Differentiable Diffusion for Dense Depth Estimation from Multi-view Images

Numair Khan
Brown University

Min H. Kim
KAIST

James Tompkin
Brown University

Abstract

We present a method to estimate dense depth by optimizing a sparse set of points such that their diffusion into a depth map minimizes a multi-view reprojection error from RGB supervision. We optimize point positions, depths, and weights with respect to the loss by differential splatting that models points as Gaussians with analytic transmittance. Further, we develop an efficient optimization routine that can simultaneously optimize the 50k+ points required for complex scene reconstruction. We validate our routine using ground truth data and show high reconstruction quality. Then, we apply this to light field and wider baseline images via self supervision, and show improvements in both average and outlier error for depth maps diffused from inaccurate sparse points. Finally, we compare qualitative and quantitative results to image processing and deep learning methods.

1. Introduction

In multi-view reconstruction problems, estimating dense depth can be difficult for pixels in smooth regions. As such, 2D diffusion-based techniques [19, 8] perform gradient-based densification using only a sparse set of depth labels in image space. These assume smoothness between points to densify the point set. Smoothness can be a good assumption; for instance, many indoor scenes have low texture walls and adhere to the basic assumption that diffusion implies. But, diffusion from noisy point samples may produce results with lower accuracy, and it can be difficult to identify and filter out noisy or erroneous points from a sparse set. However, given the correct noise-free constraints, diffusion can be shown to produce comparable or better results than state-of-the-art dense processing methods.

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

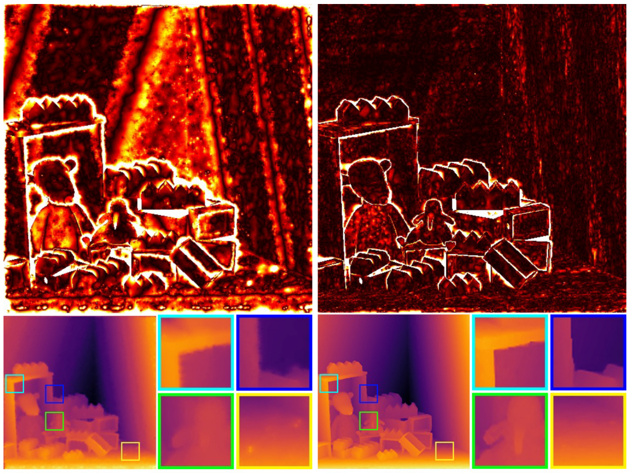


Figure 1: **Left:** Diffusion from an noisy point set produces significant errors from points found along RGB edges instead of depth edges (*top*, log absolute error). In the background, smooth depth regions have banding; in the foreground, RGB texture details are pulled into the depth causing outliers on the floor. **Right:** Our differentiable point optimization reduces error across the image, removing banding errors and minimizing texture pull.

On synthetic and real-world data across narrow-baseline light field multi-view data, and with initial results on wider-baseline unstructured data, we show that our method reduces significant diffusion errors caused by noisy or spurious points. Further, we discuss why edges are difficult to optimize via reprojection from depth maps. Finally, in comparisons to both image processing and deep learning baselines, our method shows competitive performance especially in reducing bad pixels. Put together, we show the promise of direct point optimization for diffusion-based dense depth estimation.

Data, code, and results: visual.cs.brown.edu/diffdiffdepth

2. Related Work

Sparse Depth Estimation and Densification Our problem begins with sparse depth estimates, e.g., from multiple views [27], and relates to dense depth estimation [28, 10] and depth densification or completion. Early work used cross-bilateral filters to complete missing depth samples [25]. Chen et al. learn to upsample low-resolution depth camera input and

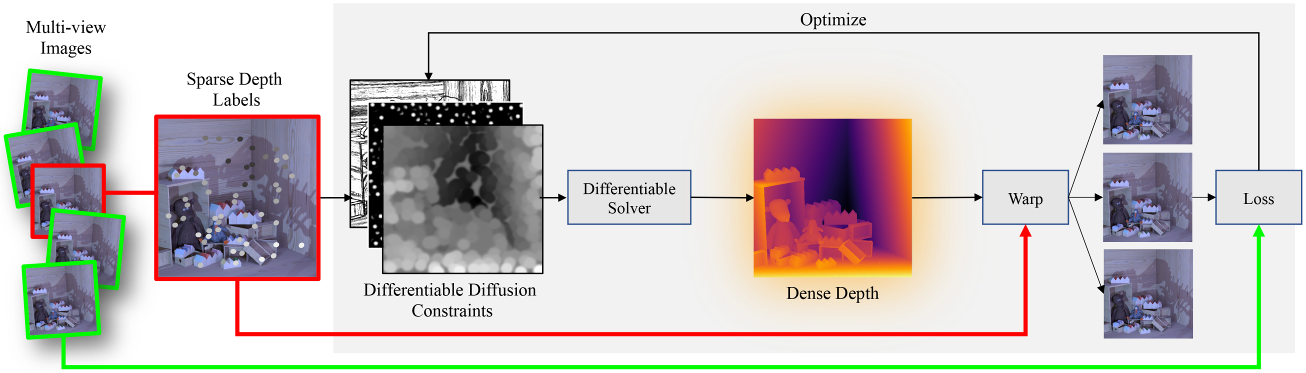


Figure 2: From a set of noisy sparse depth samples, our method uses differentiable splatting and diffusion to produce a dense depth map. Then, we optimize point position, disparity, and weight against an RGB reprojection loss. This reduces errors in the initial set of points.

regularize it from paired RGB data [5]. Imran et al. consider the problem of depth pixels being interpolated across discontinuities, and compensate by learning inter-depth object mixing [11]. Efficient computation is also addressed by Holynski and Kopf [8], 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 [18]. We consider the problem of when depth samples themselves may not be accurate, and any resulting densification without correcting the samples will lead to error.

Depth Estimation for Light Fields We demonstrate our results primarily on light fields. Khan et al. find accurate sparse Epipolar Plane Image (EPI) points using large Prewitt filters [16], then diffuses these across all views using occlusion-aware edges [15]. Zhang et al. [42] propose an EPI spinning parallelogram operator with a large support to reliably find points, and Tošić and Berkner [35] create light field scale-depth spaces with specially adapted convolution kernels. Wang et al. [36, 37] exploit angular EPI views to address the problem of occlusion, and Tao et al. [34] uses both correspondence and defocus in a higher-dimensional EPI space for depth estimation.

Beyond EPIs, Jeon et al.'s [12] method exploits defocus and depth too and builds a subpixel cost volume. Chuchwara et al. [6] present an efficient method based on superpixels and PatchMatch [3] that works well for wider-baseline views. Chen et al. [4] estimate occlusion boundaries with superpixels in the central view to regularize depth estimation.

Deep learned 'priors' can guide the estimation process. Alperovich et al. [2] showed that an encoder-decoder can be used to perform depth estimation for the central cross-hair of views. Huang et al.'s [10] work can handle an arbitrary number of uncalibrated views. Shin et al. [31] combine four different angular directions using a CNN, with data augmentation to overcome limited light field training data. Shi et al. estimate depth by fusing outputs from optical flow networks across a light field [30]. Jiang et al. [13, 14] learn to estimate depth for every pixel in a light field using a low-rank inpainting to complete disoccluded regions. Finally, most recently, Li et al. use oriented relation networks to learn depth from local EPI analysis [20].

Differentiable Rendering Unlike the approaches mentioned thus far, we use differentiable rendering to optimize sparse constraints through a differentiable diffusion process. Xu et al. [40] use differentiable diffusion based on convolutions for coarse depth refinement. We build upon radiative energy transport models that approximate transmittance through a continuously-differentiable isotropic Gaussian representation [24]. In this area, and related to layered depth images [29], recent work in differentiable rendering has addressed multi-plane transmittance for view synthesis [43, 21]. Other works consider transmittance in voxel-based representations [22] and for differentiable point cloud rendering [41, 7]. A known challenge with differentiable point clouds is backpropagating the 3D point locations through a differentiable renderer via a splatting algorithm [26]. Wiles et al. [39] present a neural point cloud renderer that allows gradients to be back propagated to a 3D point cloud for view synthesis. We propose a method to meet this challenge by directly rendering depth, and use it to show how to optimize sparse depth samples to correctly reproject RGB samples across multi-view data.

3. Depth via Differentiable Diffusion

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

3.1. Depth Diffusion

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

$$\begin{aligned} \mathrm{D}_o = \underset{\mathrm{D}}{\operatorname{argmin}} \iint_\Omega & \lambda(x,y)(\mathrm{D}(x,y) - \mathrm{S}(x,y))^2 \\ & + \vartheta(x,y) \|\nabla \mathrm{D}(x,y)\| dxdy, \end{aligned} \tag{1}$$

where $\lambda(x,y) = \sum_{\mathbf{x} \in \mathcal{P}} \delta((x,y) - \mathrm{K}\mathbf{x})$ is a sum of point masses centered at the projection of $\mathcal{P}$—the *splatting* function. The second term enforces smoothness; ϑ is low around depth edges where it is desirable to have high gradients. Solving Equation (1) in 3D is expensive and complex, needing, e.g.,

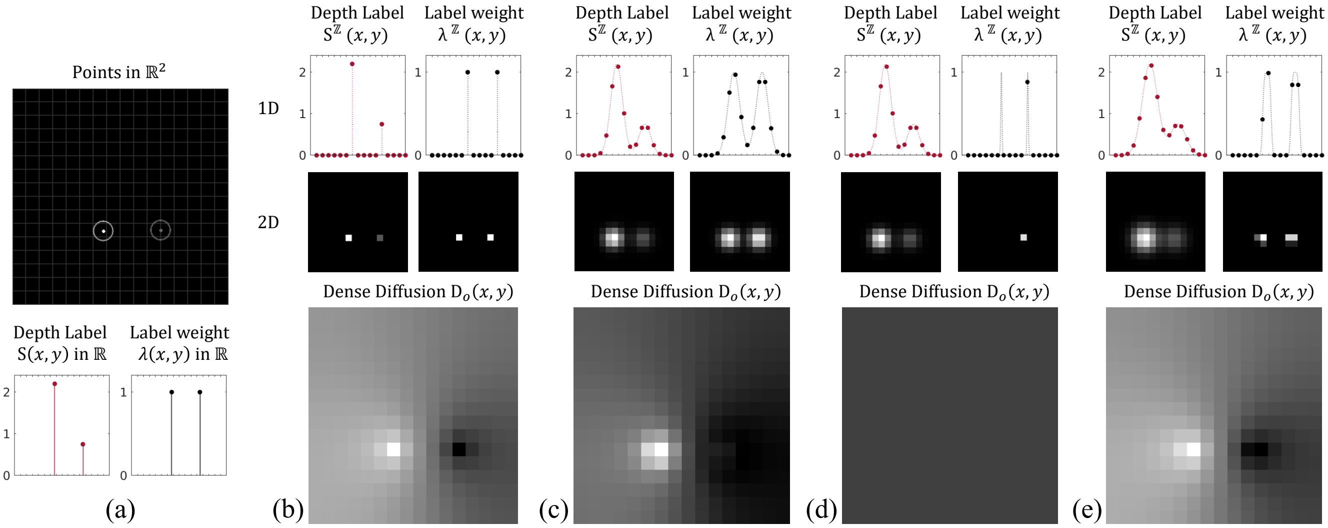


Figure 3: Depth diffusion happens in image space, so how we splat a set of scene points in $\mathbb{R}^3$ onto a pixel grid in $\mathbb{Z}^2$ has a significant impact on the results. **(a)** The image-space projection of scene points are Dirac delta functions which cannot be represented in discrete pixels. **(b)** Rounding the projected position to the closest pixel provides the most accurate splatting of depth labels for diffusion, even if it introduces position error. Unfortunately, the functional representation of the splatted point remains a non-differentiable Dirac delta. **(c)** Image-space Gaussians provide a differentiable representation, but the depth labels are not accurate. Since the label weights $\lambda^{\mathbb{Z}}$ are no longer point masses, non-zero weight is assigned to off-center 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.

voxels or a mesh. More practically, the energy in Equation (1) is minimized over a discrete pixel grid with indices x,y:

$$\mathrm{D}_o = \operatorname*{argmin}_{\mathrm{D}} \sum_{(x,y)} \Big(\lambda^{\mathbb{Z}}(x,y)\big(\mathrm{D}(x,y)-\mathrm{S}^{\mathbb{Z}}(x,y)\big)^2 + \sum_{(u,v)\in\mathcal{N}(x,y)} \vartheta^{\mathbb{Z}}(x,y)\|\mathrm{D}(u,v)-\mathrm{D}(x,y)\| \Big), \tag{2}$$

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 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 K**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 3 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.

3.2. Differentiable Image-space Representation

A common smooth representation is to model the density **x** at a three-dimensional scene point as a sum of scaled isotropic Gaussians [24, 32]. 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 3(e) where the scene point projecting onto a sub-pixel location ends up with zero pixel weight—effectively vanishing.

Our proposed representation overcomes these problems by modeling depth labels as scaled Gaussians centered at the 2D projection K**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 helps minimize the "leakage" of weight onto neighboring pixels (Fig. 3c). But unlike a delta, it is differentiable, and can be sized to match some pixel extent so that points do not vanish (Figs. 4d & 4e). Thus, we define the discrete functions:

$$\mathrm{S}^{\mathbb{Z}}(x,y) = \sum_{\mathbf{x}\in\mathcal{P}} \alpha_{\mathbf{x}}(x,y)\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y), \tag{3}$$

where $\alpha_{\mathbf{x}}(x,y)$ is a function that will merge projected labels in screen space (we will define $\alpha_{\mathbf{x}}$ in Sec. 3.3), and $\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}$ declares the label contribution at pixel (x,y) from a *single* scene point $\mathbf{x} = (X_{\mathbf{x}}, Y_{\mathbf{x}}, Z_{\mathbf{x}})$ with projection $\mathrm{K}\mathbf{x} = (x_{\mathbf{x}}, y_{\mathbf{x}})$. We define $\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}$ as:

$$\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y) = Z_{\mathbf{x}} \exp\left(-\frac{(x-x_{\mathbf{x}})^2 + (y-y_{\mathbf{x}})^2}{2\sigma^2_{\mathrm{S}}} \right). \tag{4}$$

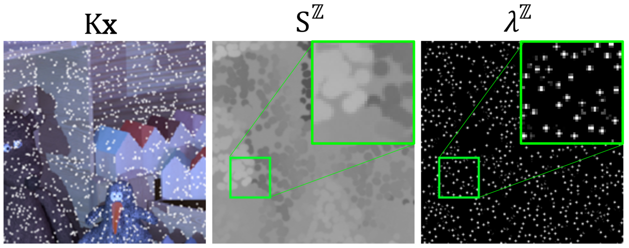


Figure 4: *Left:* The image-space projection K**x** of scene points **x** $\in \mathcal{P}$ plotted in white. *Middle:* Our differentiable labeling function $\mathrm{S}^{\mathbb{Z}}$ accurately splats depth labels while handling occlusion. *Right:* A higher-order Gaussian representation of $\lambda^{\mathbb{Z}}$ is differentiable, and provides weights that are close to point masses without any points vanishing during discretization.

Similarly, the discrete label weights are defined as:

$$\lambda^{\mathbb{Z}}(x,y) = \sum_{\mathbf{x}\in\mathcal{P}} \alpha_{\mathbf{x}}(x,y)\lambda^{\mathbb{Z}}_{\mathbf{x}}(x,y), \tag{5}$$

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

$$\lambda^{\mathbb{Z}}_{\mathbf{x}}(x,y) = w_{\mathbf{x}}\exp\left(-\frac{(x-x_{\mathbf{x}})^2+(y-y_{\mathbf{x}})^2}{2\sigma_\lambda^2}\right)^p, \tag{6}$$

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 (Sec. 3.3). Using higher-order Gaussians for depth requires differentiating a transmission integral (upcoming Eq. (7)), yet no analytic form exists for higher-order Gaussians (with an isotropic Gaussian, a representation in terms of the *lower* incomplete gamma function γ is possible, but the derivative is still notoriously difficult to estimate).

3.3. 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 (3) and (5) to accumulate values at any local pixel (x,y) that receives significant density contribution from multiple $S^{\mathbb{Z}}_{\mathbf{x}}$. This accumulation must maintain the differentiability of $\mathrm{S}^{\mathbb{Z}}$ and must ensure correct occlusion ordering so that an accurate depth label is splatted at (x,y). Using a Z-buffer to handle occlusion by overwriting depth labels and weights from back to front makes $\mathrm{S}^{\mathbb{Z}}$ non-differentiable.

We diffuse projected points in 2D; however, to motivate and illustrate the derivation of $\alpha_{\mathbf{x}}$, we will temporarily elevate our differentiable screen-space representation to $\mathbb{R}^3$ and use an orthographic projection—this provides the simplest 3D representation of our '2.5D' data labels, and allows us to formulate $\alpha_{\mathbf{x}}$ using the tools and settings of radiative energy transfer through participating media [24].

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:

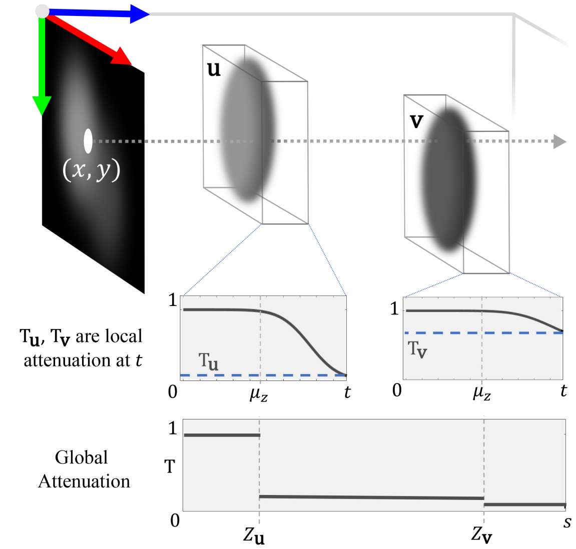

u
v
(x, y)
$\mathrm{T}_{\mathbf{u}}$, $\mathrm{T}_{\mathbf{v}}$ are local attenuation at t
1
$\mathrm{T}_{\mathbf{u}}$
0
μ_z
t
1
$\mathrm{T}_{\mathbf{v}}$
0
μ_z
t
Global Attenuation
1
T
0
$Z_{\mathbf{u}}$
$Z_{\mathbf{v}}$
s

Figure 5: 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 **u** and **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 $z < s$.

$$\mathrm{T}(x,y,s) = \exp\left(-\int_0^s \rho\sum_{\mathbf{x}\in\mathcal{P}}\exp\left(-\left(\frac{(x-x_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} + \frac{(y-y_{\mathbf{x}})^2}{2\sigma_{\mathrm{S}}^2} + \frac{(z-Z_{\mathbf{x}})^2}{2\sigma_Z^2}\right)\right)dz\right). \tag{7}$$

As $\sigma_Z \to 0$, the density contribution at any point s along the ray will come from only a single Gaussian. Furthermore, as the contribution of each Gaussian is extremely small beyond a certain distance, and as the attenuation along a ray in empty space does not change, we can redefine the bounds of the integral in a local frame of reference. Thus, we consider each Gaussian as centered at μ_z in its local coordinate frame with non-zero density only on $[0,t]$ (Figure 5). The independence of Gaussians lets us split the integral over $[0,s]$ into a sum of integrals, each over $[0,t]$ (please see supplemental document for detailed derivation). Using the product rule of exponents, we can rewrite Equation (7) as:

$$\begin{aligned}\mathrm{T}(x,y,s) &= \prod_{\mathbf{x}}\exp\left(-\int_0^t \rho\frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)}{Z_{\mathbf{x}}}\exp\left(-\frac{(z-\mu_z)^2}{2\sigma_Z^2}\right)dz\right) \\ &= \prod_{\mathbf{x}}\mathrm{T}_{\mathbf{x}}(x,y),\end{aligned} \tag{8}$$

where the product is over all $\mathbf{x} \in \mathcal{P} \mid Z_\mathbf{x} < s$. By looking again at Equation (4), we can see that $\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)/Z_\mathbf{x}$ is simply the normalized Gaussian density in xy.

Each $\mathrm{T}_\mathbf{x}$ is independent, allowing parallel calculation:

$$\begin{aligned}\mathrm{T}_\mathbf{x}(x,y) = \exp\Bigg(&\sqrt{\frac{\pi}{2}}\frac{\sigma_Z\;\rho\;\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)}{Z_\mathbf{x}} \\ &\left(-\mathrm{erf}\left(\frac{\mu_z}{\sigma_Z\sqrt{2}}\right) - \mathrm{erf}\left(\frac{t-\mu_z}{\sigma_Z\sqrt{2}}\right)\right)\Bigg) \\ = \exp&\left(c\frac{\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)}{Z_\mathbf{x}}\right),\end{aligned} \tag{9}$$

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 [24]:

$$\mathrm{S}^{\mathbb{Z}}(x,y) = \int_0^\infty \mathrm{T}(s,x,y)\mathrm{a}(s,x,y)\mathrm{P}(s,x,y)ds, \tag{10}$$

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:

$$\begin{aligned}\mathrm{S}^{\mathbb{Z}}(x,y) = \int_0^\infty \mathrm{T}(x,y,s)\sum_{\mathbf{x}\in\mathcal{P}} Z_\mathbf{x}\rho\exp\Bigg(&-\frac{(x-x_\mathbf{x})^2}{2\sigma_\mathrm{S}^2} \\ &+\frac{(y-y_\mathbf{x})^2}{2\sigma_\mathrm{S}^2}+\frac{(s-Z_\mathbf{x})^2}{2\sigma_Z^2}\Bigg)ds.\end{aligned} \tag{11}$$

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 $Z_\mathbf{x}$:

$$\begin{aligned}\mathrm{S}^{\mathbb{Z}}(x,y) &= \sum_{\mathbf{x}\in\mathcal{P}}\sum_{s\in\mathcal{N}_\mathbf{x}} \mathrm{d}s\mathrm{T}(x,y,s)\rho\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)\exp\left(-\frac{(s-Z_\mathbf{x})^2}{2\sigma_Z^2}\right) \\ &= \sum_{\mathbf{x}\in\mathcal{P}}\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y)\sum_{s\in\mathcal{N}_\mathbf{x}}\mathrm{d}s\mathrm{T}(x,y,s)\;\rho\;\exp\left(-\frac{(s-Z_\mathbf{x})^2}{2\sigma_Z^2}\right) \\ &= \sum_{\mathbf{x}\in\mathcal{P}}\alpha_\mathbf{x}(x,y)\mathrm{S}^{\mathbb{Z}}_{\mathbf{x}}(x,y).\end{aligned} \tag{12}$$

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

$$\alpha_\mathbf{x}(x,y) = \frac{\rho\mathrm{d}s}{Z_\mathbf{x}}\sum_{s\in\mathcal{N}_\mathbf{x}}\mathrm{T}(s,x,y)\;\rho\;\exp\left(-\frac{(s-Z_\mathbf{x})^2}{2\sigma_Z^2}\right). \tag{13}$$

3.4. Optimization by Gradient Descent

To restate our goal, we want to optimize the parameters $\Theta = \{\mathrm{S}^{\mathbb{Z}}, \lambda^{\mathbb{Z}}, \vartheta^{\mathbb{Z}}\}$ for dense depth diffusion (Eq. (2)). The function $\mathrm{S}^{\mathbb{Z}}(x,y)$ proposes a depth label at pixel (x,y), $\lambda^{\mathbb{Z}}(x,y)$ determines how strictly this label is applied to the pixel, and $\vartheta^{\mathbb{Z}}(x,y)$ controls the smoothness of the output depth map at (x,y). We find Θ by using gradient descent to minimize a loss function $\mathrm{L}(\Theta)$. Using our differentiable representation, we can express $\mathrm{S}^{\mathbb{Z}}$ and $\lambda^{\mathbb{Z}}$ in terms of the image-space projection of the sparse point set $\mathcal{P}$. Doing so provides strong constraints on both the initial value of these two 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 is effective and generates high-quality depth maps; we refer the reader to the supplemental document for details. This shows the potential of our differentiable sparse point optimization and diffusion method, and inform us of the contribution on the final result of the follow self-supervised loss for captured images.

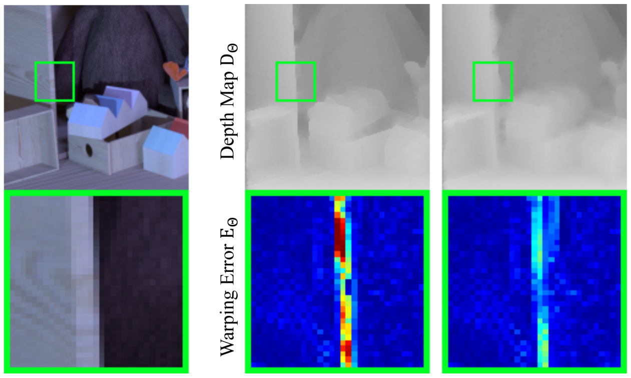
Self-supervised Loss Working with a set of multi-view images $\mathcal{I} = \{\mathrm{I}_0, \mathrm{I}_1, ..., \mathrm{I}_n\}$ allows us to define a self-supervised loss function for the optimization. Given a dense depth map D_Θ generated by diffusion with parameters Θ, we define the warping operator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

$$\mathrm{E}_\Theta(x,y) = \frac{1}{\sum_i \mathrm{M}^i_\Theta(x,y) + \epsilon}\sum_i\Big(|\mathrm{I}(x,y) - \mathcal{W}_\Theta[\mathrm{I}^i](x,y)|\;\mathrm{M}^i_\Theta(x,y)\Big), \tag{14}$$

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

We observe that E_Θ is non-zero even if we use the ground truth depth map, because small pixel errors are inevitable during the sub-pixel interpolation for warping. However, the more significant errors come from an unexpected source: the sharpness of depth edges. Depth labels are ambiguous at pixels lying on RGB edges, and limited sampling frequency blurs these edges within pixels (Figure 6). By assigning a fixed label to these pixels, sharp depth edges cause large errors. Consequently, the optimization process smooths all edges. While doing so minimizes the reprojection error, it may be desirable to have sharp depth edges for aesthetic and practical purposes, even if the edge location is slightly incorrect.

Therefore, we add a loss term to reward high gradients in E_Θ, effectively allowing the optimization to ignore errors caused by sharp depth edges. In addition, we include a smoothness term E_S similar to Ranjan et al. [23] to encourage depth to be guided by image edges, and a structural self-similarity error [38] E_SSIM which is commonly used to regularize warping error.



(a) (b) (c)

Figure 6: Our everyday intuition says that depth edges should be sharp, but a limited sampling rate blurs them in the RGB input **(a)**. This can cause unintended high error during optimization via 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 (Eq. (15)).

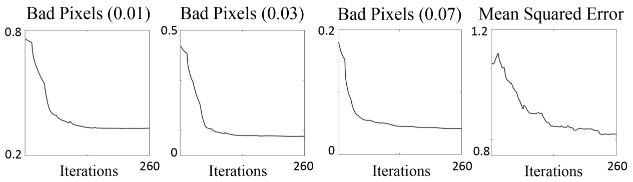


Figure 7: Over optimization iterations, mean squared error reduces and 'bad pixels' are significantly suppressed. From the corresponding error map in Figure 1 we can see that most remaining errors lie along edges, where depth is not well defined (as per Figure 6)—the ground truth depth values are pixel-rounded while the RGB image contains blurred depth edges.

Our final loss function becomes:

$$\mathrm{L}(\Theta)=\sum_{(x,y)}\Big(\mathrm{E}_\Theta(x,y)+\mathrm{E}_\mathrm{S}+\mathrm{E}_\mathrm{SSIM}-\nabla\mathrm{E}_\Theta(x,y)\Big). \quad (15)$$

3.5. Implementation

Our proposed framing of the diffusion problem allows us to express $\mathrm{S}^\mathbb{Z}$ and $\lambda^\mathbb{Z}$ as differentiable functions of the points set $\mathcal{P}$, and thus, to calculate $\partial\mathrm{L}/\partial\mathbf{x}$. Since $\mathcal{P}$ provides strong constraints on the shape of these functions, we optimize over the parameters $\mathrm{Z}_\mathbf{x}$, $\mathrm{K}\mathbf{x}$, $w_\mathbf{x}$, and $\vartheta^\mathbb{Z}$ instead of directly over Θ ($w_\mathbf{x}$ is the scaling factor from Equation (6)). To regularize the smoothness and data weights, we further define $\vartheta^\mathbb{Z}(x,y)=\exp(-Q(x,y))$ and $w_\mathbf{x}=\exp(-R(\mathbf{x}))$, for some unconstrained R and Q that are optimized. Thus, our final parameter set is $\bar{\Theta}=\{\mathrm{Z}_\mathbf{x},\mathrm{K}\mathbf{x},R,Q\}$. We initialize $R(\mathbf{x})$ to zero for all $\mathbf{x}$, and $Q(x,y)$ to the magnitude of the image gradient $\|\nabla I\|$.

Distance Both RGB and VGG16 features can be used as distances for warping loss E_Θ; we found VGG16 features to outperform RGB. VGG loss has a better notion of space from a larger receptive field and handles textureless regions better. Thus, we take each warped image in Equation (14), 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 require a trade-off between resource use and accuracy. The parameter σ_S in Equation (3) 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 3(c) and leads to incorrect diffusion results. However, ensuring a uniform weight requires having a large value of σ_S, and this may cause the labels of neighboring points to be occluded. We found that using $\sigma_\mathrm{S}=1.3$ provides a good balance between accuracy and compactness. This spreads the label density over three pixels in each direction before it vanishes, so we use a Gaussian kernel size of 7×7.

For σ_Z, we want the spread to be as small as possible. However, if the value is very small then we must use a large number of samples in $\mathcal{N}_\mathbf{x}$ when calculating the quadrature in Equation (13). 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. (5)). As the order is increased, the Gaussian becomes more similar to a box function and leaks less weight onto neighboring pixels. However, its gradients become smaller, and the loss takes longer to converge. With $p=2$, we calculate $\sigma_\lambda=0.71$ to provide the necessary density to prevent points from vanishing (Fig. 3(d)).

Routine We use Adam [17]. 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 (3) by summing over all points $\mathbf{x}$ is impossibly slow for any scene of reasonable complexity. Fortunately, in practice we only need to sum the contribution from a few points $\mathbf{x}$ at each pixel and, so, the computation of $\alpha_\mathbf{x}(x,y)$ in Equation (13) 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 (2), and back-propagate gradients through the solver and renderer in five seconds.

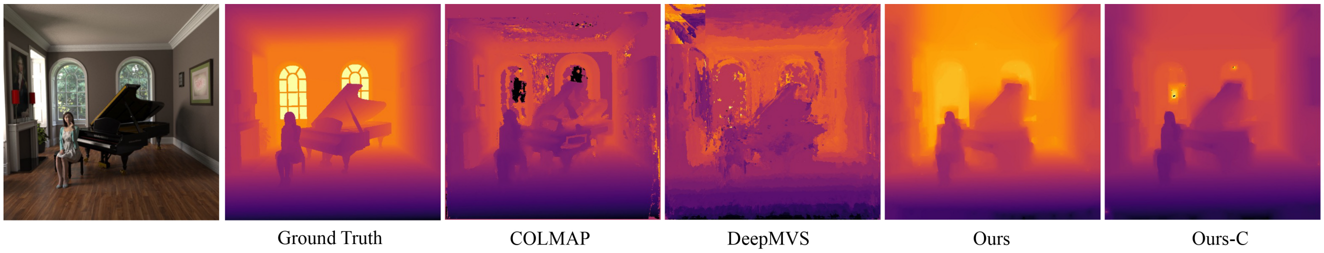


Figure 8: Depth results on the synthetic *Piano-MVS* scene. *Left to right:* Ground truth, dense reconstruction from COLMAP [28, 27], DeepMVS [10], our method using 702 sparse points in $\mathcal{P}$, and our method with 2,808 sparse points from dense COLMAP output.

Software and Hardware We implement our method in PyTorch. For diffusion, we implement a differentiable version of Szeliski's LAHBPCG solver [33]. All CPU code was run on an AMD Ryzen Threadripper 2950X 16-Core Processor, and GPU code on an NVIDIA GeForce RTX 2080Ti.

4. Experiments

4.1. Light Fields

Datasets Along with the *Dino*, *Sideboard*, *Cotton*, and *Boxes* scenes from the synthetic HCI Dataset [9], we add two new *Living Room* and *Piano* scenes with more realistic lighting, materials, and depth ranges. We path trace these with Arnold in Maya. All synthetic light fields have 9×9 views, each of 512 × 512 pixels. For real-world scenes, we use light fields from the New Stanford Light Field Archive [1]. These light fields have 17×17 views captured from a camera rig, with a wider baseline and high spatial resolution (we downsample 2× for memory).

Baselines and Metrics For our method, we use an initial point set extracted from EPI edge filters [16]. We compare to the methods of Zhang et al. [42], Khan et al. [15], Jiang et al. [13], Shi et al. [30], and Li et al. [20]. Khan et al.'s algorithm is diffusion-based, whereas the last three methods are deep-learning-based. For metrics, we use mean-squared error (MSE) and *bad pixels* (BP). BP measures the percentage of pixels with error higher than a threshold. For real-world scenes without ground truth depth, we provide a measure of performance as the reprojection error in LAB induced by depth-warping the central view onto the corner views; please see our supplemental material.

Results While learning-based methods [13, 30, 20] tend to do well on the HCI dataset, their quantitative performance degrades on the more difficult *Piano* and *Living Room* scenes (Tab. 2). A similar qualitative trend shows the learning-based methods performing worse than diffusion on the real-world light fields (Fig. 9). Our method provides more consistent overall performance on all datasets. Moreover, the existing diffusion-based method [15] has few pixels with very large errors but many pixels with small errors, producing consistently low MSE but more bad pixels. In contrast, our method consistently places in the top-three on the bad pixel metrics. We show additional results and error maps in our supplemental material. Finally, as is common, it is possible to post-process our results with a weighted median filter to reduce MSE (e.g., *Dino* 0.54 vs. 0.86) at the expense of increased bad pixels (BP(0.01) of 39.6 vs. 25.6).

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

Table 1: Quantitative results for wider-baseline unstructured five-camera cases, as the *Living Room-MVS* and *Piano-MVS* scenes.

4.2. Multi-view Stereo

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

Baselines and Metrics We compare to dense reconstruction from COLMAP [28] and to DeepMVS [10]. Out method uses the sparse output of COLMAP as the initial point set, which is considerably sparser than the initial set for light fields (500 vs. 50k). To increase the number of points, we diffuse a preliminary depth map and optimize the smoothness parameter for 50 iterations. Then, we sample this result at RGB edges. Using this augmented set, we optimize all parameters in turns of 25 iterations, repeated 5 times. In addition, we also evaluate a variant of our method, Ours-C, with sparse labels initialized from the dense COLMAP output at RGB edges.

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

Results To account for the error in least squares, Table 1 presents the minimum of ten different fits for each method. Both DeepMVS and COLMAP generate results with many invalid pixels. We assign such pixels the mean GT depth. Our method outperforms the baselines with a sparse point set (≈700 points) and generates smooth results by design that qualitatively have fewer artifacts (Fig. 8). Using 4× as many initial points (≈2,800 points) in the Our-C variant leads to additional improvements.

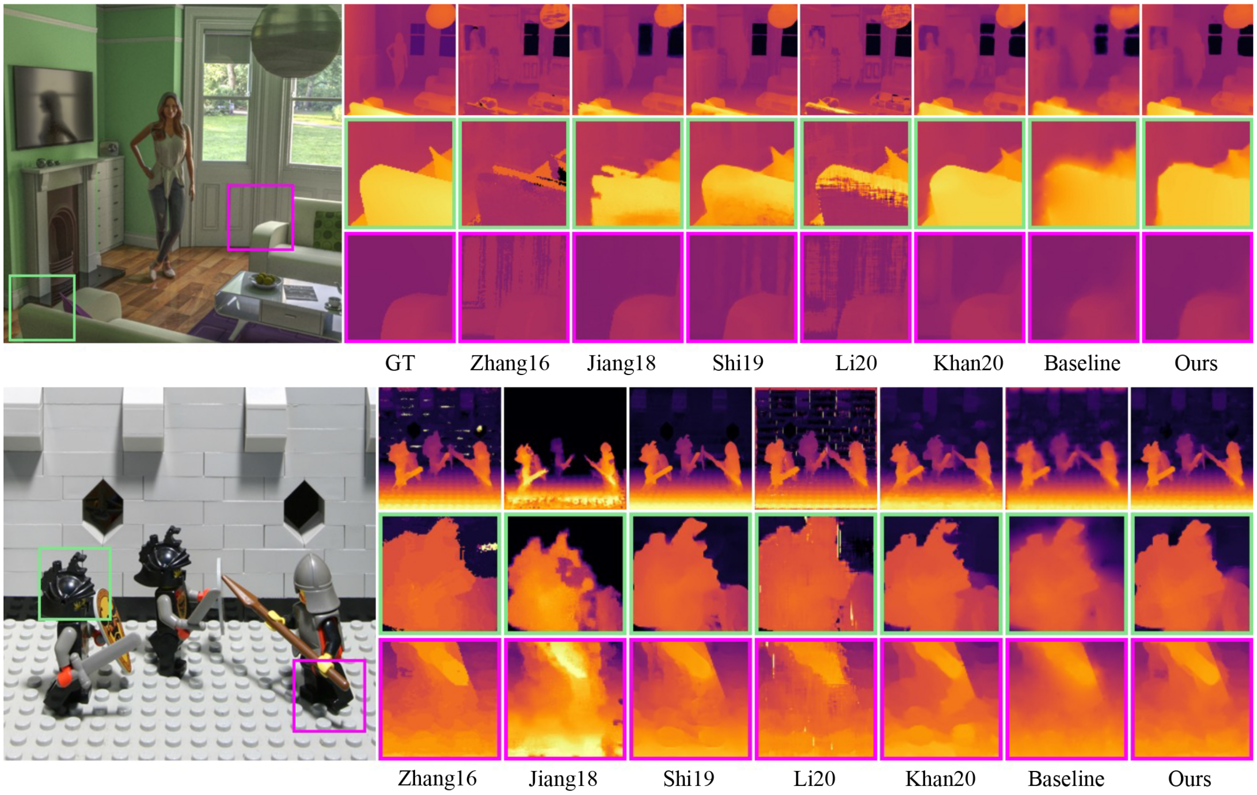


Figure 9: *Top:* Disparity results on the synthetic *Living Room* light field. *Bottom:* Disparity results on a real light field. *Left to right*: Zhang et al. [42], Jiang et al. [13], Shi et al. [30], Li et al. [20], Khan et al. [15], a baseline diffusion result without any optimization, our results and finally, for the top light field, ground truth.

Light Field	MSE * 100						BP(0.1)						BP(0.3)						BP(0.7)					
	[42]	[20]	[13]	[30]	[15]	Ours	[42]	[20]	[13]	[30]	[15]	Ours	[42]	[20]	[13]	[30]	[15]	Ours	[42]	[20]	[13]	[30]	[15]	Ours
Living Room	0.67	0.57	0.23	0.25	0.25	0.20	59.5	58.5	37.2	48.0	47.2	30.3	43.3	42.7	23.7	26.5	25.0	17.5	17.0	16.6	11.4	10.8	11.5	9.23
Piano	26.7	13.7	14.4	8.66	12.7	8.71	36.7	27.5	24.7	27.0	37.6	17.0	25.0	17.6	13.6	11.4	20.0	7.93	5.33	4.13	5.88	4.29	4.95	3.49
Light Field	MSE * 100						BP(0.01)						BP(0.03)						BP(0.07)					
	[42]	[20]	[13]	[30]	[15]	Ours	[42]	[20]	[13]	[30]	[15]	Ours	[42]	[20]	[13]	[30]	[15]	Ours	[42]	[20]	[13]	[30]	[15]	Ours
Sideboard	1.02	1.89	1.96	1.12	0.89	2.23	78.0	62.3	47.4	53.0	73.8	43.0	42.0	18.0	18.3	20.4	37.4	16.5	14.4	6.50	9.31	9.02	16.2	8.35
Dino	0.41	3.28	0.47	0.43	0.45	0.86	81.2	52.7	29.8	43.0	69.4	25.6	48.9	12.8	8.81	13.1	30.9	7.69	7.52	5.82	3.59	4.32	10.4	4.06
Cotton	1.81	1.95	0.97	0.88	0.68	3.07	75.4	58.8	25.4	38.6	56.2	31.1	34.8	14.0	6.30	9.60	18.0	7.82	4.35	4.11	2.02	2.74	4.86	4.06
Boxes	7.90	4.67	11.6	8.48	6.69	9.17	84.7	68.3	51.8	66.5	76.8	60.3	55.3	28.0	27.0	37.2	47.9	32.7	18.9	13.4	18.3	21.9	28.3	20.5

Table 2: **(Best viewed in color)** Quantitative comparison on synthetic light fields. The top three results are highlighted in gold, silver and bronze. BP(x) is the number of *bad pixels* which fall above threshold x in error. Higher BP thresholds are used for *Living Room* and *Piano* as their average error is larger for all methods: they contain specular surfaces, larger depth ranges, and path tracing noise.

5. Conclusion

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

Acknowledgements We thank the reviewers for their detailed feedback. Numair Khan thanks an Andy van Dam PhD Fellowship, and Min H. Kim acknowledges the support of Korea NRF grant (2019R1A2C3007229).

References

[1] The New Stanford Light Field Archive, 2008.

[2] A. Alperovich, O. Johannsen, and B. Goldluecke. Intrinsic light field decomposition and disparity estimation with a deep encoder-decoder network. In *26th European Signal Processing Conference (EUSIPCO)*, 2018.

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

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

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

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

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

[8] Aleksander Holynski and Johannes Kopf. Fast depth densification for occlusion-aware augmented reality. In *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, volume 37. ACM, 2018.

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

[10] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2821–2830, 2018.

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

[12] Hae-Gon Jeon, Jaesik Park, Gyeongmin Choe, Jinsun Park, Yunsu Bok, Yu-Wing Tai, and In So Kweon. Accurate depth map estimation from a lenslet light field camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1547–1555, 2015.

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

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

[15] Numair Khan, Min H. Kim, and James Tompkin. Fast and accurate 4D light field depth estimation. Technical Report CS-20-01, Brown University, 2020.

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

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

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

[19] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*, pages 689–694. 2004.

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

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

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

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

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

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

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

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

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

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

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

[31] Changha Shin, Hae-Gon Jeon, Youngjin Yoon, In So Kweon, and Seon Joo Kim. Epinet: A fully-convolutional neural network

using epipolar geometry for depth from light field images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4748–4757, 2018.

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

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

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

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

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

[37] Ting-Chun Wang, Alexei A Efros, and Ravi Ramamoorthi. Depth estimation with occlusion modeling using light-field cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 38(11):2170–2181, 2016.

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

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

[40] Yan Xu, Xinge Zhu, Jianping Shi, Guofeng Zhang, Hujun Bao, and Hongsheng Li. Depth completion from sparse lidar data with depth-normal constraints. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2811–2820, 2019.

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

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

[43] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. 37(4):65:1–12, 2018.