (Figure 7.6). By assigning a fixed label to these pixels, sharp depth edges cause large errors. Consequently, the optimization process smooths all edges. While doing so minimizes the reprojection error, it may be desirable to have sharp depth edges for aesthetic and practical purposes, even if the edge location is slightly incorrect.

Therefore, we add a loss term to reward high gradients in E_Θ, effectively allowing the optimization to ignore errors caused by sharp depth edges. In addition, we include a smoothness term E_S similar to Ranjan et al. [79] to encourage depth to be guided by image edges, and a structural

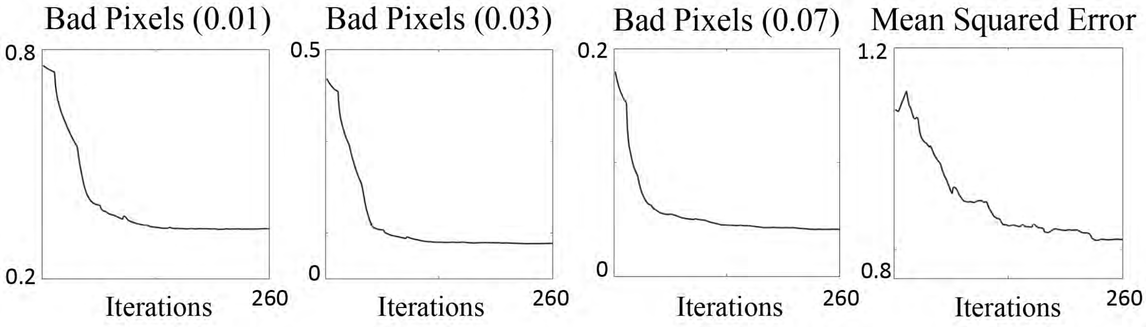


Figure 7.7: Over optimization iterations, mean squared error reduces and 'bad pixels' are significantly suppressed. From the corresponding error map in Figure **??** we can see that most remaining errors lie along edges, where depth is not well defined (as per Figure **??**)—the ground truth depth values are pixel-rounded while the RGB image contains blurred depth edges.

self-similarity error [110] $\mathrm{E}_{\mathrm{SSIM}}$ which is commonly used to regularize warping error. Our final loss function becomes:

$$\mathrm{L}(\Theta) = \sum_{(x,y)} \Big(\mathrm{E}_{\Theta}(x, y) + \mathrm{E}_{\mathrm{S}} + \mathrm{E}_{\mathrm{SSIM}} - \nabla \mathrm{E}_{\Theta}(x, y) \Big). \tag{7.16}$$

7.1.5 Implementation

Our proposed framing of the diffusion problem allows us to express $\mathrm{S}^{\mathbb{Z}}$ and $\lambda^{\mathbb{Z}}$ as differentiable functions of the points set $\mathcal{P}$, and thus, to calculate $\partial\, \mathrm{L} / \partial\, \mathbf{x}$. Since $\mathcal{P}$ provides strong constraints on the shape of these functions, we optimize over the parameters $\mathrm{Z}_{\mathbf{x}}$, $\mathrm{K}\mathbf{x}$, $w_{\mathbf{x}}$, and $\vartheta^{\mathbb{Z}}$ instead of directly over Θ ($w_{\mathbf{x}}$ is the scaling factor from Equation (7.6)). To regularize the smoothness and data weights, we further define $\vartheta^{\mathbb{Z}}(x, y) = \exp(-Q(x, y))$ and $w_{\mathbf{x}} = \exp(-R(\mathbf{x}))$, for some unconstrained R and Q that are optimized. Thus, our final parameter set is $\bar{\Theta} = \{\mathrm{Z}_{\mathbf{x}}, \mathrm{K}\mathbf{x}, R, Q\}$. We initialize $R(\mathbf{x})$ to zero for all $\mathbf{x}$, and $Q(x, y)$ to the magnitude of the image gradient $\|\nabla I\|$.

Distance Both RGB and VGG16 features can be used as distances for warping loss E_{Θ}; we found VGG16 features to outperform RGB. VGG loss has a better notion of space from a larger receptive field and handles textureless regions better. Thus, we take each warped image in Equation (7.15), run a forward pass through VGG16, then compute an L_1 distance between the 64 convolution activation maps of the first two layers. $\nabla \mathrm{E}_{\Theta}$ is computed using the 2D channel-wise mean of E_{Θ}; E_{S} and $\mathrm{E}_{\mathrm{SSIM}}$ are calculated in RGB space.

Hyperparameters This require a trade-off between resource use and accuracy. The parameter σ_{S} in Equation (7.3) determines the pixel area of a splatted depth label $Z_{\mathbf{x}}$. Ideally, we want the label

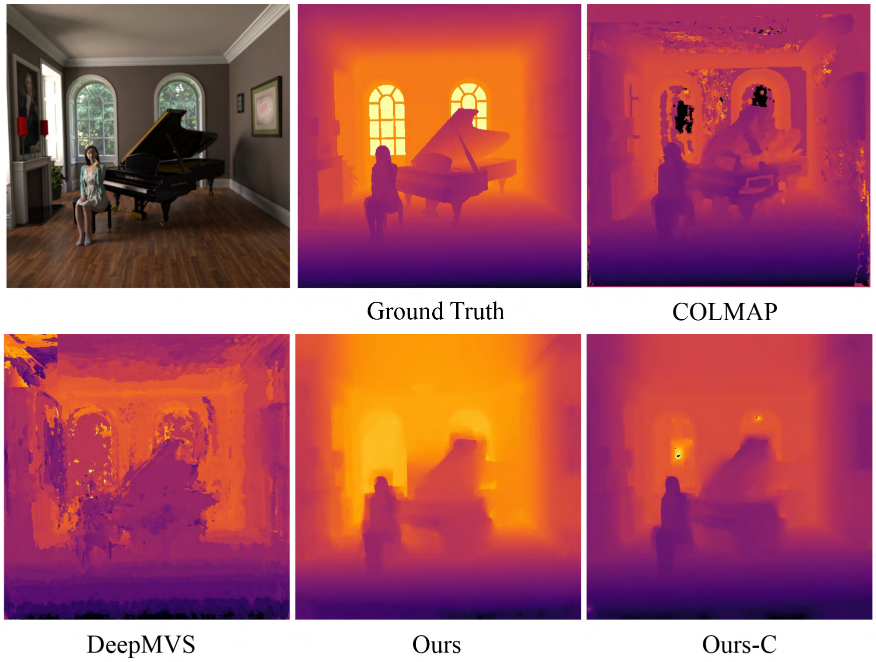


Figure 7.8: Depth results on the synthetic *Piano-MVS* scene. *Raster order:* One input image, ground truth, dense reconstruction from COLMAP [90, 89], DeepMVS [36], our method using 702 sparse points in $\mathcal{P}$, and our method with $2,808$ sparse points from dense COLMAP output.

to be $Z_\mathbf{x}$ over all pixels where $\lambda_\mathbf{x}^\mathbb{Z} > \epsilon$. The case where the label falls off while the weight is much larger than zero is illustrated in Figure 7.3(c) and leads to incorrect diffusion results. However, ensuring a uniform weight requires having a large value of σ_S, and this may cause the labels of neighboring points to be occluded. We found that using $\sigma_\mathrm{S} =$ 1.3 provides a good balance between accuracy and compactness. This spreads the label density over three pixels in each direction before it vanishes, so we use a Gaussian kernel size of 7×7.

For σ_Z, we want the spread to be as small as possible. However, if the value is very small then we must use a large number of samples in $\mathcal{N}_\mathbf{x}$ when calculating the quadrature in Equation (7.14). An insufficient number of samples causes aliasing when calculating $\alpha_\mathbf{x}$ at different pixel locations (x, y). A value of $\sigma_Z =$ 1.0 and 8 samples in each $\mathcal{N}_\mathbf{x}$ works well in practice.

We use a Gaussian of order $p =$ 2 to represent $\lambda_\mathbf{x}^\mathbb{Z}$ (Eq. (7.5)). As the order is increased, the Gaussian becomes more similar to a box function and leaks less weight onto neighboring pixels. However, its gradients become smaller, and the loss takes longer to converge. With $p =$ 2, we calculate $\sigma_\lambda =$ 0.71 to provide the necessary density to prevent points from vanishing (Fig. 7.3(d)).

Routine We use Adam [52]. We observe a lower loss when a single parameter is optimized at once. Thus, we optimize each parameter separately for 13 iterations, and repeat for 5 passes.

Efficiency The set of edge pixels require to represent a high-resolution image can run into the tens of thousands, and naively optimizing for this many points is expensive. This is true both of computation time and of memory. Calculating $\mathrm{S}^{\mathbb{Z}}$ in Equation (7.3) by summing over all points $\mathbf{x}$ is impossibly slow for any scene of reasonable complexity. Fortunately, in practice we only need to sum the contribution from a few points $\mathbf{x}$ at each pixel and, so, the computation of $\alpha_{\mathbf{x}}(x, y)$ in Equation (7.14) is serialized by depth only for points in a local neighborhood. By splitting the image plane into overlapping tiles, non-local points $\mathrm{K}\mathbf{x}$ can be rendered in parallel. The amount of overlap equals the kernel size in xy, and is needed to account for points that may lie close to the boundary in neighboring tiles. Using this parallelization scheme, we can render more than 50k points in correct depth order, solve the diffusion problem of Equation (7.2), and back-propagate gradients through the solver and renderer in five seconds.

Software and Hardware We implement our method in PyTorch. For diffusion, we implement a differentiable version of Szeliski's LAHBPCG solver [100]. All CPU code was run on an AMD Ryzen Threadripper 2950X 16-Core Processor, and GPU code on an NVIDIA GeForce RTX 2080Ti.

7.2 Experiments

7.2.1 Light Fields

Datasets Along with the *Dino*, *Sideboard*, *Cotton*, and *Boxes* scenes from the synthetic HCI Dataset [35], we add two new *Living Room* and *Piano* scenes with more realistic lighting, materials, and depth ranges. We path trace these with Arnold in Maya. All synthetic light fields have 9×9 views, each of 512 × 512 pixels. For real-world scenes, we use light fields from the New Stanford Light Field Archive [56]. These light fields have 17×17 views captured from a camera rig, with a wider baseline and high spatial resolution (we downsample 2× for memory).

Baselines and Metrics For our method, we use an initial point set extracted from EPI edge filters [49]. We compare to the methods of Zhang et al. [125], Jiang et al. [46], Shi et al. [93], Li et al. [58] and our own non-differentiable version from Chapter 6 (Ours-N). The former three methods are deep-learning-based. For metrics, we use mean-squared error (MSE) and *bad pixels* (BP). BP measures the percentage of pixels with error higher than a threshold. For real-world scenes without ground truth depth, we provide a measure of performance as the reprojection error in LAB induced by depth-warping the central view onto the corner views; please see our supplemental material.

Table 7.1: Quantitative results for wider-baseline unstructured five-camera cases, as the *Living Room-MVS* and *Piano-MVS* scenes.

	MSE				Q25			
	D-MVS	COLMAP	Ours	Ours-C	D-MVS	COLMAP	Ours	Ours-C
Living Room-MVS	1.99	1.37	0.30	0.17	64.9	4.44	14.8	4.22
Piano-MVS	1.51	2.56	0.81	0.69	6.87	42.60	2.15	1.37
Average	1.75	1.97	0.56	0.43	35.9	23.50	8.48	2.80

Results While learning-based methods [46, 93, 58] tend to do well on the HCI dataset, their quantitative performance degrades on the more difficult *Piano* and *Living Room* scenes (Table 7.2). A similar qualitative trend shows the learning-based methods performing worse than diffusion on the real-world light fields (Fig. 7.9). Our method provides more consistent overall performance on all datasets. Moreover, our non-differentiable diffusion-based method has few pixels with very large errors but many pixels with small errors, producing consistently low MSE but more bad pixels. In contrast, our differentiable method consistently places in the top-three on the bad pixel metrics. We show additional results and error maps in our supplemental material. Finally, as is common, it is possible to post-process our results with a weighted median filter to reduce MSE (e.g., *Dino* 0.54 vs. 0.86) at the expense of increased bad pixels (BP(0.01) of 39.6 vs. 25.6).

7.2.2 Multi-view Stereo

Datasets We path trace *Living Room-MVS* and *Piano-MVS* datasets at 512×512. Each scene has five unstructured views with a mean baseline of approximately 25cm.

Baselines and Metrics We compare to dense reconstruction from COLMAP [90] and to DeepMVS [36]. Out method uses the sparse output of COLMAP as the initial point set, which is considerably sparser than the initial set for light fields (500 vs. 50k). To increase the number of points, we diffuse a preliminary depth map and optimize the smoothness parameter for 50 iterations. Then, we sample this result at RGB edges. Using this augmented set, we optimize all parameters in turns of 25 iterations, repeated 5 times. In addition, we also evaluate a variant of our method, Ours-C, with sparse labels initialized from the dense COLMAP output at RGB edges.

For metrics, we again use MSE, and also report the 25th percentile of absolute error as Q25. As the depth output of each method is ambiguous up to a scale, we estimate a scale factor for each result using a least squares fit to the ground truth at 500 randomly sampled valid depth pixels.

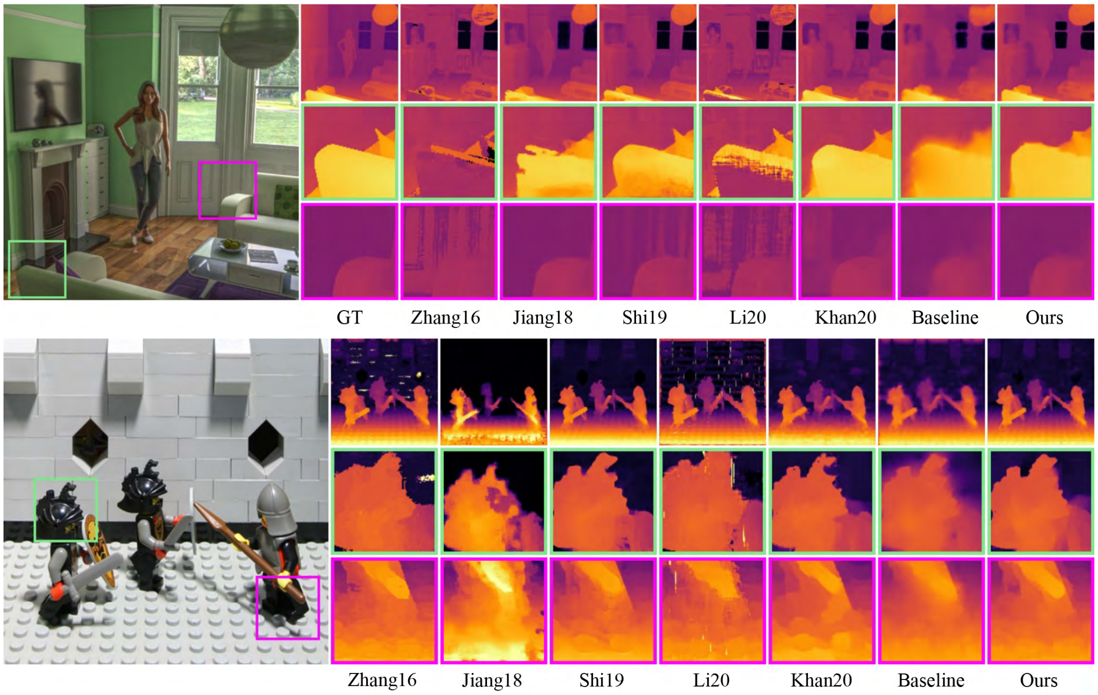


Figure 7.9: *Top:* Disparity results on the synthetic *Living Room* light field. *Bottom:* Disparity results on a real light field. *Left to right*: Zhang et al. [125], Jiang et al. [46], Shi et al. [93], Li et al. [58], Khan et al. [?], a baseline diffusion result without any optimization, our results and finally, for the top light field, ground truth.

Table 7.2: **(Best viewed in color)** Quantitative comparison on synthetic light fields. The top three results are highlighted in gold, silver and bronze. BP(x) is the number of *bad pixels* which fall above threshold x in error. Higher BP thresholds are used for *Living Room* and *Piano* as their average error is larger: they contain specularities, larger depth ranges, and path-tracing noise.

Light Field	MSE * 100						BP(0.1)						BP(0.3)						BP(0.7)					
	[125]	[58]	[46]	[93]	Ours-N	Ours	[125]	[58]	[46]	[93]	Ours-N	Ours	[125]	[58]	[46]	[93]	Ours-N	Ours	[125]	[58]	[46]	[93]	Ours-N	Ours
Living Room	0.67	0.57	0.23	0.25	0.25	0.20	59.5	58.5	37.2	48.0	47.2	30.3	43.3	42.7	23.7	26.5	25.0	17.5	17.0	16.6	11.4	10.8	11.5	9.23
Piano	26.7	13.7	14.4	8.66	12.7	8.71	36.7	27.5	24.7	27.0	37.6	17.0	25.0	17.6	13.6	11.4	20.0	7.93	5.33	4.13	5.88	4.29	4.95	3.49
Light Field	MSE * 100						BP(0.01)						BP(0.03)						BP(0.07)					
	[125]	[58]	[46]	[93]	Ours-N	Ours	[125]	[58]	[46]	[93]	Ours-N	Ours	[125]	[58]	[46]	[93]	Ours-N	Ours	[125]	[58]	[46]	[93]	Ours-N	Ours
Sideboard	1.02	1.89	1.96	1.12	0.89	2.23	78.0	62.3	47.4	53.0	73.8	43.0	42.0	18.0	18.3	20.4	37.4	16.5	14.4	6.50	9.31	9.02	16.2	8.35
Dino	0.41	3.28	0.47	0.43	0.45	0.86	81.2	52.7	29.8	43.0	69.4	25.6	48.9	12.8	8.81	13.1	30.9	7.69	7.52	5.82	3.59	4.32	10.4	4.06
Cotton	1.81	1.95	0.97	0.88	0.68	3.07	75.4	58.8	25.4	38.6	56.2	31.1	34.8	14.0	6.30	9.60	18.0	7.82	4.35	4.11	2.02	2.74	4.86	4.06
Boxes	7.90	4.67	11.6	8.48	6.69	9.17	84.7	68.3	51.8	66.5	76.8	60.3	55.3	28.0	27.0	37.2	47.9	32.7	18.9	13.4	18.3	21.9	28.3	20.5

Results To account for the error in least squares, Table 7.1 presents the minimum of ten different fits for each method. Both DeepMVS and COLMAP generate results with many invalid pixels. We assign such pixels the mean GT depth. Our method outperforms the baselines with a sparse point set (≈ 700 points) and generates smooth results by design that have fewer artifacts (Fig. 7.8). Using $4\times$ as many initial points ($\approx 2,800$ points) in the Our-C variant leads to additional improvements.

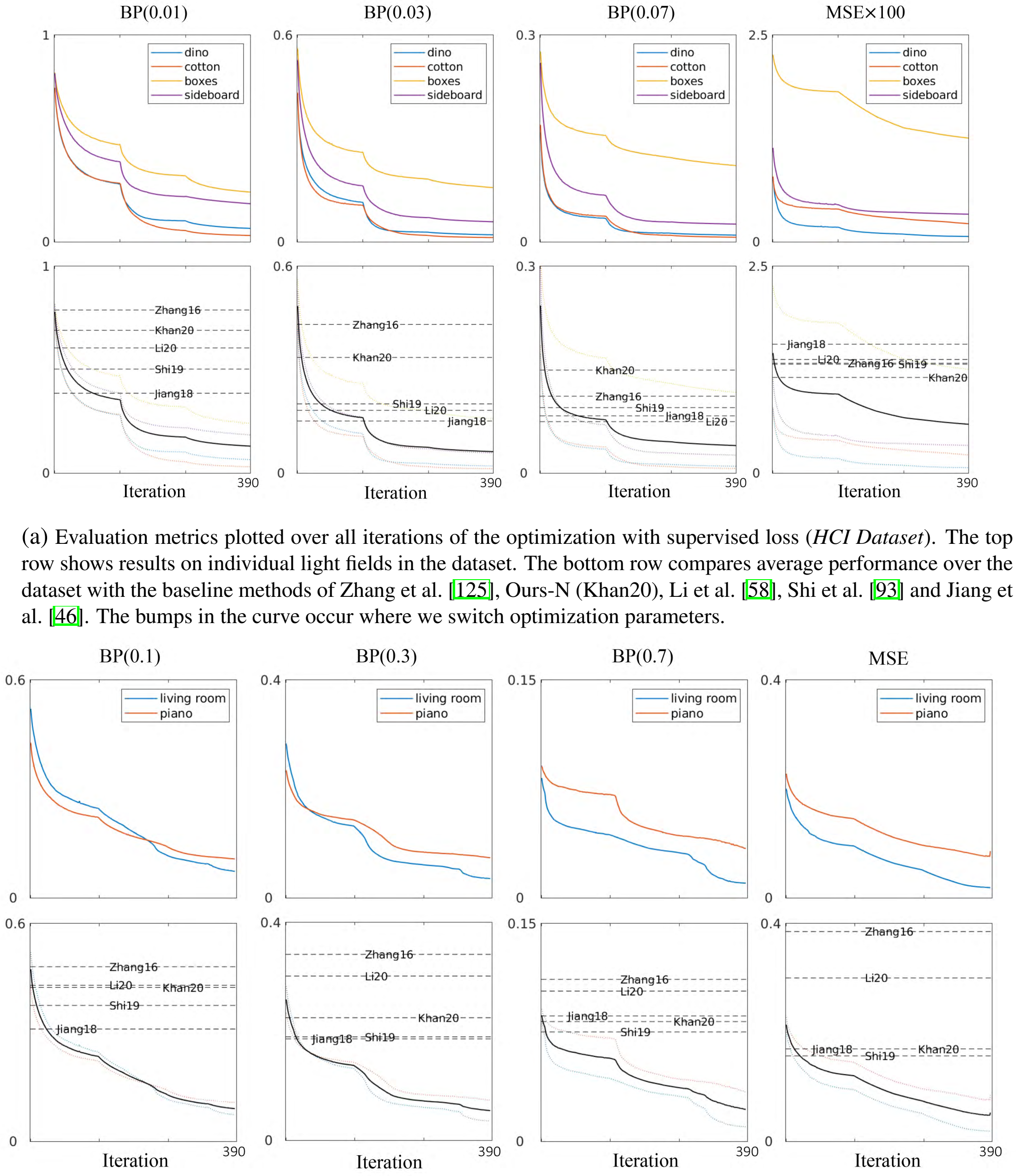
7.2.3 Supervised Loss

We describe a validation of our differentiable rendering and diffusion approach using a supervised loss against ground truth data. In Figure 7.10, we present MSE and bad pixel metrics over iterations of the optimization for both the HCI dataset and our new *living room* and *piano* realistic scenes. For comparison, we also mark the performance of five existing methods. In Figure 7.11, this experiment shows that our approach can produce errors close to zero, and validates the potential of such an approach in the best case.

Still, why do the errors not reduce to zero? As Figure 7.11 shows, the presence of outliers in the original point set and the lack of labels in regions with fine detail prevents the diffusion from entirely eliminating errors. Such sources of error occur in all six light fields to varying degrees.

7.3 Summary

We present a method to differentiably render and diffuse a sparse depth point set so that we can directly optimize the parameters of our multi-view edge model via a multi-view RGB reprojection loss. While we recover dense depth maps, our approach can be interpreted as a point denoiser for diffusion, as related to volume rendering via radiative transfer, or as a kind of differentiable depth-image-based rendering. We discuss higher-order weighting term design choices that make this possible, demonstrate our method's ability to reduce error in bad pixels, and discuss why remaining errors are difficult to optimize via reprojection from depth maps. In comparisons to both image processing and deep learning baselines, our method shows competitive performance, especially in reducing bad pixels. Moreover, by using the method to encode and then decode a precomputed depth map, we establish the completeness of our multi-view depth model.



(a) Evaluation metrics plotted over all iterations of the optimization with supervised loss (*HCI Dataset*). The top row shows results on individual light fields in the dataset. The bottom row compares average performance over the dataset with the baseline methods of Zhang et al. [125], Ours-N (Khan20), Li et al. [58], Shi et al. [93] and Jiang et al. [46]. The bumps in the curve occur where we switch optimization parameters.

(b) Evaluation metrics for the *Piano* and *Living Room* light fields with supervised loss.

Figure 7.10: We validate our image-space representation and optimization using ground truth depth to supervise the optimization using the same routine as Section 3.5. This generates high-quality results on all four evaluation metrics (MSE is plotted on a logarithmic scale). This confirms the potential of our differentiable sparse point optimization and diffusion method.

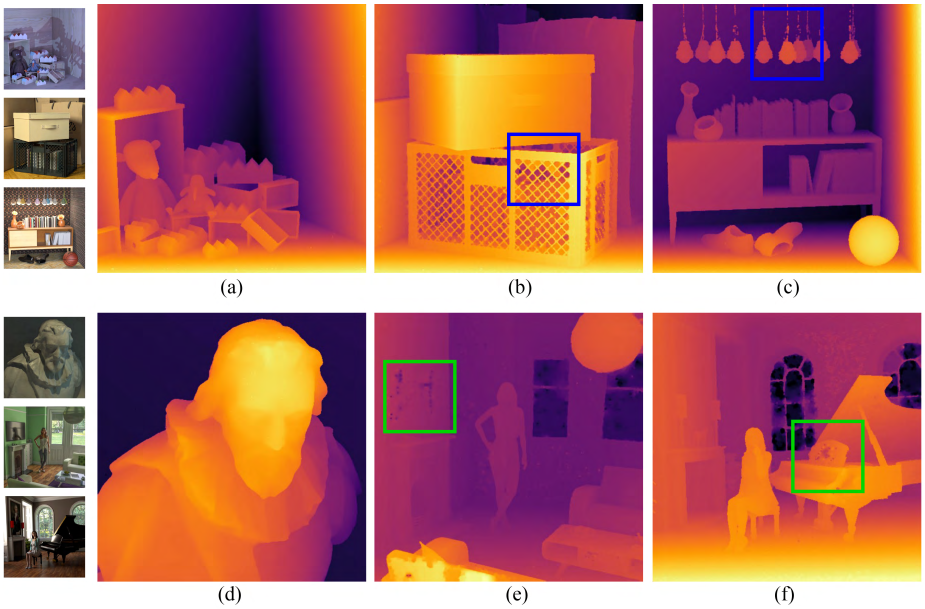


Figure 7.11: Disparity maps generated by our method using supervised loss. **(a)–(f)**: *Boxes*, *Dino*, *Sideboard*, *Cotton*, *Living Room* and *Piano*. The presence of outliers in the original point set (green boxes), and the lack of labels in regions with fine detail (blue boxes) prevents the diffusion from entirely eliminating errors. Such sources of error occur in all six light fields to varying degrees.

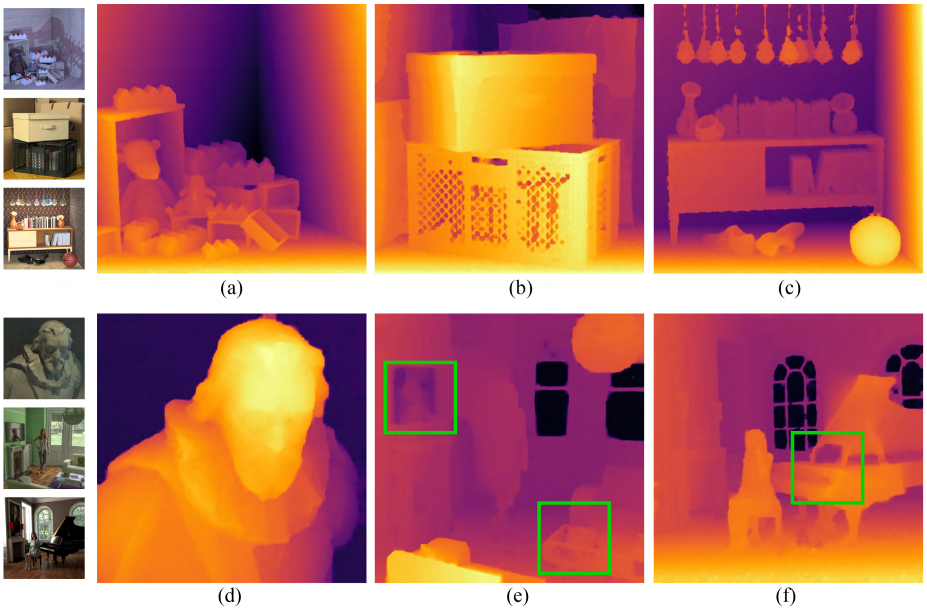


Figure 7.12: Our disparity maps with self-supervised loss. As expected, specular surfaces (green boxes) are especially difficult to label correctly with a self-supervised reprojection loss; Figure C.3 shows that all methods suffer in these areas.

Chapter 8

Conclusion

This dissertation showed that multi-view edges encode all relevant information for supporting higher level computational photography tasks that rely on depth reconstruction. This was demonstrated by proposing a representation for multi-view depth that satisfies the criteria of concision, explicitness, and completeness in particular. Given that different applications are variously dependent on the quality dimensions of depth reconstruction, we defined completeness for computational photography task using occlusion edges and view-consistency in addition to the more commonly used accuracy metric. Inspired by the work of Elder [25], the proposed representation was formulated as a sparse model based on multi-view edges. We demonstrated how the parameters of the model can be estimated from a structured light field and from unstructured multi-view images.

The proposed representation has explicitness in that it provides more useful information than the multi-view RGB input. This includes edges, occlusion surfaces and depth boundaries. The representation has concision as it stores depth labels for each edge pixel only once, and exploits a smoothness assumption to deal with traditionally noise-prone areas such as specular and texture-free regions. Finally, the ability of the model to capture all relevant information for higher level tasks — its completeness — is demonstrated through two application-specific methods for inverting the edge code and retrieving a dense 2D depth map. In addition, a variant of the edge model is presented that is differentiable and can be optimized via gradient descent guided by a multi-view reprojection loss. We observe that the reconstructed dense depth is better than, or comparable to, state-of-the-art methods on each of the three metrics: accuracy, occlusion-edge localization and view-consistency. This provides strong evidence that multi-view depth edges are indeed complete with respect to computational photography tasks.

8.1 Discussion

The proposed model may be considered from two points of view: first, like Elder's [25] representation, our model provides a concise, explicit, and complete encoding of multi-view depth. It represents the implicit structural information in multi-view images explicitly. It can also be used to provide a concise representation of a pre-computed 2D depth map with minimal loss of information (Section 7.2.3). In addition, since we also propose methods for extracting sparse depth labels from multi-view RGB input directly, the model, coupled with the reconstruction algorithms, describes a multi-view depth estimation approach.

8.1.1 Depth Encoding

While the results in Section 7.2.3 would *indicate* the completeness of our model, they are not claimed as *proof* of completeness since they fail to achieve a perfect reconstruction. However, this may be attributed to certain implementational constraints which limit the model's ability to encode and reconstruct all depth features. While the optimization process proposed in Section 7.1.4 can discard noisy labels, it cannot add new labels to the original point set $\mathcal{P}$. Our evaluation is, thus, limited by our original choice of edges: a good set of multi-view edges may be more complete than a bad one; an empty set will never be complete. Put another way, the null hypothesis that multi-view edges are incomplete is invalidated by a statistically significant, not necessarily perfect, result. Since we do not establish statistical significance, we settle for the weaker claim of *indicating* completeness. In fact, the lack of robust experimental design mechanisms and rigorous proofs for establishing the significance of results is a problem that hounds the computer vision community in general, where focus is more on practical improvements as gauged by established datasets. This may be an area for researchers to address as the field matures. More immediately for our purposes, an avenue of future research would be to allow the differentiable representation to add samples, thereby, also optimizing the edge set.

As a depth encoding, our model has applications in light field compression algorithms [65, 45]. The property of conciseness requires that any redundant information in the input should be discarded. While the proposed model does achieve this, it by no means discards *all* redundant information. That is, the correlation between elements of the model is never zero. Nonetheless, the model does expose this residual redundancy in a manner that is highly conducive to compression as the parameters of neighboring edge pixels are likely to be highly correlated [25]. Thus, while we have described the explicitness of our method in terms of edges, occlusion surfaces and depth boundaries above, we may also say that our method makes the redundancy of the input explicit.

8.1.2 Depth Estimation

In Section 7.1.5 we observed the limitations of an RGB reprojection loss in supervising depth reconstruction. In addition to blurring edges (Figure 7.6), such a loss is unable to reconstruct specular surfaces, assigning them instead to an incorrect virtual depth plane that minimizes reprojection error. This is true of any correspondence based depth estimation method [90, 89] including those based on optical flow. However, while incorrect in geometric terms, the virtual plane proves useful for simulating view-dependent effects in novel view synthesis [69, 26]. Thus, it may be argued that the RGB reprojection error is used as a fourth metric — in addition to accuracy, view-consistency, and occlusion edges — to quantify the completeness of our representation. However, this would expand the scope of the representation beyond scene depth since, by this definition, the ground truth depth is itself not complete. Hence, we do not include it.

As we discuss below, our method is not designed to handle non-Lambertian effects such as refraction. We do observe, however, that since non-Lambertian surfaces violate the EPI linearity assumption, our multi-view edge estimation pipeline (Section 4.1.5) can reliably discard them as outliers. Such a surface is then assigned a depth value diffused from the closest non-refractive edge, leading to fronto-parallel surface reconstruction. This effect is observed in the reconstruction of the crystal orb in the right-most column of Figure B.9. Thus, by using the smoothness assumption, our method is able to fall back on a piece-wise planar reconstruction for refractive surfaces.

Smoothness can be a good assumption for surfaces in general; for instance, many indoor scenes have low texture walls and adhere to the basic assumption that diffusion implies. However, this is less true of outdoor or natural scenes which have more textural detail. In such cases, the quality of the reconstructed depth is strongly dependent on the ability of the multi-view edge detector to capture high frequency — and potentially low amplitude — details. In addition, since smoothness relies on diffusion from neighbors, it fails to deal with the so-called “island problem” where a value must be propagated across depth boundaries [51]. This can be observed in the grill in the middle row of B.6. This failure does not, however, imply that our model is incomplete. The depth of a featureless “island” cannot be ambiguously determined by any method, and our estimate is as accurate (or inaccurate) as any other. Imagine looking at a textureless surface through a small hole — it is impossible to uniquely determine the depth of the textureless surface using stereo cues alone. Since the depth information of such an “island” region is not encoded in the input, it cannot be retrieved without using some prior on the surface configurations.

8.1.3 Non-Lambertian Surfaces

Non-Lambertian surfaces have traditionally proven to be a banana skin for surface reconstruction algorithms. Our multi-view edge detector (Section 4.1) relies on the regularity of EPI structure induced by Lambertian surfaces and is, thus, unable to work with specular or refractive surfaces. Ben-Ezra and Nayar [7] present a model-based algorithm for recovering the shape of transparent parametric surfaces under certain simplifying constraints. For more general settings, the problem is severely under-constrained, and learnt priors may offer a viable solution.

8.1.4 Light Field View Sampling

The presence of a smooth edge in an EPI depends on a sufficiently high sampling rate of light field views; a low sampling rate will generate highly aliased edges which can be hard to detect. Chaiet al. [13] and Mildenhallet al. [69] have studied the sampling rate requirements of light fields in the context of novel view synthesis. Their work provides explicit sampling guidelines for alias free view synthesis by quadr-ilinear sampling. Extending these results to our EPI edge detector to get an analytic bound for the sampling rate is not straight-forward. However, Figure A.4 shows that our method is fairly robust to low sampling rates.

We formulate our multi-view edge model with a focus on structured light fields; that is, light fields with views uniformly sampled along a straight line. Nonetheless, the model can also be applied to irregularly sampled light fields as long as the views lie on a straight line and the baselines are known. In this latter case, we can create a sparse EPI where only a subset of pixel rows are valid, but which nonetheless preserves the regular linear structure required for our edge detector to work.

8.2 Future Work

Future work could seek to incorporate data-based priors to improve the quality of multi-view edges for light fields (Chapter 4.1). While we found that EPI edge detection using a pre-trained neural network fails to achieve the requisite accuracy, it may be possible to learn EPI features for our proposed Prewitt filters to operate on rather than using LAB color space features as we presently do. This idea of using learnt features in a traditional algorithm—edge detection in our case— is similar in spirit to Wiles's [117] rendering of learnt point cloud features.

Furthermore, the differentiable representation of our multi-view edge model and diffusion-based smooth reconstruction method (Chapter 7) can be used in a CNN to learn model parameters directly from multi-view RGB images rather than optimizing them via gradient descent. The former approach has the potential to be much faster.

Finally, as the field of computational photography evolves and new applications are introduced, it may be necessary to update and evaluate the list of metrics we have proposed to quantify completeness. Future applications will likely place novel demands on multi-view depth reconstruction.

Bibliography

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[2] Edward H. Adelson and James R. Bergen. *The plenoptic function and the elements of early vision*, volume 2. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of Technology, 1991.

[3] A. Alperovich, O. Johannsen, and B. Goldluecke. Intrinsic light field decomposition and disparity estimation with a deep encoder-decoder network. 2018.

[4] Horace B Barlow et al. Possible principles underlying the transformation of sensory messages. *Sensory communication*, 1(01), 1961.

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

[6] Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. Frequency bias in neural networks for input of non-uniform density. In *International Conference on Machine Learning*, pages 685–694. PMLR, 2020.

[7] Moshe Ben-Ezra and Shree K Nayar. What does motion reveal about transparency? In *Computer Vision, IEEE International Conference on*, volume 3, pages 1025–1025. IEEE Computer Society, 2003.

[8] Pravin Bhat, Larry Zitnick, Michael Cohen, and Brian Curless. Gradientshop: A gradient-domain optimization framework for image and video filtering. In *ACM Transactions on Graphics (TOG)*, 2009.

[9] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, David Kriegman, and Ravi Ramamoorthi. Deep 3d capture: Geometry and reflectance from sparse multi-view images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[10] Robert C Bolles, H Harlyn Baker, and David H Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International journal of computer vision*, 1(1):7–55, 1987.

[11] Neill D. F. Campbell, George Vogiatzis, Carlos Hernandez, and Roberto Cipolla. Automatic object segmentation from calibrated images. In *Proceedings of the Conference for Visual Media Production*, CVMP, pages 126–137. IEEE Computer Society, 2011.

[12] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, June 1986.

[13] Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic sampling. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 307–318, 2000.

[14] Jie Chen, Junhui Hou, Yun Ni, and Lap-Pui Chau. Accurate light field depth estimation with superpixel regularization over partially occluded regions. *IEEE Transactions on Image Processing*, 27(10):4889–4900, 2018.

[15] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. *Advances in neural information processing systems*, 29:730–738, 2016.

[16] Zhao Chen, Vijay Badrinarayanan, Gilad Drozdov, and Andrew Rabinovich. Estimating depth from rgb and sparse sensing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 167–182, 2018.

[17] Xinjing Cheng, Peng Wang, and Ruigang Yang. Depth estimation via affinity learned with convolutional spatial propagation network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 103–119, 2018.

[18] Jaesung Choe, Sunghoon Im, Francois Rameau, Minjun Kang, and In So Kweon. Volumefusion: Deep depth fusion for 3d scene reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16086–16095, 2021.

[19] Inchang Choi, Orazio Gallo, Alejandro Troccoli, Min H Kim, and Jan Kautz. Extreme view synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7781–7790, 2019.

[20] A. Chuchvara, A. Barsi, and A. Gotchev. Fast and accurate depth estimation from sparse light fields. *IEEE Transactions on Image Processing*, 29:2492–2506, 2020.

[21] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996.

[22] Peng Dai, Yinda Zhang, Zhuwen Li, Shuaicheng Liu, and Bing Zeng. Neural point cloud rendering via multi-plane projection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7830–7839, 2020.

[23] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 145–156, 2000.

[24] Maximilian Diebold and Bastian Goldluecke. Epipolar plane image refocusing for improved depth estimation and occlusion handling. 2013.

[25] James H Elder. Are edges incomplete? *International Journal of Computer Vision*, 34(2-3):97–122, 1999.

[26] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2367–2376, 2019.

[27] David Forsyth and Jean Ponce. *Computer vision: A modern approach.* Prentice hall, 2011.

[28] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[29] Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F Cohen. The Lumigraph. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54. ACM, 1996.

[30] Hyunho Ha, Seung-Hwan Baek, Giljoo Nam, and Min H. Kim. Progressive acquisition of svbrdf and shape in motion. *Computer Graphics Forum*, 2020.

[31] Matthieu Hog, Neus Sabater, and Christine Guillemot. Light field segmentation using a ray-based graph structure. In *ECCV*, 2016.

[32] Matthieu Hog, Neus Sabater, and Christine Guillemot. Superrays for efficient light field processing. *IEEE Journal of Selected Topics in Signal Processing*, 11(7):1187–1199, 2017.

[33] Matthieu Hog, Neus Sabater, and Christine Guillemot. Dynamic super-rays for efficient light field video processing. In *BMVC*, 2018.

[34] Aleksander Holynski and Johannes Kopf. Fast depth densification for occlusion-aware augmented reality. 37(6), 2018.

[35] Katrin Honauer, Ole Johannsen, Daniel Kondermann, and Bastian Goldluecke. A dataset and evaluation methodology for depth estimation on 4d light fields. In *Asian Conference on Computer Vision*, pages 19–34. Springer, 2016.

[36] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deep-MVS: Learning multi-view stereopsis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2821–2830, 2018.

[37] Hayato Ikoma, Cindy M. Nguyen, Christopher A. Metzler, Yifan Peng, and Gordon Wetzstein. Depth from defocus with learned optics for imaging and occlusion-aware depth estimation. *IEEE International Conference on Computational Photography (ICCP)*, 2021.

[38] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2462–2470, 2017.

[39] Saif Imran, Yunfei Long, Xiaoming Liu, and Daniel Morris. Depth coefficients for depth completion. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12438–12447. IEEE, 2019.

[40] Aaron Isaksen, Leonard McMillan, and Steven J Gortler. Dynamically reparameterized light fields. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 297–306, 2000.

[41] Shahram Izadi, Richard A Newcombe, David Kim, Otmar Hilliges, David Molyneaux, Steve Hodges, Pushmeet Kohli, Jamie Shotton, Andrew J Davison, and Andrew Fitzgibbon. Kinectfusion: real-time dynamic 3d surface reconstruction and interaction. In *ACM SIGGRAPH 2011 Talks*, pages 1–1. 2011.

[42] Adrian Jarabo, Belen Masia, Adrien Bousseau, Fabio Pellacini, and Diego Gutierrez. How do people edit light fields? *ACM Transactions on Graphics (SIGGRAPH 2014)*, 33(4), 2014.

[43] Adrian Jarabo, Belen Masia, and Diego Gutierrez. Efficient propagation of light field edits. In *In Proc. of SIACG'11*, pages 75–80, 2011.

[44] Hae-Gon Jeon, Jaesik Park, Gyeongmin Choe, Jinsun Park, Yunsu Bok, Yu-Wing Tai, and In So Kweon. Accurate depth map estimation from a lenslet light field camera. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1547–1555, 2015.

[45] Xiaoran Jiang, Mikael Le Pendu, and Christine Guillemot. Light field compression using depth image based view synthesis. In *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 19–24. IEEE, 2017.

[46] Xiaoran Jiang, Mikaël Le Pendu, and Christine Guillemot. Depth estimation with occlusion handling from a sparse set of light field views. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 634–638. IEEE, 2018.

[47] Xiaoran Jiang, Jinglei Shi, and Christine Guillemot. A learning based depth estimation framework for 4d densely and sparsely sampled light fields. In *Proceedings of the 44th International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2019.

[48] Neel Joshi and C Lawrence Zitnick. Micro-baseline stereo. *Technical Report MSR-TR-2014–73*, page 8, 2014.

[49] Numair Khan, Qian Zhang, Lucas Kasser, Henry Stone, Min Hyuk Kim, and James Tompkin. View-consistent 4d light field superpixel segmentation. In *International Conference on Computer Vision (ICCV) 2019*. IEEE, 2019.

[50] Changil Kim, Henning Zimmer, Yael Pritch, Alexander Sorkine-Hornung, and Markus H Gross. Scene reconstruction from high spatio-angular resolution light fields. *ACM Trans. Graph.*, 32(4):73–1, 2013.

[51] Incheol Kim and Min H Kim. Non-local haze propagation with an iso-depth prior. In *International Joint Conference on Computer Vision, Imaging and Computer Graphics*, pages 213–238. Springer, 2017.

[52] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[53] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *European conference on computer vision*, pages 82–96. Springer, 2002.

[54] Johannes Kopf, Kevin Matzen, Suhib Alsisan, Ocean Quigley, Francis Ge, Yangming Chong, Josh Patterson, Jan-Michael Frahm, Shu Wu, Matthew Yu, Peizhao Zhang, Zijian He, Peter Vajda, Ayush Saraf, and Michael Cohen. One shot 3d photography. 39(4), 2020.

[55] J. Ku, A. Harakeh, and S. L. Waslander. In defense of classical image processing: Fast depth completion on the cpu. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 16–22, 2018.

[56] Stanford Graphics Laboratory. The new stanford light field archive, 2008.

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

[58] Kunyuan Li, Jun Zhang, Rui Sun, Xudong Zhang, and Jun Gao. Epi-based oriented relation networks for light field depth estimation. *British Machine Vision Conference*, 2020.

[59] Qinbo Li and Nima Khademi Kalantari. Synthesizing light field from a single image with variable mpi and two network fusion. *ACM Transactions on Graphics*, 39(6), 12 2020.

[60] Rui Li and Wolfgang Heidrich. Hierarchical and view-invariant light field segmentation by maximizing entropy rate on 4d ray graphs. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019.

[61] Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu, and William T Freeman. Learning the depths of moving people by watching frozen people. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4521–4530, 2019.

[62] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. 38(4):65:1–14, July 2019.

[63] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM Transactions on Graphics (TOG)*, 39(4):71–1, 2020.

[64] Ziyang Ma, Kaiming He, Yichen Wei, Jian Sun, and Enhua Wu. Constant time weighted median filtering for stereo matching and beyond. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 49–56, 2013.

[65] Marcus Magnor and Bernd Girod. Data compression for light-field rendering. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(3):338–343, 2000.

[66] Abhimitra Meka, Christian Haene, Rohit Pandey, Michael Zollhoefer, Sean Fanello, Graham Fyffe, Adarsh Kowdle, Xueming Yu, Jay Busch, Jason Dourgarian, Peter Denny, Sofien Bouaziz, Peter Lincoln, Matt Whalen, Geoff Harvey, Jonathan Taylor, Shahram Izadi, Andrea Tagliasacchi, Paul Debevec, Christian Theobalt, Julien Valentin, and Christoph Rhemann. Deep reflectance fields - high-quality facial reflectance field inference from color gradient illumination. volume 38, July 2019.

[67] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[68] Hajime Mihara, Takuya Funatomi, Kenichiro Tanaka, Hiroyuki Kubo, Yasuhiro Mukaigawa, and Hajime Nagahara. 4d light field segmentation with spatial and angular consistencies. In *Proceedings of the International Conference on Computational Photography (ICCP)*, 2016.

[69] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.

[70] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pages 405–421. Springer, 2020.

[71] Giljoo Nam, Joo Ho Lee, Hongzhi Wu, Diego Gutierrez, and Min H. Kim. Simultaneous acquisition of microscale reflectance and normals. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2016)*, 35(6), 2016.

[72] Peer Neubert and Peter Protzel. Superpixel benchmark and comparison. In *Proc. Forum Bildverarbeitung*, volume 6, 2012.

[73] In Kyu Park, Kyoung Mu Lee, et al. Robust light field depth estimation using occlusion-noise aware data costs. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2484–2497, 2017.

[74] Jeong Joon Park, Richard Newcombe, and Steve Seitz. Surface light field fusion. In *2018 International Conference on 3D Vision (3DV)*, pages 12–21. IEEE, 2018.

[75] Eric Penner and Li Zhang. Soft 3d reconstruction for view synthesis. 36(6), 2017.

[76] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[77] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. *ArXiv preprint*, 2021.

[78] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[79] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J Black. Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12240–12249, 2019.

[80] Martin Rerabek and Touradj Ebrahimi. New light field image dataset. In *8th International Conference on Quality of Multimedia Experience (QoMEX)*, number CONF, 2016.

[81] Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. A versatile scene model with differentiable visibility applied to generative pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 765–773, 2015.

[82] Christian Richardt, Carsten Stoll, Neil A. Dodgson, Hans-Peter Seidel, and Christian Theobalt. Coherent spatiotemporal filtering, upsampling and rendering of RGBZ videos. *Computer Graphics Forum (Proceedings of Eurographics)*, 31(2), May 2012.

[83] Christian Richardt, James Tompkin, and Gordon Wetzstein. *Capture, Reconstruction, and Representation of the Visual Real World for Virtual Reality*, pages 3–32. Springer International Publishing, Cham, 2020.

[84] Gernot Riegler and Vladlen Koltun. Free view synthesis. In *European Conference on Computer Vision*, pages 623–640. Springer, 2020.

[85] Gernot Riegler and Vladlen Koltun. Stable view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12216–12225, 2021.

[86] Mark A Ruzon and Carlo Tomasi. Color edge detection with the compass operator. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference On.*, volume 2, pages 160–166. IEEE, 1999.

[87] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002.

[88] Alexander Schick, Mika Fischer, and Rainer Stiefelhagen. Measuring and evaluating the compactness of superpixels. In *Proceedings of the 21st international conference on pattern recognition (ICPR2012)*, pages 930–934. IEEE, 2012.

[89] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[90] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[91] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 1, pages 519–528. IEEE, 2006.

[92] Jonathan Shade, Steven Gortler, Li wei He, and Richard Szeliski. Layered depth images. pages 231–242, July 1998.

[93] Jinglei Shi, Xiaoran Jiang, and Christine Guillemot. A framework for learning depth from a flexible subset of dense and sparse light field views. *IEEE Transactions on Image Processing*, 28(12):5867–5880, 2019.

[94] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[95] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020.

[96] Carsten Stoll, Nils Hasler, Juergen Gall, Hans-Peter Seidel, and Christian Theobalt. Fast articulated motion tracking using a sums of gaussians body model. In *2011 International Conference on Computer Vision*, pages 951–958. IEEE, 2011.

[97] David Stutz. Superpixel segmentation: An evaluation. In Juergen Gall, Peter Gehler, and Bastian Leibe, editors, *Pattern Recognition*, volume 9358 of *Lecture Notes in Computer Science*, pages 555 – 562. Springer International Publishing, 2015.

[98] David Stutz, Alexander Hermans, and Bastian Leibe. Superpixels: an evaluation of the state-of-the-art. *Computer Vision and Image Understanding*, 166:1–27, 2018.

[99] Murali Subbarao and Gopal Surya. Depth from defocus: A spatial domain approach. *International Journal of Computer Vision*, 13(3):271–294, 1994.

[100] Richard Szeliski. Locally adapted hierarchical basis preconditioning. In *ACM SIGGRAPH 2006 Papers*, pages 1135–1143. 2006.

[101] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020.

[102] Michael W Tao, Sunil Hadap, Jitendra Malik, and Ravi Ramamoorthi. Depth from combining defocus and correspondence using light-field cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 673–680, 2013.

[103] James Tompkin, Samuel Muff, James McCann, Hanspeter Pfister, Jan Kautz, Marc Alexa, and Wojciech Matusik. Joint 5d pen input for light field displays. In *The 28th Annual ACM Symposium on User Interface. Software and Technology, UIST'15*, November 2015.

[104] Ivana Tosic and Kathrin Berkner. Light field scale-depth space transform for dense depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 435–442, 2014.

[105] Vision.middlebury.edu. Middlebury stereo evaluation.

[106] Ting-Chun Wang, Manmohan Chandraker, Alexei A. Efros, and Ravi Ramamoorthi. Svbrdf-invariant shape and reflectance estimation from light-field cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[107] Ting-Chun Wang, Alexei A Efros, and Ravi Ramamoorthi. Occlusion-aware depth estimation using light-field cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3487–3495, 2015.

[108] Ting-Chun Wang, Alexei A Efros, and Ravi Ramamoorthi. Depth estimation with occlusion modeling using light-field cameras. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2170–2181, 2016.

[109] Tsun-Hsuan Wang, Fu-En Wang, Juan-Ting Lin, Yi-Hsuan Tsai, Wei-Chen Chiu, and Min Sun. Plug-and-play: Improve depth estimation via sparse data propagation. *arXiv preprint arXiv:1812.08350*, 2018.

[110] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[111] Sven Wanner and Bastian Goldluecke. Globally consistent depth labeling of 4d light fields. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 41–48. IEEE, 2012.

[112] Sven Wanner, Stephan Meister, and Bastian Goldluecke. Datasets and benchmarks for densely sampled 4d light fields. In *VMV*, pages 225–226, 2013.

[113] Sven Wanner, Christoph Straehle, and Bastian Goldluecke. Globally consistent multi-label assignment on the ray space of 4d light fields. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[114] Silvan Weder, Johannes Schonberger, Marc Pollefeys, and Martin R. Oswald. Routedfusion: Learning real-time depth map fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[115] David Weikersdorfer, David Gossow, and Michael Beetz. Depth-adaptive superpixels. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*, pages 2087–2090. IEEE, 2012.

[116] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy. High performance imaging using large camera arrays. In *ACM SIGGRAPH 2005 Papers*, pages 765–776. 2005.

[117] Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. Synsin: End-to-end view synthesis from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7467–7477, 2020.

[118] Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. Deep view synthesis from sparse photometric images. *ACM Transactions on Graphics (TOG)*, 38(4):1–13, 2019.

[119] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. *European Conference on Computer Vision (ECCV)*, 2018.

[120] Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent mvsnet for high-resolution multi-view stereo depth inference. *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[121] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. 38(6), November 2019.

[122] Kaan Yucer, Changil Kim, Alexander Sorkine-Hornung, and Olga Sorkine-Hornung. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 249–257. IEEE, 2016.

[123] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020.

[124] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape-from-shading: a survey. *IEEE transactions on pattern analysis and machine intelligence*, 21(8):690–706, 1999.

[125] Shuo Zhang, Hao Sheng, Chao Li, Jun Zhang, and Zhang Xiong. Robust depth estimation for light field via spinning parallelogram operator. *Computer Vision and Image Understanding*, 145:148–159, 2016.

[126] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph.*, 37(4), July 2018.

[127] Hao Zhu, Qi Zhang, and Qing Wang. 4D light field superpixel and segmentation. In *IEEE CVPR*, 2017.

Appendix A

Expanded Results for Piece-Wise Constant Reconstruction via 4D Superpixels

We present additional information and experimental results to better characterize the performance of the tested methods:

Section A.1 Parameter variation of weight of clustering features in the k-means baseline and in our method. This shows that increasing intensity and color weight increases boundary performance but decreases view consistency performance.

Section A.2 We show performance trends of all tested methods as the baseline of the light field increases and the number of views decreases.

Section A.3 Per dataset metric scores for the HCI dataset, rather than the average presented in the main paper. These show the relative characteristics of the different scenes.

A.1 Parameter Variation

The baseline k-means-based segmentation method clusters each pixel in the central view based on a vector $f = (x, y, d, L^*, a^*, b^*)$ of spatial, depth, and CIELAB color features. For our evaluation in the paper, each feature was assigned the same weight parameters as used in the spatio-angular segmentation stage of our method. Figures A.1-A.3 explore the effect of varying the feature weights on the accuracy, compactness, and consistency metrics for our method and the k-means baseline.

For reference, results for the LFSP [127] algorithm using both ground-truth (LFSP-GT) and Wang et al.'s [107, 108] (LFSP-Wang) method for disparity estimation are shown on each plot.

For the results presented in Section 5.2.4, we chose a spatial feature weight of 1 to increase accuracy. At this level, our superpixels are approximately as compact as those of LFSP. However, overall, at this weight our superpixel boundaries are sometimes less smooth than LFSP; here, compactness is not a sufficient metric to describe the differences in boundary curvature.

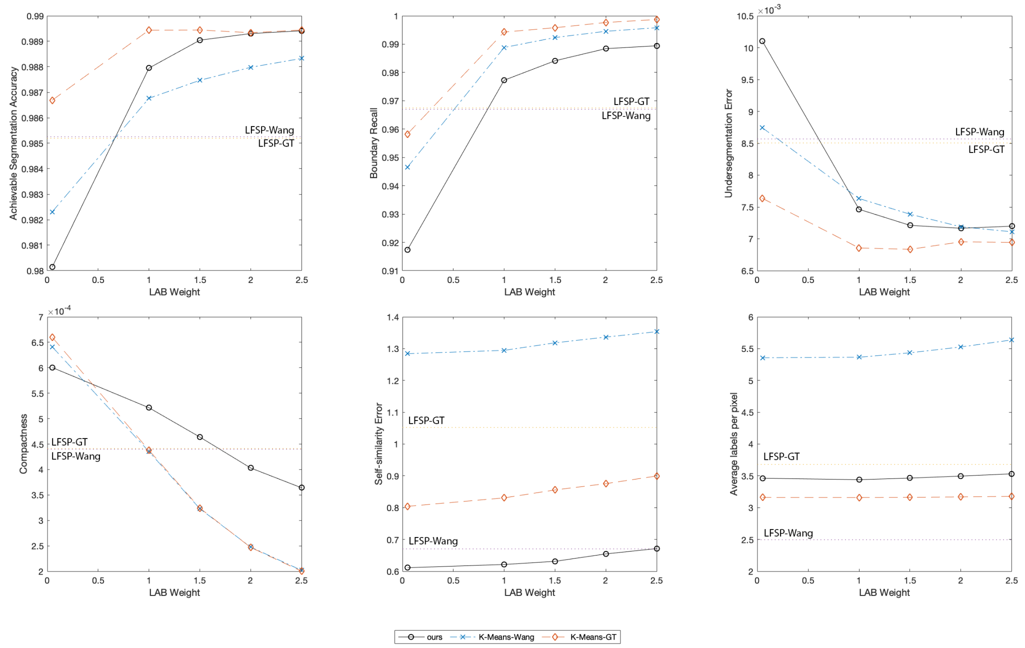
As spatial feature weight ω_{xy} increases, our superpixels become increasingly compact and so more regular in shape (bottom left, Fig. A.2 (a)). At comparable compactness to LFSP, we have improved boundary recall and achievable accuracy, and lower undersegmentation error. For view consistency we have lower self similarity error and fewer labels per pixel.

A.2 Large disparity light fields

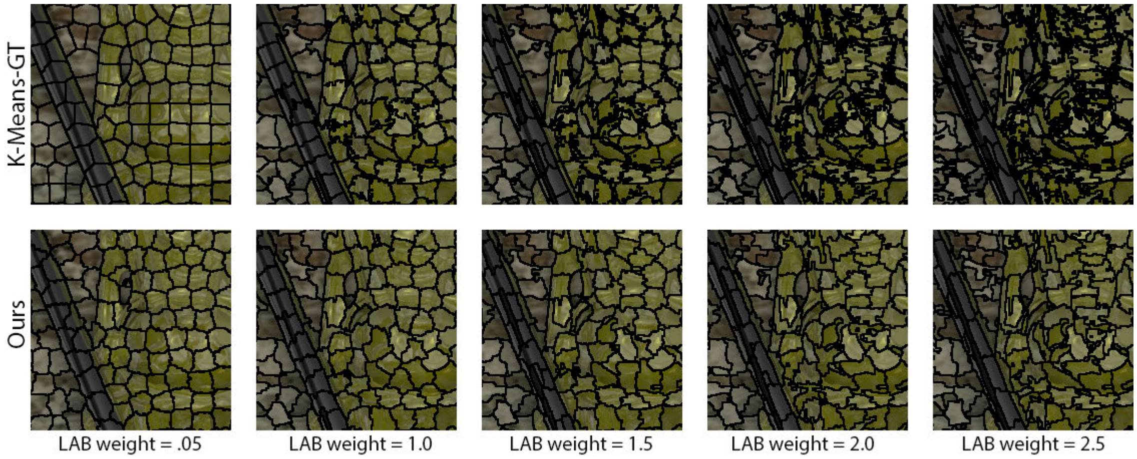
We incrementally remove views from the HCI light fields (superpixel size $= 20$, $\omega_{xy} = 1$; Fig. A.4). The performance of our method follows the same trend as LFSP and k-means, while meeting our goal of greater view-consistency.

A.3 HCI Per-Scene Quantitative Metrics

For completeness, we include the per-scene qualitative measures on the HCI dataset across Figures A.5 to A.7. We can see varying complexity across the datasets, e.g., papillon has relatively easier boundaries, while horses has difficult boundary segmentation with text in the background. Different techniques also perform better or worse on different datasets, e.g., our approach does well on still life, but less well on papillon due to our weaker regularization for smooth untextured regions.

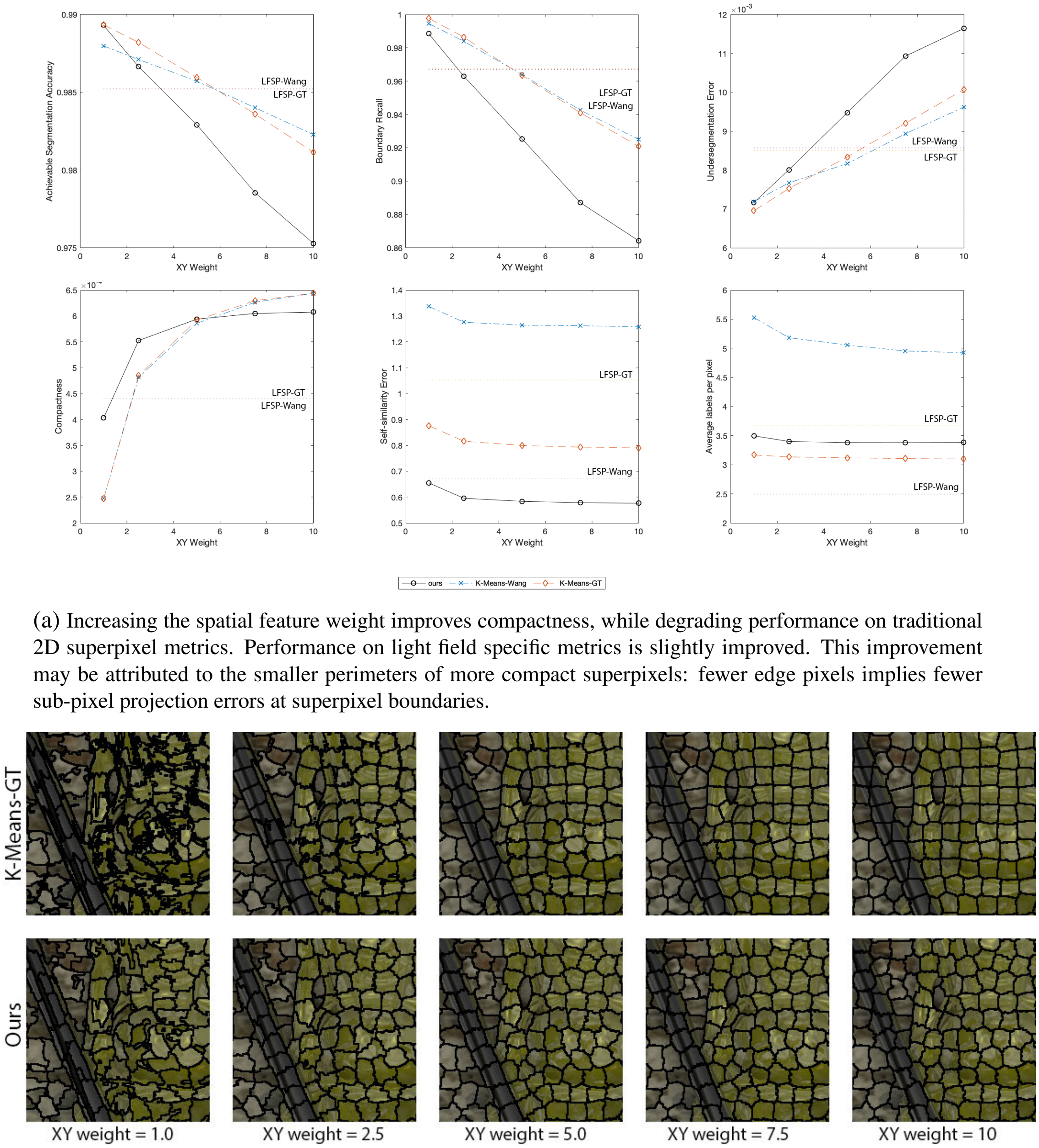


(a) Increasing the CIELAB color weight improves performance on traditional 2D superpixel metrics but decreases compactness, while slightly degrading performance on the light field specific metrics. Note that, as the *average labels per pixel* metric does not include non-central occluded pixels, this baseline performs strongly (as discussed in the main paper).



(b) A larger CIELAB color weight generates less compact superpixels.

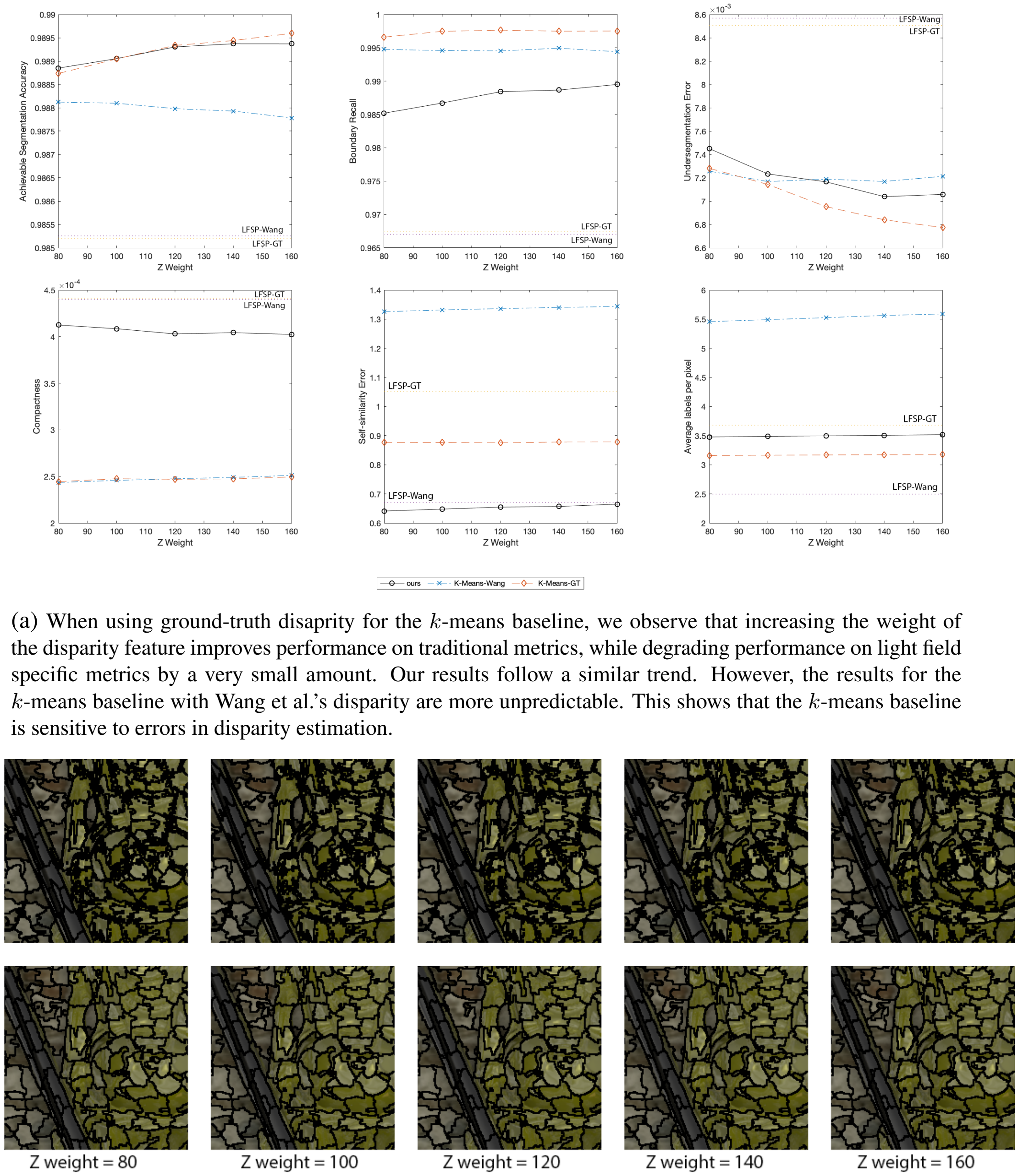
Figure A.1: An evaluation of the effect of feature weight on clustering, as a demonstration of the trade-off between boundary following and spatial/depth regularity. The superpixel size is fixed at 20.



(a) Increasing the spatial feature weight improves compactness, while degrading performance on traditional 2D superpixel metrics. Performance on light field specific metrics is slightly improved. This improvement may be attributed to the smaller perimeters of more compact superpixels: fewer edge pixels implies fewer sub-pixel projection errors at superpixel boundaries.

(b) A larger spatial weight generates more compact superpixels, at the expense of boundary accuracy.

Figure A.2: An evaluation of the XY spatial feature weight on clustering. The superpixel size is fixed at 20.



(a) When using ground-truth disaprity for the k-means baseline, we observe that increasing the weight of the disparity feature improves performance on traditional metrics, while degrading performance on light field specific metrics by a very small amount. Our results follow a similar trend. However, the results for the k-means baseline with Wang et al.'s disparity are more unpredictable. This shows that the k-means baseline is sensitive to errors in disparity estimation.

(b) A larger CIELAB color weight generates less compact superpixels.

Figure A.3: An evaluation of the disparity feature weight on clustering. The superpixel size is fixed at 20.

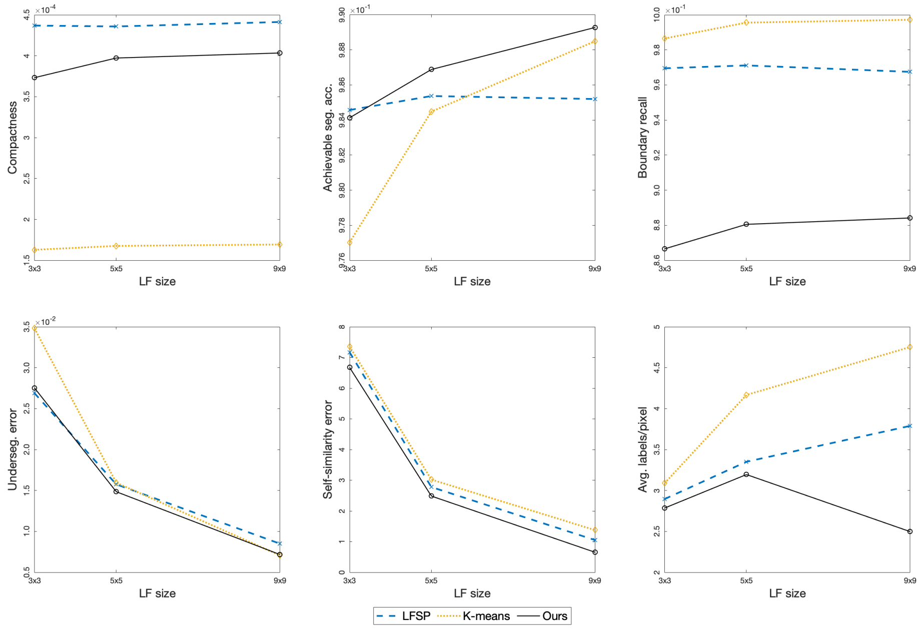


Figure A.4: Performance on large disparity light fields. As the number of views decreases and the baseline increases, the performance trends found at 9×9 views tend to be maintained.

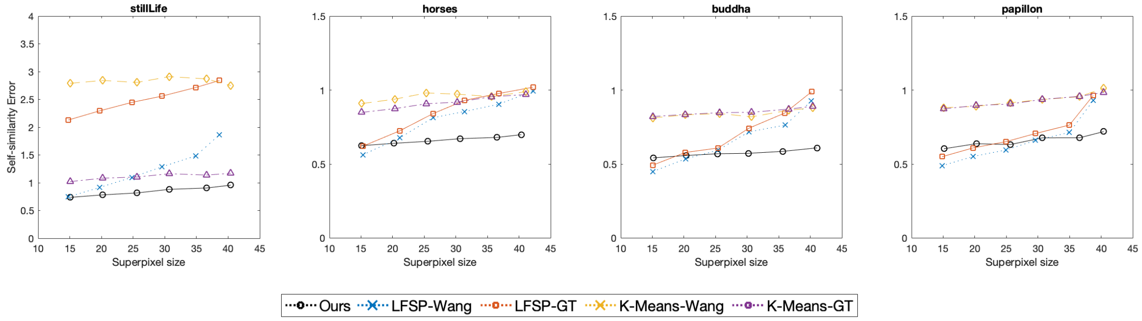


Figure A.5: Self-similarity error measured over all light fields in the HCI dataset. This error provides a measure of the consistency of superpixel shape across views; smaller errors indicate greater consistency.

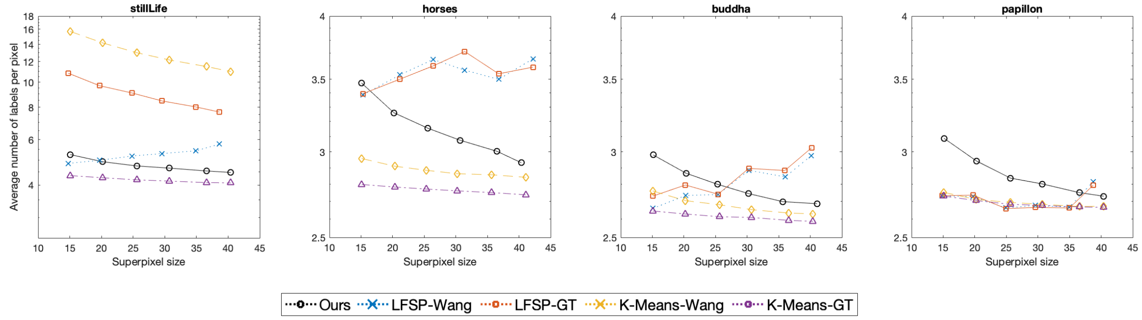


Figure A.6: The average number of labels per pixel measured over all light fields in the HCI dataset. Smaller values indicate that a greater number of pixels have a consistent label across views.

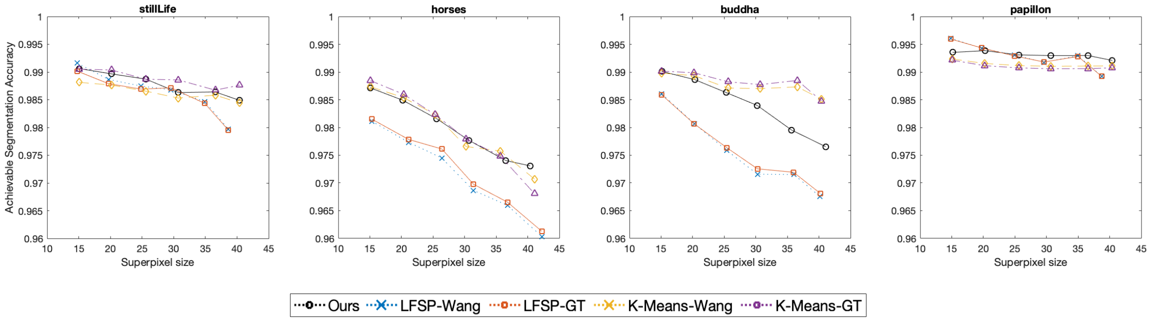


Figure A.7: The achievable segmentation accuracy measured over all light field in the HCI dataset. The achievable segmentation accuracy describes how well the segments align with the ground truth labels. Hence, it provides a measure of the percentage of correctly labeled pixels. While ASA is not the same as object segmentation accuracy, it provides an upper bound on the accuracy of an object-level segmentation based on the current oversegmentation.

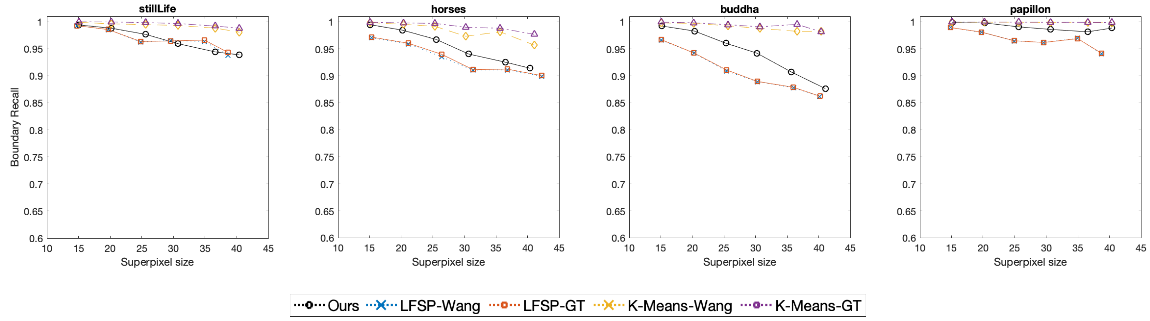


Figure A.8: Boundary recall measured over all light fields in the HCI dataset. Boundary recall measures the fraction of ground truth edges which fall within a distance d of one or more super pixel boundary. Intuitively, the higher the boundary recall, the better the superpixels adhere to object boundaries.

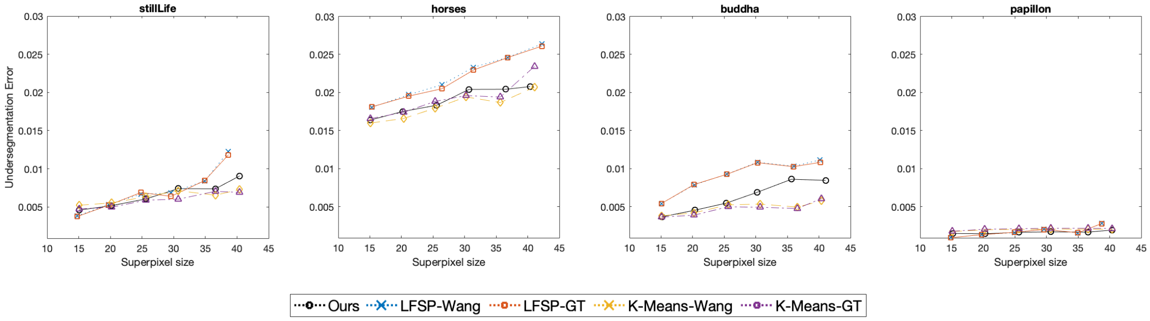


Figure A.9: Undersegmentation Error measured over all light fields in the HCI dataset. Undersegmentation error measures the percentage of superpixels that extend over ground truth segment borders

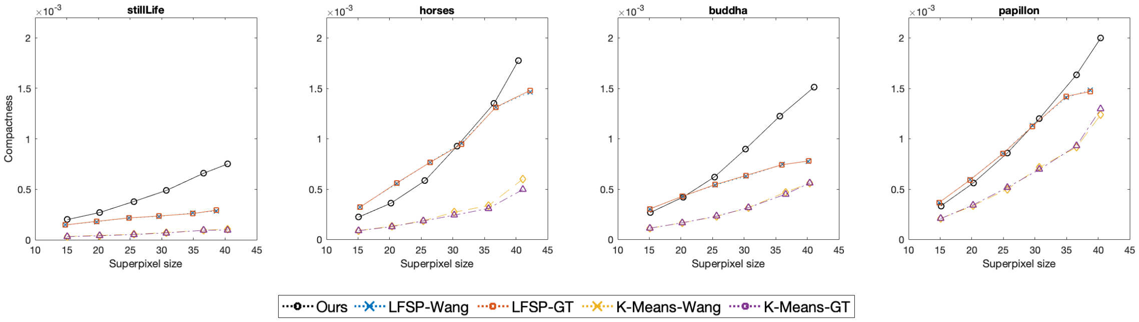


Figure A.10: Compactness measured over all superpixels in the light fields of the HCI dataset. The compactness is related to the ratio of area to perimeter, and larger values signify smoother superpixel boundaries.

Appendix B

Expanded Analysis of Edge-Aware Bidirectional Diffusion

We include additional discussion covering the benefits over naive diffusion, consistency over views within the 4D light field, tolerance to depth label errors and edge blur, robustness to hyperparameter variation, details of dataset preprocessing, and an example of textures within dark backgrounds in the Stanford dataset (Section B.1). Next, we present error maps comparing reprojection loss versus our bidirectional diffusion approach (Section B.3), and error maps versus ground truth for the HCI dataset (Section B.4). Finally, we show additional qualitative results on the Stanford dataset (Section B.9) and an additional editing example (Figure B.10).

B.1 Additional Discussion

Naive Diffusion. In Figure B.1, we demonstrate visually that naively diffusing disparity labels can be problematic because edge localization is ambiguous.

Multi-view Depth and Error. As ground truth disparity is only provided for the central view of the HCI data set, and as the Stanford data set has no ground truth depth, we did not include quantitative error evaluation across '4D' views. Qualitatively, our method tends to produce results that are consistent across views (Figure B.2).

Disparity Noise and Blur Tolerance. To show our robustness, we evaluate our method on noisy disparity labels (Figure B.3) and low-gradient edges (Figure B.4). Our method provides greater robustness to disparity errors than naive diffusion, and provides greater robustness via MSE to low-gradient (or blurry) edges than two learning-based baselines.

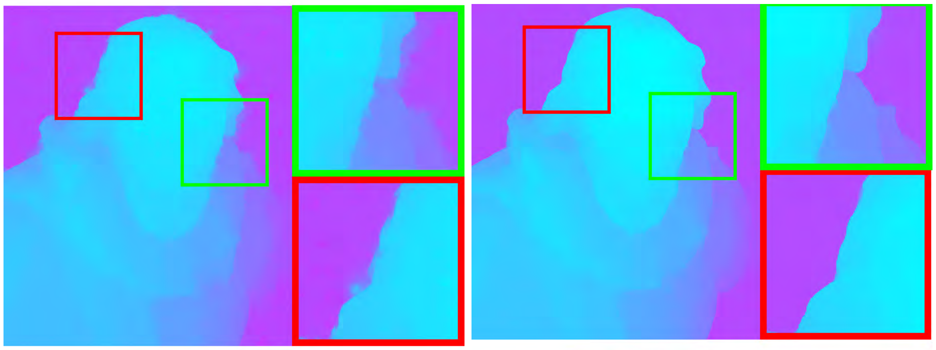


Figure B.1: *Left*: Naïvely diffusing disparity labels causes artifacts around edges due to ambiguity in the localization of labels around edges. *Right*: Estimating the diffusion gradient removes this ambiguity and yields sharp depth edges.

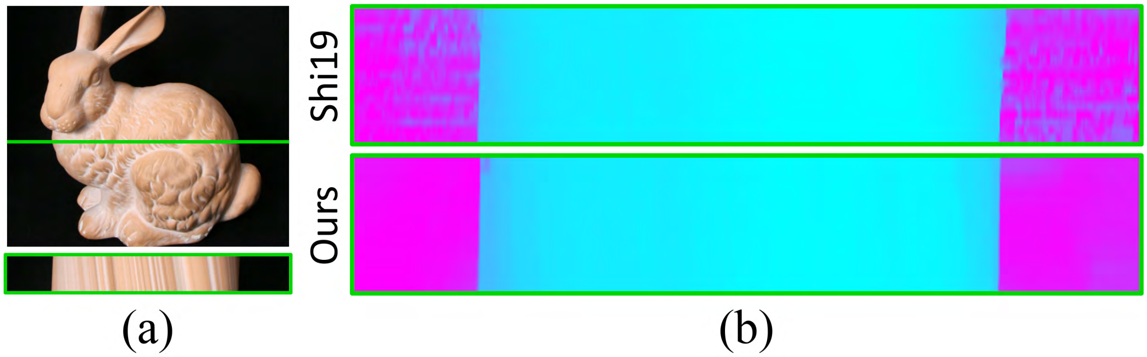


Figure B.2: **(a)** We visualize depth consistency for the highlighted epipolar line. **(b)** Our results are more consistent than Shi et al. [93] across views (EPIs are scaled vertically for clarity).

Lenslet Distortion and EPFL Lytro Dataset. The Lytro light fields in the EPFL dataset are provided decoded as MATLAB files. In general, while our method can handle small amounts of distortion, the EPI-based edge detection stage expectedly fails when EPI features are no longer linear. This is true for the edge views of Lytro light fields. As such, we only use the central 7×7 views of the EPFL scenes for all experiments.

Black Backgrounds and Stanford Dataset. Our EPI edge detector aggregates information from all three channels in CIE LAB color space, which allows it to detect even faint edges. Thus, it captures the subtle background texture on the black cloth in the Stanford dataset examples of single objects; typically, this detail is not visible to the naked eye. This feature of our work also explains why we do not incorrectly detect false edges in the Bunny scene (Figure 6.3).

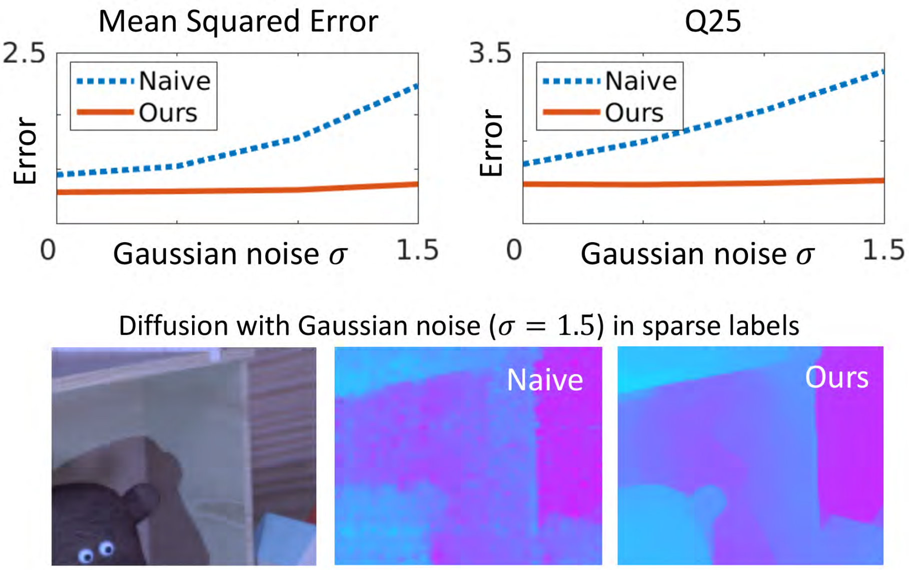


Figure B.3: Robustness of our method to noise in disparity labels. We compare our method to naive diffusion on the *Dino* light field.

B.2 Expanded Results

We present qualitative results on the HCI dataset in Figure B.6, and expanded results on the real-world Stanford scenes in Figure B.9. Our method produce stronger depth edges compared to the baselines, and our smoothness regularization leads to fewer artifacts in textureless regions.

B.3 Diffusion Gradients as Self-supervised Loss

As in Section 6.2, we compare our method to a reprojection error loss. In Figure B.7, to complement the quantitative MSE numbers in the main paper, we demonstrate the qualitative improvement from our bidirectional diffusion gradient approach in comparison.

B.4 Error Maps

We visualize the absolute disparity error of all baselines and our method in Figure B.8. The baseline methods produce larger errors around depth edges compared to our approach. This can be seen in the fewer regions of red for our method compared to the baselines. The corresponding dense disparity maps are shown in Figure 6.8. Qualitatively, our results are comparable to the learning-based baselines [58, 93, 46] with fewer extreme errors around edges.

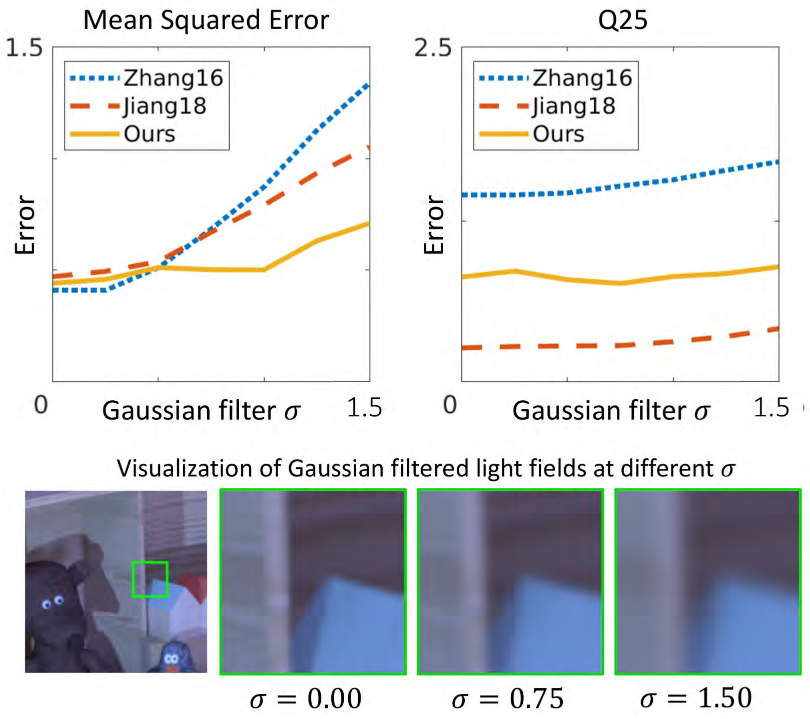


Figure B.4: Robustness of our method to low-gradient edges (*Dino* light field; we compare to the methods of Zhang et al. [125] and Jiang et al. [46] which have the best MSE and Q25 on this light field, respectively).

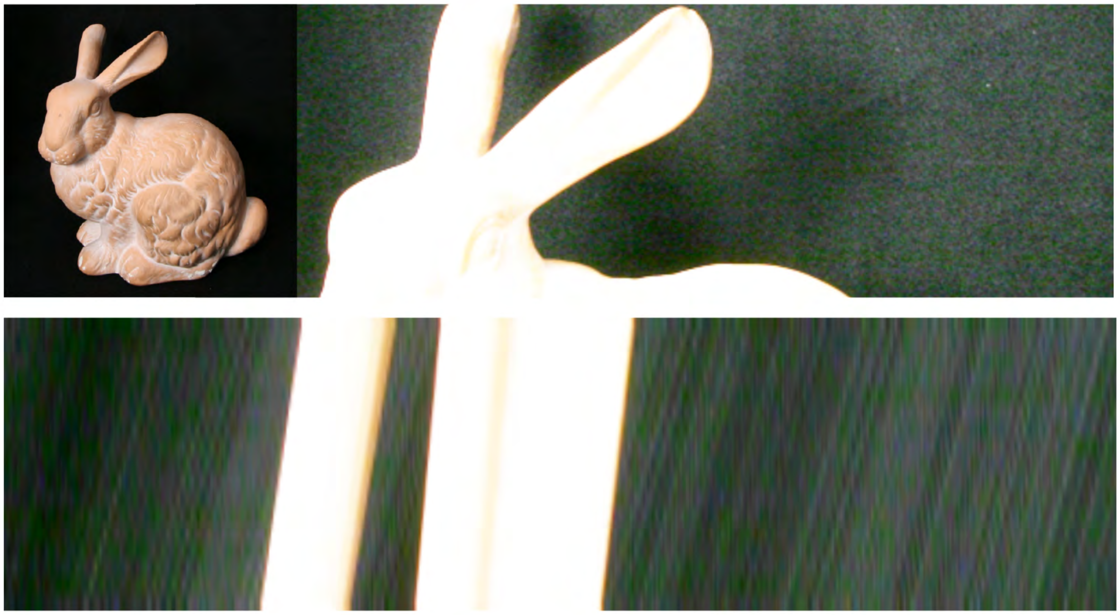


Figure B.5: *Top:* On the Stanford Bunny scene, enhanced image contrast shows the texture of the cloth in the seemingly black background. *Bottom:* In EPI space (scaled vertically for clarity) the texture appears as sloped lines, providing background disparity to methods that can exploit this subtle information.

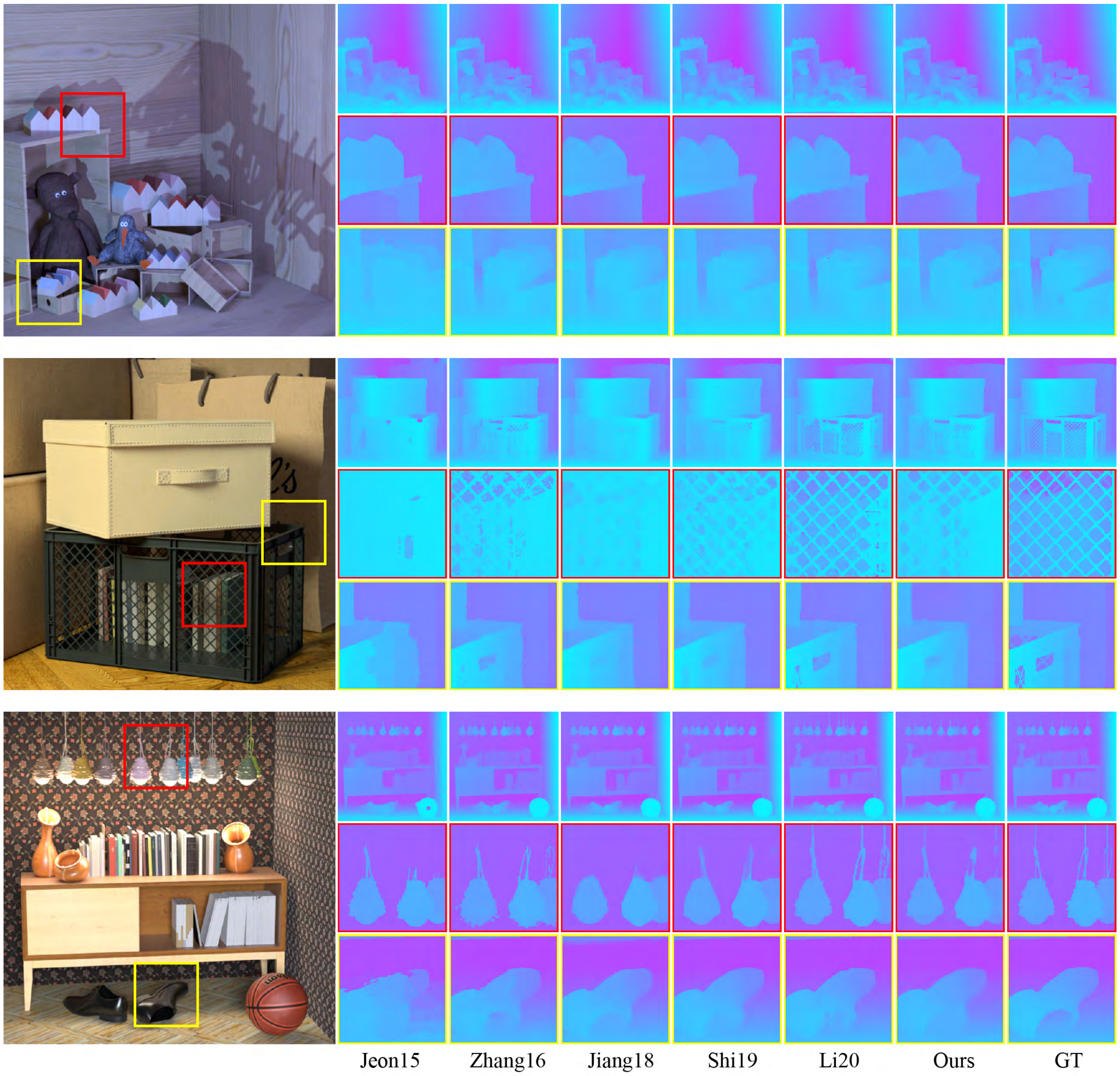


Figure B.6: Results on the synthetic light fields of the HCI dataset. *Left to right:* Jeon et al. [44], Zhang et al [125], Jiang et al. [46], Shi et al. [93], Li et al. [58], our method, and finally, the ground truth. Qualitatively, our results are comparable to the learning-based baselines [58, 93, 46] with fewer extreme errors around edges.

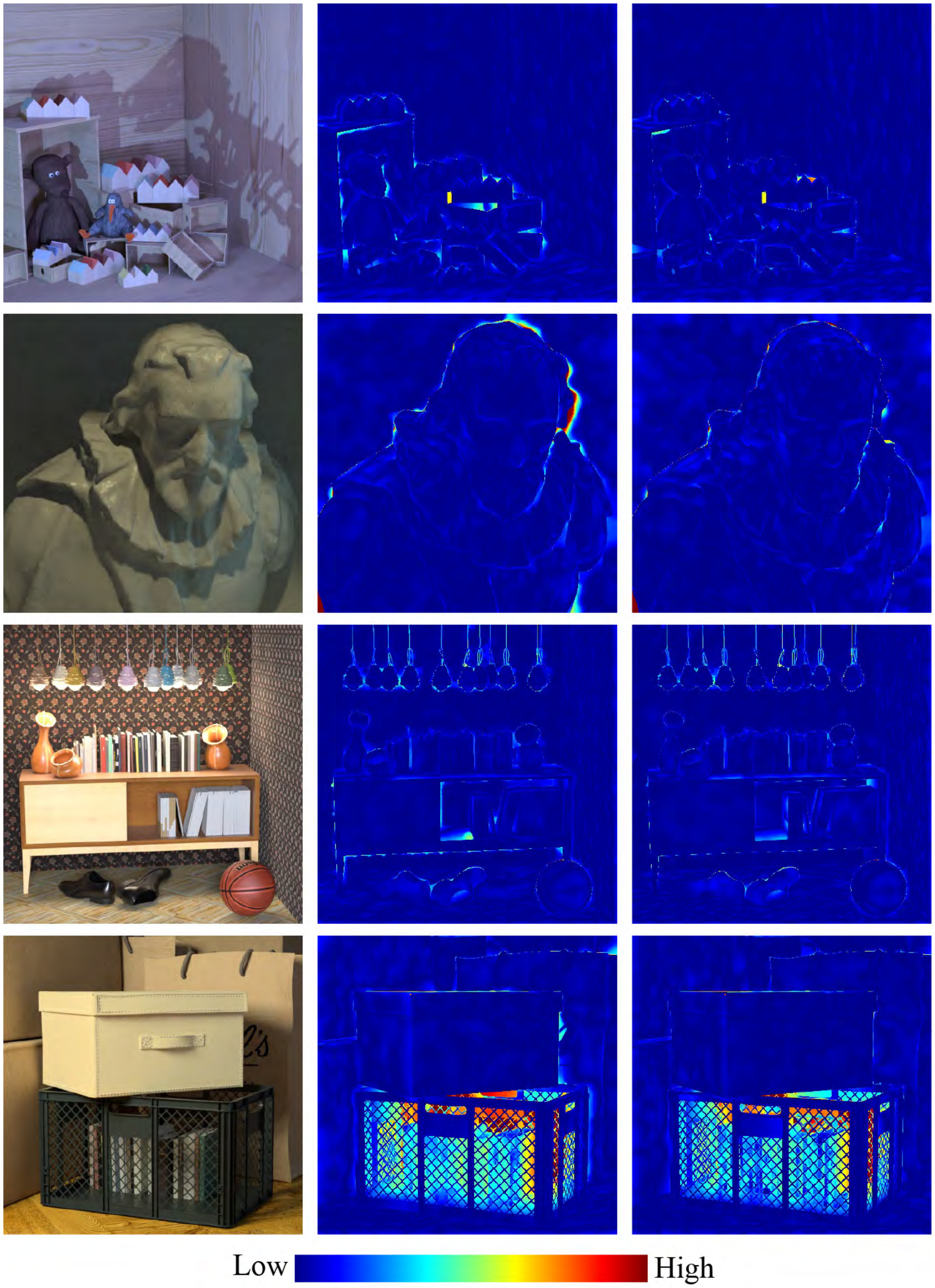


Figure B.7: Multiview reprojection error (*center*) as self-supervised loss for depth edge localization, compared to our bidirectional diffusion gradients (*right*). We show absolute disparity error. Our method has lower error around edges.

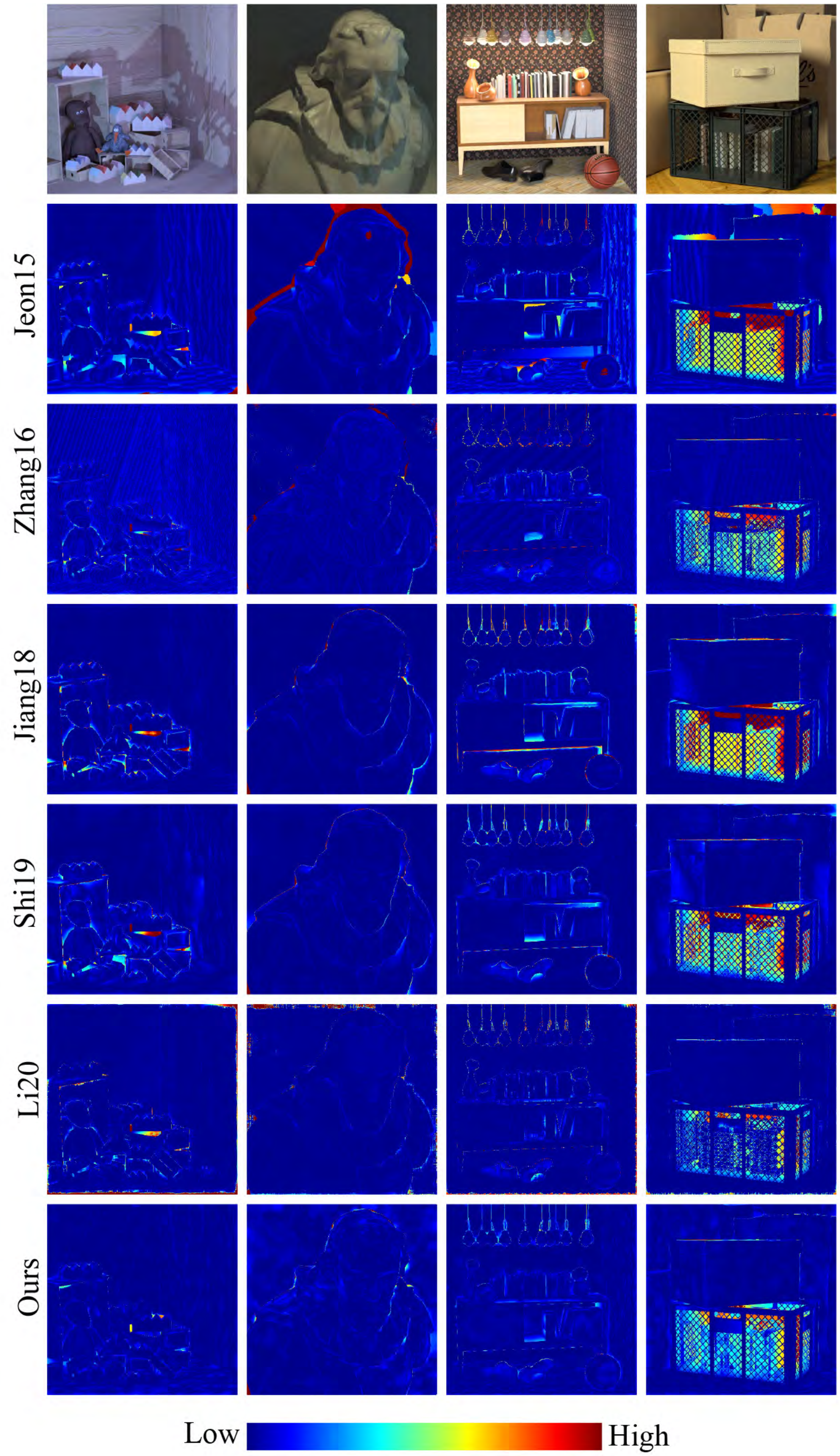


Figure B.8: A visualization of the absolute disparity error for all baselines. *Top to bottom:* Jeon et al. [44], Zhang et al [125], Jiang et al. [46], Shi et al. [93], Li et al. [58], and our method.

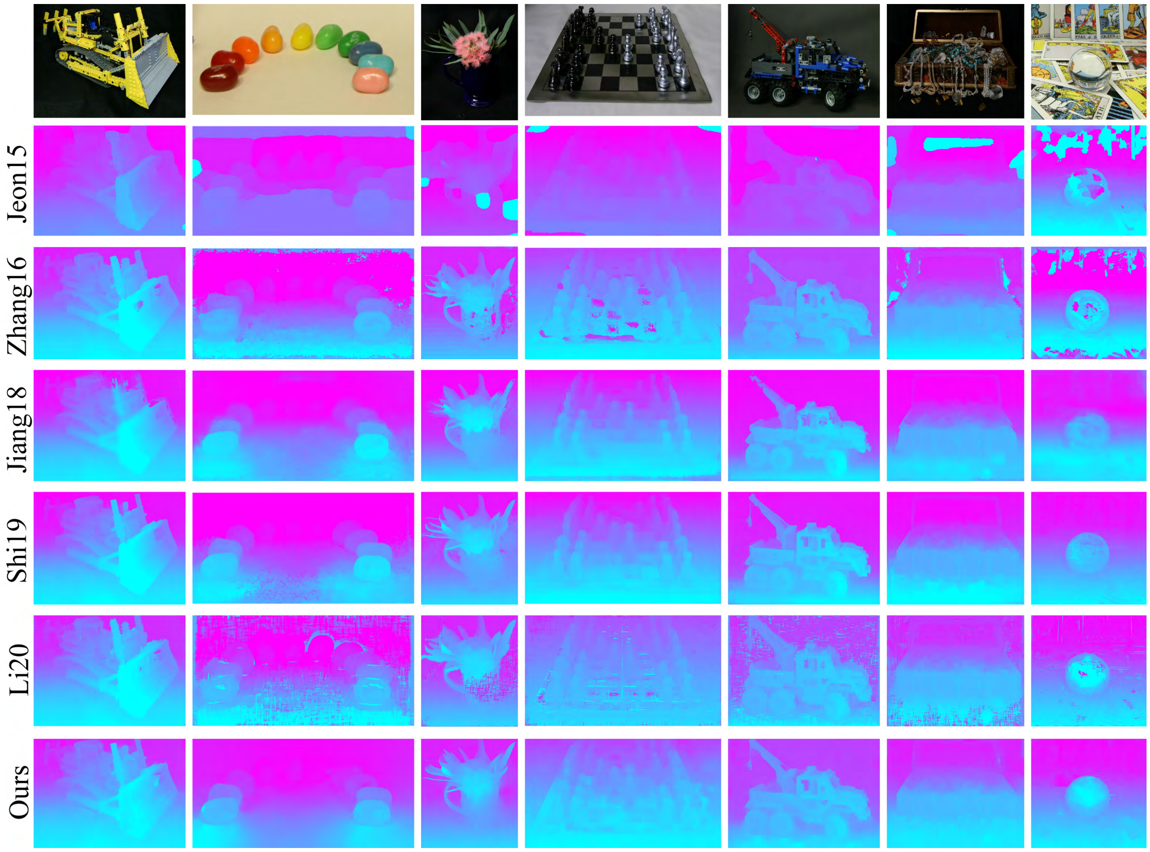


Figure B.9: Results on light fields from the Stanford dataset. *Top to bottom:* Jeon et al. [44], Zhang et al [125], Jiang et al. [46], Shi et al. [93], Li et al. [58] and our method.

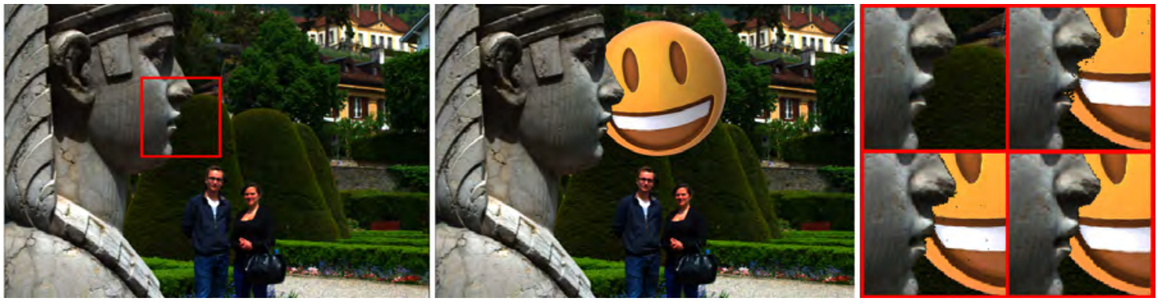


Figure B.10: Additional light field editing results. *Left:* input scene. *Center:* Our editing results. *Right, clockwise from top-left*: Detail of the unmodified light field image, Zhang et al. [125]'s editing result, Shi et al. [93]'s editing result, and our result with fewer artifacts.

Appendix C

Expanded Analysis of Differentiable Diffusion

C.1 Expanded Results

In Table C.1, we compare performance on the real-world Stanford and EPFL light fields, with the methods of Zhang at al. [125], Li et al [58], Jiang et al. [46], Shi et al. [93], and the central-view results of Chapter 6 (Ours-N). No ground truth depth data exists for these scenes. As a proxy for depth accuracy we use reprojection error $\times 10^{-2}$ in RGB color space induced by warping the central view onto the corner views using the estimated disparity map.

Our approach is competitive or better than other methods except on the *Chess* light field, which exhibits strong specular effects due to polished metal materials. However, featureless backgrounds cause our edges to be diffuse (Figure C.2).

We also include error maps for all light field datasets, along with depth maps for all example light fields listed in the main and supplemental documents. Please see Figures C.3 and C.4 for synthetic scenes, Figures C.5 and C.6 for the Stanford scenes, and Figures C.7 and C.8 for the EPFL scenes. For the additional results presented here, we only run a single pass over each parameter.

C.2 Noise Handling

We evaluate the robustness of our method to noise by randomly adding Gaussian noise of increasing variance to 50% of the original points. We find our method robust to position noise and still capable under significant disparity noise. Figure C.1 shows results for the *Dino* scene

Table C.1: **(Best viewed in color)** Quantitative comparison of our method against five baseline methods on real-world Stanford (top) and EPFL (bottom) datasets. As a metric, we use reprojection error $\times 10^{-2}$ as a proxy for depth accuracy as no ground truth exists. The top three results are highlighted in gold, silver, and bronze.

Light Fields	Reprojection Error					
	[125]	[58]	[46]	[93]	Ours-N	Ours
Bulldozer	2.06	1.90	3.57	1.87	2.44	1.67
Bunny	1.18	1.08	1.57	1.08	1.10	0.96
Chess	1.72	3.19	1.63	1.54	1.66	2.22
Eucalyptus	1.19	1.03	1.55	1.11	1.14	1.01
Jelly Beans	1.36	1.78	1.71	1.40	1.68	1.27
Lego	3.13	3.50	4.98	2.57	3.19	2.38
Tarot	15.1	12.1	17.1	10.5	17.2	9.98
Treasure	2.51	1.89	2.35	1.96	2.18	1.87
Truck	1.49	1.39	1.80	1.33	1.37	1.24
Average	3.30	3.10	4.03	2.60	3.55	2.51
Bikes	2.26	4.85	2.24	5.13	2.39	2.24
Grid	3.33	6.46	3.05	6.84	3.20	2.87
Silos	1.77	3.34	1.85	3.41	2.01	1.69
Sphynx	2.37	4.80	2.31	5.21	2.50	2.30
Average	2.43	4.86	2.36	5.14	2.53	2.28

C.3 Point Sparsity

In Figure C.9, we evaluate the performance of the optimization routine by varying the point set size. We observe that for the bad pixels metrics, the routine converges at the same minima even when only half the original point set is used. For MSE, having more points is helpful. This is expected as a larger point set is more helpful in filtering the effect of outliers.

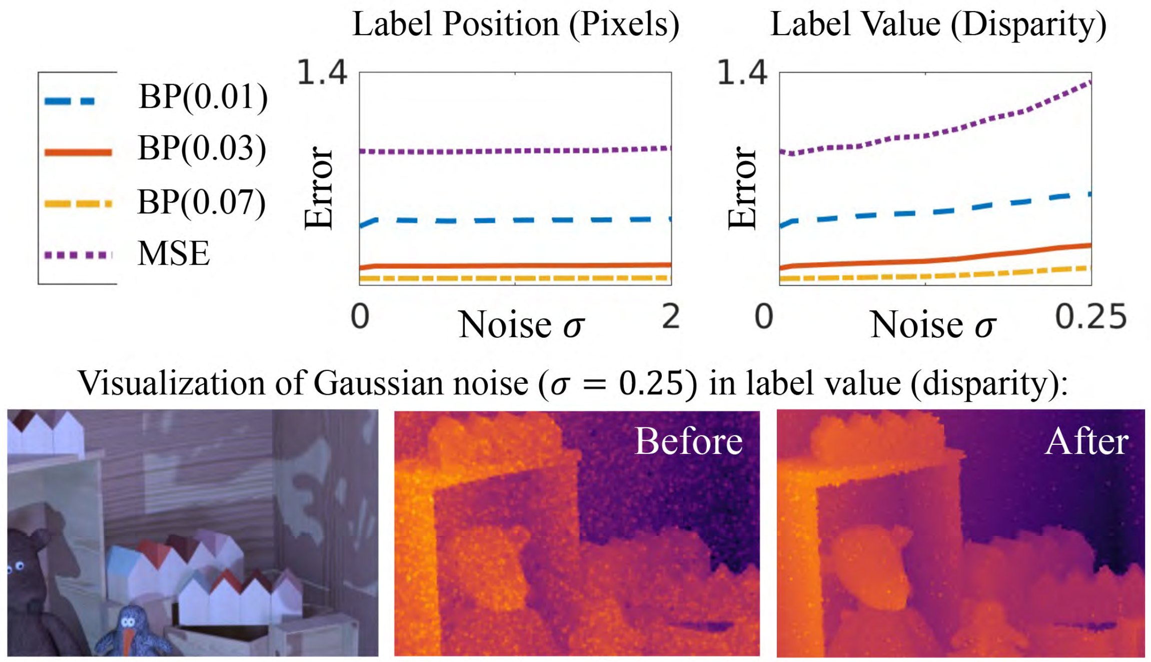


Figure C.1: Robustness of our method to noise (*Dino* light field). The ground truth scene has mean disparity -0.1 with SD 0.73.

C.4 Ray Attenuation Formula

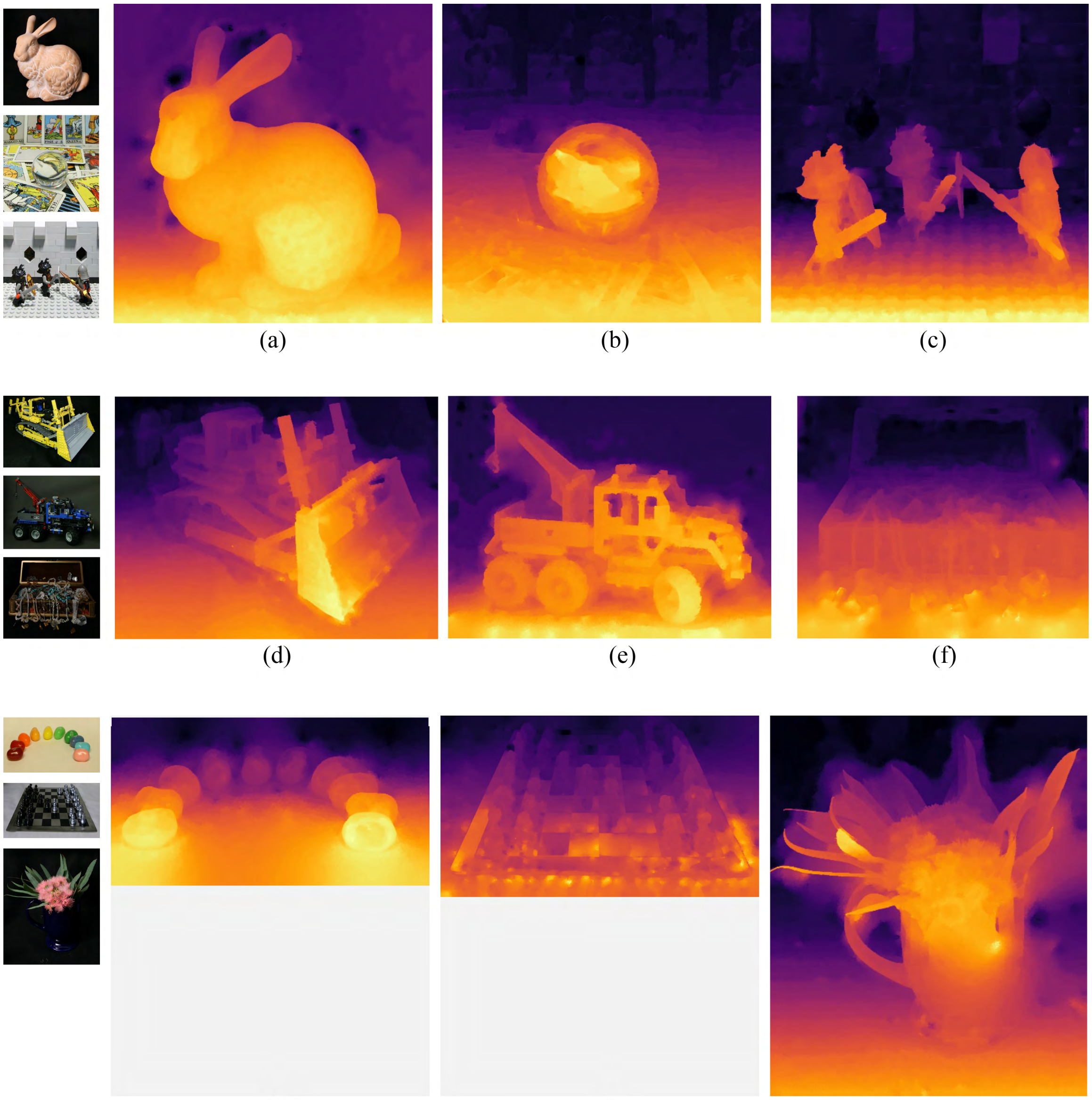
The attenuation factor T at distance s from the image plane is defined as

$$
\begin{aligned}
\mathrm{T}(x,y,s) &= \exp\left(-\int_0^s \rho \sum_{\mathbf{x}\in\mathcal{P}} \exp\left(-\left(\frac{(x-x_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} + \frac{(y-y_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} + \frac{(z-Z_{\mathbf{x}})^2}{2\sigma_Z^2}\right)\right) dz\right) \\
&= \exp\left(-\int_0^s \rho \sum_{\mathbf{x}\in\mathcal{P}} \exp\left(-\frac{(x-x_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} - \frac{(y-y_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} - \frac{(z-Z_{\mathbf{x}})^2}{2\sigma_Z^2}\right) dz\right) \\
&= \exp\left(-\int_0^s \rho \sum_{\mathbf{x}\in\mathcal{P}} \exp\left(-\frac{(x-x_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} - \frac{(y-y_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2}\right) \exp\left(-\frac{(z-Z_{\mathbf{x}})^2}{2\sigma_Z^2}\right) dz\right) \\
&= \exp\left(-\int_0^s \rho \sum_{\mathbf{x}\in\mathcal{P}} \frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)}{Z_{\mathbf{x}}} \exp\left(-\frac{(z-Z_{\mathbf{x}})^2}{2\sigma_Z^2}\right) dz\right) \quad \text{(From Equation 7.4)} \\
&= \exp\left(-\sum_{\mathbf{x}\in\mathcal{P}} \int_0^s \rho\, \frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)}{Z_{\mathbf{x}}} \exp\left(-\frac{(z-Z_{\mathbf{x}})^2}{2\sigma_Z^2}\right) dz\right)
\end{aligned}
$$

As $\sigma_Z \rightarrow 0$, the density contribution at any point s along the ray will come from only a single Gaussian. Thus, we can write

$$\begin{aligned}
\mathrm{T}(x,y,s) &= \exp\Bigg(-\int_0^{s_1} \rho\, \frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}_1}(x,y)}{Z_{\mathbf{x}_1}} \exp\Bigg(-\frac{(z-Z_{\mathbf{x}_1})^2}{2\sigma_Z^2} \Bigg)\, dz - \int_0^{s_2} \rho\, \frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}_2}(x,y)}{Z_{\mathbf{x}_2}} \exp\Bigg(-\frac{(z-Z_{\mathbf{x}_2})^2}{2\sigma_Z^2} \Bigg)\, dz - ... \\
&\quad - \int_0^{s_n} \rho\, \frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}_{\mathrm{n}}}(x,y)}{Z_{\mathbf{x}_{\mathrm{n}}}} \exp\Bigg(-\frac{(z-Z_{\mathbf{x}_{\mathrm{n}}})^2}{2\sigma_Z^2} \Bigg)\, dz \Bigg)
\end{aligned}$$

$$\begin{aligned}
\mathrm{T}(x,y,s) &= \exp\Bigg(-\int_0^{t} \rho\, \frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}_1}(x,y)}{Z_{\mathbf{x}_1}} \exp\Bigg(-\frac{(z-\mu_z)^2}{2\sigma_Z^2} \Bigg)\, dz - \int_0^{t} \rho\, \frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}_2}(x,y)}{Z_{\mathbf{x}_2}} \exp\Bigg(-\frac{(z-\mu_z)^2}{2\sigma_Z^2} \Bigg)\, dz - ... \\
&\quad - \int_0^{t} \rho\, \frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}_{\mathrm{n}}}(x,y)}{Z_{\mathbf{x}_{\mathrm{n}}}} \exp\Bigg(-\frac{(z-\mu_z)^2}{2\sigma_Z^2} \Bigg)\, dz \Bigg) \quad \text{(Figure 7.5 describes the bounds } [0,t]\text{.)} \\
&= \prod_{\mathbf{x}} \exp\Bigg(-\int_0^{t} \rho \frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)}{Z_{\mathbf{x}}} \exp\Bigg(-\frac{(z-\mu_z)^2}{2\sigma_Z^2} \Bigg)\, dz \Bigg) \\
&= \prod_{\mathbf{x}} \mathrm{T}_{\mathbf{x}}(x,y)
\end{aligned}$$



(a) (b) (c)

(d) (e) (f)

(h) (i) (j)

Figure C.2: Our results on the real world light fields of the Stanford dataset.

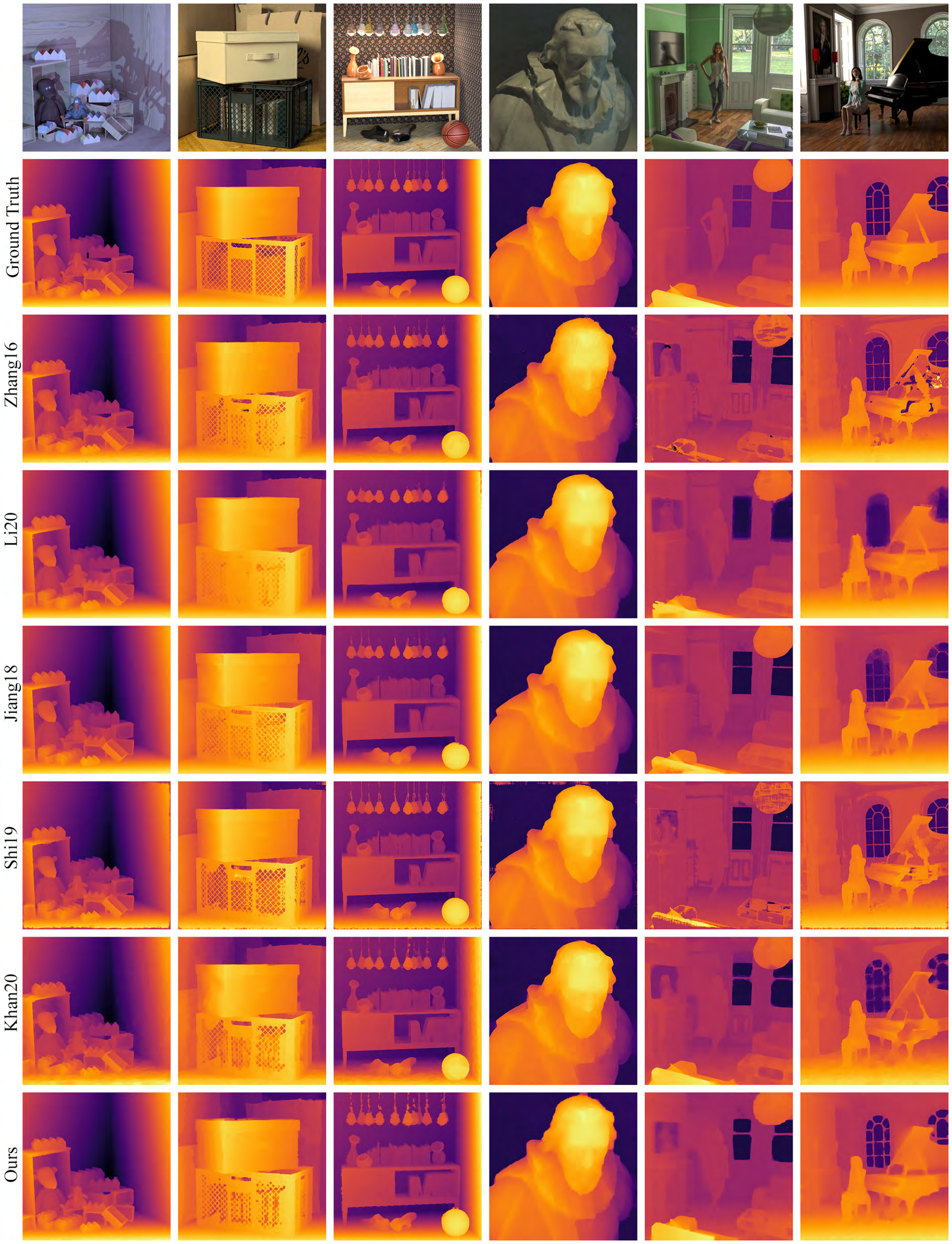


Figure C.3: Qualitative comparison of our method with all baselines on the HCI dataset, and the light fields *Living Room* and *Piano*. From top to bottom: Zhang et al. [125], Li et al. [58], Jiang et al. [46], Shi et al. [93], our non-differentiable method of Chapter 6 (Khan20), and our differentiable approach.

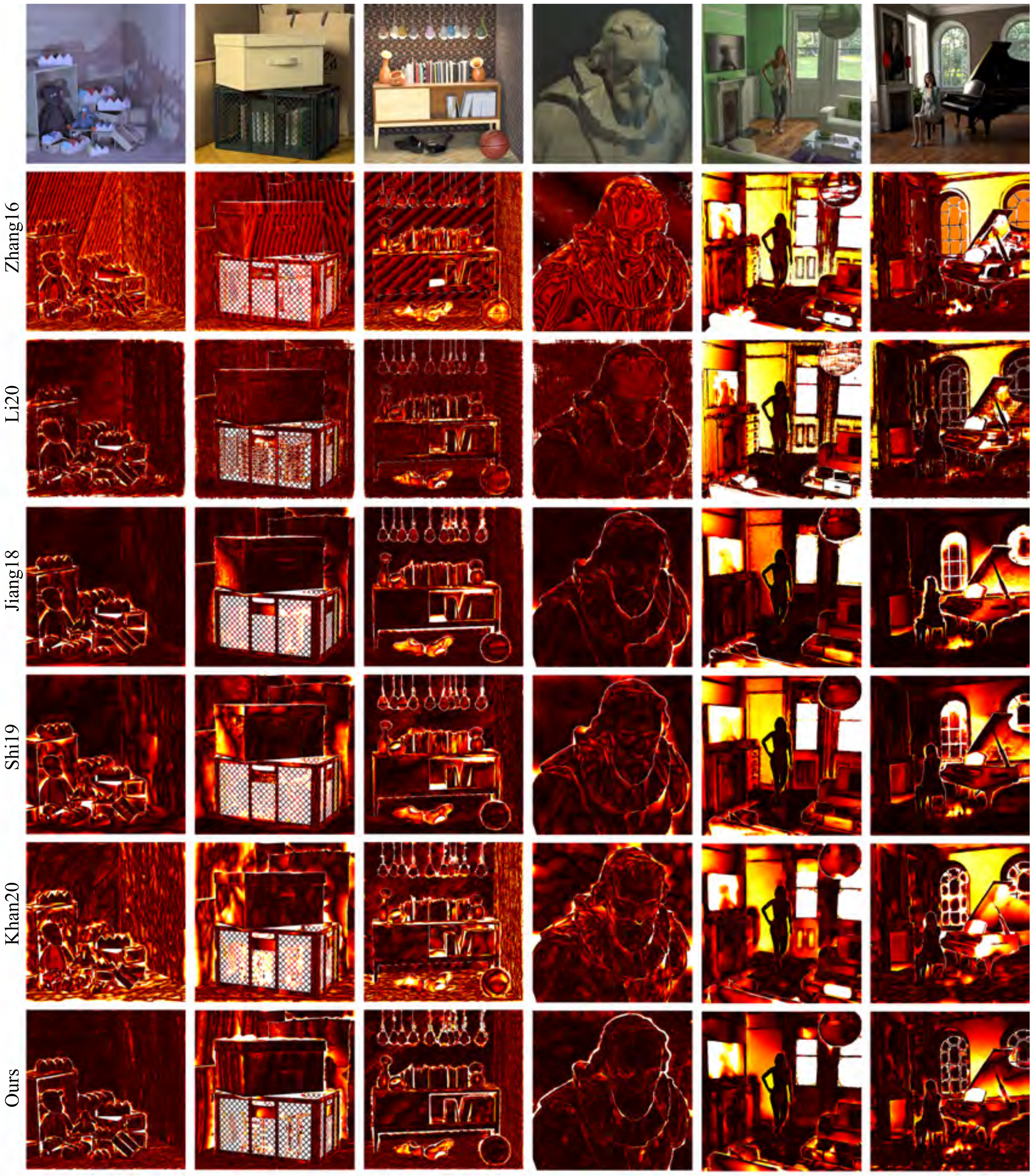


Figure C.4: A visualization per-pixel L1 error for all methods (lighter regions have higher error). From top to bottom: Zhang et al. [125], Li et al. [58], Jiang et al. [46], Shi et al. [93], our non-differentiable method of Chapter 6 (Khan20), and our differntiable approach.

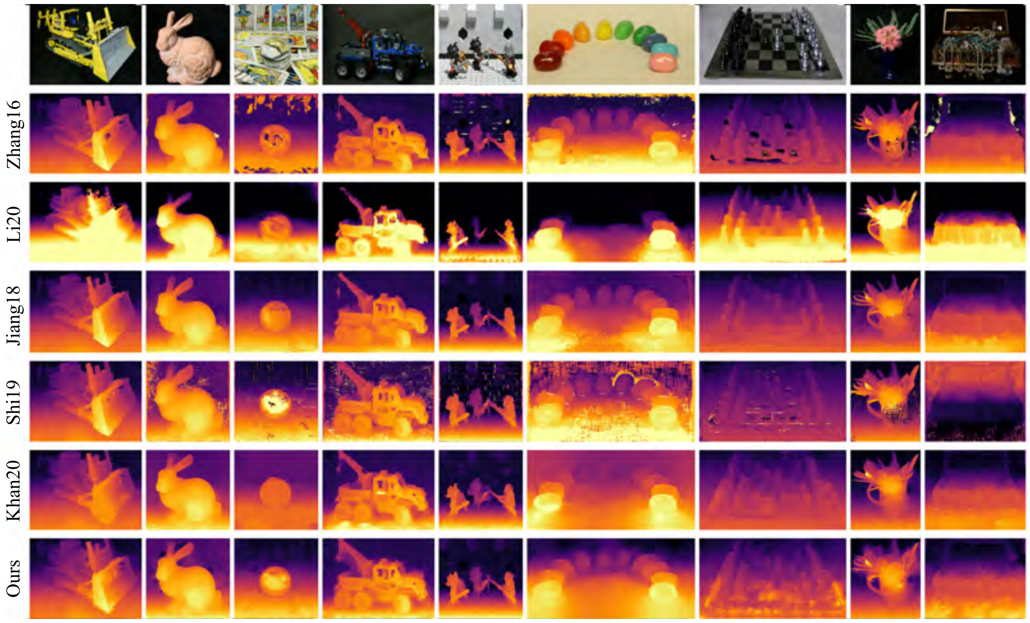


Figure C.5: Qualitative comparison of our method with all baselines on the Stanford dataset. From top to bottom: Zhang et al. [125], Li et al. [58], Jiang et al. [46], Shi et al. [93], our non-differentiable method of Chapter 6 (Khan20), and our differentiable approach.

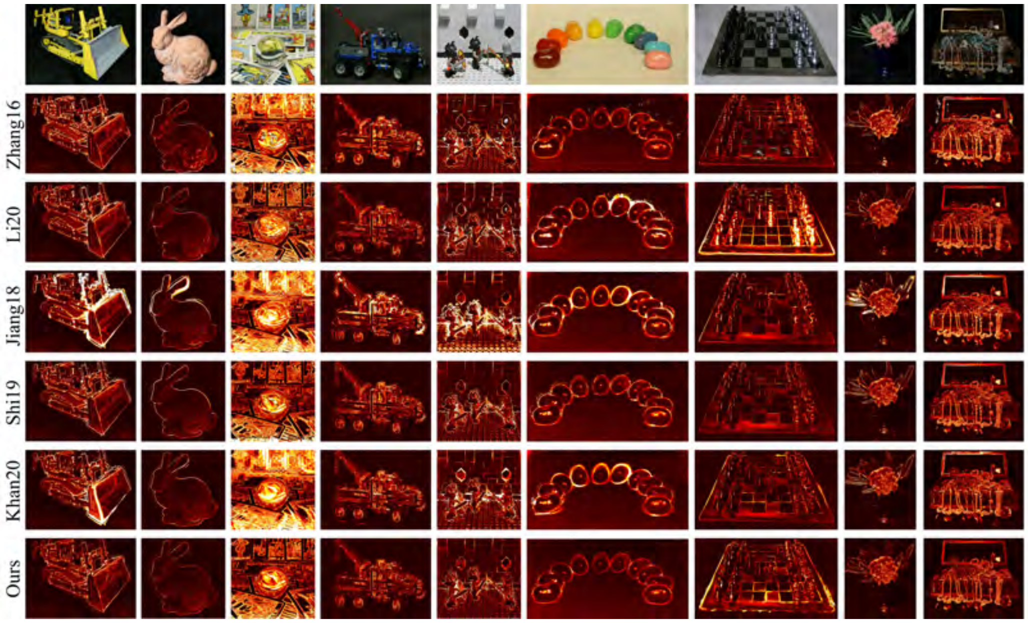


Figure C.6: A visualization of the reprojection error for all methods on the Stanford dataset (lighter regions have higher error). From top to bottom: Zhang et al. [125], Li et al. [58], Jiang et al. [46], Shi et al. [93], our non-differentiable method of Chapter 6 (Khan20), and our differentiable approach.

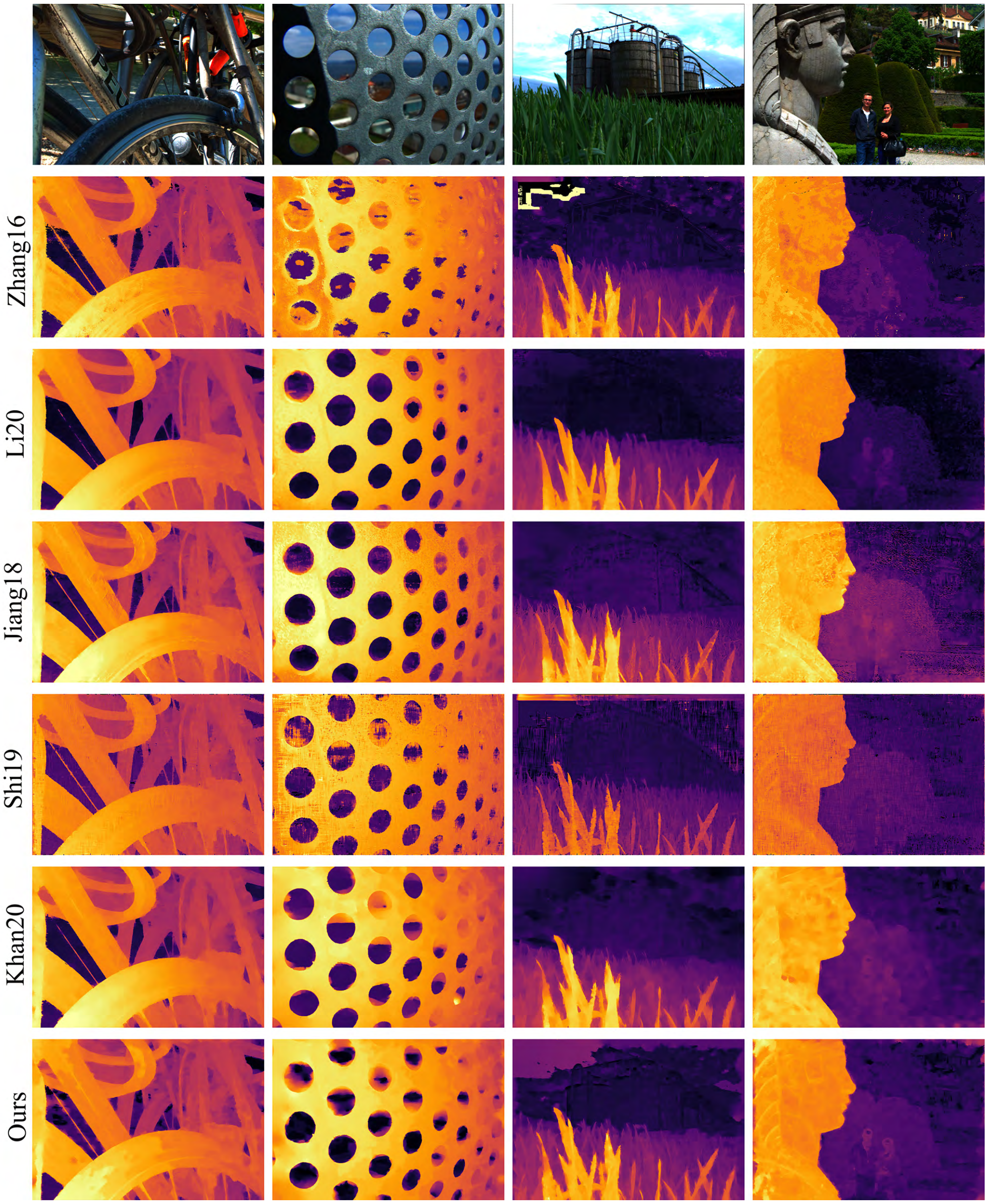


Figure C.7: Qualitative comparison of our method with all baselines on the EPFL dataset. From top to bottom: Zhang et al. [125], Li et al. [58], Jiang et al. [46], Shi et al. [93], our non-differentiable method of Chapter 6 (Khan20), and our differentiable approach.

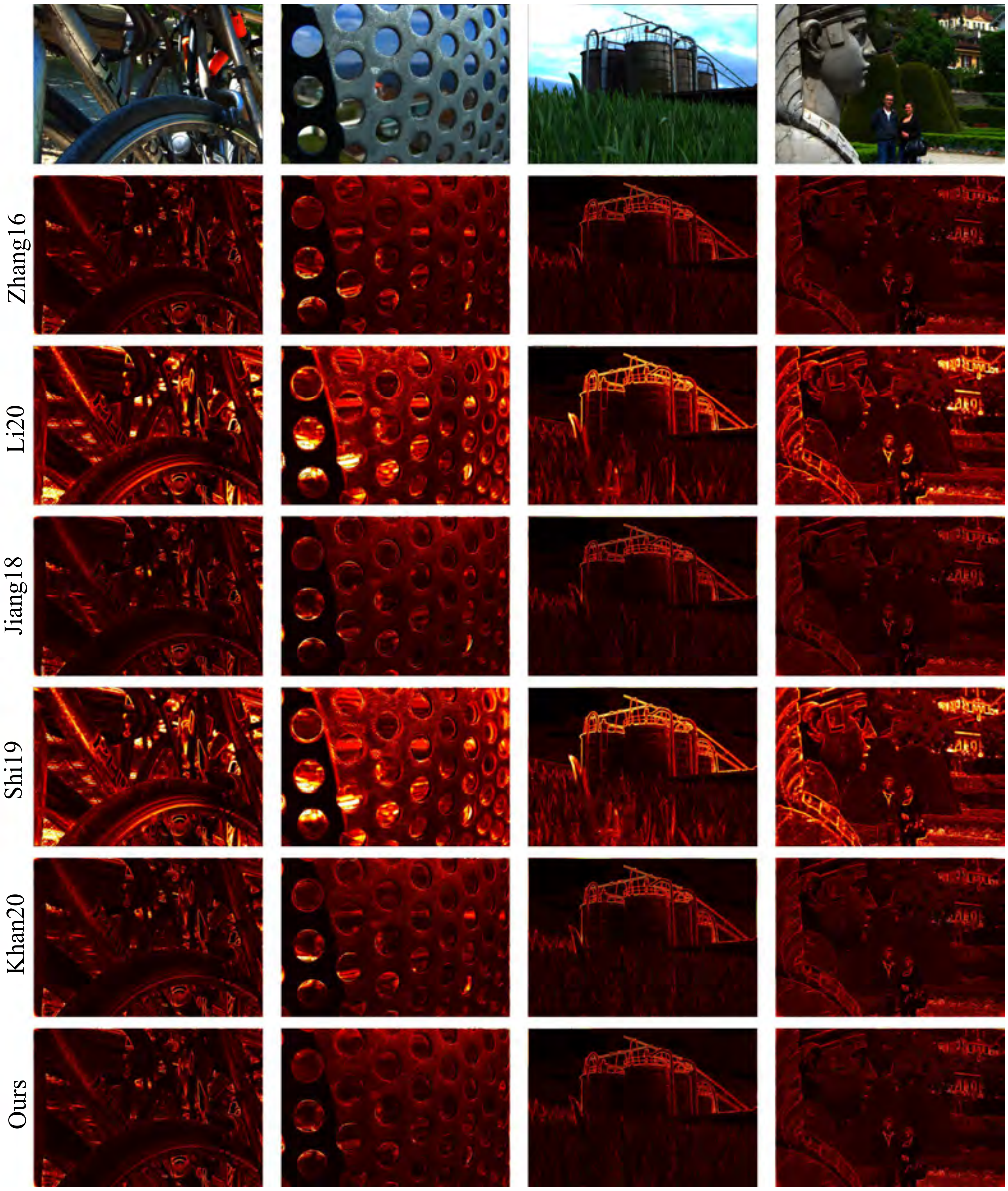


Figure C.8: A visualization of the reprojection error for all methods on the EPFL dataset (lighter regions have higher error). From top to bottom: Zhang et al. [125], Li et al. [58], Jiang et al. [46], Shi et al. [93], our non-differentiable method of Chapter 6 (Khan20), and our differentiable approach.

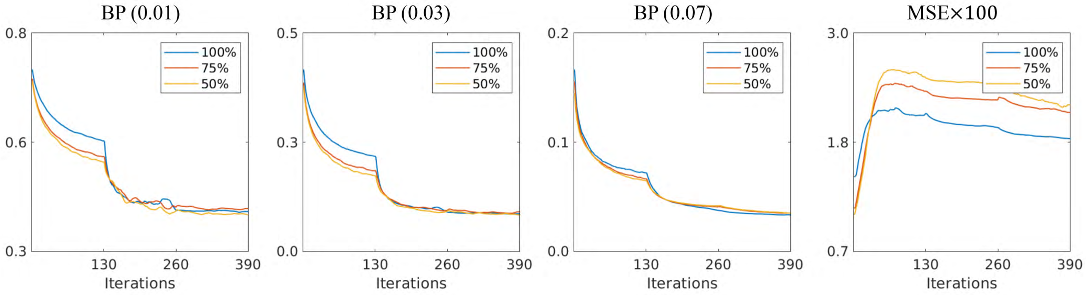


Figure C.9: We evaluate optimization performance with varying point set size. We report average performance over all iterations on *cotton* and *dino* using 100%, 75% and 50% of the original point set.