Are Multi-view Edges Incomplete for Depth Estimation?

Numair Khan[1], Min H. Kim[2], James Tompkin[1*]

[1]Brown University, Providence, RI, United States.
[2]KAIST, Daejeon, South Korea.

*Corresponding author(s). E-mail(s): james_tompkin@brown.edu;
Contributing authors: numair_khan@brown.edu; minhkim@vclab.kaist.ac.kr;

Abstract

Depth estimation tries to obtain 3D scene geometry from low-dimensional data like 2D images. This is a vital operation in computer vision and any general solution must preserve all depth information of potential relevance to support higher-level tasks. For scenes with well-defined depth, this work shows that multi-view edges can encode all relevant information—that multi-view edges are complete. For this, we follow Elder's complementary work on the completeness of 2D edges for image reconstruction. We deploy an image-space geometric representation: an encoding of multi-view scene edges as constraints and a diffusion reconstruction method for inverting this code into depth maps. Due to inaccurate constraints, diffusion-based methods have previously underperformed against deep learning methods; however, we will reassess the value of diffusion-based methods and show their competitiveness without requiring training data. To begin, we work with structured light fields and Epipolar Plane Images (EPIs). EPIs present high-gradient edges in the angular domain: with correct processing, EPIs provide depth constraints with accurate occlusion boundaries and view consistency. Then, we present a differentiable representation form that allows the constraints and the diffusion reconstruction to be optimized in an unsupervised way via a multi-view reconstruction loss. This is based around point splatting via radiative transport, and extends to unstructured multi-view images. We evaluate our reconstructions for accuracy, occlusion handling, view consistency, and sparsity to show that they retain the geometric information required for higher-level tasks.

Keywords: Edges, depth reconstruction, diffusion, light fields, multi-view reconstruction.

1 Introduction

Depth estimation is a vital first step in many computer vision tasks such as novel view synthesis [1–4], scene editing [5–7], lighting and material estimation [8], and augmented reality [9]. The problem has a long history with a wide variety of proposed solutions. These include photometric stereo [10], shape from shading [11], depth from defocus [12, 13], active illumination [10, 14–17], and deep-learning-based methods including monocular settings [18–21].

The most popular and widely studied approach is still binocular and multi-view passive stereo depth estimation [22, 23]. This is due to its ability to work in many environments and lighting conditions, its immunity to interference from competing active illumination signals, and its ability to generate depth at the same resolution as the color input. In their basic form, stereo depth methods use epipolar constraints to perform a correspondence search at each pixel in neighboring images. However, this search is computationally expensive

and is susceptible to failure in textureless, specular, and disoccluded regions. The baseline between neighboring cameras can also have a significant impact on quality: Small baselines reduce accuracy as the change in disparity relative to depth is low, and large baselines make it difficult to find corresponding points in neighboring images [24].

Nonetheless, stereo depth estimation remains popular, especially with the recent proliferation of camera sensors. Most smartphones now have at least two back-facing cameras, e.g., the Google Pixel 7 Pro has three, the Light L16 had sixteen, and light field cameras may have many more. A large number of sensors leads to increased—often prohibitive—data and computational costs. But it also enables new applications in computational photography, including ones that require depth, with the quality of depth often correlating with the quality achieved in the computational photography task. While depth accuracy is less important for frontal-scene novel view synthesis [25], depth accuracy is critical for tasks that require measurement such as 3D reconstruction, tasks that edit scenes such as light field painting [5, 26], or tasks that rely on correct surface normals like relighting [10] or material estimation [8].

Beyond accuracy, we must also consider other properties of a representation for depth. For instance, a depth map contains discrete samples on a regular grid, but this may be redundant if neighboring samples do not vary. For 2D appearance images, Elder described the explicitness, concision, and completeness of a given representation [27], where completeness captures all information of potential relevance to any higher-level visual task. From this, Elder presented a representation of 2D images based on sparse 2D texture edges and a diffusion reconstruction step. This provides an explicit, concise, and complete image-space representation that is accurate in its reconstruction of an original image.

From this inspiration, our paper shows that sparse multi-view edges can similarly provide a complete representation of scene depth; that is, multi-view edges encode all relevant information to support higher-level tasks that rely on depth estimation. These can represent depth for all input views with correct occlusion, maintaining the explicitness of depth structures via occlusion edges. Further, their concision is particularly useful: beyond their sparseness helping to reduce data costs from many input camera views, multi-view edges can help us to ignore estimating depth in low-confidence textureless regions and instead rely on diffusion to fill in the gaps.

First, we describe how to quantify completeness by the three metrics of accuracy, occlusion-edge accuracy, and view-consistency, and then define the proposed sparse multi-view edge representation for scene depth (Section 2). Then, we show how to estimate the parameters of the representation for structured 4D light field images (Section 3). Two-plane parameterized light fields are a good starting point because multi-view edges are well-defined via gradients in epipolar plane images (EPIs). Evaluating the representation for completeness requires measuring the loss of information during encoding. Thus, we show how to decode the representation into piece-wise smooth multi-view depth maps using diffusion, where the representation provides constraints upon the diffusion operation (Section 4). We evaluate the representation for completeness, observing that our encoding/decoding approach finds a good balance between the three metrics (Section 5).

However, even though our multi-view edge representation is in principle complete, in practice errors in the diffusion constraints may make it less effective. This is one reason why diffusion-based methods for depth estimation have declined in favor versus deep-learning-based methods. To reconsider this situation, we present a differentiable encoding variant that allows us to optimize representation parameters directly to improve quality with respect to the three metrics (Section 6). Using Gaussian splatting and radiative transport, this differentiable approach optimizes constraints with respect to a multi-view reprojection loss, and lets us relax our capture scenario to unstructured multi-view images. Optimizing the representation directly also lets us easily control sparsity by assessing the value of each multi-view edge. This produces accurate and compact representations for scene depth that can be competitive with deep learning methods (Section 7), where the representation is especially advantageous for sparsity and discreteness that are difficult to represent with CNNs.

Code and video results are available online at https://visual.cs.brown.edu/incompletedepth.

2 Defining a Representation

2.1 Criteria

We begin by describing properties we would like in a representation and how to measure them. To evaluate an image representation, Elder [27] determined explicitness, concision, and completeness criteria. We contextualize these to the specific setting of multi-view scene reconstruction and consider them criteria for evaluating a general-purpose image space geometric representation.

Explicitness

Important structural information should be explicitly represented. Adelson [28] describes this as representing "things" not "stuff." For instance, edges are more explicit than intensity values. Multi-view color images implicitly store the geometric structure of a scene. Depth maps explicitly store geometric structure, and this is more easily usable by later tasks, e.g., through depth map gradients we can define occlusion boundaries. Another example is spatio-angular segmentation masks [6, 29] that define piece-wise constant object surfaces. Both representations are more explicit than multi-view RGB images.

Concision

Any redundant information in the input should be discarded. This property—based on Barlow's efficient coding hypothesis [30]—is especially important for multi-view input, as multi-view images implicitly encode depth in their angular dimension with high redundancy. Many methods [23, 31–38] exploit this redundancy for higher-quality depth reconstruction than traditional stereo. However, suppose we store the reconstructed result as a depth map per input view. This retains redundancy and so per-view depth maps are not concise. Concision also relates to the sparsity of a needed representation, where textureless regions might require little representation.

Completeness

The representation should encode all relevant information to be able to support a variety of higher-level tasks. Different tasks are enabled as depth reconstruction quality increases, e.g., approximate reconstructions can suffice for novel view synthesis tasks [25, 39] but not for normal or BRDF estimation [40, 41]. Only high quality reconstruction everywhere makes stringent higher-level tasks possible.

2.2 Measurement

Elder also lists *generality*, *reliability* and *precision* as evaluative criteria for a representation. We believe these concepts are subsumed within three metrics for a representation's reconstructed depth.

Accuracy

Accuracy refers to the *correctness* of the estimated depth maps in metric terms. Correct reconstruction is a broad goal of 3D reconstruction and so accuracy is a prime metric in benchmarks [42–45]. 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.

Occlusion Edge Accuracy

For some tasks, like compositing a new scene element behind an existing one, the mean error over *all* pixels may be less relevant than the error specifically for pixels upon depth boundaries. The accuracy of occlusion edge reconstruction is measured by restricting MSE, MAE, Q25, and bad pixel $\mathrm{BP}(\cdot)$ to the vicinity of depth edges defined by gradients in ground truth. We also show precision-recall curves of these edges.

View-Consistency

View consistency in multi-view depth estimation requires the globally-consistent reconstruction of each view I_i as represented by a depth map $D_i : \mathbb{R}^2 \to \mathbb{R}$ in camera space coordinates. 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 or for output on a light field display [5, 46].

We measure view consistency by reprojecting a depth map onto a reference view and computing the variance. Let $D_0, D_1, \ldots, D_n$ represent the

depth maps for n views warped onto a target view (u, v). The view consistency at pixel s in view (u, v) is given by:

$$\mathcal{C}_{(u,v)}(s) = \frac{1}{n}\sum_{i=0}^{n}(D_i(s) - \mu_s)^2, \quad (1)$$

$$\mu_s = \frac{1}{n}\sum_{i=0}^{n} D_i(s), \quad (2)$$

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

$$\mathcal{C}_{(u,v)} = \frac{1}{S}\sum_{s=0}^{S}\mathcal{C}_{(u,v)}(s). \quad (3)$$

This form allows consistency to be evaluated for both synthetic and real world scenes.

Discussion

A fundamental trade-off exists between these three metrics. Maximizing consistency penalizes general and occlusion edge accuracy: in the extreme case, a single depth value for all pixels would provide the highest consistency with low accuracy and no occlusion edges. A continuous or smooth range of depth values allows greater precision and accuracy but leads to lower gradient edges. A complete depth representation should be general enough to optimize each metric separately.

2.3 A Complete Multi-view Edge Representation for Depth

Next, let us begin to define our representation by considering Elder's edge-based representation for images [27]. This consists of four components:

1. The 2D pixel location (x_i, y_i) of edges within an image,
2. The brightness and contrast (b_i, c_i) of intensity values at each edge pixel,
3. A 2D *gradient vector* (g_i) that indicates the direction perpendicular to an edge, and
4. A blur scale (r_i) describing the extent and attenuation of the edge.

Elder's paper shows results for grayscale images, but the concepts extend to each spectral channel.

Assuming that we can recover the representation from an image, the original image can be reconstructed with low error through anisotropic diffusion (Figure 1). The representation is more compact than an image (≈2–10% of the input pixels) as constant or smoothly-varying intensity values are not stored; instead, edges are blurred by the diffusion process according to their direction and scale to fill in these gaps. This representation can be used by tasks like editing by adding, varying, or removing points from the representation, such as removing windows from an image of a house as shown in Elder's succeeding paper [47].

Fig. 1: *Top:* Elder showed that a representation of image edges including position, brightness and contrast, gradient, and blur scale is complete as it lets us reconstruct the image via a diffusion process with low error. *Left to right:* Original image, edge locations, reconstructed image; reproduced from [27]. *Bottom:* We show that a similar approach is possible for multi-view images and provides a complete representation for scene geometry via depth.

With this inspiration, let us formulate a representation of sparse geometry 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 that allows the edge point to be uniquely identified and localized in multiple views.

Our model consists of:

1. The 2D subpixel location (x_i, y_i) of an edge in the central camera view,
2. A depth value (d_i) for the edge,
3. A 2D *surface vector* (s_i) that indicates the direction of occlusion, and
4. A confidence value (c_i) of the edge being a depth edge rather than a texture edge.

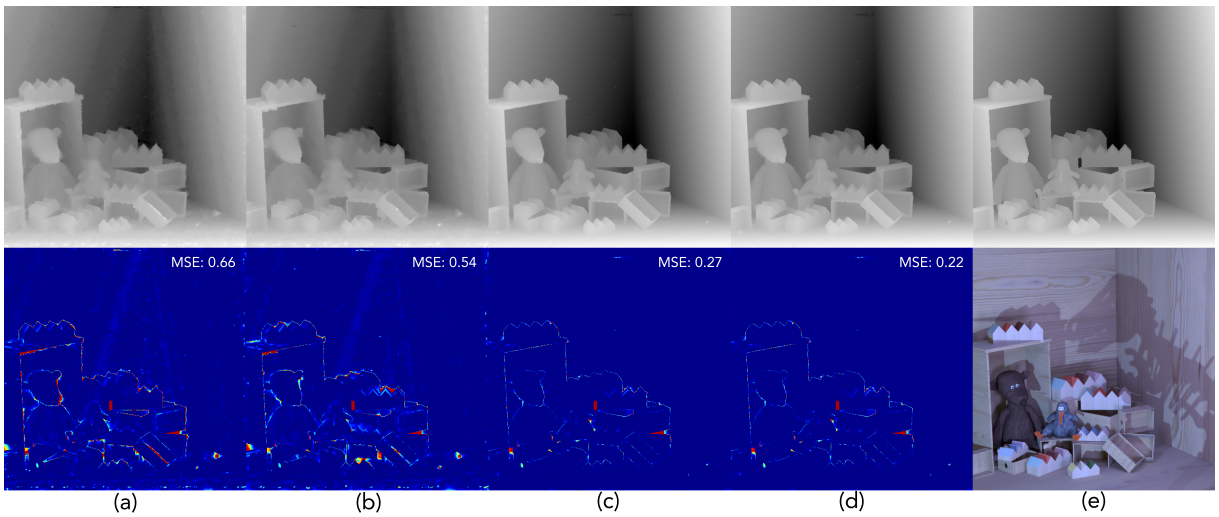


Fig. 2: *Model completeness on synthetic scene.* With correct constraints set in our multi-view edge model, diffusion-based reconstruction generates high-quality results. The drastic improvement between (b) and (c) also shows the potential of providing better constraints within an optimization approach (Section 6). **(a)** Direct diffusion of sparse depth labels with no constraint optimization. **(b)** Our initial reconstruction approach using bidirectional diffusion (Section 4). **(c)** Using an oracle to optimize the surface vector parameter. **(d)** Using an oracle to optimize both the surface vectors and the sparse edge set. **(e)** Ground truth depth and input light field view. *Please zoom in to the PDF to see the detail.*

Elder's single-view edge model [27] encodes photometric information, whereas our multi-view edge model encodes geometric information. It assumes that smoothly-varying depth regions can be interpolated from sharp neighboring edges via their surface vectors and confidences that define the direction and contribution to the local region's depth estimate (cf. to image edges blurring to contribute). Subpixel edge locations and continuous depth values are required when projecting the representation into multiple cameras at different positions, and special attention will be required to handle aliasing given the limited sensor resolution.

This representation makes structural features of the input explicit. This includes edges that provide information about changes in the visual composition of a scene, and occlusion surfaces and depth boundaries that together provide information about the 3D structure of a scene. Moreover, this information encoding is concise. Redundant measurements are avoided and noisy estimates in smooth and texture-less regions—ambiguous areas for most correspondence-based dense depth estimation methods—are discarded.

Reconstruction Methods

A reconstruction method allows us to convert our edge model into a depth map. Using only the first two model parameters along with a piece-wise constant depth reconstruction can produce view-consistent results. This can be posed as a 4D light field superpixel segmentation task [48] and is useful for tasks such as view interpolation that might not require as-highly-accurate depth estimates.

However, for tasks like light field editing or compositing, accuracy is important and requires the use of all four parameters of our model along with a smooth reconstruction method. For our work, this is achieved by *guided diffusion* of depth values from edges in both the angular and spatial domain through the solution of a constrained optimization problem [49]. The solution minimizes an energy E that encourages adherence to the sparse edge depth values—the so-called *data term* E_d—while reducing the gradient everywhere via a *smoothness term* E_s:

$$E = \lambda_d \, E_d + \lambda_s \, E_s. \tag{4}$$

Terms are traded by hyperparameters λ_d and λ_s.

Demonstration of Completeness

To highlight the potential accuracy of this model, we implement a brute-force optimization of representation parameters given a ground truth depth map on a synthetic scene (Figure 2). The multi-view edge model can represent highly accurate depth maps via diffusion-based reconstruction, so long as the model parameters are correct.

Why an Image-space Representation?

The 2D subpixel locations and depth values of our multi-view edges effectively describe a 3D point cloud. This can be projected into 2D and, with the surface vector and confidence parameters, diffused at any particular image resolution. Thus, the representation can offer a one-to-one correspondence between color pixels and geometry, which is useful for tasks like image editing, AR, and mixed reality that operate on image pixels. Moreover, they can typically more efficiently capture fine details than a mesh or a voxel grid, the latter being expensive in terms of resolution. Neural fields may offer these advantages but are computationally expensive to compute and difficult to edit once computed [50].

Limitations of Scope

Multi-view images encode both photometric and geometric information; we focus on the geometric information encoded in our proposed model. For this, we show the completeness and concision properties empirically and do not quantify the explicitness of our model. Our work uses edges and points recovered assuming the scene is composed of Lambertian surfaces. In practice, we demonstrate empirically that this model is often sufficient for editing tasks of real-world scenes captured with narrow baseline camera systems. Finally, we assume that edges that lead to depth estimates are not blurred.

3 Recovering Model Parameters for Light Fields

We consider the case of structured light fields as might be captured by a lenslet-array camera (e.g., Lytro). In Section 6, we will show examples that extend this to unstructured multi-view images. For now, structured light fields provide simpler ways to extract multi-view edge information.

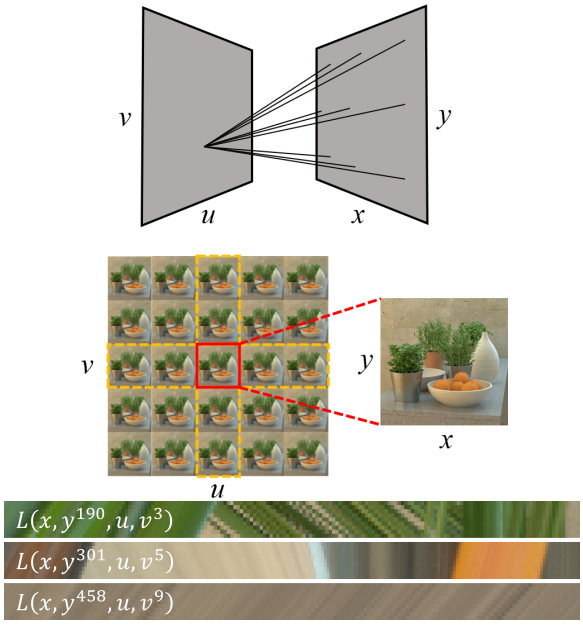


Fig. 3: $LF(x, y, u, v)$ defines a two-plane light field parameterization with central views $\mathcal{H}, \mathcal{V}$ outlined in dashed yellow. Light rays are shown as black lines (top). We show epipolar plane images (EPIs) sliced from the 4D volume below.

Given a 4D light field $LF(x, y, u, v)$ (Figure 3), we define the central horizontal row of views $\mathcal{H} = LF(x, y, u, v_c)$ and central vertical column of views $\mathcal{V} = LF(x, y, u_c, v)$. We call $\mathcal{H}, \mathcal{V}$ the 'cross-hair' views. Each view $I \in \mathcal{H}$ contains a set of epipolar plane images (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 an EPI, where the depth of the point determines the slope of the line.

3.1 Finding 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 EPI. For robust occlusion handling, we must accurately detect the intersections of lines in EPIs (Figure 4). However, classical edge detectors like Canny [51] and Compass [52] often generate curved or noisy responses at line intersections, which makes later line fitting and occlusion localization difficult. Instead,

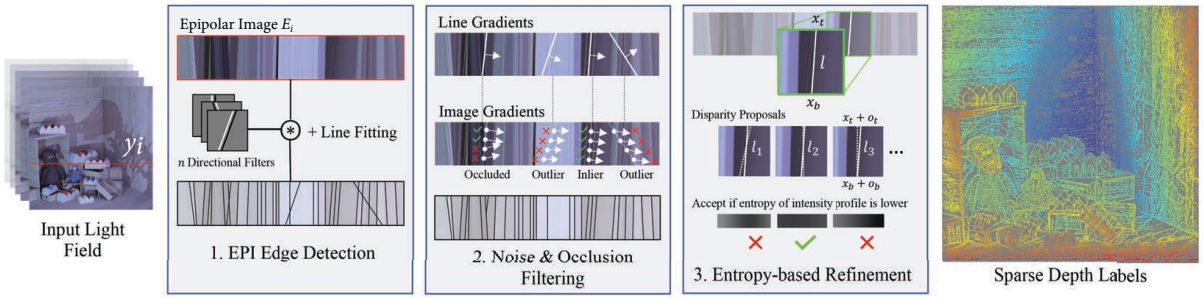


Fig. 4: EPI edges provide both the location and disparity labels of a sparse point set $\mathcal{P}$. Thus, the first stage of multi-view edge estimation consists of EPI edge detection and line fitting. In the second stage, we compare the direction of each EPI line with underlying image gradients to 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.

we propose an EPI-specific method to detect a line set $\mathcal{L}$ for each EPI of the central horizontal and vertical views of our light field. *Note:* We describe line detection for the central horizontal views; central vertical views follow similarly.

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 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 [53]. Our final C map has clean intersections.

3.2 Multi-view Edge Refinement

3.2.1 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.

3.2.2 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:

$$\sum_{\mathbf{s} \in S_l} \mathbb{1}\left(\frac{\nabla I(\mathbf{s})(\nabla l)^T}{\|\nabla I(\mathbf{s})\| \|\nabla l\|} > cos(\tau_f) \right) < k, \qquad (5)$$

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 $1 \leq c \leq$ 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.

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:

$$v(l) = \mathbb{1}\left(\frac{\nabla I(\mathbf{s}_c)(\nabla l)^T}{\|\nabla I(\mathbf{s}_c)\|\|\nabla l\|} > cos(\tau_v)\right), \quad (6)$$

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

3.2.3 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 $\mathcal{L}_c = \{l \in \mathcal{L} \mid v(l) = 1\}$.

$$E(l) = \sum_{\mathbf{s} \in S_l} -P(I(\mathbf{s})) \log_2(P(I(\mathbf{s}))), \quad (7)$$

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

We minimize $E(l)$ through 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 $l_j = (x_t + o_t, x_b + o_b)$ (Figure 4). This is accepted with probability one if $E(l_j) < E(l_{j-1})$. We use $t = 0.88$, $\alpha = 0.15$, and run ten search iterations.

Then, the resulting disparity estimates are 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 $\mathscr{S}$ around p:

$$f(p_d) = \frac{1}{W} \sum_{q \in \mathscr{S}} \mathcal{N}_{\sigma_s}(\|p_s - q_s\|)\mathcal{N}_{\sigma_d}(p_d - q_d) \cdot \mathcal{N}_{\sigma_c}(\|p_c - q_c\|)p_d, \quad (8)$$

where the normalization factor W is given by

$$W = \sum_{q \in \mathscr{S}} \mathcal{N}_{\sigma_s}(\|p_s - q_s\|)\mathcal{N}_{\sigma_d}(p_d - q_d) \cdot \mathcal{N}_{\sigma_c}(\|p_c - q_c\|). \quad (9)$$

In theory, the parameters σ_c and σ_d depend on the scene content and the maximum disparity. In practice, 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.

3.3 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. In Figure 5(a), two different scene configurations generate a similar EPI and, thus, similar multi-view edge parameters. The problem is exacerbated by the fact that in practice we do not even know the location of the discontinuity—that is, the edge—precisely in pixels when using Prewitt filters: Figure 5(b) shows that a Prewitt filter convolved with a set of pixels representing an edge will generate a non-zero activation along two-pixel columns.

Using a model without surface vector parameters results in significant errors around edges when using a smooth reconstruction method. 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 [9] deal with this by assuming that sparse labels do not lie on depth edges so that neighboring pixels have a similar

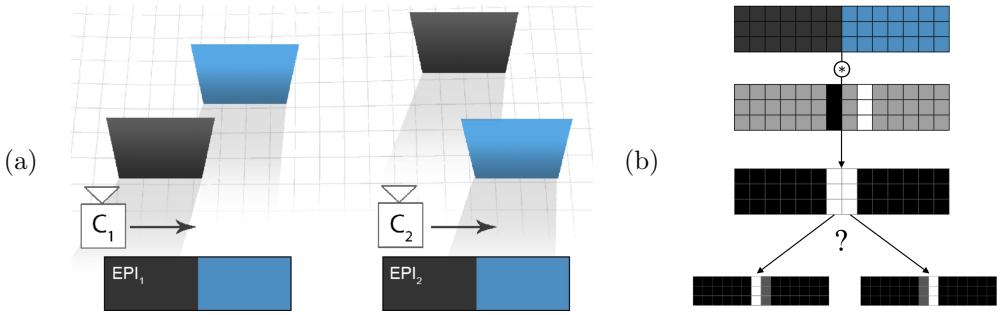


Fig. 5: Sparse labels at edges are difficult to propagate because the edge is weakly localized at the boundary of two projected surfaces. As a result, labels may be assigned to the incorrect side of a depth boundary. **(a)** Two different scene configurations captured with cameras C_1 and C_2 may generate 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.

label. Yucer et al. [54] 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 a method that uses a bidirectional 'backward-forward' diffusion process to generate a surface vector parallel to the image gradient.

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. [54] compare the variation in texture on both sides of an edge as the view changes: the background 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 a 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 (Figure 10). 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:

$$\hat{D}[\mathcal{A}] = \underset{D}{\operatorname{argmin}} \sum_{p \in \mathcal{A}} E_d(p) + \sum_{(p,q) \in \mathscr{S}} E_s(p,q), \tag{10}$$

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

$$E_d(p) = \lambda_d(p) \big\| S[\mathcal{A}](p) - D(p) \big\|, \tag{11}$$

$$E_s(p,q) = \lambda_s(p) \big\| D(p) - D(q) \big\|, \tag{12}$$

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

Equation (10) represents a standard Poisson problem, and we solve it using an implementation of the LAHBPCG solver [55] by posing the constraints in the gradient domain [56]. We begin by defining two sets formed from opposite offset directions $\nabla I(p)$ and $-\nabla I(p)$:

$$\mathcal{P}_f = \{p + \nabla I(p) \;\forall\; p \in \mathcal{P}\}, \tag{13}$$

$$\mathcal{P}_b = \{p - \nabla I(p) \;\forall\; p \in \mathcal{P}\}, \tag{14}$$

where $\nabla I(p)$ is the gradient of the central light field view at point p. Then, we solve Equation (10) for both offset directions $\hat{D}[\mathcal{P}_f]$ and $\hat{D}[\mathcal{P}_b]$ using

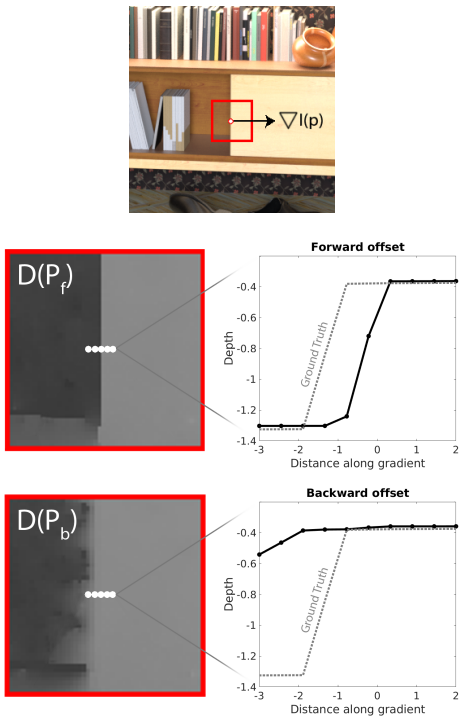


Fig. 6: *Top:* Given an edge point p with image gradient $\nabla I(p)$ and depth label p_d, we would like to determine which side of the edge to propagate p_d. We generate images $\hat{D}[\mathcal{P}_f]$ (*middle*), and $\hat{D}[\mathcal{P}_b]$ (*bottom*) by solving a Poisson optimization problem with diffusion direction $p + \nabla I(p)$ and $p - \nabla I(p)$ respectively. The correct diffusion direction (*middle*; forward) 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.

data and smoothness weights:

$$\lambda_d(p) = \begin{cases} 10^6 & \text{if } p \in \mathcal{A}, \\ 0 & \text{otherwise,} \end{cases} \tag{15}$$

$$\lambda_s(p) = \frac{1}{\|\nabla I(p)\| + \epsilon}. \tag{16}$$

Given both solutions, we compare the normalized depth profile around each point $p \in \mathcal{P}$ along $\nabla I(p)$ in $\hat{D}[\mathcal{P}_f]$ and $\hat{D}[\mathcal{P}_b]$. Figure 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 $\mathbf{F} = [-1\ -1\ +1\ +1]$.

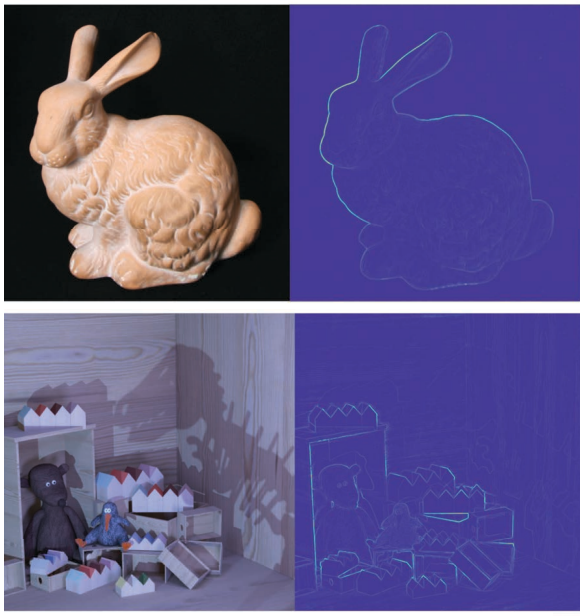


Fig. 7: Estimated depth edge confidence λ_s, which successfully ignores texture edges on the bunny in the top scene and shadow edges on the dinosaur skeleton in the bottom scene.

3.4 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 7). Texture edge gradient remains low in both passes. 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.

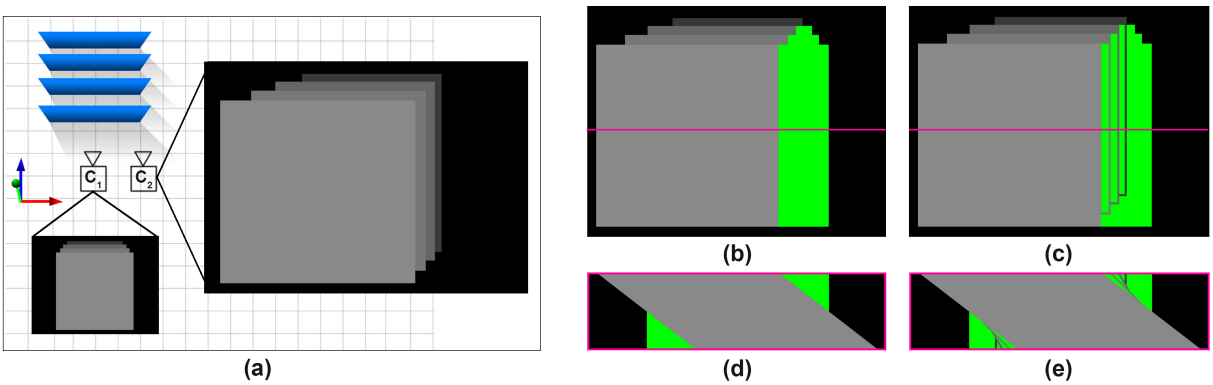


Fig. 8: **(a)** Ground-truth disparity maps from two different camera positions C_1 and C_2. **(b)** Naively attempting to generate the output of C_2 by reprojecting C_1 results in large holes (in green). **(c)** Our method uses depth edges to guide disparity propagation in such disoccluded regions. The EPIs corresponding to the highlighted row are shown in **(d)** and **(e)**. The EPI in **(e)** constitutes our depth EPI D_o.

4 Occlusion-aware 4D Depth Reconstruction

Section 3 estimates accurate view-consistent depth for multi-view edges; next, we present an occlusion-aware reconstruction method to invert our multi-view edge encoding into depth maps. By showing that our representation encodes all relevant information required for generating accurate and view-consistent depth reconstructions with strong occlusion edges, we establish completeness. Practically, this accomplishes light field depth estimation. Methods often strive for geometric accuracy without considering occlusion edges, which are especially important for handling visibility in computational photography applications. Moreover, aggregating information across many light field views produces high accuracy depth, but most approaches estimate only a depth map for the central view. We show how to estimate depth for every pixel in the light field, e.g., for editing a light field photograph where every output view will be seen on a light field display.

Challenges

Our reconstruction method is an anisotropic diffusion process that fills in missing regions. Depth diffusion is a long-standing problem in which it is difficult to ensure both consistency and correctness in non-central-view disoccluded regions because these regions are sampled less by the camera. As such, methods that estimate depth for every light field pixel are often not strictly occlusion aware, or are expensive [31, 34] due to the extra angular dimension increasing data and computation costs [34, 57]. Researchers have tried to overcome this barrier by learning data-driven priors with deep learning. Jiang et al. [35, 58] presented the first practical view consistent method based on deep learning. Learning requires training data and may overfit to scenes or capture scenarios [59]. Our multi-view edge encoding and smooth reconstruction method is a counterpart first principles method with no learned priors.

One solution to reconstruct per-view depth from our multi-view edge encoding would compute a disparity map for each sub-aperture view separately. However, this is challenging for boundary views and is inefficient in terms of redundant computation and due to the spatial domain constraints or regularization required to ensure consistency across views. Another solution might calculate a disparity map for a source view and then reproject it into all other views. However, this fails to handle scene points that are not visible in the source view. Such points cause holes for disocclusions, or lead to inaccurate disparity estimates when the points lie on an occluding surface. While most methods try to deal with holes through inpainting, occluding surfaces are more difficult to deal with as the occluding surface may have a depth value (or label) not seen in the original view. Thus, techniques like diffusion alone are insufficient to prove or disprove the completeness of our edge model.

Approach

Our proposed method deals with this issue of depth consistency in subviews of light fields via an occlusion-aware diffusion process (Figure 8). As our occlusion-aware multi-view edges persist across views, we can use them as reliable guides for an angular inpainting process that fills any holes in reprojected views. As inpainting occurs in the angular rather than the spatial domain of the light field, this ensures depth view consistency by design while accounting for the visibility of points in disoccluded regions. This avoids trying to constrain or regularize view consistency after estimating depth spatially, and so aids efficiency.

4.1 Central View Depth Estimation

From our multi-view edge model, we have 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 $\mathcal{A} = \{p \pm \nabla I(p) \; \forall \; p \in \mathcal{P}\}$ be the sparse set of points $\mathcal{P}$ offset by the surface vector. We use $S[\mathcal{A}]$ 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:

$$\hat{D}[\mathcal{A}] = \underset{D}{\operatorname{argmin}} \sum_{p \in \mathcal{A}} E_d(p) + \sum_{(p,q) \in \mathscr{S}} E_s(p,q), \tag{17}$$

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

$$E_d(p) = \lambda_d(p) \big\| S[\mathcal{A}](p) - D(p) \big\|_2^2, \tag{18}$$

$$E_s(p,q) = \lambda_s(p) \big\| D(p) - D(q) \big\|_2^2, \tag{19}$$

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

$$\lambda_d(p) = \omega \, \exp(a \lambda_e(p)) \tag{20}$$

$$\lambda_e(p) = \max_{\{\hat{D}[\mathcal{P}_f], \hat{D}[\mathcal{P}_b]\}} \| N_p \circledast \mathbf{F} \|. \tag{21}$$

where N_p is a set of pixels around p, and $\mathbf{F}$ is the step filter $[-1 \; -1 \; +1 \; +1]$. The parameters are set to $\omega = 1.5 \times 10^2$ and $a = 3$ for all scenes. The smoothness term $\lambda_s(p)$ is provided by the depth edge confidence at every pixel (Section 3.4).

Equation (17) represents a standard Poisson problem, and we solve it using the Locally Adaptive Hierarchical Basis Preconditioning Conjugate Gradient (LAHBPCG) solver [55] by posing the constraints in the gradient domain as proposed by Bhat et al. [56]. The result $\hat{D}[\mathcal{A}]$ provides a dense depth estimate for the central view.

4.2 Cross-hair View Projection

Our EPI line-fitting algorithm works on EPIs in the central cross-hair views—that is, the central row and column of light field images. Computing this on other rows and columns can be expensive, and the central set is usually sufficient to detect visible surfaces in the light field [60]. 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. [61], using a weighted median filter [62] 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.

4.3 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 (Section 3.1) that are robust to occlusions in a single view as they exist 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 8). Again, we formulate diffusion as a constrained quadratic optimization problem:

$$\hat{D} = \underset{D}{\operatorname{argmin}} \sum_{p \in D} E_d(p) + \sum_{(p,q) \in \mathcal{S}} E_s(p,q), \quad (22)$$

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:

$$E_d(p) = \lambda_d(p) \left\| D(p) - D_o(p) \right\|_2^2, \quad (23)$$

$$E_s(p,q) = \lambda_s(p,q) \left\| D(p) - D(q) \right\|_2^2, \quad (24)$$

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

$$\lambda_s(p,q) = \frac{c}{\|\nabla I(p)\| + \epsilon}, \quad (25)$$

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

$$\lambda_d(p) = \begin{cases} 15 & \text{if } p \in \mathcal{C}, \\ \lambda_e(p) & \text{if } p \in \mathcal{L}, \\ 0 & \text{otherwise,} \end{cases} \quad (26)$$

where $\lambda_e(p)$ is the edge-importance weight from Equation 21, 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 (22) defines the optimal disparity map $\hat{D}$ as one that minimizes divergence from the labeled data (Equation. (23)) while being as smooth as possible. Equation (24) 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 (25)). This allows pixels across edges to have a disparity difference without being penalized. The data weight (Equation (26)) is determined empirically and works for all datasets.

Equation (22) is again a standard Poisson optimization. We solve this using the LAHBPCG solver [55] by posing the data and smoothness constraints in the gradient domain [56].

4.4 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 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 light field view.

5 Evaluating the Reconstruction

At this point, we evaluate our approach for recovering the multi-view edge model and for reconstructing depth from this model (Figure 9).

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 [64]. 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 [65] and the New Stanford Light Field Archive [66]. The EPFL light fields are captured with a Lytro Illum and consist of 15×15 views of 434×625 pixels each. As edge views tend to be noisy, we only use the central 7×7 views. The Stanford Archive scenes 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.

Baselines

We compare ours to three non-learning-based methods: the defocus and correspondence cues methods by Jeon et al. [57] and Wang et al. [33], and the spinning parallelogram operator of Zhang

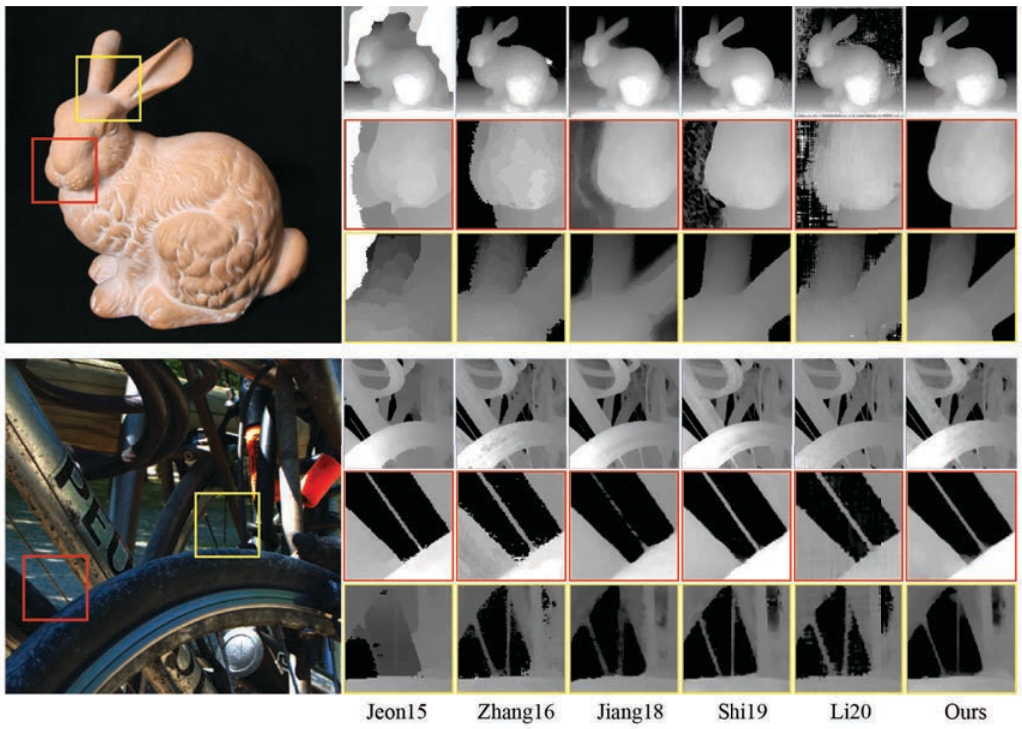


Fig. 9: Occlusion edges in disparity maps. *Top:* Stanford dataset light field captured with a camera rig. *Bottom:* EPFL light field from a Lytro Illum. *Left to right:* Jeon et al. [57], Zhang et al [34], Jiang et al. [35], Shi et al. [63], and ours.

et al. [34]. We also compare with the learning-based methods of Jiang et al.. [35], Shi et al.. [63], and Li et al. [59]. Both Shi et al. and Jiang et al. use the deep-learning-based Flownet 2.0 [67] 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 [9]: this uses COLMAP, which fails on typical skew-projected light field data.

5.1 Quantitative Metrics

Occlusion Edge Accuracy

Qualitatively, our method produces sharper and more accurate occlusion edges than state-of-the-art light field depth estimation methods.

Figure 10 visualizes occlusion boundaries as depth gradients. The learning-based methods of Shi et al. and Li et al. generate spurious boundaries in textureless regions, and the approach of Yucer et al. [54] fails without thousands of views. We also evaluate our edges quantitatively on four scenes from the synthetic HCI Dataset [64] via ground truth disparity for the central view (Figure 11 and Table 2). Although our Q25 error is higher, our method has high boundary-recall precision and a lower average MSE 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 (Section 3.2) in both vertical and horizontal EPIs. In practice, we found no advantage of doing so.

Diffusion Gradients as Self-supervision

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*

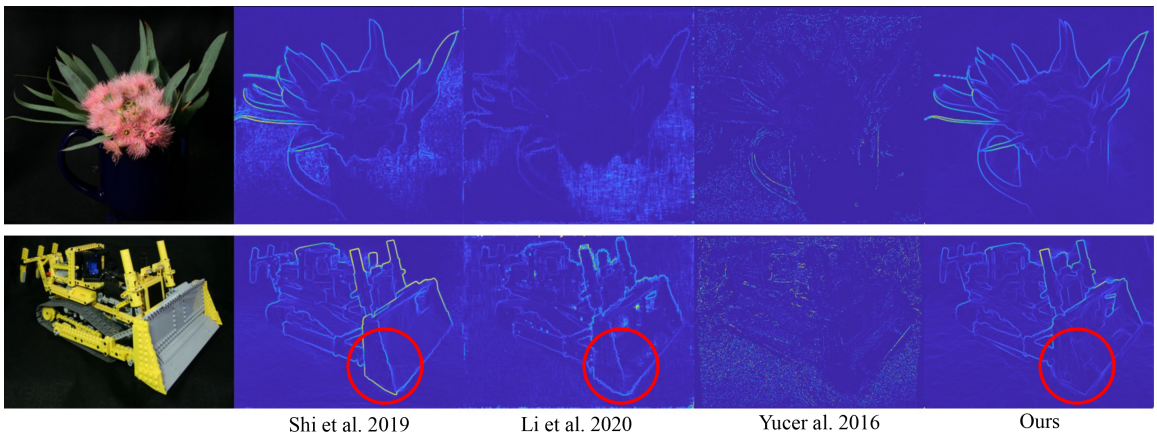


Fig. 10: Visualizing occlusion edges as gradients of disparity maps. *Left to right:* Shi et al. [63], Li et al [59], Yucer et al. [54], and ours. *Bottom row, red circle:* the learning-based methods hallucinate a strong depth edge on the plow even though it is in contact with the black ground cloth at the same depth. Yucer et al.'s method fails in the absence of many views.

Table 1: 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 Field	MSE × 100		Q25	
	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**

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 1 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.

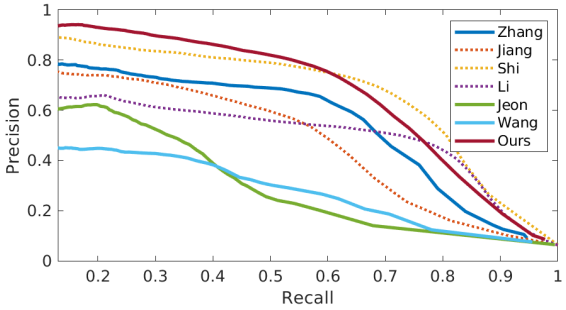


Fig. 11: Average precision-recall curves of depth boundaries for all baselines (HCI dataset). Learning-based methods are shown as dotted lines. Our approach consistently outperforms traditional algorithms [33, 34, 57] and the learning-based method of Jiang et al. [35], while outperforming Shi et al. [63] and Li et al. [59] at medium-to-low recall rates. Area under curve is highest for both our method and Shi et al., at 0.63.

Accuracy

Table 2 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 error. For baseline techniques to have higher MSE but fewer

Table 2: 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 mean squared error; Q25 is 25th percentile of the absolute error.

Light Field	MSE × 100							Q25							Run time (s)						
	[57]	[34]	[35]	[63]	[59]	[33]	Ours	[57]	[34]	[35]	[63]	[59]	[33]	Ours	[57]	[34]	[35]	[63]	[59]	[33]	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

bad pixels means they must have larger outliers. This is confirmed by the error plots in Figure A5.

View Consistency

Figure 12 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 a 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 13 plots runtime versus view consistency across our three datasets. Our method produces comparable or better consistency and is quicker, being 2–4× faster than Jiang et al.'s methods per view for equivalent error.

Qualitative

Figures A5 and A6 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 a 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.

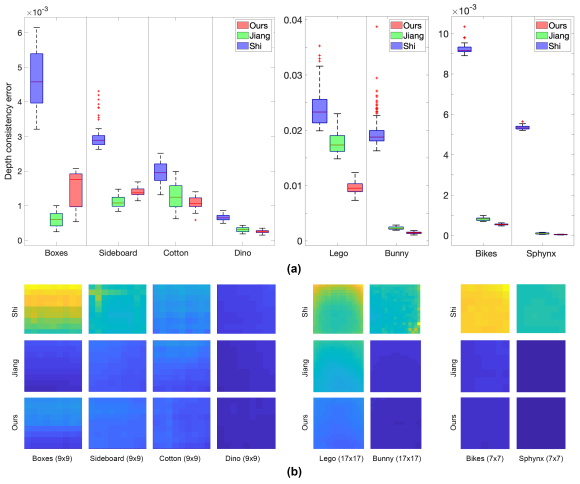


Fig. 12: Quantitative view consistency comparison of our method and Jiang et al. [35] and Shi et al. [63]. While the method of Jiang et al. enforces cross-view consistency, Shi et al. operates on each view individually and has no explicit consistency constraint. **(a)** For each light field, we plot summary statistics over $\mathcal{C}_{(u,v)}$ for all views (u, v) in the light field (Equation (3)). **(b)** The angular distribution of the error over all views.

While these results do seem to indicate that our edge model encodes all relevant information required for generating accurate and view-consistent depth reconstructions with strong occlusion edges, the accuracy criterion is not always satisfied. In particular, as we estimate depth explicitly only around potential occlusion boundaries our method has lower accuracy in non-edge regions, reflected by the Q25 error. In the next section, we describe a differentiable variant of our model that addresses this shortcoming.

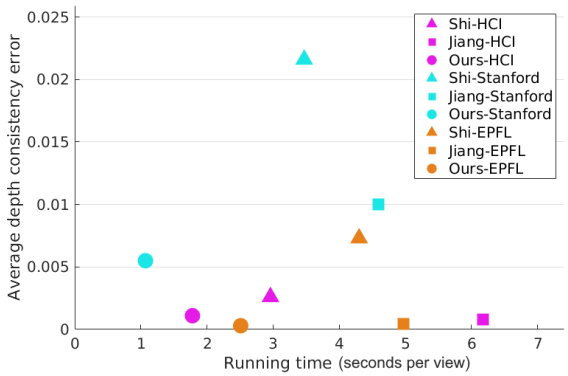


Fig. 13: Average depth consistency error and runtimes for the three assessed datasets. Our method runs consistently faster than the baselines while having comparative or better consistency. The errors across datasets are shown in absolute terms.

6 A Differentiable Model Implementation

Even with our efforts, it can be difficult to identify and filter out noisy or erroneous points from a sparse edge set, and diffusion from noisy points produces results with lower accuracy. However, with correct noise-free constraints, Section 2.3 showed that our edge model and reconstruction produce extremely low error and are significantly better than current dense processing methods.

So, how can we handle noisy edges? We present a method to optimize edge constraints *through* a set of linear equations representing the solution to the standard Poisson problem of depth diffusion (Figure 14). For this, we develop a differentiable and occlusion-aware image-space representation for a sparse set of scene edges that allows us to solve the inverse problem efficiently using gradient descent. This section expands upon EPI edges to consider unstructured multi-view images too in which only sparse points in correspondence are easily discovered: we treat each point as a Gaussian to be splatted into the camera, then we use the setting of radiative energy transfer through participating media to model the occlusion interaction between Gaussians. This lets us optimize over point position, depth, and weight parameters via reprojection error from RGB images.

In Section 7, we show that this method reduces diffusion errors caused by noisy or spurious points, and allows us to optimize a sparse point set. Further, we discuss why edges are difficult to optimize via reprojection from depth maps. Finally, in comparison to both image processing and deep learning baselines, our method shows competitive performance, especially in reducing bad pixels.

6.1 Depth via Differentiable Diffusion

Given a set $\mathcal{I} = \{\mathrm{I}_0, \mathrm{I}_1, ..., \mathrm{I}_n\}$ of n multi-view images 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. We achieve this by optimizing the set of scene points so that their diffused image minimizes a reprojection error across $\mathcal{I}$.

We begin by restating the task of reconstruction via diffusion within the context of a sparse set of 3D scene points $\mathcal{P}$ and a camera. 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 want a dense depth map D_o by penalizing the difference from the sparse labels S while promoting smoothness by minimizing the gradient $\nabla\mathrm{D}$:

$$\mathrm{D}_o = \underset{\mathrm{D}}{\operatorname{argmin}} \iint_\Omega \lambda(x,y)\left(\mathrm{D}(x,y) - \mathrm{S}(x,y)\right)^2 + \vartheta(x,y)\|\nabla\mathrm{D}(x,y)\|\,dx\,dy, \tag{27}$$

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 (27) in 3D is expensive and complex, needing for example voxels or a mesh. More practically, the energy in Equation (27) is minimized over a discrete pixel grid with indices x, y:

$$\mathrm{D}_o = \underset{\mathrm{D}}{\operatorname{argmin}} \sum_{(x,y)} \Bigg(\lambda^\mathbb{Z}(x,y)\left(\mathrm{D}(x,y) - \mathrm{S}^\mathbb{Z}(x,y)\right)^2 + \sum_{(u,v)\in\mathcal{N}(x,y)} \vartheta^\mathbb{Z}(x,y)\|\mathrm{D}(u,v) - \mathrm{D}(x,y)\| \Bigg), \tag{28}$$

where $\mathcal{N}(x,y)$ defines a four-pixel neighborhood around (x,y), and $\lambda^\mathbb{Z}$, $\vartheta^\mathbb{Z}$ and $\mathrm{S}^\mathbb{Z}$ are, respectively,

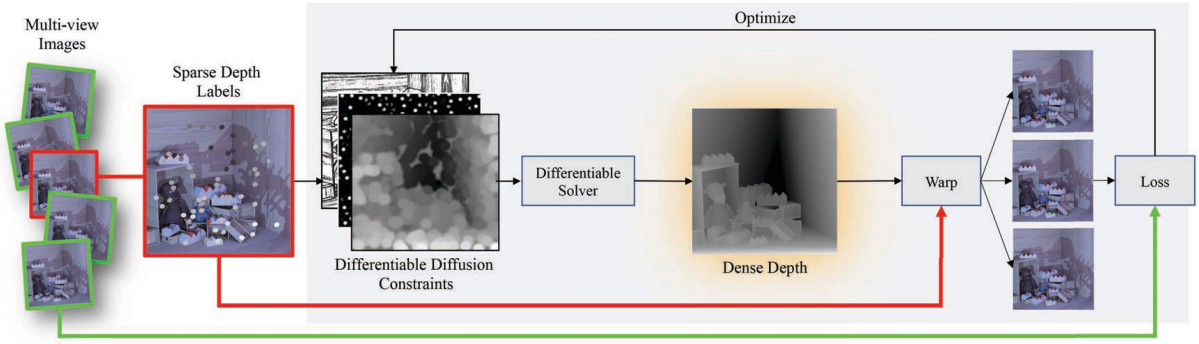


Fig. 14: From a set of noisy sparse depth samples, our method uses differentiable splatting and diffusion to produce a dense depth map. Then, we optimize point position, disparity, and weight against an RGB reprojection loss. This reduces errors in the initial set of points.

the discrete counterparts of the splatting function λ, the local smoothness weight ϑ, and the depth label in $\mathbb{R}^2$, S.

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 $\mathbf{K}\mathbf{x}$ 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 15 shows, we require 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.

6.2 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 [68, 69]. 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 15(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 $\mathbf{K}\mathbf{x}$ 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. Thus, its behavior is closer to that of a delta function, and it minimizes the "leakage" of weight onto neighboring pixels (Figure 15c). But unlike a delta, it is differentiable and can be sized to match some pixel extent so that points do not vanish (Figures 15d & 15e). Thus, we define the discrete functions:

$$\mathrm{S}^{\mathbb{Z}}(x,y) = \sum_{\mathbf{x}\in\mathcal{P}} \alpha_{\mathbf{x}}(x,y)\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y), \tag{29}$$

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. 6.3), 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 $\mathbf{K}\mathbf{x} = (x_{\mathbf{x}}, y_{\mathbf{x}})$. We define $\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}$ as:

$$\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y) = Z_{\mathbf{x}} \exp\left(-\frac{(x-x_{\mathbf{x}})^2 + (y-y_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} \right). \tag{30}$$

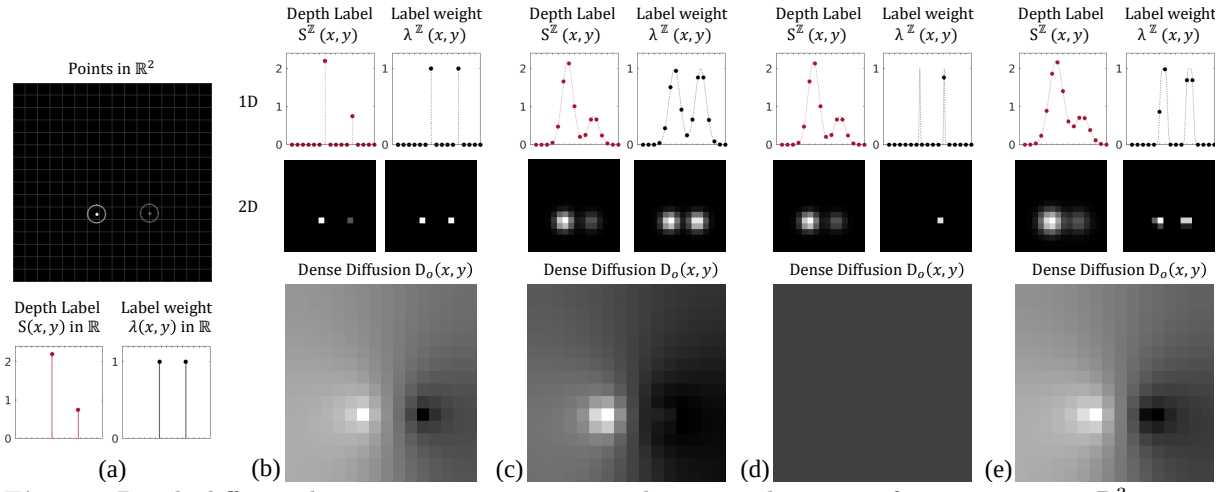


Fig. 15: Depth diffusion happens in image space, so how we splat a set of scene points in $\mathbb{R}^3$ onto a pixel grid in $\mathbb{Z}^2$ has a significant impact on the results. **(a)** The image-space projection of scene points are Dirac delta functions which cannot be represented in discrete pixels. **(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 depth labels. **(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 also being differentiable.

Similarly, the discrete label weights are defined as:

$$\lambda^{\mathbb{Z}}(x,y) = \sum_{\mathbf{x}\in\mathcal{P}} \alpha_{\mathbf{x}}(x,y)\lambda^{\mathbb{Z}}_{\mathbf{x}}(x,y), \tag{31}$$

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

$$\lambda^{\mathbb{Z}}_{\mathbf{x}}(x,y) = w_{\mathbf{x}}\exp\left(-\frac{(x-x_{\mathbf{x}})^2+(y-y_{\mathbf{x}})^2}{2\sigma_\lambda^2}\right)^p, \tag{32}$$

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

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 (Section 6.3). Using higher-order Gaussians for depth requires differentiating a transmission integral (upcoming Equation (33)),

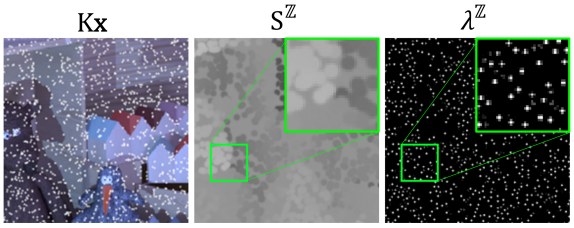


Fig. 16: *Left:* The image-space projection K**x** of scene points $\mathbf{x} \in \mathcal{P}$ plotted in white. *Middle:* Our differentiable labeling function $S^{\mathbb{Z}}$ accurately splats depth labels while handling occlusion. *Right:* A higher-order Gaussian representation of $\lambda^{\mathbb{Z}}$ is differentiable, and provides weights that are close to point masses without any points vanishing during discretization.

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).

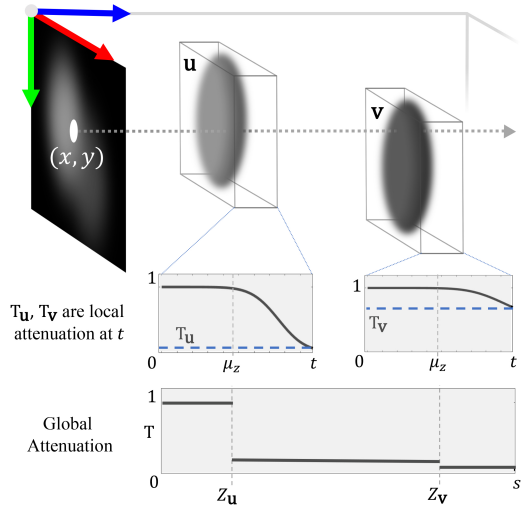


Fig. 17: We estimate depth labels at points overlapping in xy using a radiative transfer formulation with Gaussians in orthographic space. If σ_Z is small, the influence of the points $\mathbf{u}$ and $\mathbf{v}$ in scene space is restricted to small windows around $Z_{\mathbf{u}}$ and $Z_{\mathbf{v}}$. As $\sigma_Z \to 0$, we assume the density contribution at any point s along a ray comes from a single Gaussian. This allows the attenuation effect of each Gaussian to be calculated independently. The global attenuation function at s can be calculated as the product of local attenuation for all points with $z < s$.

6.3 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 (29) and (31) 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 setting of radiative energy transfer through participating media [68].

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:

$$\mathrm{T}(x, y, s) = \exp\Bigg(- \int_0^s \rho \sum_{\mathbf{x} \in \mathcal{P}} \exp\Bigg(- \Big(\frac{(x - x_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} + \frac{(y - y_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} + \frac{(z - Z_{\mathbf{x}})^2}{2\sigma_Z^2} \Big) \Bigg) dz \Bigg). \tag{33}$$

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 17). 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 (33) as:

$$\begin{aligned} \mathrm{T}(x, y, s) &= \prod_{\mathbf{x}} \exp\Bigg(- \int_0^t \rho \frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x, y)}{Z_{\mathbf{x}}} \cdot \\ &\qquad \exp\Bigg(- \frac{(z - \mu_z)^2}{2\sigma_Z^2} \Bigg) dz \Bigg) \\ &= \prod_{\mathbf{x}} \mathrm{T}_{\mathbf{x}}(x, y), \end{aligned} \tag{34}$$

where the product is over all $\mathbf{x} \in \mathcal{P} \mid Z_{\mathbf{x}} < s$. By looking again at Equation (30), 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:

$$\mathrm{T}_{\mathbf{x}}(x,y) = \exp\left(\sqrt{\frac{\pi}{2}}\frac{\sigma_Z\ \rho\ \mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)}{Z_{\mathbf{x}}} \left(-\operatorname{erf}\left(\frac{\mu_z}{\sigma_Z\sqrt{2}}\right) - \operatorname{erf}\left(\frac{t-\mu_z}{\sigma_Z\sqrt{2}}\right)\right)\right) \tag{35}$$
$$= \exp\left(c\frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)}{Z_{\mathbf{x}}}\right),$$

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 [68]:

$$\mathrm{S}^{\mathbb{Z}}(x,y) = \int_0^{\infty} \mathrm{T}(s,x,y)\mathrm{a}(s,x,y)\mathrm{P}(s,x,y)\,ds, \tag{36}$$

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:

$$\mathrm{S}^{\mathbb{Z}}(x,y) = \int_0^{\infty} \mathrm{T}(x,y,s) \sum_{\mathbf{x}\in\mathcal{P}} Z_{\mathbf{x}}\,\rho \exp\left(-\frac{(x-x_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} + \frac{(y-y_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} + \frac{(s-Z_{\mathbf{x}})^2}{2\sigma_Z^2}\right) ds. \tag{37}$$

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 $\mathbf{x}$, and estimate the integral by sampling s at step length $\mathrm{d}s$ over a small interval $\mathcal{N}_{\mathbf{x}}$ around each $Z_{\mathbf{x}}$:

$$\mathrm{S}^{\mathbb{Z}}(x,y) = \sum_{\mathbf{x}\in\mathcal{P}} \sum_{s\in\mathcal{N}_{\mathbf{x}}} \mathrm{d}s\,\mathrm{T}(x,y,s)\ \rho\ \mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)\cdot \exp\left(-\frac{(s-Z_{\mathbf{x}})^2}{2\sigma_Z^2}\right)$$
$$= \sum_{\mathbf{x}\in\mathcal{P}} \mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y) \sum_{s\in\mathcal{N}_{\mathbf{x}}} \mathrm{d}s\,\mathrm{T}(x,y,s)\ \rho\ \cdot \exp\left(-\frac{(s-Z_{\mathbf{x}})^2}{2\sigma_Z^2}\right)$$
$$= \sum_{\mathbf{x}\in\mathcal{P}} \alpha_{\mathbf{x}}(x,y)\ \mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y). \tag{38}$$

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

$$\alpha_{\mathbf{x}}(x,y) = \frac{\rho\,\mathrm{d}s}{Z_{\mathbf{x}}} \sum_{s\in\mathcal{N}_{\mathbf{x}}} \mathrm{T}(s,x,y)\ \rho\ \cdot \exp\left(-\frac{(s-Z_{\mathbf{x}})^2}{2\sigma_Z^2}\right). \tag{39}$$

6.4 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 (Equation (28)). 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 $\mathcal{P}$. This provides strong constraints on both the initial value of these functions and on how they are updated at each step of the optimization, 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 generates high-quality depth maps, and shows the potential of our differentiable sparse point optimization and diffusion method. Please see the appendix for details.

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

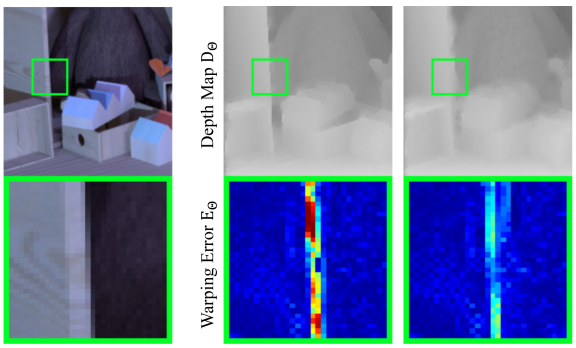


Fig. 18: Our everyday intuition says that depth edges should be sharp, but a limited sampling rate blurs them in the RGB input **(a)**. This can cause unintended high error during optimization via losses computed on RGB reprojections. In **(b)**, the depth edge is sharp, but reprojecting it into other views via warping 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 (Equation (41)).

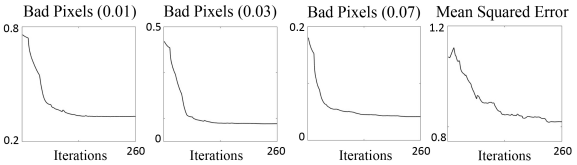


Fig. 19: Over optimization iterations, mean squared error reduces and 'bad pixels' are significantly suppressed. Most remaining errors lie along edges, where depth is not well defined (Figure 18).

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 calculated as:

$$E_\Theta(x,y) = \frac{1}{\sum_i M^i_\Theta(x,y) + \epsilon} \cdot \sum_i \left(|I(x,y) - \mathcal{W}_\Theta[I^i](x,y)| \; M^i_\Theta(x,y) \right), \quad (40)$$

where $M^i_\Theta(x,y)$ 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 18). 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. [70] to encourage depth to be guided by image edges, and a structural self-similarity error [71] E_{SSIM} which is used to regularize warping error. Our final loss function is:

$$L(\Theta) = \sum_{(x,y)} \Big(E_\Theta(x,y) + E_S + E_{SSIM} - \nabla E_\Theta(x,y) \Big). \quad (41)$$

7 Evaluating the Differentiable Approach

Additional Datasets

Along with the previous scenes (Section 5), we add new synthetic *Living Room* and *Piano* scenes with more realistic lighting, materials, and depth ranges. We path trace these with Arnold in Maya at 512×512 pixels. Each light field has 9×9 views, and each multi-view image set has five unstructured views with a mean baseline of ≈ 25 cm.

7.1 Light Fields

While learning-based methods [35, 59, 63] tend to do well on the HCI dataset, their quantitative performance degrades on the more difficult *Piano* and *Living Room* scenes (Table 4). A similar qualitative trend shows the learning-based methods performing worse than diffusion on the real-world light fields (Figure 21). Our method provides more

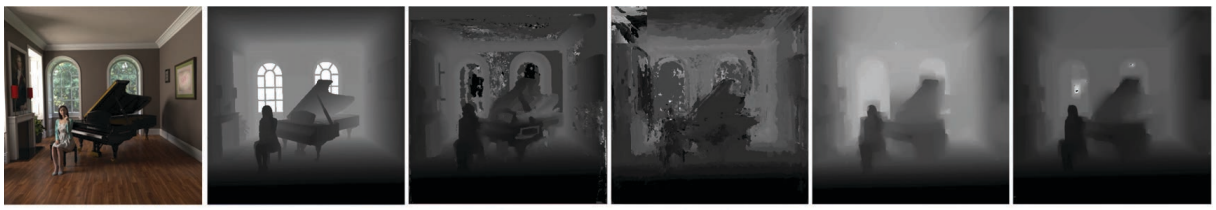


Fig. 20: Depth results on the synthetic *Piano-MVS* scene. *Left to right:* Ground truth, dense reconstruction from COLMAP [23, 72], DeepMVS [36], our method using 702 sparse points in $\mathcal{P}$, and our method with $2,808$ sparse points from dense COLMAP output.

Table 3: Quantitative results for wider-baseline unstructured five-camera cases, as the *Living Room-MVS* and *Piano-MVS* scenes.

MVS	MSE				Q25			
	D-MVS	C-Map	Ours	Ours-C	D-MVS	C-Map	Ours	Ours-C
Living Room	1.99	1.37	0.30	0.17	64.9	4.44	14.8	4.22
Piano	1.51	2.56	0.81	0.69	6.87	42.6	2.15	1.37
Average	1.75	1.97	0.56	0.43	35.9	23.5	8.48	2.80

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.

For breadth of comparison, we also compare to Jin and Hou's learning-based self-supervised method [73] and a state-of-the-art supervised learning approach in Wang et al.'s cost-constructor method [74]. Like our method, Jin and Hou also do not need any labels, but unlike our method, it is trained on data. MSE performance is slightly worse than our non-differentiable approach and comparable to our differentiable approach, but with more extreme outliers as shown by the BP(0.07) metric. Wang et al.'s method provides superior performance on the HCI dataset, but must be re-trained for each new dataset to account for different disparity ranges and dilation rates in their cost construction step.

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). We report results with this filter.

7.2 Multi-view Stereo

Baselines and Metrics

We compare to dense reconstruction from COLMAP [23] and to DeepMVS [36]. Our 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 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 the least squares fit the ground truth at 500 randomly sampled valid depth pixels.

Results

To account for the error in the least squares, Table 3 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 (only ≈ 700 points) and generates smooth results that qualitatively have fewer artifacts (Figure 20). Using $4\times$ as many initial points ($\approx 2,800$ points) in the Our-C variant leads to additional improvements.

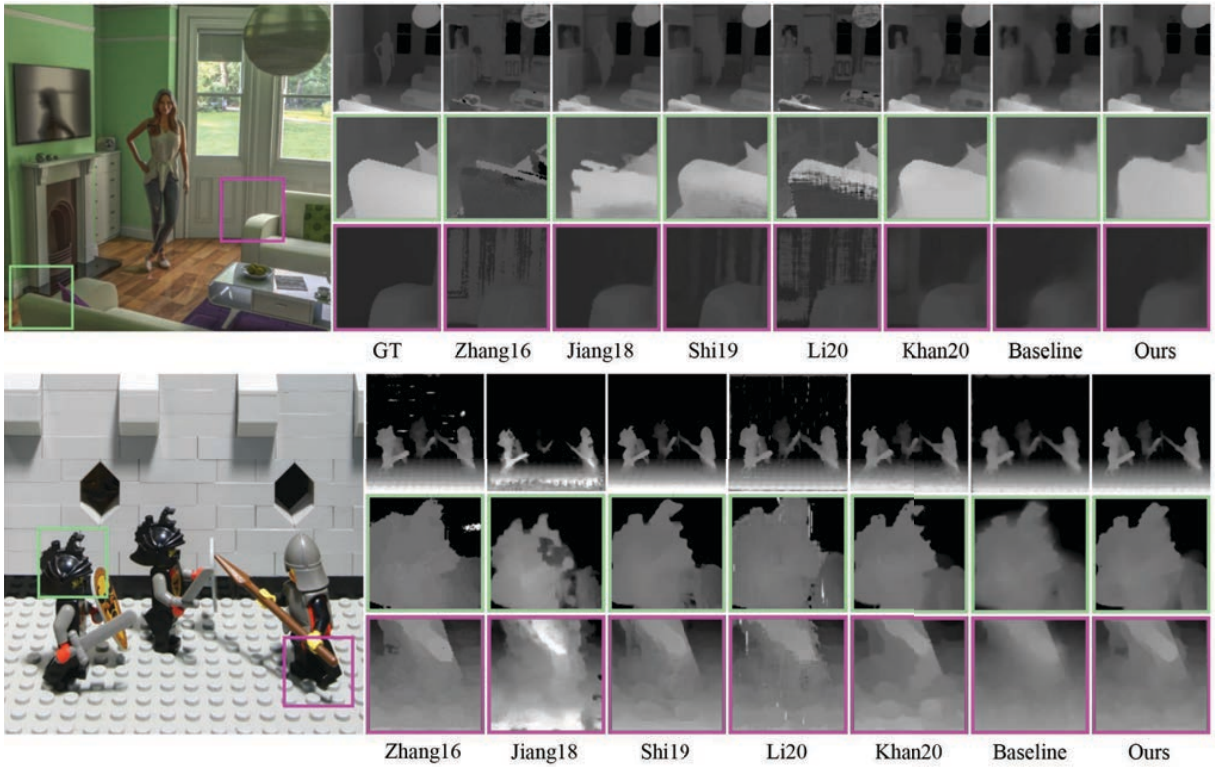


Fig. 21: *Top:* Disparity results on the synthetic *Living Room* light field. *Bottom:* Disparity results on a real light field. *Left to right*: Zhang et al. [34], Jiang et al. [35], Shi et al. [63], Li et al. [59], our non-differentiable approach as 'Khan20' (Sections 3 & 4), a baseline diffusion result without any optimization, our differentiable constraint results. The top synthetic light field also adds ground truth to the far left.

Table 4: Quantitative comparison on synthetic HCI 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. 'Ours†' denotes our non-differentiable method, 'Ours∇' denotes our differentiable method with an unsupervised loss. *Note:* [73] do not report BP(0.01) nor BP(0.03). (Best viewed in color.)

Light Field	[34]	[59]	[35]	[63]	[73]	[74]	Ours†	Ours∇	[34]	[59]	[35]	[63]	[73]	[74]	Ours†	Ours∇
	MSE * 100								BP(0.01)							
Sideboard	1.02	1.89	1.96	1.12	1.79	0.54	0.89	2.23	78.0	62.3	47.4	53.0	-	-	73.8	43.0
Dino	0.41	3.28	0.47	0.43	0.69	0.08	0.45	0.86	81.2	52.7	29.8	43.0	-	-	69.4	25.6
Cotton	1.81	1.95	0.97	0.88	0.80	0.16	0.68	3.07	75.4	58.8	25.4	38.6	-	-	56.2	31.1
Boxes	7.90	4.67	11.6	8.48	7.45	2.89	6.69	9.17	84.7	68.3	51.8	66.5	-	-	76.8	60.3
	BP(0.03)								BP(0.07)							
Sideboard	42.0	18.0	18.3	20.4	-	-	37.4	16.5	14.4	6.50	9.31	9.02	14.2	3.35	16.2	8.35
Dino	48.9	12.8	8.81	13.1	-	-	30.9	7.69	7.52	5.82	3.59	4.32	8.25	1.00	10.4	4.06
Cotton	34.8	14.0	6.30	9.60	-	-	18.0	7.82	4.35	4.11	2.02	2.74	8.46	0.31	4.86	4.06
Boxes	55.3	28.0	27.0	37.2	-	-	47.9	32.7	18.9	13.4	18.3	21.9	26.2	10.7	28.3	20.5

Table 5: Quantitative comparison on more realistic 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 for all methods: they contain specular surfaces, larger depth ranges, and path tracing noise. 'Ours†' denotes our non-differentiable method, and 'Ours∇' denotes our differentiable method with an unsupervised loss.

Light Field	[34]	[59]	[35]	[63]	Ours†	Ours∇	[34]	[59]	[35]	[63]	Ours†	Ours∇
	MSE * 100						BP(0.1)					
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
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
	BP(0.3)						BP(0.7)					
Living Room	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	25.0	17.6	13.6	11.4	20.0	7.93	5.33	4.13	5.88	4.29	4.95	3.49

7.3 Sparsity Evaluation

We use the differentiable model to extract a minimal set of points that satisfies our completeness criterion, thereby achieving compactness of representation as well. This is accomplished by optimizing the projected data weight $\lambda^{\mathbb{Z}}$ to be high for points that have a high contribution to the result, and low for less important points and outliers. To achieve such a gradation, we run the optimization by selecting at each step only those points with a weight larger than one to use for dense depth diffusion. To prevent points from being trapped in a discarded state, and thus receiving no gradients, we add a random jitter to the weights at each iteration. Moreover, this forces the optimization to push the weight of important points higher so that they can never be randomly discarded, and vice versa for outliers and superfluous points.

We demonstrate the results of our sparsification approach using a supervised loss optimized on the light fields of the HCI dataset. We run the optimization for 2500 steps for each light field and only optimize the data weight parameter. We evaluate two variants of our approach:

Ours$N\nabla$ selects the N points with the highest weight, allowing us to generate a point set with a fixed size budget (e.g., for storage cost). For a fair comparison to other sparsification methods that cannot set proportional weights, we do not use our optimized weight values (Figure 22); using them leads to a slight performance increase.

Ours$T\nabla$ selects all points with weight greater than a threshold of one during optimization, and also uses the optimized weight values of those selected points instead of ignoring them. Rather than a controller for a size budget like *Ours*$N\nabla$, this can be thought of as a strategy to produce a controller for quality.

We also compare our approach to three naive methods: sparsification by random sampling, by stratified sampling in screen space (2D), and by bucketing and averaging points into voxels on a regular 3D grid (3D).

Table 6 shows the percentage improvement in each metric over the baseline from Section 3, and Table 7 shows the number of points used per scene. In general, all methods yield some improvement, indicating the presence of outliers and superfluous points in the baseline. However, sampling using our differentiable diffusion approach produces a lower mean-squared error and bad pixel metrics than the other methods for the same number of points (Figure 22). Further, the naive sparsification baselines sometimes decrease performance, while our approaches empirically always improve performance. Comparing our two methods, *Ours*$T\nabla$ leads to greater improvement than *Ours*$N\nabla$, e.g., an average MSE decrease of 38.4% rather than 33.72%, though more points are retained on average (15,975 vs. a fixed 7,500).

8 Discussion

When we consider the maximal possible performance achievable of our method, when considering the results in Figure 2, we see a considerably minimized error, but not a perfect reconstruction.

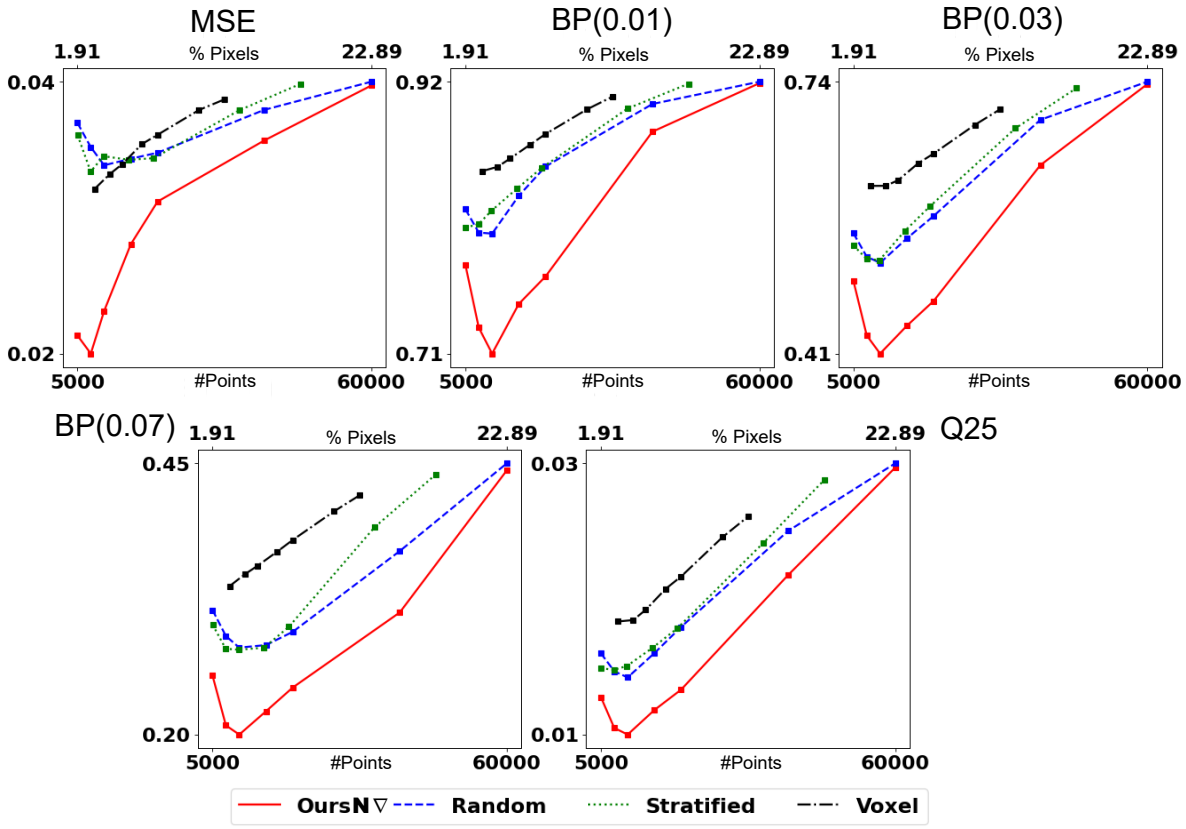


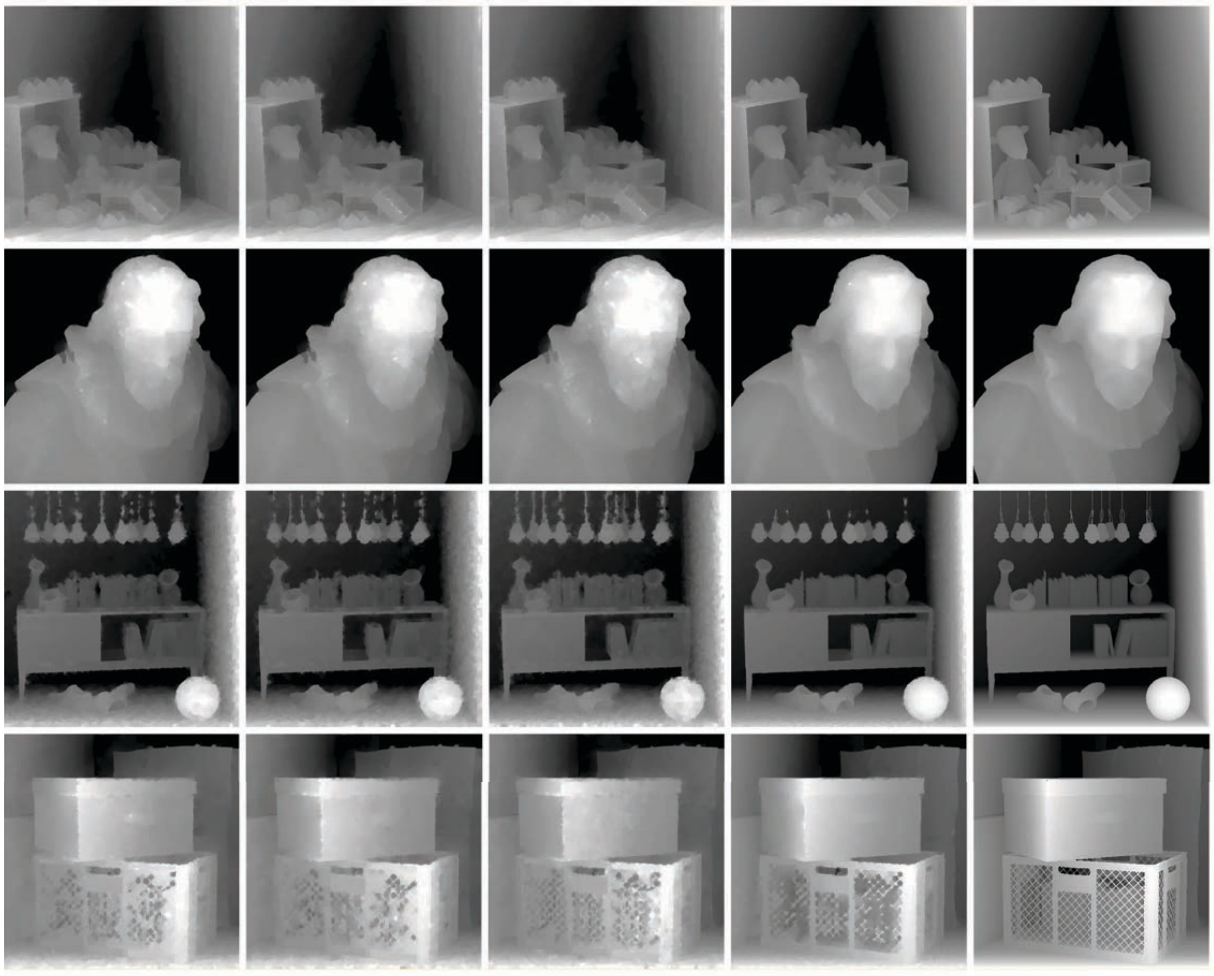
Fig. 22: *Our sparsification approach achieves better performance than naive baselines.* Evaluating the quality of the depth reconstruction from edge sets obtained using four sparsification methods on the four light fields of the HCI dataset. The error is along the y-axis. The x-axis labels show both the number of edge samples (*bottom*) and the percentage of total pixels in a single light field view (*top*).

Table 6: *Sparsification* improves *performance by removing redundant confounding points.* The percentage change in each metric over the baseline presented in Section 3, using 25–85k points per scene (Table 7). We evaluate uniform random sampling *Rnd*, 2D tile-based stratified sampling *StrS*, 3D voxel-based sampling *VoxS*, and two variants of our differentiable diffusion-based approach: *OursN*∇ selects the N points with highest confidence, whereas *OursT*∇ selects all points with confidence greater than a threshold of one. We use $N = 7500$ for *OursN*∇ and *Rnd* where we have direct control over the output set size, and aim for a roughly similar number of points in *StrS* and *VoxS* through hyper-parameter adjustment. The exact number of points in the sparse set output by each method is provided in Table 7. (Best viewed in color.)

Light Field	MSE * 100					BP(0.01)					BP(0.03)					BP(0.07)				
	Rnd	StrS	VoxS	OursN∇	OursT∇	Rnd	StrS	VoxS	OursN∇	OursT∇	Rnd	StrS	VoxS	OursN∇	OursT∇	Rnd	StrS	VoxS	OursN∇	OursT∇
Sideboard	-13.6	-16.7	-2.11	-52.1	-53.6	-4.59	-5.12	-0.34	-24.3	-25.3	-11.0	-12.8	0.22	-42.9	-45.7	-15.0	-16.6	2.12	-57.5	-60.8
Dino	-7.99	-11.6	-17.9	-45.7	-49.9	-3.38	-2.82	-4.23	-9.90	-14.0	-7.74	-7.61	-8.77	-26.8	-30.2	-17.2	-19.2	-16.3	-45.2	-44.1
Cotton	9.16	0.40	6.11	-2.48	-15.1	-21.9	-19.2	-5.43	-21.7	-27.4	-44.5	-43.1	-6.85	-43.0	-52.6	-28.4	-32.2	40.4	-26.1	-43.0
Boxes	1.75	-1.90	-6.22	-34.6	-35.0	-0.35	-0.27	2.43	-8.88	-9.15	0.12	-1.51	5.01	-13.7	-14.3	6.90	0.74	5.14	-16.5	-17.6

Table 7: *Edge set size can be significantly reduced without reducing performance through sparsification.* The size of the edge set generated by each sparsification method. *Base* is the original sparse set generated by the first-principles method described in Section 3. We present the percentage relative to the number of pixels in a single 512×512 view of the light field. However, it should be noted that our sparse multi-view edge set is complete for reconstructing the geometric information contained in *all* the views. Hence, the actual compactness achieved by our model is much higher.

Light Field	Number of multi-view edge samples / Percentage of total pixels in single view					
	RndSamp	2DStratSamp	VoxelSamp	OursN∇	OursT∇	Base
Sideboard	7500/2.86%	7552/2.88%	14178/5.41%	7500/2.86%	9382/3.58%	50623/19.3%
Dino	7500/2.86%	7544/2.88%	10759/4.10%	7500/2.86%	28195/10.8%	85807/32.7%
Cotton	7500/2.86%	7551/2.88%	10749/4.10%	7500/2.86%	19653/7.46%	60921/23.2%
Boxes	7500/2.86%	7238/2.76%	8453/3.22%	7500/2.86%	6670/2.54%	25332/9.66%



Random 2D stratified Voxel stratified OursT∇ Ground truth

Fig. 23: Qualitative comparison of the depth maps obtained using different sparse sampling methods. Our approach to sparsification maintains smooth surfaces and sharp edges better than naive methods.

This can be attributed to certain implementational constraints that limit the model's ability to encode and reconstruct all depth features: while the optimization process proposed in Section 6.4 can discard noisy labels, it cannot add new labels to the original edge set $\mathcal{P}$. Such errors are somewhat mitigated by starting with many points and sparsifying them. However, our evaluation is ultimately limited by our original edge set: a good set of edges will be more complete than a bad one; the empty set will never be complete.

Another source of errors is the so-called "island problem" where a value must be propagated across depth boundaries [75]. This can be observed in our results for the *boxes* scene in the grill (Figure A5). However, this failure does not imply that our model is incomplete. The depth of a featureless "island" cannot be ambiguously determined by any method, and our estimate is as 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 surface configurations.

Furthermore, as stated, our model assumes Lambertian surfaces. However, since non-Lambertian surfaces violate EPI linearity, our multi-view edge estimation pipeline can reliably discard them as outliers. Such a surface is then assigned a depth value diffused from the closest non-refractive edge, leading to a fronto-parallel surface reconstruction. Thus, diffusion allows our method to fall back on a piece-wise planar reconstruction for refractive surfaces.

Finally, the baseline of a camera system affects multi-view edge recovery. We have shown our approach on narrow and wide baseline light fields (synthetic, Lytro, and Stanford), and on front-facing multi-view stereo scenes, but larger baselines from sparse setups may cause trouble in reliably finding and propagating edges. While our approach is occlusion-aware, reliably finding occluded elements in very sparse point sets in wide multi-view stereo setups is difficult. As such, we did not test our method on sparse wide baseline setup datasets as might be found in the Tanks and Temples [76], DTU [77], or ETH3D [78] datasets. For example, our existing multi-view point sets use 700–2800 points, as compared to 7,500 minimally for light fields. On these additional datasets, point sparseness would only increase.

9 Conclusion

Multi-view edges encode all relevant information for supporting higher-level tasks that rely on depth reconstruction. Inspired by the work of Elder [27], this was demonstrated by proposing a representation for multi-view depth as a sparse model based on multi-view edges that satisfies the criteria of explicitness, concision, and completeness. 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. The completeness of the representation—its ability to capture all relevant information for higher-level tasks—was demonstrated through estimating representation parameters—an edge code—from a structured light field and by a reconstruction method that inverts the edge code to retrieve a dense 2D depth map. Given that different applications are variously dependent on the quality dimensions of depth reconstruction, we defined completeness for computational photography tasks using occlusion edges and view consistency in addition to the more commonly used accuracy metric.

However, practical use can still be limited by the inaccuracy of the representation parameters, leading to errors in the diffusion process. As such, we created a differentiable variant of the representation that redefines multi-view edges as Gaussians in 3D space that can be splatted to a camera in an occlusion-aware way via radiative transport. Then, representation parameters can be optimized via gradient descent guided by a multi-view reprojection loss. This loosens the requirement on estimating multi-view edges from structured light field and additionally allows optimizing point clouds recovered from unstructured multi-view images. Further, the differentiable form of the reconstruction can also optimize the concision of the representation via sparsity.

In comparative evaluation, we observe that the reconstructed dense depth is comparable to

existing methods on each of the three metrics: accuracy, occlusion-edge localization, and view-consistency, and is comparatively good at reducing bad pixels via its reprojection loss. First, these findings provide strong evidence that multi-view depth edges are indeed complete. Second, they suggest that compact, efficient, and well-defined edge codes have value amid deep learning. Third, they provide evidence that this value is derived from the underlying structural information contained within images—that of high-confidence depth estimates from well-defined multi-view edges—and not contained within images—where we must make assumptions in regions of low confidence (be they data-driven or otherwise). As a way of handling low confidence regions through sparsity, diffusion-based methods are still a reasonable smoothing approach to take, and that careful constraint estimation and optimization ultimately determines the final quality.

Acknowledgements

Numair Khan thanks an Andy van Dam PhD Fellowship at Brown University, Min H. Kim acknowledges the MSIT/IITP of Korea (RS-2022-00155620, 2022-0-00058, and 2017-0-00072), the Samsung Research Funding Center (SRFC-IT2001-04), and the NIRCH of Korea (2021A02P02-001), and James Tompkin thanks Cognex and US NSF CAREER-2144956.

References

[1] Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The Lumigraph. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 43–54 (1996)

[2] Choi, I., Gallo, O., Troccoli, A., Kim, M.H., Kautz, J.: Extreme view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 7781–7790 (2019)

[3] Riegler, G., Koltun, V.: Free view synthesis. In: European Conference on Computer Vision, pp. 623–640 (2020). Springer

[4] Riegler, G., Koltun, V.: Stable view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 12216–12225 (2021)

[5] Jarabo, A., Masia, B., Bousseau, A., Pellacini, F., Gutierrez, D.: How do people edit light fields? ACM Transactions on Graphics (SIGGRAPH 2014) **33**(4) (2014)

[6] Mihara, H., Funatomi, T., Tanaka, K., Kubo, H., Mukaigawa, Y., Nagahara, H.: 4d light field segmentation with spatial and angular consistencies. In: Proceedings of the International Conference on Computational Photography (ICCP) (2016)

[7] Luo, X., Huang, J.-B., Szeliski, R., Matzen, K., Kopf, J.: Consistent video depth estimation. ACM Transactions on Graphics (TOG) **39**(4), 71–1 (2020)

[8] Ha, H., Baek, S.-H., Nam, G., Kim, M.H.: Progressive acquisition of svbrdf and shape in motion. Computer Graphics Forum (2020). https://doi.org/10.1111/cgf.14087

[9] Holynski, A., Kopf, J.: Fast depth densification for occlusion-aware augmented reality **37**(6) (2018)

[10] Xu, Z., Bi, S., Sunkavalli, K., Hadap, S., Su, H., Ramamoorthi, R.: Deep view synthesis from sparse photometric images. ACM Transactions on Graphics (TOG) **38**(4), 1–13 (2019)

[11] Zhang, R., Tsai, P.-S., Cryer, J.E., Shah, M.: Shape-from-shading: a survey. IEEE transactions on pattern analysis and machine intelligence **21**(8), 690–706 (1999)

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

[13] Ikoma, H., Nguyen, C.M., Metzler, C.A., Peng, Y., Wetzstein, G.: Depth from defocus with learned optics for imaging and occlusion-aware depth estimation. IEEE International Conference on Computational

Photography (ICCP) (2021)

[14] Bi, S., Xu, Z., Sunkavalli, K., Kriegman, D., Ramamoorthi, R.: Deep 3d capture: Geometry and reflectance from sparse multi-view images. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)

[15] Debevec, P., Hawkins, T., Tchou, C., Duiker, H.-P., Sarokin, W., Sagar, M.: Acquiring the reflectance field of a human face. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, pp. 145–156 (2000)

[16] Meka, A., Haene, C., Pandey, R., Zollhoefer, M., Fanello, S., Fyffe, G., Kowdle, A., Yu, X., Busch, J., Dourgarian, J., Denny, P., Bouaziz, S., Lincoln, P., Whalen, M., Harvey, G., Taylor, J., Izadi, S., Tagliasacchi, A., Debevec, P., Theobalt, C., Valentin, J., Rhemann, C.: Deep reflectance fields - high-quality facial reflectance field inference from color gradient illumination, vol. 38 (2019). https://doi.org/10.1145/3306346.3323027. http://gvv.mpi-inf.mpg.de/projects/DeepReflectanceFields/

[17] Nam, G., Lee, J.H., Wu, H., Gutierrez, D., Kim, M.H.: Simultaneous acquisition of microscale reflectance and normals. ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2016) **35**(6) (2016). https://doi.org/10.1145/2980179.2980220

[18] Chen, W., Fu, Z., Yang, D., Deng, J.: Single-image depth perception in the wild. Advances in neural information processing systems **29**, 730–738 (2016)

[19] Li, Z., Dekel, T., Cole, F., Tucker, R., Snavely, N., Liu, C., Freeman, W.T.: Learning the depths of moving people by watching frozen people. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4521–4530 (2019)

[20] Ranftl, R., Lasinger, K., Hafner, D., Schindler, K., Koltun, V.: Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) (2020)

[21] Ranftl, R., Bochkovskiy, A., Koltun, V.: Vision transformers for dense prediction. ArXiv preprint (2021)

[22] Seitz, S.M., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), vol. 1, pp. 519–528 (2006). IEEE

[23] Schönberger, J.L., Zheng, E., Pollefeys, M., Frahm, J.-M.: Pixelwise view selection for unstructured multi-view stereo. In: European Conference on Computer Vision (ECCV) (2016)

[24] Joshi, N., Zitnick, C.L.: Micro-baseline stereo. Technical Report MSR-TR-2014–73, 8 (2014)

[25] Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (TOG) (2019)

[26] Jarabo, A., Masia, B., Gutierrez, D.: Efficient propagation of light field edits. In: In Proc. of SIACG'11, pp. 75–80 (2011)

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

[28] Adelson, E.H., Bergen, J.R.: The Plenoptic Function and the Elements of Early Vision vol. 2. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of Technology, ??? (1991)

[29] Hog, M., Sabater, N., Guillemot, C.: Light field segmentation using a ray-based graph structure. In: ECCV (2016)

[30] Barlow, H.B., et al.: Possible principles

underlying the transformation of sensory messages. Sensory communication **1**(01) (1961)

[31] Wanner, S., Goldluecke, B.: Globally consistent depth labeling of 4d light fields. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 41–48 (2012). IEEE

[32] Wang, T.-C., Efros, A.A., Ramamoorthi, R.: Occlusion-aware depth estimation using light-field cameras. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3487–3495 (2015)

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

[34] Zhang, S., Sheng, H., Li, C., Zhang, J., Xiong, Z.: Robust depth estimation for light field via spinning parallelogram operator. Computer Vision and Image Understanding **145**, 148–159 (2016)

[35] Jiang, X., Le Pendu, M., Guillemot, C.: Depth estimation with occlusion handling from a sparse set of light field views. In: 2018 25th IEEE International Conference on Image Processing (ICIP), pp. 634–638 (2018). IEEE

[36] Huang, P.-H., Matzen, K., Kopf, J., Ahuja, N., Huang, J.-B.: DeepMVS: Learning multi-view stereopsis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2821–2830 (2018)

[37] Yao, Y., Luo, Z., Li, S., Fang, T., Quan, L.: Mvsnet: Depth inference for unstructured multi-view stereo. European Conference on Computer Vision (ECCV) (2018)

[38] Yao, Y., Luo, Z., Li, S., Shen, T., Fang, T., Quan, L.: Recurrent mvsnet for high-resolution multi-view stereo depth inference. Computer Vision and Pattern Recognition (CVPR) (2019)

[39] Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. ACM Trans. Graph. **37**(4) (2018)

[40] Wang, T.-C., Chandraker, M., Efros, A.A., Ramamoorthi, R.: Svbrdf-invariant shape and reflectance estimation from light-field cameras. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)

[41] Park, J.J., Newcombe, R., Seitz, S.: Surface light field fusion. In: 2018 International Conference on 3D Vision (3DV), pp. 12–21 (2018). IEEE

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

[43] Vision.middlebury.edu: Middlebury Stereo Evaluation. https://vision.middlebury.edu/stereo/

[44] Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2012)

[45] Menze, M., Geiger, A.: Object scene flow for autonomous vehicles. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2015)

[46] Tompkin, J., Muff, S., McCann, J., Pfister, H., Kautz, J., Alexa, M., Matusik, W.: Joint 5d pen input for light field displays. In: The 28th Annual ACM Symposium on User Interface. Software and Technology, UIST'15 (2015)

[47] Elder, J.H., Goldberg, R.M.: Image editing in the contour domain. IEEE Trans. Pattern Anal. Mach. Intell. **23**(3), 291–296 (2001). https://doi.org/10.1109/34.910881

[48] Khan, N., Zhang, Q., Kasser, L., Stone, H., Kim, M.H., Tompkin, J.: View-consistent 4d

light field superpixel segmentation. In: International Conference on Computer Vision (ICCV) 2019 (2019). IEEE

[49] Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. In: ACM SIGGRAPH 2004 Papers, pp. 689–694 (2004)

[50] Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., Sridhar, S.: Neural fields in visual computing and beyond. Computer Graphics Forum (2022). https://doi.org/10.1111/cgf.14505

[51] Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **8**(6), 679–698 (1986). https://doi.org/10.1109/TPAMI.1986.4767851

[52] Ruzon, M.A., Tomasi, C.: Color edge detection with the compass operator. In: Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference On., vol. 2, pp. 160–166 (1999). IEEE

[53] Kim, C., Zimmer, H., Pritch, Y., Sorkine-Hornung, A., Gross, M.H.: Scene reconstruction from high spatio-angular resolution light fields. ACM Trans. Graph. **32**(4), 73–1 (2013)

[54] Yucer, K., Kim, C., Sorkine-Hornung, A., Sorkine-Hornung, O.: Depth from gradients in dense light fields for object reconstruction. In: 2016 Fourth International Conference on 3D Vision (3DV), pp. 249–257 (2016). IEEE

[55] Szeliski, R.: Locally adapted hierarchical basis preconditioning. In: ACM SIGGRAPH 2006 Papers, pp. 1135–1143 (2006)

[56] Bhat, P., Zitnick, L., Cohen, M., Curless, B.: Gradientshop: A gradient-domain optimization framework for image and video filtering. In: ACM Transactions on Graphics (TOG) (2009)

[57] Jeon, H.-G., Park, J., Choe, G., Park, J., Bok, Y., Tai, Y.-W., So Kweon, I.: Accurate depth map estimation from a lenslet light field camera. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1547–1555 (2015)

[58] Jiang, X., Shi, J., Guillemot, C.: 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)

[59] Li, K., Zhang, J., Sun, R., Zhang, X., Gao, J.: Epi-based oriented relation networks for light field depth estimation. British Machine Vision Conference (2020)

[60] Wanner, S., Straehle, C., Goldluecke, B.: Globally consistent multi-label assignment on the ray space of 4d light fields. In: IEEE Conference on Computer Vision and Pattern Recognition (2013)

[61] Shih, M.-L., Su, S.-Y., Kopf, J., Huang, J.-B.: 3d photography using context-aware layered depth inpainting. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2020)

[62] Ma, Z., He, K., Wei, Y., Sun, J., Wu, E.: Constant time weighted median filtering for stereo matching and beyond. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 49–56 (2013)

[63] Shi, J., Jiang, X., Guillemot, C.: 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)

[64] Honauer, K., Johannsen, O., Kondermann, D., Goldluecke, B.: A dataset and evaluation methodology for depth estimation on 4d light fields. In: Asian Conference on Computer Vision, pp. 19–34 (2016). Springer

[65] Rerabek, M., Ebrahimi, T.: New light field image dataset. In: 8th International Conference on Quality of Multimedia Experience (QoMEX) (2016)

[66] Laboratory, S.G.: The New Stanford Light Field Archive (2008). http://lightfield.stanford.edu/

[67] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2462–2470 (2017)

[68] Rhodin, H., Robertini, N., Richardt, C., Seidel, H.-P., Theobalt, C.: A versatile scene model with differentiable visibility applied to generative pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 765–773 (2015)

[69] Stoll, C., Hasler, N., Gall, J., Seidel, H.-P., Theobalt, C.: Fast articulated motion tracking using a sums of gaussians body model. In: 2011 International Conference on Computer Vision, pp. 951–958 (2011). IEEE

[70] Ranjan, A., Jampani, V., Balles, L., Kim, K., Sun, D., Wulff, J., Black, M.J.: 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, pp. 12240–12249 (2019)

[71] Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)

[72] Schönberger, J.L., Frahm, J.-M.: Structure-from-motion revisited. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)

[73] Jin, J., Hou, J.: Occlusion-aware unsupervised learning of depth from 4-d light fields. IEEE Transactions on Image Processing **31**, 2216–2228 (2022)

[74] Wang, Y., Wang, L., Liang, Z., Yang, J., An, W., Guo, Y.: Occlusion-aware cost constructor for light field depth estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 19809–19818 (2022)

[75] Kim, I., Kim, M.H.: Non-local haze propagation with an iso-depth prior. In: International Joint Conference on Computer Vision, Imaging and Computer Graphics, pp. 213–238 (2017). Springer

[76] Knapitsch, A., Park, J., Zhou, Q.-Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (ToG) **36**(4), 1–13 (2017)

[77] Aanæs, H., Jensen, R.R., Vogiatzis, G., Tola, E., Dahl, A.B.: Large-scale data for multiple-view stereopsis. International Journal of Computer Vision, 1–16 (2016)

[78] Schops, T., Schonberger, J.L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., Geiger, A.: A multi-view stereo benchmark with high-resolution images and multi-camera videos. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3260–3269 (2017)

[79] Khan, N., Kim, M.H., Tompkin, J.: View-consistent 4d light field depth estimation. British Machine Vision Conference (2020)

[80] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[81] Park, I.K., Lee, K.M., *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)

[82] Diebold, M., Goldluecke, B.: Epipolar plane image refocusing for improved depth estimation and occlusion handling (2013)

[83] Tosic, I., Berkner, K.: Light field scale-depth space transform for dense depth estimation. In: Proceedings of the IEEE Conference on

Computer Vision and Pattern Recognition Workshops, pp. 435–442 (2014)

[84] Tao, M.W., Hadap, S., Malik, J., Ramamoorthi, R.: Depth from combining defocus and correspondence using light-field cameras. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 673–680 (2013)

[85] Kolmogorov, V., Zabih, R.: Multi-camera scene reconstruction via graph cuts. In: European Conference on Computer Vision, pp. 82–96 (2002). Springer

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

[87] Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: PatchMatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics (Proc. SIGGRAPH) **28**(3) (2009)

[88] Chen, J., Hou, J., Ni, Y., Chau, L.-P.: Accurate light field depth estimation with superpixel regularization over partially occluded regions. IEEE Transactions on Image Processing **27**(10), 4889–4900 (2018)

[89] Wang, T.-H., Wang, F.-E., Lin, J.-T., Tsai, Y.-H., Chiu, W.-C., Sun, M.: Plug-and-play: Improve depth estimation via sparse data propagation. arXiv preprint arXiv:1812.08350 (2018)

[90] Cheng, X., Wang, P., Yang, R.: Depth estimation via affinity learned with convolutional spatial propagation network. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 103–119 (2018)

[91] Alperovich, A., Johannsen, O., Goldluecke, B.: Intrinsic light field decomposition and disparity estimation with a deep encoder-decoder network. In: European Signal Processing Conference (EUSIPCO) (2018)

[92] Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., Courville, A.: On the spectral bias of neural networks. In: International Conference on Machine Learning, pp. 5301–5310 (2019). PMLR

[93] Basri, R., Galun, M., Geifman, A., Jacobs, D., Kasten, Y., Kritchman, S.: Frequency bias in neural networks for input of non-uniform density. In: International Conference on Machine Learning, pp. 685–694 (2020). PMLR

[94] Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European Conference on Computer Vision, pp. 405–421 (2020). Springer

[95] Zhang, K., Riegler, G., Snavely, N., Koltun, V.: Nerf++: Analyzing and improving neural radiance fields. arXiv preprint arXiv:2010.07492 (2020)

[96] Tancik, M., Srinivasan, P.P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J.T., Ng, R.: Fourier features let networks learn high frequency functions in low dimensional domains. arXiv preprint arXiv:2006.10739 (2020)

[97] Sitzmann, V., Martel, J., Bergman, A., Lindell, D., Wetzstein, G.: Implicit neural representations with periodic activation functions. Advances in Neural Information Processing Systems **33** (2020)

[98] Weder, S., Schonberger, J., Pollefeys, M., Oswald, M.R.: Routedfusion: Learning real-time depth map fusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)

[99] Choe, J., Im, S., Rameau, F., Kang, M., Kweon, I.S.: Volumefusion: Deep depth fusion for 3d scene reconstruction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 16086–16095 (2021)

[100] Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 303–312 (1996)

[101] Izadi, S., Newcombe, R.A., Kim, D., Hilliges, O., Molyneaux, D., Hodges, S., Kohli, P., Shotton, J., Davison, A.J., Fitzgibbon, A.: Kinectfusion: real-time dynamic 3d surface reconstruction and interaction. In: ACM SIGGRAPH 2011 Talks, pp. 1–1 (2011)

[102] Kopf, J., Matzen, K., Alsisan, S., Quigley, O., Ge, F., Chong, Y., Patterson, J., Frahm, J.-M., Wu, S., Yu, M., Zhang, P., He, Z., Vajda, P., Saraf, A., Cohen, M.: One shot 3d photography **39**(4) (2020)

[103] Richardt, C., Stoll, C., Dodgson, N.A., Seidel, H.-P., Theobalt, C.: Coherent spatiotemporal filtering, upsampling and rendering of RGBZ videos. Computer Graphics Forum (Proceedings of Eurographics) **31**(2) (2012). https://doi.org/10.1111/j.1467-8659.2012.03003.x

[104] Chen, Z., Badrinarayanan, V., Drozdov, G., Rabinovich, A.: Estimating depth from rgb and sparse sensing. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 167–182 (2018)

[105] Imran, S., Long, Y., Liu, X., Morris, D.: Depth coefficients for depth completion. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12438–12447 (2019). IEEE

[106] Ku, J., Harakeh, A., Waslander, S.L.: In defense of classical image processing: Fast depth completion on the cpu. In: 2018 15th Conference on Computer and Robot Vision (CRV), pp. 16–22 (2018). https://doi.org/10.1109/CRV.2018.00013

Appendix A

A.1 Variable Table

Please reference Table A1 at the back of this document for a list of all variables used within the paper and their meaning, definition, and values where appropriate.

A.2 Implementation Software and Hardware

Our non-differentiable method was implemented in MATLAB (except the C++ Poisson solver), as were the three non-data-driven comparison algorithms and parts of Jiang et al. All networks were implemented in TensorFlow. Our differentiable method was implemented in PyTorch. For diffusion, we implement a differentiable version of Szeliski's LAHBPCG solver [55]. All CPU code was run on an AMD Ryzen Threadripper 2950X 16-Core Processor, and GPU code on an NVIDIA GeForce RTX 2080Ti.

A.3 Supervised Loss

In the main paper, we describe a validation of our differentiable rendering and diffusion approach using a supervised loss against ground truth data. In Figure A3, 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 A4, 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 A4 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.

A.4 Differentiable Implementation Details

Our proposed framing of the diffusion problem allows us to express $S^{\mathbb{Z}}$ 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

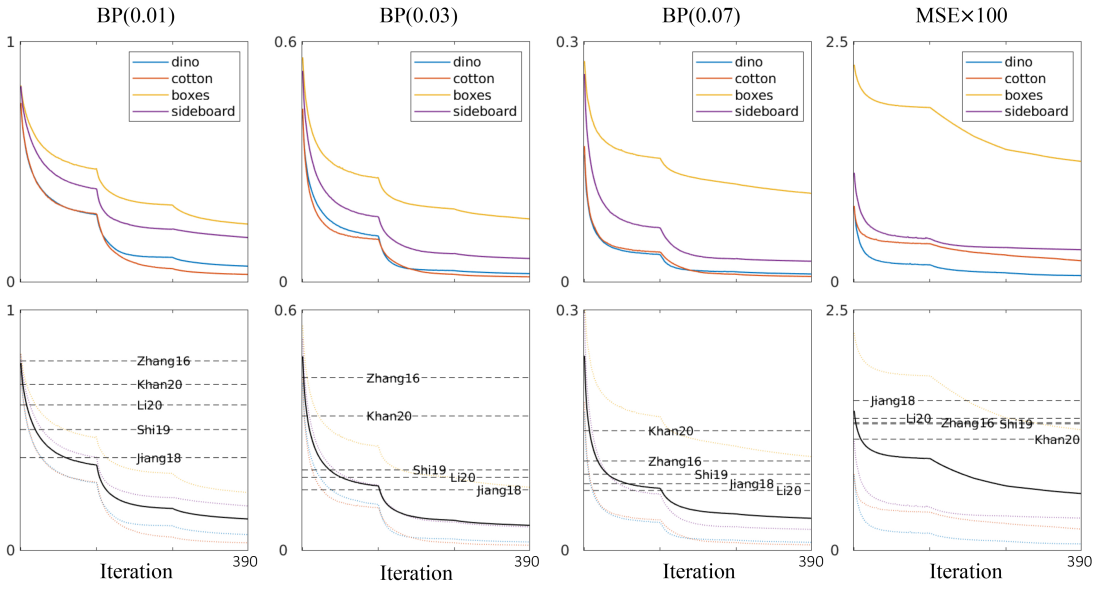


Fig. A1: Evaluation metrics plotted over all iterations of the optimization with supervised loss (*HCI Dataset*). The top row shows results on individual light fields in the dataset. The bottom row compares the average performance over the dataset with the baseline methods of Zhang et al. [34], Khan et al. [79], Li et al. [59], Shi et al. [63] and Jiang et al. [35]. The bumps in the curve occur where we switch optimization parameters.

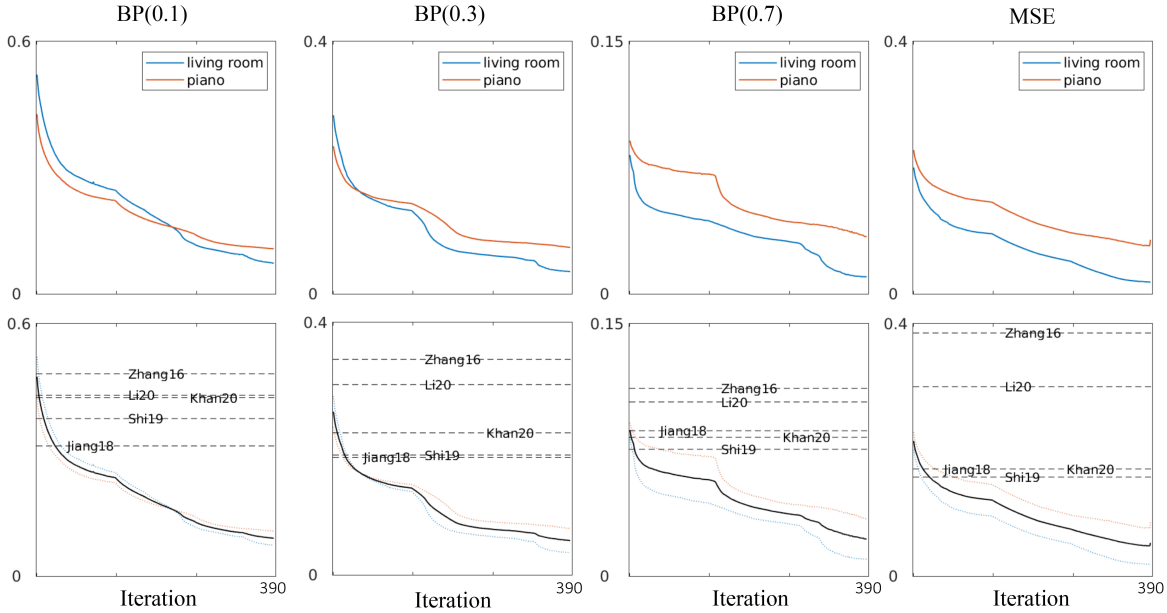


Fig. A2: Evaluation metrics for the *Piano* and *Living Room* light fields with supervised loss.

Fig. A3: We validate our image-space representation and optimization using ground truth depth to supervise the optimization using the same routine as Section 6.4. 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.

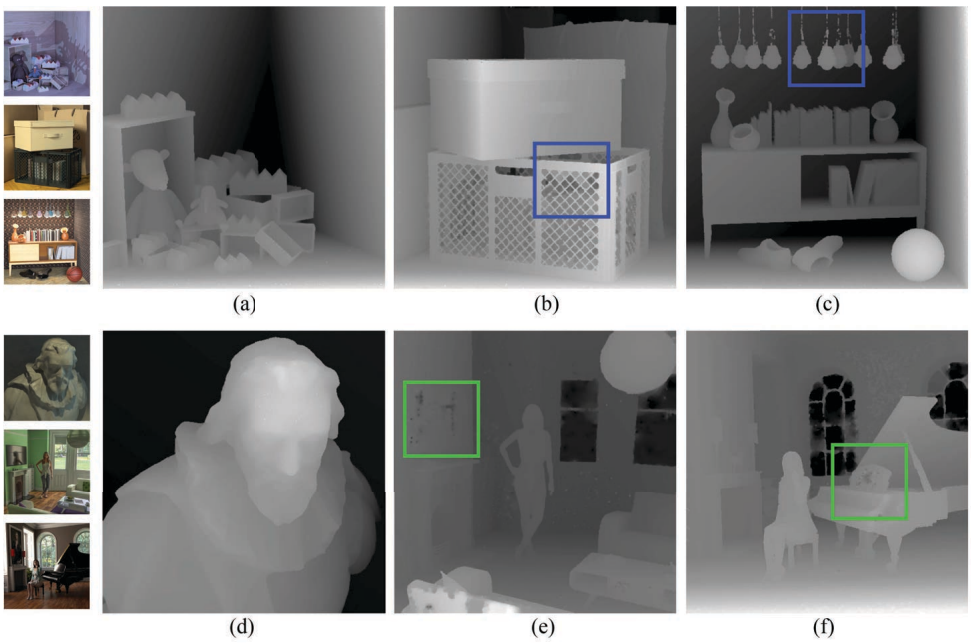


Fig. A4: Disparity maps generated by our method using supervised loss. **(a)**–**(f)**: *Boxes*, *Dino*, *Sideboard*, *Cotton*, *Living Room* and *Piano*. The presence of outliers in the original point set (green boxes), and the lack of labels in regions with fine detail (blue boxes) prevents the diffusion from entirely eliminating errors. Such sources of error occur in all six light fields to varying degrees.

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 (32)). 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 a 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 (40), 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 E_SSIM are calculated in RGB space.

Hyperparameters

This requires a trade-off between resource use and accuracy. The parameter σ_S in Equation (29) determines the pixel area of a splatted depth label $Z_{\mathbf{x}}$. Ideally, we want the label to be $Z_{\mathbf{x}}$ over all pixels where $\lambda^{\mathbb{Z}}_{\mathbf{x}} > \epsilon$. The case where the label falls off while the weight is much larger than zero is illustrated in Figure 15(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 (39). 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 $\lambda^{\mathbb{Z}}_{\mathbf{x}}$ (Eq. (31)). 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. 15(d)).

Routine

We use Adam [80]. 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 (29) 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 (39) 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 (28), and back-propagate gradients through the solver and renderer in five seconds.

A.5 Additional Results for Non-differentiable Approach

Please see Figures A5 and A6 for additional qualitative results.

A.6 Additional Related Work

Many researchers and their work have inspired our approach. We describe in greater detail how these works address depth estimation.

Light Field Depth Estimation

Light field depth estimation methods typically seek to exploit the regular structure of an EPI [81]. Wanner and Goldluecke's [31] 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 [82] 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 [34] spinning parallelogram operator works in a similar fashion on EPIs, but has larger support than the 3×3 Scharr filters used by Wanner and Goldluecke [31] and Diebold and Goldluecke [82], and provides more accurate estimates. This approach is similar to Tošić and Berkner's [83] convolution with a set of specially adapted kernels to create light field scale-depth spaces. Wang et al. [32] [33] 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 [84] work considers higher dimensional representations of EPIs that allows them to use both correspondence and defocus for depth estimation. These

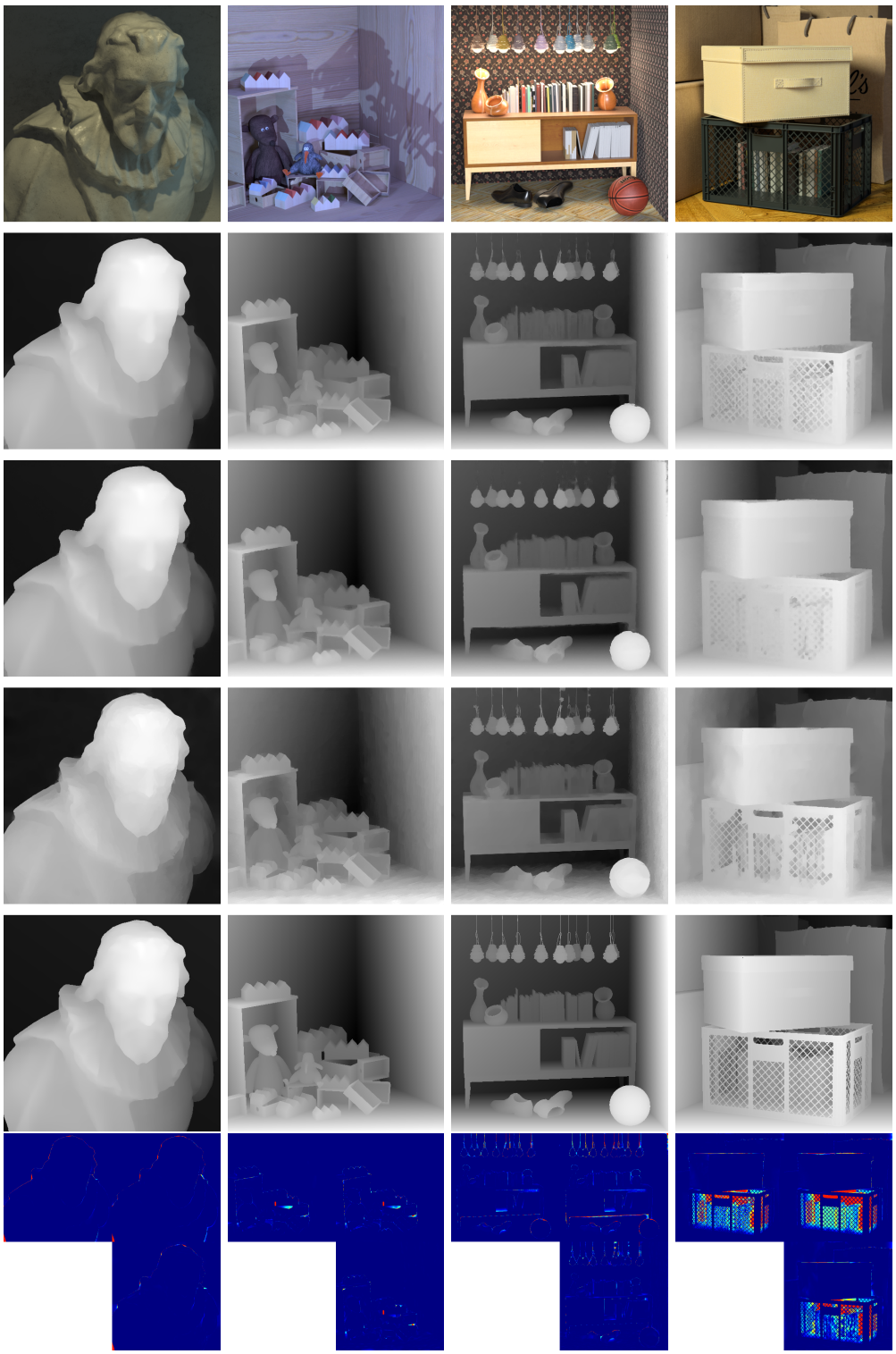


Fig. A5: HCI dataset. Top to bottom: light field central view, Shi et al. [63], Jiang et al. [35], our non-differentiable method, ground truth depth, error maps in clockwise order (Shi, Jiang, Ours). In general, our method has a lower mean squared error (MSE) with fewer large outliers (please zoom into 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 texture variations.

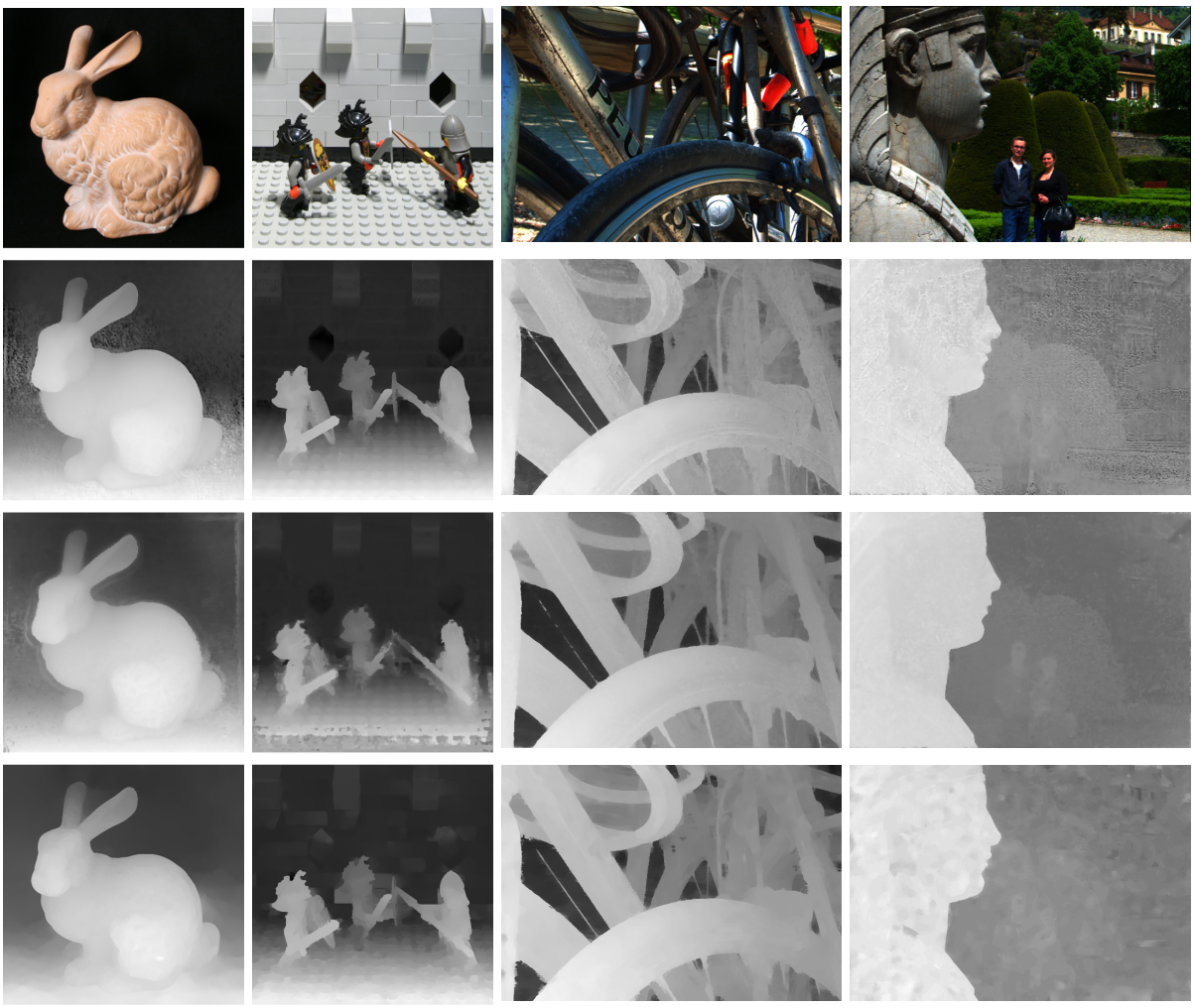


Fig. A6: Real-world light fields from the Stanford (*left pair*) and EPFL (*right pair*) datasets. *Top to bottom:* central RGB view, Shi et al. [63], Jiang et al. [35], and our non-differentiable method. While our method has more bad pixels and can be sensitive in narrow baseline cases (*far right:* 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.

latter two works use graph cuts [85] 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. [57] 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. [86]. They use an over-segmentation of each view to get initial depth proposals that are then iteratively improved using PatchMatch [87]. Their work demonstrates the use of superpixels for higher-level vision tasks. Closely related to our method of edge-aware bidirectional diffusion is the work of Holynski and Kopf [9] who present an efficient method for depth densification from a sparse set of points for augmented reality applications. Similar to our smooth reconstruction method, Yucer et al. [54] 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. [88] estimate accurate occlusion boundaries by using superpixels in the central view to regularize the depth estimation process. In general, densification methods [10, 89, 90] 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. [91] 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 handles an arbitrary number of uncalibrated views. Jiang et al. [35, 58] fuse the disparity estimates at four corner views estimated using a deep-learning optical-flow method. Shi et al. [63] build on this by adding a refinement network to the fusion pipeline. Finally, self-supervised approaches using learning also exist, for instance, that of Jin and Hou [73].

Depth Occlusion Edge Estimation

High-accuracy depth edges are vital for image editing tasks; however, correctly localizing depth edges proves difficult. CNNs trained on a mean loss over all pixels fail to capture high frequencies (also due in part to spectral bias [92, 93] and the averaging effect implicit in convolution). Methods such as Neural Radiance Fields [94] also inherently have a smoothness bias that lets them avoid degenerate solutions that may result from the shape-radiance ambiguity [95] and can require positional encoding for high-frequency details [96, 97]. Similar smoothing artifacts can be observed in depth fusion approaches [98, 99], especially those based on averaging signed distance functions [100, 101]. Many depth estimation methods, including top performers on the Middlebury Stereo Dataset [43], mitigate the blurriness of depth edges by using a discrete range of depth values. Others enforce strong occlusion edges with a weighted median filter on depth [61, 62, 102].

Sparse Depth Reconstruction

Early work in the field of depth densification and completion from sparse inputs used cross-bilateral filters to complete missing depth samples [103]. Chen et al. learn to upsample low-resolution depth camera input and regularize it from paired RGB data [104]. Imran et al. consider the problem of depth pixels being interpolated across discontinuities, and compensate by learning inter-depth object mixing [105]. Efficient computation is also addressed by Holynski and Kopf [9], 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 [106]. 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.

Table A1: Reference table for all variables used throughout the manuscript.

Variable	Interpretation	Type, specification, or value
Light field parameterization		
LF	Light field.	
x, y, u, v	Light field ray intersections in two-plane model.	
$\mathcal{H}$	Central row of views in light field.	$LF(x, y, u_c, v)$
$\mathcal{V}$	Central column of views in light field.	$LF(x, y, u, v_c)$
I	An image sampled from a light field.	RGB 8-bit
I	A 2D perspective image in 3rd column and 5th row of light field.	$LF(x, y, 2, 4)$
E_i	Epipolar image in light field; horizontal.	$E_i(x, u) = I(x, y_i, u)$
E_j	Epipolar image in light field; vertical.	$E_j(y, v) = I(x_j, y, v)$
Model parameterization—all parameters shared across all scenes.		
Line fitting		
w, h	Epipolar image (EPI) width and height.	Per light field, e.g., $h = 9$.
l	Parametric model of a line within an EPI.	
λ	Perpendicular distance in confidence map within which to discard other lines.	$h/5$
k	Number of samples that a line must lie across to be included.	$h/4$
τ_f	Line filtering threshold.	$\pi/13$
τ_v	Line visibility threshold.	$\pi/10$
Entropy refinement		
t	Line visibility threshold.	0.88
α	Line visibility threshold.	0.15
	Number of iterations.	10
Disparity trilateral filtering		
σ_s	Spatial filter bandwidth.	10
σ_d	Disparity filter bandwidth.	0.1
σ_c	Color filter bandwidth (Lab).	0.5
Diffusion—all parameters shared across all scenes.		
p	Point in set.	
$\mathcal{A}$	Point set.	
λ_d	Bi-directional solve data term.	10^6 if $p \in \mathcal{A}$; 0 otherwise.
λ_s	Bi-directional solve smoothness term.	$1/\|\nabla I(p)+\epsilon\|$
r, ϵ	Weighted median filter parameters	7, 10^{-6}
$E_d(\cdot)$	Data term in optimization defined over discrete pixels	
$E_s(\cdot, \cdot)$	Smoothness term in optimization defined over pairs of pixels	
Differential optimization		
$\mathcal{I}$	Set of n multi-view images	$\mathcal{I} = \{\mathrm{I}_0, \mathrm{I}_1, ..., \mathrm{I}_n\}$
$\mathcal{P}$	Noisy scene points in $\mathbb{R}^3$	
$\mathrm{S}(\cdot, \cdot)$	Sparse depth labels in continuous image space	$\mathrm{S} : \mathbb{R}^2 \to \mathbb{R}$
$\lambda(\cdot, \cdot)$	Splatting function in continuous image space	$\lambda : \mathbb{R}^2 \to \mathbb{R}$
$\vartheta(\cdot, \cdot)$	Local smoothness weight in continuous image space	$\vartheta : \mathbb{R}^2 \to \mathbb{R}$
$\mathrm{S}^{\mathbb{Z}}(\cdot, \cdot)$	Sparse depth labels in discrete pixel space	$\mathrm{S} : \mathbb{Z}^2 \to \mathbb{R}$
$\lambda(\cdot, \cdot)^{\mathbb{Z}}$	Splatting function in discrete pixel space	$\lambda : \mathbb{Z}^2 \to \mathbb{R}$
$\vartheta(\cdot, \cdot)^{\mathbb{Z}}$	Local smoothness weight in discrete pixel space	$\vartheta : \mathbb{Z}^2 \to \mathbb{R}$
σ_s	The spatial spread of the Gaussian label	1.3
σ_λ	The spatial spread of the super-Gaussian weight	0.71
σ_Z	The spatial spread of a 3D Gaussian in the depth dimension, used for modeling occlusion of point labels	1.0
p	Order of higher-order Gaussian (*super*-Gaussian) labeling funciton	2