Learning to Predict Scene-Level Implicit 3D from Posed RGBD Data

Nilesh Kulkarni, Linyi Jin, Justin Johnson, David F. Fouhey
University of Michigan

Abstract

We introduce a method that can learn to predict scene-level implicit functions for 3D reconstruction from posed RGBD data. At test time, our system maps a previously unseen RGB image to a 3D reconstruction of a scene via implicit functions. While implicit functions for 3D reconstruction have often been tied to meshes, we show that we can train one using only a set of posed RGBD images. This setting may help 3D reconstruction unlock the sea of accelerometer+RGBD data that is coming with new phones. Our system, D2-DRDF, can match and sometimes outperform current methods that use mesh supervision and shows better robustness to sparse data.

1. Introduction

Consider the image in Figure 1. From this ordinary RGB image, we can understand the complete 3D geometry of the scene, including the floor and walls behind the chairs. Our goal is to enable computers to recover this geometry from a single RGB image. To this end, we present a method that does so while learning *only* on posed RGBD images.

The task of reconstructing the full 3D of a scene from a single previously unseen RGB image has long been known to be challenging. Early work on full 3D relied on voxels [6, 17] or meshes [19], but these representations fail on scenes due to topology and memory challenges. Implicit functions (or learning to map each point in $\mathbb{R}^3$ to a value like the distance to the nearest surface) have shown substantial promise at overcoming these challenges. When conditioned on an image, these have led to several successful methods.

Unfortunately, the implicit function status quo mainly ties implicit function reconstruction methods to mesh supervision. This symbiosis has emerged since meshes give excellent direct supervision. However, methods are limited to training with an image-aligned mesh that is usually watertight (and often artist-created) [37, 41, 49] and occasionally non-watertight but professionally-scanned [28, 66].

We present a method, Depth to DRDF (D2-DRDF), that breaks the implicit-function/mesh connection and can train an effective single RGB image implicit 3D reconstruction

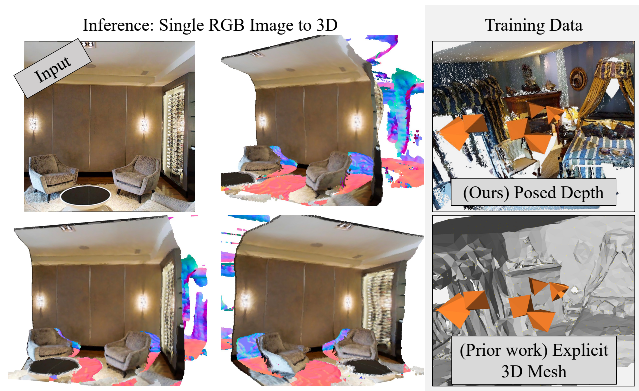


Figure 1. We propose a method, D2-DRDF, that reconstructs full 3D from a single RGB image. During inference (left), our method uses implicit functions to reconstruct the complete 3D including visible and occluded surfaces (visualized as surface normals) such as the occluded wall and empty floor. For training, our method uses *only* RGBD images and poses, unlike most prior works that need an explicit and often watertight 3D mesh.

system using a set of RGBD images with known pose. We envision that being able to entirely skip meshing will enable the use of vast quantities of lower-quality data from consumers (e.g., from increasingly common consumer phones with LiDAR scanners and accelerometers) as well as robots. In addition to permitting training, the bypassing of meshing may enable adaptation in a new environment on raw data without needing an expert to ensure mesh acquisition.

Our key insight is that we can use segments of observed free space in depth maps in other views to constrain distance functions. We show this using the Directed Ray Distance Function (DRDF) [28] that has recently shown good performance in 3D reconstruction using implicit functions and has the benefit of not needing watertight meshes for training. Given an input *reference* view, the DRDF breaks the problem of predicting the 3D surface into a set of independent 1D distance functions, each along a ray through a pixel in the reference view and accounting for only surfaces on the ray. Rather than use an ordinary unsigned distance function, the DRDF signs the distance using the location of the camera's pinhole and the intersections along the ray. While [28] showed their method could be trained on *non*-watertight meshes, their method is still dependent on meshes. In our paper, we show that the DRDF can be cleanly supervised using *auxiliary views* of RGBD observations and their poses. We derive constraints on the DRDF that power loss functions for training our system. While the losses on any one image are insufficient to produce a reconstruction, they provide a powerful signal when accumulated across thousands of training views.

We evaluate our method on realistic indoor scene datasets and compare it with methods that train with full mesh supervision. Our method is competitive and sometimes even better compared to the best-performing mesh-supervised method [29] with full, professional captures. As we degrade scan completeness, our method largely maintains its performance while mesh-based methods perform substantially worse. We conclude by showing that fine-tuning of our method on a handful of images enables a simple, effective method for fusing and completing scenes from a handful of RGBD images.

2. Related Work

Our approach infers a complete 3D scene from a single RGB image using implicit functions that are supervised via posed RGBD scans. This touches on several long-term goals of 3D computer vision that we discuss below.

Reconstructing Scenes from a Single Image. At test time, our system produces a *full* 3D reconstruction from a single RGB image, including occluded regions. This means that our desired output goes beyond 2.5D properties such as depth [4, 12], surface normals [12, 14, 60], or other intrinsic-image properties [24, 52]. Works on attempting to reconstruct complete 3D usually have focused on reconstructing objects from image by using voxels [6, 17] or meshes [18, 19], point clouds [13, 33] or CAD models [23]. These are usually trained on synthetic datasets like ShapeNet [2] or scene datasets [16, 48, 55] and do not generalize well to realistic scenes. Another line of works tries to learn holistic structures [40, 64] or planar surfaces [25, 34, 63] from realistic scanned mesh data [1, 8] Creating realistic mesh based data for scenes requires post-processing using Poisson surface reconstruction [9, 26] which leads to deviation from raw captures. Manually aligning them is expensive hence 3D object-aligned datasets like [54, 57] are scarce and limited in diversity. Our method avoids these limitations by directly operating on the raw captured RGBD data.

Reconstructing Scenes from Posed Scans. There has been considerable work on using multiview RGBD data at inference time to produce 3D reconstructions, starting with analytic techniques [7, 9, 22, 30] and now using learning [21, 56, 59]. We use some similar tools to this community: for instance, the ray distances we use are known in this community as a projected distance functions [7]. However, their works solve a fundamentally different problem by using posed RGBD data at test time: their goal is to produce a reconstruction from a set of posed RGBD images from one particular scene; our goal is to use a large dataset of posed RGBD data to train a neural network that can map a new single RGB image to a 3D reconstruction.

Our goal of learning to predict reconstructions from a single previously unseen image also distinguishes our work from NeRF [38] and other similar radiance field approaches [3]. Their goal is to learn a radiance field for a *particular* scene from a set of posed scans. There are methods that try to predict this radiance field from a single image [62]; we compare with a model using similar losses and find that an objective specialized for 3D works better.

Implicit Functions for 3D Reconstruction. We perform reconstruction with implicit functions. Implicit functions have used for shape and scene modeling as level sets [36], signed distance functions [41], occupancy function [37, 44, 49], distance functions [5, 28, 53] and other modifications like [61]. Our work has two key distinctions. First, many approaches [5, 53, 61] *fit* an implicit function to one shape (i.e., there is no generalization to new shapes). In contrast, our work produces reconstructions from previously unseen RGB images. Second, the methods that predict an implicit function from a new image [28, 66] assume access to a mesh at training time. On other hand, our work assumes access only to posed RGBD data for training.

The most similar work to ours is [28] that also aims to learn to reconstruct scenes from a single image by training on realistic data. While [28] shows how to reduce supervision requirements by enabling the use of non-watertight meshes, their method is still limited to mesh supervision. We show how to use posed RGBD data for supervision instead of using an image-aligned mesh. This substantially reduces the requirements for collecting training data.

3. Pixel-Aligned 3D Reconstruction & DRDF

We propose a method to predict full 3D scene structure from a previously unseen RGB image. At inference time, our method receives a single image with no depth information. We refer to this view as the *reference view*. As output, the method produces an estimate of a *distance function* at each point in a pre-defined set of 3D points. Given this volume of predicted distances, one can *decode* the predicted distance into a set of surfaces: e.g., if the predicted distance were the unsigned distance to the nearest surface, one could declare all points with sufficiently small predicted distance to be surfaces. At training time, our network is given supervision for predicting its distance function. Previously, supervision been done via a mesh that provides oracle distance calculations. In §4, we show how to derive supervision from posed RGBD images at *auxiliary views* instead.

The approaches in the paper follow pixelNerf [62] for predicting a distance at a 3D point $\mathbf{x}$. The network accepts

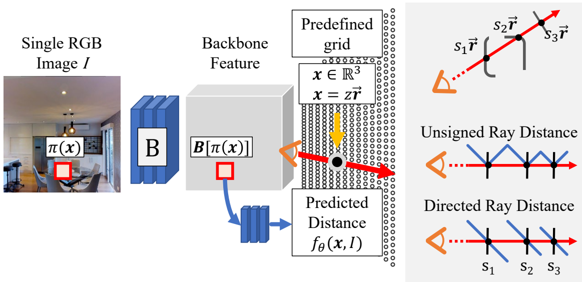


Figure 2. Model and DRDF overview. (Left) Given a single input RGB image I and query point $\mathbf{x}$ on a ray $\vec{\mathbf{r}}$ in a pre-defined grid, all models, like [62] extract image features at the projection $\pi(\mathbf{x})$ of $\mathbf{x}$ with a backbone and then predict the distance using a MLP on the backbone features and a positional encoding of $\mathbf{x}$. (Right) Given a ray $\vec{\mathbf{r}}$ through the scene, ray distance functions are defined as the distance to intersections of the ray with the scene. This permits analyzing the 3D scene as a set of 1D distances per ray. The Directed Ray Distance Function is further signed to be positive before and negative after an intersection.

an image I and 3D point $\mathbf{x}$ and makes a prediction $f_\theta(\mathbf{x}, I)$. As shown in Fig. 2, the network extracts a convolutional feature from a backbone $\mathbf{B}$ at the projection of $\mathbf{x}$, denoted $\pi(\mathbf{x})$. The image feature is concatenated to a positional encoding of $\mathbf{x}$ and put into a MLP to produce the final prediction $f_\theta(\mathbf{x}, I)$. By repeating this inference for a set of 3D points, the network can infer the full 3D of the scene.

Training. In a conventional setup, the network $f_\theta(\mathbf{x}, I)$ is trained by obtaining samples from a mesh and directly training the network via regression. Given a mesh that is co-aligned with an image, one can obtain an n tuples consisting of an image I_i, 3D point $\mathbf{x}_i$ and ground-truth distance d_i. Then the network is trained by minimizing the empirical risk $\frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(f_\theta(\mathbf{x}_i, I_i), d_i)$ for some loss $\mathcal{L}$ like L1.

What Distance Function Should We Predict? There are multiple distance functions in the literature, and the above formulation enables predicting any of them. To explain distinctions, assume we are given a point $\mathbf{x}$ at which we want to evaluate a distance and a scene $\mathcal{S} \subseteq \mathbb{R}^3$ that we want to evaluate the distance to.

In a standard distance function, one defines a distance from a point $\mathbf{x}$ to the *entire* scene $\mathcal{S}$. For instance, the *Unsigned Distance Function* (UDF) is defined as $\min_{\mathbf{x}' \in \mathcal{S}} ||\mathbf{x} - \mathbf{x}'||$. In a *ray-based distance function* [7, 28], the distance is defined from $\mathbf{x}$ only to the points in the scene that lie on the ray $\vec{\mathbf{r}}$ from the reference view pinhole towards $\mathbf{x}$. In other words, the ray distance function is defined as the distance to the intersections of the ray with the scene. For instance, the *Unsigned Ray Distance Function* (URDF) is defined as $\min_{\mathbf{x}' \in \mathcal{S}, \mathbf{x}' \text{ on } \vec{\mathbf{r}}} ||\mathbf{x} - \mathbf{x}'||$. This definition means that when predicting the distance at a point $\mathbf{x}$, all of the points that define the ray distance function project to the same pixel, $\pi(\mathbf{x})$, which in turn means that neural networks do not need as large receptive fields [28].

Since a ray-based distance function is defined *only* by scene points that intersect a ray, we can reduce the 3D problem to a 1D problem, as illustrated in Fig. 2. Rather than represent each 3D point as a 3D point, we represent it as the distance along the ray $\vec{\mathbf{r}}$: the vector $\mathbf{x}$ is represented via a scalar z such that $\mathbf{x} = z\vec{\mathbf{r}}$. Then, if the i^{th} intersection along the ray is $s_i\vec{\mathbf{r}}$ for $s_i \in \mathbb{R}$, the URDF is defined as the distance to the nearest intersection along that ray, or $d_{\text{UR}}(z) = \min_{i=1,\ldots,k} |z - s_i|$.

Directed Ray Distance Function. The Directed Ray Distance Function (DRDF) is a ray distance function that incorporates a sign into the URDF. In particular, the DRDF is defined as $d_{\text{DR}}(z) = \text{dir}(z) d_{\text{UR}}(z)$, where the predicate $\text{dir}(z)$ is positive if the nearest intersection is ahead of z along the ray from the pinhole and negative otherwise. This is pictured in Fig. 2. The presence of both positive and negative components leads to better behavior under uncertainty [28].

Critical Properties of DRDFs. Two critical properties that we use to define supervision are that at z, the distance to the nearest intersection is $|d_{\text{DR}}(z)|$ and the nearest intersection on the ray is at $z + d_{\text{DR}}(z)$.

4. Depth-Supervised DRDF Reconstuction

Having introduced the setup in §3, we now explain how to train a network to predict an image-conditioned DRDF *without* access to an underlying mesh but instead access to posed RGBD images. Concretely, given a point $z\vec{\mathbf{r}}$, we aim to define a loss function $\mathcal{L}$ that evaluates a network's prediction of the DRDF $f_\theta((z\vec{\mathbf{r}}), I)$.

Our key insight is that we can *sometimes* see this point $z\vec{\mathbf{r}}$ in other views and these observations of $z\vec{\mathbf{r}}$ provide a constraints on what the value of the DRDF can be. Consider, for instance, the point 3m out from reference view camera pinhole along the red ray in Fig. 3. If we project this point into the auxiliary view, we know that its distance to the camera is closer to auxiliary view's pinhole than the depthmap observed at the auxiliary view. Moreover, we can see that a segment of the ray is visible, starting at the point $s\vec{\mathbf{r}}$ and ending at the point $e\vec{\mathbf{r}}$ with the ending point $e\vec{\mathbf{r}}$ closer to $z\vec{\mathbf{r}}$ than $s\vec{\mathbf{r}}$ is.

We can derive supervision on $d_{\text{DR}}(z)$ from the segment of visible free space observable in the auxiliary view by using the fact that the nearest intersection at z is $|d_{\text{DR}}(z)|$ units away. Since e is nearer than s, and there are *no* intersections in the free space between s and e, we know that there cannot be an intersection within $(e - z)$ of z. This gives an inequality that $|d_{\text{DR}}(z)| \geq (e - z)$: otherwise there would be an intersection in the visible free space. If the ends of the segment are actual intersections where the ray visibly intersects an object, we obtain an equality constraint on $d_{\text{DR}}(z)$ since we actually see the nearest intersection. In other cases, the ray simply disoccludes or is occluded, and we can only have an inequality. We denote intersection start/end events as $\mathcal{I}$

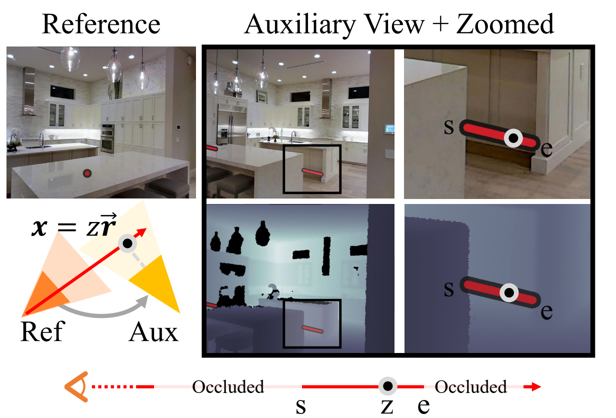


Figure 3. A red ray originates in the reference camera. Given a point $\mathbf{x}$ that is z units along the ray $\vec{\mathbf{r}}$, we can test if $z\vec{\mathbf{r}}$ is visible in an auxiliary view by comparing its projected depth with the RGBD image. The point z is visible, along with a segment of the ray from s to e units. Since the distance to the nearest intersection is $|d_{\text{DR}}(z)|$ and we know there are no intersections within $(e - z)$ units of z, we can constrain that $|d_{\text{DR}}(z)| \geq (e - z)$.

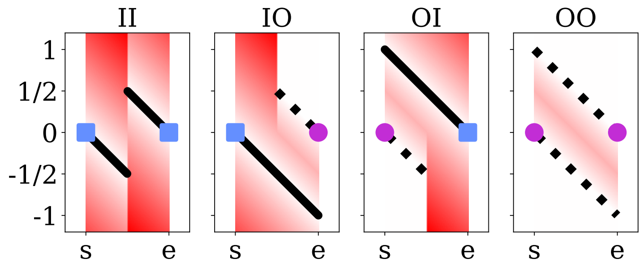


Figure 4. Loss functions for a starting s and ending e events spaced a unit apart. Intersection events (blue ■) define the DRDF precisely, and we penalize the network from deviating. Occlusion events (**purple** ●) provide bounds that we penalize the network for violating. We plot l_s as a line from the start event and l_e as a line from the end event. Loss: 0 ▭ 1.7

and occlusion start/end events as $\mathcal{O}$ and distinguish them by checking if the projected z-value of the ray in auxiliary view is sufficiently close to the recorded auxiliary view's depthmap value.

4.1. Losses

We now convert the concept of using freespace to constrain the DRDF into concrete losses. Recall that our goal is to evaluate a prediction $y = f_\theta(z\vec{\mathbf{r}}, I)$ from our network and that the point at z along $\vec{\mathbf{r}}$ is in some segment of visible freespace from s to e along $\vec{\mathbf{r}}$. Our goal is to penalize the prediction y. Since there are two start/end event classes ($\mathcal{I}$, $\mathcal{O}$), there are four segment types: $\mathcal{II}$, $\mathcal{IO}$, $\mathcal{OI}$, and $\mathcal{OO}$. We show several of these segments in Figure 5.

To assist the loss definitions, we define variables $l_s = s - z$, and $l_e = e - z$ which we plot in Fig. 4 as a function of z. l_s is the value of d_{DR} if the closest intersection is at s, and l_e is the value of d_{DR} if the closest intersection is at e. The values l_s, l_e define equalities for intersections and inequalities for occlusions.

$\mathcal{L}_{\mathcal{II}}$**:** $\mathcal{II}$ **segment.** Given a segment bounded by two intersections, the nearest intersections are known exactly as l_s or l_e depending on whether $z < \frac{s+e}{2}$ or not. We penalize the ℓ_1 error between the prediction y and the known DRDF, or $\mathcal{L}_{\mathcal{II}}(y) = |y - l_s|$ if $z < \frac{s+e}{2}$ and $|y - l_e|$ otherwise. This penalty is is zero only when y is equal to the known DRDF.

$\mathcal{L}_{\mathcal{OO}}$**:** $\mathcal{OO}$ **segment.** Given a segment bounded by two occlusion events, the exact DRDF is not known, but the visible free space rules out potential values. Since $|d_{\text{DR}}(z)|$ is the distance to the nearest intersection, $d_{\text{DR}}(z)$ cannot lie in $[l_s, l_e]$ since such a value would imply an intersection in free space between s and e. We penalize incursions into $[l_s, l_e]$ with a ℓ_1 penalty: if we denote halfway between l_s and l_e as h, then this can be done as $\mathcal{L}_{\mathcal{OO}}(y) = \max(0, l_e - h - |y - h|)$. The resulting penalty is zero if $y \leq l_s$ or if $y \geq l_e$.

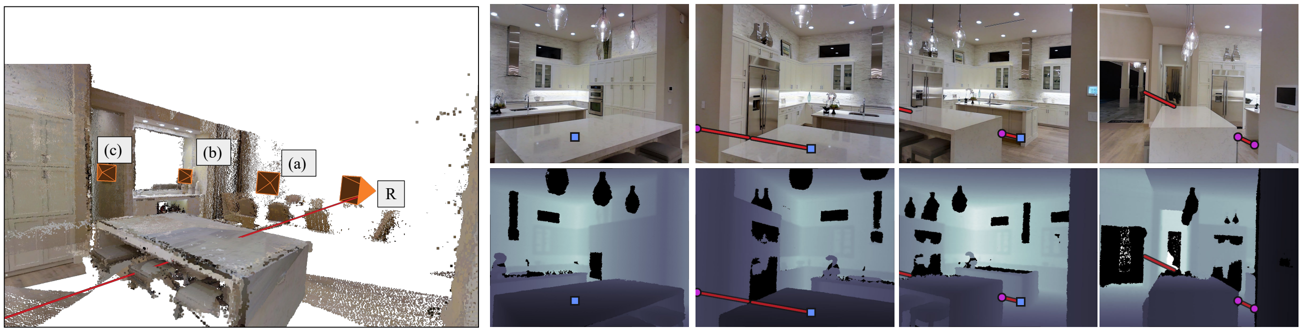
$\mathcal{L}_{\mathcal{IO}}$**:** $\mathcal{IO}$ **segment.** When the segment is bounded by an intersection event followed by an occlusion event, the situation is more complex and we define $\mathcal{L}_{\mathcal{IO}}$ piecewise. In the first half of the segment($z < \frac{s+e}{2}$), d_{DR} is exactly known, and so we can use an ℓ_1 penalty like the $\mathcal{II}$ case, so $\mathcal{L}_{\mathcal{IO}}(y) = |y - l_s|$. In the second half, there are two options. If the nearest intersection is s, then $d_{\text{DR}}(z) = l_s$. Otherwise, the nearest intersection is unknown but after e and so $d_{\text{DR}}(z) > l_e$ must hold (since $d_{\text{DR}}(z) \leq l_e$ would imply the existence of an intersection before e). We take the minimum of errors for the two cases: ℓ_1 distance to l_s and a ℓ_1 penalty function $\max(0, l_e - y)$, resulting in $\mathcal{L}_{\mathcal{IO}}(y) = \min(\max(0, l_e - y), |y - l_s|)$. This part of the penalty is zero if either $y = l_s$ or if $y > l_e$.

$\mathcal{L}_{\mathcal{OI}}$**:** $\mathcal{OI}$ **segment.** The $\mathcal{L}_{\mathcal{OI}}$ loss is defined symmetrically to $\mathcal{L}_{\mathcal{IO}}$, simply by exchanging the role of s and e. In addition to occluded regions, this loss occurs in the reference view up to the depthmap, where a disocclusion into the reference camera's view is followed by an intersection.

To assist the network, we add two auxiliary losses that are true statistically: $\mathcal{L}_{\text{sep}}$ represents a prior that surfaces tend to be separated by distances and $\mathcal{L}_{\text{ent}}$ captures a property of the DRDF that is true in the limit if our observations are randomly chosen.

$\mathcal{L}_{\text{sep}}$**: Minimum Separation Loss.** Since the cameras never sees the insides of objects, there is no incentive to predict inside objects. This prevents the generation of zero crossings, e.g., after the first intersection. To assist the network, we add a loss $\mathcal{L}_{\text{sep}}$ that assumes that surfaces are separated by a minimum distance unless there is evidence otherwise. We continue the DRDF's known value for $t = 0.2m$ before and after each intersection event to make a continued value c. We then penalize $\mathcal{L}_{\text{sep}}(y) = |y - c|$, so long as there is no conflicting free space evidence from another view.

$\mathcal{L}_{\text{ent}}$**: Sign Entropy Loss.** The occlusion-based constraints



Scene Point Cloud, Cameras, and Ray | Reference (R) | View (a) | View (b) | View (c)

Figure 5. A red ray originates from the reference camera and intersects the table before entering the island. This ray is seen by three other *auxiliary* views (a, b, c). For each view we show just one of the many segments in that view for readability, showing intersections as blue ■s and occlusions as purple ●s. The segments in views (a) and (b) are of the form $\mathcal{O}\mathcal{I}$, where the $\mathcal{O}$ comes from the ray entering the frustum in (a) and disoccluding in (b). In view (c), we show an $\mathcal{O}\mathcal{O}$ segment. We discuss the penalties for these segments in §4.1

can be satisfied by making y positive or negative. In theory, after the first intersection, d_{DR} is positive half of the time. Learning to produce the signs uniformly with gradient descent is difficult due to the large loss between the acceptable solutions. To encourage a uniform distribution of signs, we would like to maximize the entropy of the distribution of the signs of the predictions (achieved when the distribution is uniform). Since sign is non-differentiable, we optimize a differentiable surrogate. Suppose Y is the set of predictions at points that are occluded in the reference view and H is the binary entropy $H(p) = p\log(p) + (1-p)\log(1-p)$. Then our loss is $\mathcal{L}_{\text{ent}} = H(\sum_{y\in Y}\sigma(y/\tau)/|Y|)$ where σ functions like a soft-sign. As seen in the supplement, $\mathcal{L}_{\text{ent}}$ can be minimized by distributing Y symmetric about 0. Using $\mathcal{L}_{\text{ent}}$ improves the prediction and Scene F1 by 2 points.

4.2. Implementation Details

View Selection. For every candidate auxiliary view we compute the fraction of visible points in this view that are occluded in the reference view. Auxiliary views with large number of such points provide supervision for key occluded regions in the reference view. We sample up to 20 auxiliary views per reference image from this set of views.

Sampling Strategy. Given a set of fixed auxiliary views and a reference view, we sample over 200 rays per input image with 512 points per ray. We re-balance this set by sampling $20K$ points that are visible and $20K$ points that are occluded in reference view. We rebalance as most points are from the region between the camera and the first hit.

Combining segments from different views. We merge information from multiple posed RGBD images to produce a concise merged set of non-overlapping segments along the ray. This prevents double-counting losses (e.g., if a region of the ray is seen by multiple auxiliary views). We safely merge segments that provide the same information: e.g., if one depthmap provides an $\mathcal{O}\mathcal{O}$ segment that is contained within another depthmap's $\mathcal{I}\mathcal{I}$ segment, then the $\mathcal{O}\mathcal{O}$ segment can be safely dropped since $\mathcal{L}_{\mathcal{O}\mathcal{O}}(y) \leq \mathcal{L}_{\mathcal{I}\mathcal{I}}(y)$. When segments disagree (e.g., due to inaccurate poses), we keep the segment with more auxiliary views in agreement. This approach handles merging $\mathcal{L}_{\text{sep}}$: $\mathcal{L}_{\text{sep}}$ is seen by no auxiliary views, so any visible freespace overrules it.

Network Architecture. We follow [28] to facilitate fair comparison. Additionally, we clamp the outputs of our network to be $\in [-1, 1]$ by applying a $\tanh$ activation and adjust the loss to account for this clipping.

Training. We follow a two-stage training procedure. We first train with only the reference view followed by adding auxiliary views. In the first stage, we train for 100 epochs minimizing $\mathcal{L}_{\mathcal{O}\mathcal{I}}$ and $\mathcal{L}_{\text{sep}}$. We then train for 100 epochs with auxiliary losses, minimizing a sum of the segment losses $\mathcal{L}_{\mathcal{I}\mathcal{I}}$, $\mathcal{L}_{\mathcal{I}\mathcal{O}}$, $\mathcal{L}_{\mathcal{O}\mathcal{I}}$, $\mathcal{L}_{\mathcal{O}\mathcal{O}}$, and $\mathcal{L}_{\text{sep}}$ as well as $\lambda\mathcal{L}_{\text{ent}}$ with λ set to 0.1 to balance loss scales. We minimize the loss with AdamW [27, 35] as the optimizer with learning rate warmup for 0.5% of the iterations followed by cosine learning rate decay with maximum value 3×10^{-4}. Our models are implemented using PyTorch [42] and visualizations in this paper use PyTorch3D [31, 47].

5. Experiments

We evaluate our method to address three experimental questions. First, we examine how training with RGBD data compares with mesh supervision. Next, we test how RGBD and mesh supervision compare when one has less complete scans. Finally, we show that our method can quickly adapt to multiple posed RGBD inputs.

Metrics. Throughout, we follow [28] and use two metrics that evaluate predictions against a ground-truth mesh. Following [50, 58], these metrics are based on: Accuracy/Acc (the fraction of predicted points that are within t to a ground-truth point), Completeness/Cmp (the fraction of ground-truth points that are within t to a predicted point), as well as F1 (the harmonic mean of Accuracy and Completeness). t for both Acc and Cmp is 0.5m. In

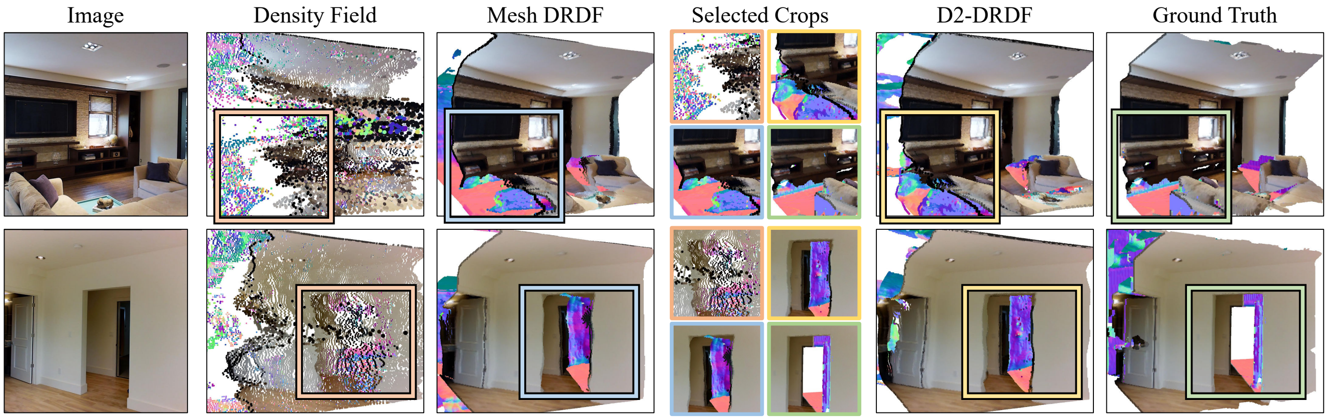


Figure 6. Comparison with baselines. 3D outputs generated by all methods trained on Matterport3D. We color the first intersection with image colors and occluded intersections with computed surface normals. We highlight regions of interest in the reconstructions in selected crops. D2-DRDF achieves results on par with Mesh DRDF while the density fields baselines fails to model the occluded parts faithfully. In row 1, our method recovers the back of sofa, and a hidden room behind the hallway in row 2. Surface Normal Map

(Scene Acc/Cmp/F1), we evaluate the predicted mesh of the full scene against the ground-truth, using 10K samples per mesh. In **(Ray Occ. Acc/Cmp/F1)**, We evaluate the performance on occluded points, evaluating per-ray and then averaging. We define occluded points for both the ground truth and prediction as any surface past the first intersection. Ray Occ. is a challenging metric as mistakes in one ray cannot be accounted for in another ray.

Datasets. We use Matterport3D [1] as our primary dataset following an identical setup to [29]. We choose Matterport3D because it has substantial occluded geometry (unlike ScanNet [8]) and was captured by a real scanner (unlike 3DFront [15], which is synthetic). We note that we use the *raw images* captured by the scanner rather than re-renders. We follow the split from [28], which splits train/val/test by house into 60/15/15 houses. After filtering and selecting images, there are 15K/1K/1K input images for each split.

5.1. Mesh Prediction Results

Baselines. Our primary point of comparison is **(1) Mesh DRDF [28]**, which learns to predict the DRDF from direct mesh supervision. For context, we also report the baselines from [28] and summarize them: **(2) LDI [51]** predicts a set of 4 layered depthmaps, where the first represents the depth and the next three represent occluded intersections; **(3) UDF [5]** predicts an unsigned scene distance function; **(4) URDF [5]** predicts an unsigned **ray** distance function; **(5) ORF** predicts an occupancy function, or whether the surface is within a fixed distance. We note that for each of these approaches, there are a number of variants (e.g., of finding intersections in a URDF along a ray). We report the highest performance reported by [28], who document extensive and detailed tuning of these baselines.

Our final baseline, **(6) Density Field [62]** tests the value of predicting a DRDF compared to a density. Like our method, Density Field only uses posed RGBD data for supervision and does not depend on mesh supervision. We adapt pixelNerf [62] to our setting, modifying the implementation from [45] to permit training from a single reference view as input and multiple auxiliary views for su-

Figure 7. Novels Views Comparison between D2-DRDF and Ground Truth(GT) from novel views. Rows 1, 2 are from unseen images on Matterport3D [1] and 3,4 from Omnidata [11]. Our method trained with *only* RGBD data recovers occluded empty floors, kitchen cabinets (row 1) and sides of kitchen island(row 3).

Table 1. Matterport3D [1] Acc/Comp/F1Score. We separate methods that use ground-truth mesh from ones using posed RGBD by a horizontal line. D2-DRDF is comparable to the best Mesh supervised method DRDF, and is better than all other mesh based methods on Scene and Ray F1 scores.

Method		Scene Acc	Scene Cmp	Scene F1	Ray Acc	Ray Cmp	Ray F1
Mesh	LDI [51]	66.2	72.4	67.4	13.9	42.8	19.3
	UDF [5]	58.7	76.0	64.7	15.5	23.0	16.6
	ORF	73.4	69.4	69.6	26.2	20.5	21.6
	URDF [5]	74.5	67.1	68.7	24.9	20.6	20.7
	DRDF [28]	75.4	72.0	71.9	28.4	30.0	27.3
D2-DRDF		73.7	73.5	72.1	28.2	22.6	25.1
Density Field [62]		45.8	80.2	57.5	24.8	14.0	17.9

pervision. In addition to the standard color loss, we supervised the network to match the ground-truth auxiliary depth RGBD depth as in [10]. At inference time, we integrate along the ray to decode a set of intersections as in [10].

Qualitative Results. We show qualitative results from our method and the baselines in Fig 6. Our methods outputs are comparable to the outputs of the mesh based DRDF that is trained with much stronger supervision. The density fields approach struggles to model the occluded hits and has a lots of floating blobs in the empty space. We show additional results from D2-DRDF in Fig. 7 where it recovers occlude kitchen cabinets, empty floor space behind kitchen island, sides of kitchen island, and hollow bed.

Quantitative Results. We report results in Table 1. Without access to ground-truth meshes, our approach slightly *outperforms* Mesh-based DRDF on Scene F1 and approaches its performance on Ray Occ. Our approach matches it in accuracy (i.e., the fraction of predicted points that are correct) but does worse on completeness. We hypothesize that this is due to mesh-based techniques obtaining supervision in adjacent rooms. Nonetheless, our approach outperforms all other baselines besides mesh-supervised DRDF by a large margin (25.1 vs 21.6 F1). D2-DRDF has substantially higher performance compared to the Density Field baseline. This shows that density fields representation has difficulties in learning from a supervision of less views with large occlusions.

5.2. Training on Incomplete Data

One advantage to using posed RGBD data compared to meshes is that it opens the door to learning from more data that is of lower quality. One can use lots of lower-quality scene captures that would produce poor meshes due to substantial amounts of incomplete data. In the case of direct mesh supervision methods, having an incomplete mesh may lead to incorrect signals for a mesh-based system: for instance, the network may learn to predict that an intersection *does not* exist simply because it was not scanned. In contrast, for D2-DRDF, missing data simply increases the fraction of points without supervision. We now test this hypothesis by reducing the number of views in datasets.

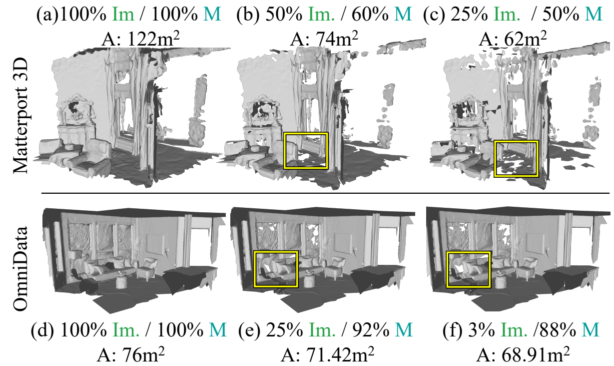


Figure 8. Mesh Degradation We show examples of how drop in images creates holes, reduces mesh area available for supervision. In (c), after retaining $\approx\frac{1}{4}$ of image data, we only lose $\approx\frac{1}{2}$ the mesh area. Im.: Image Coverage, M: Mesh Coverage, A: Area

Optimistic Degradation Setup (ODS). We simulate the degradation of dataset collection by subsampling views. To avoid conflating errors in training with suboptimal meshing with incomplete data, we *optimistically* degrade the meshes to provide an upper bound on supervision. We assume that the mesh with fewer views is identical to the mesh from all views, minus triangles with no vertices in any view. We show ODS mesh examples in Fig. 8.

We degrade meshes by selecting $1/2^i$ views per dataset for an increasing i. While reducing the views linearly impacts the sample count for RGBD training setup, it has a non-linear impact on mesh completeness since a triangle is removed only if *all* of the views seeing it are removed (which is unlikely until most views are removed). For any given image retention (Im.) %, mesh coverage (M) % degrades less giving an edge to mesh based methods.

Usually, when dealing with limited data, we use a method called *Screened Poisson Reconstruction*(SPR) [26]. However, SPR does not perform well when there is not enough data available. To avoid conflating errors caused by poor quality inadequate meshing, we establish an upper limit on the performance of methods that rely on direct supervision. Our ODS strategy is much better than using SPR, but it cannot be used in real-world scenarios. We employ Open3D's[65]'s SPR with hyper-parameters similar to [1] to reconstruct meshes.

Datasets. We evaluate on Matterport3D [1] and apply our method as is without any modifications on OmniData [11] which has a substantially different image view distribution, more rooms and more floors compared to Matterport3D.

Quantitative Results. We compare models trained on different amount of data available for supervision by using metrics defined in §5.1. In Tab. 2 we compare against the

Table 2. Robustness to Sparse Data Performance on partial data on Matterport3D[1]. We compare (**SPR and ODS**) trained DRDF [28] which uses mesh supervision and (**Depth**) Depth-based DRDF (ours), which uses posed RGBD supervision. In each row, we degrade the training data and report test performance of the trained model. At 100% data there is no M degradation for ODS or SPR. Our approach is more robust to drop in Im.: at 50% view sparsity (Im.), models using ODS or SPR suffer substantial performance drops. Scene F1 drop by 16.3 for SPR; 3.5 for ODS; 2.1 for Depth (ours).

	ODS	Scene F1			Ray Occ F1		
Im. %	M %	SPR	ODS	Depth	SPR	ODS	Depth
100	100	71.9	71.9	72.1	27.3	27.3	25.1
50	56	55.6	68.4	70.0	21.4	23.6	24.4
25	43	56.8	66.8	70.0	21.5	21.2	24.9

Table 3. Robustness to Sparse Data Performance of ODS (mesh) and Depth (ours) based DRDF on partial Omnidata [11] following the same setup as Table 2. RGBD-based training is substantially more robust to partial data.

	ODS	Scene F1			Ray Occ F1		
Im. %	M %	SPR	ODS	Depth	SPR	ODS	Depth
25	86	63.9	77.2	72.8	26.2	40.3	32.1
12.5	83	62.8	75.3	70.9	26.1	37.1.	29.3
6.3	78	40.9	73.4	71.8	5.7	32.6	28.1
3	69	42.9	69.8	70.4	3.8	20.3	26.7

Mesh-DRDF trained ODS & SPR meshes. For any given view sampling level, the supervised M % area is high, resulting in stronger supervision for methods trained with ODS than RGBD. However, on Matterport3D, our method outperforms DRDF on all metrics at 25% Im. and outperforms all other baseline methods in Scene F1 at 100% Im..

In Tab. 3 we show robustness trends on OmniData. At 25% Im./ 86% M completion, mesh-based does better. We hypothesize this gain is due to better handling of estimates beyond the room. However, as Im. reduces mesh degrades, resulting steep fall for ODS DRDF: with 3% Im./ 69% M, MeshDRDF's Ray Occ F1 drops by 20 points; ours is reduced by *just* 5.4. Moreover for SPR DRDF, at low Im. values, meshing performs dismally and the poor meshing performance translates into poor reconstruction performance: at 6% Im., training DRDF on SPR meshes produces a Ray F1 of just 5.7% and a scene F1 of 40.9%.

5.3. Adapting With Multiple Inputs

Since D2-DRDF can *directly* train on posed RGBD images, this enables test-time adaptation given a few auxiliary posed RGBD images. We start with the pre-trained model from §5.1 and then fine-tune for 500 iterations.

Dataset and Metrics. We generate 300 quadruplets of scenes consisting of a reference view as well as *three* auxiliary RGBD images with poses. These three auxiliary views

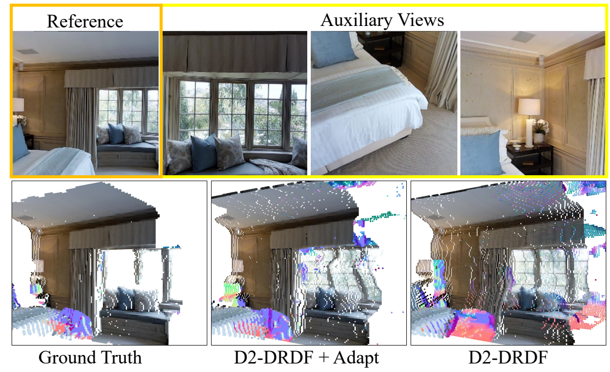


Figure 9. Adaptation to a few images We optimize our pre-trained model on the three auxiliary views (yellow) and a reference view (orange). The optimization lets the model fix reconstructions in occluded regions (*e.g.* bedside wall) as seen D2-DRDF + Adapt.

Table 4. Scene, Ray F1 Score for Adaptation Using D2-DRDF for adaptation produces better reconstructions that outperform the baselines by a large margin on Ray Occ F1 scores (by 8.9 points).

	Scene			Ray		
Method	Acc	Cmp	F1	Acc	Cmp	F1
D2-DRDF (Full)	79.2	76.4	76.0	36.2	33.6	34.9
D2-DRDF (Depth)	78.4	70.5	72.5	33.0	21.4	26.0
D2-DRDF (Scratch)	66.4	70.3	66.0	23.7	27.4	25.4
Density Field [62]	77.3	74.9	74.6	12.8	9.9	11.2

are randomly sampled from views that overlap with occluded parts of the reference view (see supp.). We evaluate inferred 3D using the metrics as §5.1.

Baselines. The baselines from §5.1 cannot operate in these settings, since they require meshes for training. Therefore we compare against a number of depth-map-based methods as well as ablations to give context to our results: **(1) Density Field** fine-tunes the density field baseline model from §5.1; **(2) Depth-Pretrained** fine-tunes D2-DRDF starting with the model from the first stage of training; **(3) Scratch Training** fine-tunes D2-DRDF from scratch.

Results. Our loss and penalty formulations lend themselves well to test-time adaptation as shown in Fig. 9. After adaptation, D2-DRDF can resolve uncertainties like the corner of the bed and wall. Table 4 shows D2-DRDF does the best.

6. Conclusion

We presented a method for learning to predict 3D from a single image using implicit functions while requiring only posed RGBD supervision. We believe our method can unlock new avenues with posed RGBD data becoming available from both consumers as well as robotic agents.

Acknowledgments. We thank our colleagues for the wonderful discussions on the project (in alphabetical order). Richard Higgins, Sarah Jabour, Dandan Shan, Karan Desai, Mohammed Banani, and Chris Rockwell for feedback. Shengyi Qian for the help with ViewSeg code used to implement the PixelNeRF baseline.

Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any Toyota entity.

References

[1] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *3DV*, 2017. 2, 6, 7, 8, 14, 15

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2

[3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *ECCV*, 2022. 2

[4] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. *NeurIPS*, 2016. 2

[5] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In *NeurIPS*, 2020. 2, 6, 7

[6] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016. 1, 2

[7] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312, 1996. 2, 3

[8] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 2, 6

[9] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)*, 36(4):1, 2017. 2

[10] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *CVPR*, 2022. 7

[11] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A scalable pipeline for making multi-task mid-level vision datasets from 3d scans. In *ICCV*, 2021. 6, 7, 8, 15, 19, 20

[12] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *ICCV*, 2015. 2

[13] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2017. 2

[14] David F Fouhey, Abhinav Gupta, and Martial Hebert. Data-driven 3d primitives for single image understanding. In *ICCV*, 2013. 2

[15] Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Cao Li, Qixun Zeng, Chengyue Sun, Yiyun Fei, Yu Zheng, Ying Li, Yi Liu, Peng Liu, Lin Ma, Le Weng, Xiaohang Hu, Xin Ma, Qian Qian, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 3d-front: 3d furnished rooms with layouts and semantics. *ICCV*, 2021. 6

[16] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *ICCV*, 2021. 2

[17] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*. Springer, 2016. 1, 2

[18] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *ICCV*, 2019. 2

[19] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *CVPR*, 2018. 1, 2

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 12

[21] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. In *CVPR*, 2018. 2

[22] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011. 2

[23] Hamid Izadinia, Qi Shan, and Steven M Seitz. Im2cad. In *CVPR*, 2017. 2

[24] Michael Janner, Jiajun Wu, Tejas D Kulkarni, Ilker Yildirim, and Josh Tenenbaum. Self-supervised intrinsic image decomposition. *NeurIPS*, 2017. 2

[25] Ziyu Jiang, Buyu Liu, Samuel Schulter, Zhangyang Wang, and Manmohan Chandraker. Peek-a-boo: Occlusion reasoning in indoor scenes with plane representations. In *CVPR*, 2020. 2

[26] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006. 2, 7, 14

[27] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 5

[28] Nilesh Kulkarni, Justin Johnson, and David F. Fouhey. Directed ray distance function for 3d scene reconstruction. In *ECCV*, 2022. 1, 2, 3, 5, 6, 7, 8, 12

[29] Nilesh Kulkarni, Ishan Misra, Shubham Tulsiani, and Abhinav Gupta. 3d-relnet: Joint object and relational network for 3d prediction. In *ICCV*, 2019. 2, 6

[30] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. *IJCV*, 2000. 2

[31] Christoph Lassner and Michael Zollhöfer. Pulsar: Efficient sphere-based neural rendering. *arXiv:2004.07484*, 2020. 5

[32] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018. 12

[33] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *AAAI*, volume 32, 2018. 2

[34] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. Planercnn: 3d plane detection and reconstruction from a single image. In *CVPR*, 2019. 2

[35] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019. 5

[36] Ravi Malladi, James A Sethian, and Baba C Vemuri. Shape modeling with front propagation: A level set approach. *TPAMI*, 1995. 2

[37] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 1, 2

[38] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020. 2

[39] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging {AI} applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 561–577, 2018. 12

[40] Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image. In *CVPR*, 2020. 2

[41] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 1, 2

[42] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019. 5

[43] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019. 12

[44] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020. 2

[45] Shengyi Qian, Alexander Kirillov, Nikhila Ravi, Devendra Singh Chaplot, Justin Johnson, David F Fouhey, and Georgia Gkioxari. Recognizing scenes from novel viewpoints. *arXiv preprint arXiv:2112.01520*, 2021. 6

[46] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 12

[47] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 5

[48] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *ICCV*, 2021. 2

[49] Shunsuke Saito, Zeng Huang, Ryota Natsume, Shigeo Morishima, Angjoo Kanazawa, and Hao Li. Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In *ICCV*, 2019. 1, 2

[50] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, 2006. 5

[51] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242, 1998. 6, 7

[52] Jian Shi, Yue Dong, Hao Su, and Stella X Yu. Learning non-lambertian object intrinsics across shapenet categories. In *CVPR*, 2017. 2

[53] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *NeurIPS*, 2020. 2

[54] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015. 2

[55] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. *CVPR*, 2017. 2

[56] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *CVPR*, 2021. 2

[57] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *CVPR*, 2018. 2

[58] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *CVPR*, 2019. 5

[59] Kaixuan Wang and Shaojie Shen. Mvdepthnet: Real-time multiview depth estimation neural network. In *3DV*, 2018. 2

[60] X. Wang, David F. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *CVPR*, 2015. 2

[61] Jianglong Ye, Yuntao Chen, Naiyan Wang, and Xiaolong Wang. Gifs: Neural implicit function for general shape representation. In *CVPR*, 2022. 2

[62] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021. 2, 3, 6, 7, 8

[63] Zehao Yu, Jia Zheng, Dongze Lian, Zihan Zhou, and Shenghua Gao. Single-image piece-wise planar 3d reconstruction via associative embedding. In *CVPR*, 2019. 2, 12

[64] Cheng Zhang, Zhaopeng Cui, Yinda Zhang, Bing Zeng, Marc Pollefeys, and Shuaicheng Liu. Holistic 3d scene understanding from a single image with implicit representation. In *CVPR*, 2021. 2

[65] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*,

2018. 7, 14, 15

[66] Shizhan Zhu, Sayna Ebrahimi, Angjoo Kanazawa, and Trevor Darrell. Differentiable gradient sampling for learning implicit 3d scene reconstructions from a single image. In *ICLE*, 2022. 1, 2

A. Overview

We discuss crucial details required to implement the results from our paper in §B. Then we discuss the under constrained nature of our segment penalties and how multiple DRDFs satisfy them in §C. In §D we discuss the details of the entropy like loss function followed by additional evaluations and qualitative results in §E in §F respectively.

B. Implementation Details

We provide additional implementation details for replicating the results in this paper. We will the release code.

B.1. Data Pre-processing

We select up to 20 auxiliary views for every reference view from the complete dataset. These auxiliary views are preprocessed along with reference view to create a cached dataset of ray signature to allow faster training. We get supervision for occluded segments of the ray if there is an occluded intersection or if the part of occluded segment is observed from an auxiliary view. Therefore, it is important to sample points from the auxiliary depth map and then convert these points to full rays in the reference view. Such a strategy guarantees that we create rays with more than one OI segment (the first hit) to train our models. After pre-computation using this strategy we have access to a large repository of rays originating from every reference image.

Detecting Events & Noise in Depth data. The process of detecting events and creating segments along the ray needs to be robust so as to not allow bad segments. It is critical that we do not allow erroneous intersections to jump into the ray signature due to missing data. Rays originating from the camera are terminated at $8m$ from the camera as this is maximum extent of the scene we reconstruct (for fair comparison to baselines [28]). We linearly sample 512 points along this ray and perform event detection for these points.

Given an auxiliary view (π) and the a ray, $\vec{\mathbf{r}}$ for every point, $\mathbf{x}$, we compute the z coordinate in the view frame of the auxiliary view. We also record the depth map value at the projection $\pi(\mathbf{x})$. We sweep along the ray and detect sign changes for difference between the z-coordinate and the recorded depth. A smooth sign change from $+ve$ to $-ve$ or vice versa indicates that there is an intersection at the sign change that is observed from this auxiliary view. However, a sign change with a discontinuity leads to an occlusion event implying the ray entering or exiting the auxiliary view. Since depth data is noisy, for a cluster of missing depth values we label the start as an exit occlusion event and the end as an entry occlusion event (if the ray becomes visible).

Conflicting segments. Detecting events using multiple auxiliary views leads a large set of segments along a particular ray. In order to get an unified segment we drop redundant information *e.g.* we drop an $\mathcal{OO}$ that is subsumed by an $\mathcal{II}$. Our event detection system is conservative in detecting segments and in case of conflicts we only keep the ray segment that has evidence from most auxiliary views. Since we are training on a large dataset of scene it is convenient to only keep segments that are accurate given the evidence from auxiliary views.

B.2. Training

We train our networks in two stages. During the first stage we only train the network to predict the DRDF values until the first intersection. This first stage allows the second stage of the learning process to learn about the occluded scene geometry. Such strategy also lends itself well to use networks bootstrapped on large collection of paired RGB and Depth data such as [32, 46]. We train for half the number of epochs in the first stage and then train on the whole scene including occluded points for the rest half of the epochs (100 + 100).

Architecture. Our architecture is similar to pixelNerf [28, 63]. We use a pre-trained version of ResNet-34 [20] from PyTorch [43], and a 5-Layer MLP with ResNet style skip connections. Our final activation for the last layer is $\tanh$ and it bounds the predicted values in range $[-1, 1]$. The MLP takes input the positional embedding (36 dimensions) and pyramid of image features at different spatial resolutions (512 dimensions). Each of the 5 hidden layer of our MLP has 1024 hidden units and the final layer predicts a single scalar value that is the directed ray distance.

Sampling Strategy. In a given reference view with access to only a sparse set of few auxiliary views we do not observe large sections of the occluded geometry. For any given ray in the reference view the predominant segments are the one that are visible in the reference view that capture the first hit. We address this bias in the type of segments by re-balancing the points sampled on all the segments. We randomly sample up to 400 rays from our preprocessed dataset. For every ray we create linearly spaced 512 points up to $8m$. We then re-sample points from this large collection of 400×512 points to keep only 50% points that are visible in the reference view (*i.e.* before the first hit) and 50% points that are occluded from the reference view.

B.3. Inference

At inference we consider the input image and a predefined grid in the view frame of size $H \times W \times D$. This grid has $H \times W$ pixel aligned rays. For each ray we decode the ray to a set of intersections along the ray. We speed up parallel decoding for all the rays with the Ray Library [39]. We use $H = W = D = 128$.

C. D2-DRDF Penalty Functions

The penalty functions discussed in §4 provide a sparse set of supervisions for our network and now we demonstrate that for a particular ray in a simplified setting. Consider a ray with following segments $\mathcal{OI}$, $\mathcal{OO}$, $\mathcal{II}$ in order. The $\mathcal{O}$ events provide a weaker constraints as compared to the $\mathcal{I}$. Throughout whenever there is an $\mathcal{I}$ it leads to equality constraints in points on the segment closer to the event, while an $\mathcal{O}$ results in an inequality constraint. In Fig 10 (top-left) we show the complete penalty function combined across various segments for points along the ray. We show that $\mathcal{I}$ events lead to a singular solution (shown as a single white line) while $\mathcal{O}$ events lead to multiple regions of zero penalty (white regions). In Fig 10 we also show multiple possible DRDFs that satisfy the penalty function but are not the ones we want. It is key to see that all the DRDF (1-5)are exactly the same for points on the ray (z) that are closest to the $\mathcal{I}$ while vary largely for points close to the $\mathcal{O}$.

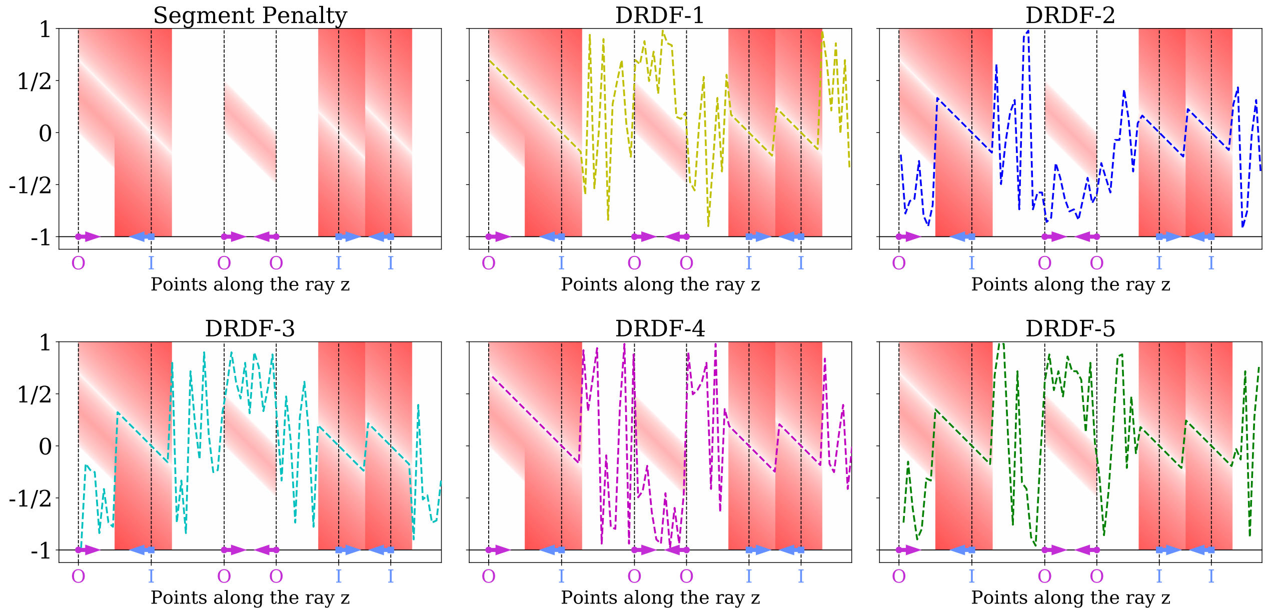


Figure 10. Under constrained penalty segments . We consider a ray that has three segments $\mathcal{OI}$, $\mathcal{OO}$, $\mathcal{II}$ and goes from *left* to *right*. On the **first** plot we show penalty segments for possible DRDF values $\in [-1, 1]$ on the Y-axis vs points along the ray z. The regions in white have zero penalty and red regions have a high value. . For any particular z these segments give us partial information on the possible values of DRDF for certain parts of the ray. We now show 5 different possible DRDF functions that all satisfy the penalty plot on top-right. Since some of our segment penalties have inequality constraints there are multiple values possible ($\mathcal{OO}$). All DRDFs (1-5) match exactly in regions close to an intersection where we have equality constraints (*e.g.* $\mathcal{II}$ segment). For regions along the ray not bound by segments DRDF is unconstrained. Penalty legend 0 1.7

However when we train our network our model observers lot of sample of rays from varied different rays, this access to large scale data encourages a singular DRDF that explains all the events while also being simple when dealing with inequality constraint (occam's razor). Since neural networks are continuous functions they discourage predictions with high frequency. The key ideas that fuel our approach is not events on a single ray but using data across multiple rays and scenes. Since we use a single neural network to fit to our train data it allows us to learn a bigger set of priors which are useful to create a final DRDF and move away from all possible solutions to these given constraints. Now, in §D we discuss another regularizer we use to encourage our network to predict sometimes positive DRDF values and sometimes negative DRDF values for occluded segment.

D. Entropy-Like Loss

In our loss function we use an entropy loss ($\mathcal{L}_{\text{ent}}$) to encourage the predicted DRDFs from segment penalties to behave like DRDFs learnt with mesh supervision. One of the key properties of DRDF is the number of samples that have a negative DRDF value is equal to number of samples that have a positive value. This statistic is only true when we are looking at a dataset of rays. Moreover, since the $\mathcal{OO}$ imposes an penalty function that discourages changing sign hence reduces diversity, our entropy like loss allows the network to easily adapt to this and jump across large loss values. Below we show some analysis on the behavior of the entropy loss.

Given a prediction $\mathbf{y} \in \mathbb{R}^n$ over n values, we optimize a differentiable surrogate for the binary entropy of the *signs* of the prediction. This objective aims to ensure that the predictions are equally negative and positive. Both our ideal objective and surrogate objective can be minimized in the limit as $n \to \infty$ by sampling $\mathbf{y}$ from a symmetric distribution centered on zero (e.g., a uniform one over $[-1, 1]$).

Ideally, we would like to minimize the negative of the entropy of the signs, or

$$p \log(p) + (1-p) \log(1-p) \;\text{ with }\; p = \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{y}_i), \tag{1}$$

where $H(\cdot) : \mathbb{R} \to \{0, 1\}$ is the Heaviside function mapping a number to its sign. Equation 1 has a unique minimum in p, namely $\frac{1}{2}$, which is achieved when exactly half of the components of $\mathbf{y}$ are positive and half are negative. The requirement of equal positives and negatives can be satisfied in a large variety of ways.

The Heaviside function is, of course, not differentiable, and so we use a differentiable surrogate and minimize

$$p \log(p) + (1-p) \log(1-p) \;\text{ with }\; p = \frac{1}{N} \sum_{i=1}^{N} \sigma(\mathbf{y}_i), \tag{2}$$

where $\sigma(\cdot) : \mathbb{R} \to [0, 1]$ is a sigmoid function with a temperature. The sigmoid functions like a soft sign function where 0 corresponds to negative values and 1 to positive values. Just like the binary one, Equation 2 has a single global minimum in p at $\frac{1}{2}$ and a family of minimums in $\mathbf{y}$.

One minimum in $\mathbf{y}$ is created by generating symmetric values, where each component in $\mathbf{y}$ has a unique corresponding component with the same magnitude but flipped sign, e.g., if there is a 0.75m prediction, then there must also be a -0.75m prediction. More generally, suppose if n is even, and we order $\mathbf{y}$ such that $\mathbf{y}_i = -\mathbf{y}_{n/2+i}$ for all $1 \leq i < \frac{n}{2}$. Then $(\sigma(\mathbf{y}_i) + \sigma(\mathbf{y}_{n/2+i})) = 1$, and so $p = \frac{1}{N}\sum_{i=1}^{n/2}(\sigma(\mathbf{y}_i) + \sigma(\mathbf{y}_{n/2+i})) = \frac{1}{2}$. This setting would happen in the limit if the components of $\mathbf{y}$ were symmetrically distributed over an interval $[-a, a]$ for $a \in \mathbb{R}^+$.

Of course, the surrogate function we minimizes permits other solutions that balance out the right way. For instance, given on minimizer $\mathbf{y}$ one can generate another minimizer by adding a δ_1 in one component and adding an appropriate δ_2 in another (i.e., $\mathbf{y}'_i = \mathbf{y}_i + \delta_i$). This entails picking δ_1 and δ_2 such that

$$\sigma(\mathbf{y}_1 + \delta_1) - \sigma(\mathbf{y}_1) = -\left(\sigma(\mathbf{y}_2 + \delta_2) - \sigma(\mathbf{y}_2)\right), \quad (3)$$

or that sum remains unmodified. Given a chosen a chosen δ_1, some algebra reveals that

$$\delta_2 = \sigma^{-1}\left(\sigma(\mathbf{y}_1) - \sigma(\mathbf{y}_1 + \delta_1) + \sigma(\mathbf{y}_2)\right) - \mathbf{y}_2. \quad (4)$$

Thus, a whole family of minimizers that do not match pairs of samples or have balanced signs is possible. However, the entropy-like loss is not the only function minimized, and the network must also minimize the data term.

E. Quantitative Evaluations

In addition to F1 scores reported in the main paper we provide the complete results in Table 5, 7 for behavior of D2-DRDF under different levels of sparse data. Please refer to §5.2 of the main paper for additional details on evaluation, and dataset creation. With decreasing amount of data, the Mesh DRDF baseline suffers a significant drop in completion (cmp) score on both scene and ray based metrics. This leads to precipitous drop in performance for F1 score (-6.1 points) as compared to D2-DRDF which only see a drop of (-0.2 points). We observe similar trends in Table 7 on OmniData.

As described in the main paper, in practice in sparse view settings we have to leverage mesh reconstruction algorithms like SPR to generated meshes from posed RGBD data. Training methods with this data leads to a subpar performance w.r.t to ODS mesh data. For completeness, we provide the full evaluation in Tab. 6, 8 for methods trained with SPR mesh data.

Table 5. Matterport3D: Robustness to sparse data *vs* ODS DRDF. Scene Acc/Cmp/F1 and Ray Acc/Cmp/F1 scores on different amounts of Matterport3D dataset. ODS DRDF's Cmp scores drop precipitously result in loss of F1 score. D2-DRDF is more stable and robust to amount of data.

		Scene						Ray					
		ODS DRDF			D2-DRDF			ODS DRDF			D2-DRDF		
Im. %	M %	Acc	Cmp	F1	Acc	Cmp	F1	Acc	Cmp	F1	Acc	Cmp	F1
100	100	75.4	72	71.9	73.7	73.5	72.1	28.4	30.0	27.3	28.2	22.6	25.1
50	56	73.0	67.9	68.4	66.6	76.9	70.0	28.4	20.2	23.6	22.4	26.8	24.4
25	43	72.1	65.6	66.8	64.1	80.5	70.0	27.3	17.3	21.2	21.4	29.7	24.9

Table 6. Matterport3D: Robustness to sparse data *vs* SPR DRDF. Scene Acc/Cmp/F1 and Ray Acc/Cmp/F1 scores on different amounts of Matterport3D dataset. SPR DRDF's Cmp scores drop precipitously result in loss of F1 score. D2-DRDF is more stable and robust to amount of data.

	Scene						Ray					
	SPR DRDF			D2-DRDF			SPR DRDF			D2-DRDF		
Im. %	Acc	Cmp	F1	Acc	Cmp	F1	Acc	Cmp	F1	Acc	Cmp	F1
100	75.4	72	71.9	73.7	73.5	72.1	28.4	30.0	27.3	28.2	22.6	25.1
50	51.2	65.9	55.6	66.6	76.9	70.0	16.3	31.2	21.4	22.4	26.8	24.4
25	51.9	67.7	56.8	64.1	80.5	70.0	16.1	32.7	21.5	21.4	29.7	24.9

Table 7. Omnidata: Robustness to sparse data *vs.* ODS DRDF. Scene Acc/Cmp/F1 and Ray Acc/Cmp/F1 metrics on different amounts of Omnidata dataset. ODS DRDF's Cmp scores drop precipitously resulting in loss of Ray F1 scores. D2-DRDF is more stable and robust to amount of data.

		Scene						Ray					
		ODS DRDF			D2-DRDF			ODS DRDF			D2-DRDF		
Im. %	M %	Acc	Cmp	F1	Acc	Cmp	F1	Acc	Cmp	F1	Acc	Cmp	F1
25	86	82.9	74.1	77.2	69.2	79.1	72.8	43.9	37.3	40.3	29.3	35.6	32.1
12.5	83	81.4	72.3	75.3	65.9	79.2	70.9	41.8	33.4	37.1	26.3	33.0	29.3
6.3	78	80.5	69.5	73.4	68.8	77.5	71.8	40.0	27.5	32.6	27.4	28.8	28.1
3	69	80.5	63.7	69.8	63.7	81.1	70.4	37.1	14.0	20.3	24.4	29.5	26.7

Table 8. Omnidata: Robustness to sparse data *vs.* SPR DRDF. Scene Acc/Cmp/F1 and Ray Acc/Cmp/F1 metrics on different amounts of Omnidata dataset. SPR DRDF's scores are significantly worse as compared to D2-DRDF

	Scene						Ray					
	SPR DRDF			D2-DRDF			SPR DRDF			D2-DRDF		
Im. %	Acc	Cmp	F1	Acc	Cmp	F1	Acc	Cmp	F1	Acc	Cmp	F1
25	65.9	64.4	63.9	69.2	79.1	72.88	23.7	29.3	26.2	29.3	35.6	32.1
12.5	66.1	62.3	62.8	65.9	79.2	70.9	24.0	28.5	26.1	26.3	33.0	29.3
6.3	45.9	39.2	40.9	68.8	77.5	71.8	16.4	3.5	5.7	27.4	28.8	28.1
3	47.6	41.5	42.9	63.7	81.1	70.4	18	2.1	3.8	24.4	29.5	26.7

E.1. Mesh Degradation under sparse views

SPR from sparse views. All baseline models trained with mesh supervision from *Screened Poisson Reconstruction* used only the subset of views with depth maps. We use depth maps at 512×512 resolution and convert to posed point clouds. Our final point cloud for the scene is a combination of all the view point clouds. We estimate per point normals using Open3D's [65] nearest neighbour normal estimation. The estimated normals along with the unprojected point cloud is used as input to the SPR at an oct-tree depth of 9. This is the same standard used in reconstructing houses in Matterport3D[1].

ODS vs. SPR. Our optimistic degradation setup is an upper bound on the perfromance for methods that train with mesh supervision. In practice a fair comparison would require DRDF baseline trained with meshes reconstructed using *Screened Poisson Reconstruction*[26]. Meshes generated using SPR are of much lower quality when compared to ODS meshes. In Fig. 11 we see

SPR reconstrution compared the ODS on Matterport3D at only 25% image views. The recovered SPR meshes from Open3D's open source implementation [65] have a much lower quality as compared to meshes created by ODS. SPR meshes fail to keep the details of the scene, showcasing that in practice training methods that require mesh supervision is impractical when there are only a sparse set of posed RGBD views.

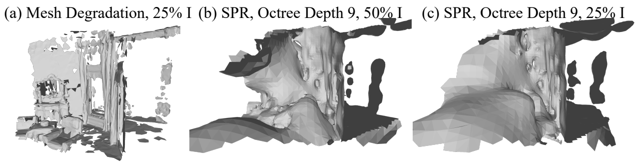


Figure 11. Screened Poisson Reconstruction (SPR) (a): degradation with ODS on 25% image data; (b), (c): SPR reconstructed mesh with 50% and 25% image data respectively. The meshes from SPR (b,c) have lots of reconstruction errors and miss on details in the scene whereas the ODS mesh (a) has holes but with reasonable geometry

Overall across both Matteport3D and OmniData we observe that methods trained with SPR mesh achieve a much lower scene F1 and ray F1 scores as compared to same approaches trained with ODS meshes.

F. Qualitative Results

We show additional qualitative results on Matterport and Omnidata.

Matterport [1] Novel Views. In Fig. 13, 14 we show qualitative outputs of D2-DRDF model trained on Matterport[1] dataset. We color the occluded reconstructed regions with a surface normal maps from Fig 12.

Matterport [1] Comparison to Baselines. In Fig. 15 we show qualitative comparison with baseline methods. The outputs of D2-DRDF model trained on Matterport[1] are comparable to DRDF model trained with Mesh supervision. The density field baseline trained with posed RGBD data fails at modeling the occluding geometry at test-time. We color the occluded reconstructed regions with a surface normal maps from Fig 12.

Omnidata [11] Comparison to Baselines. In Fig. 16, 17 we show qualitative outputs from D2-DRDF model trained on the OmniData. We color the occluded reconstructed regions with a surface normal maps from Fig 12.

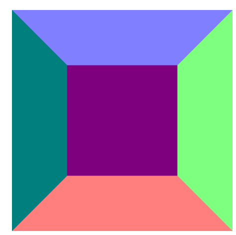


Figure 12. Surface Normal Legend We use this surface normal palette to color occluded points reconstructed by all the methods. The surface normals are computed in the camera frame of reference. In Fig. 13, 14, 15, 16, 17 we show reconstructed empty floors are colored in pink. The occluded side walls, kitchen cabinets, walls in rooms, other side of kitchen islands are colored in green or purple

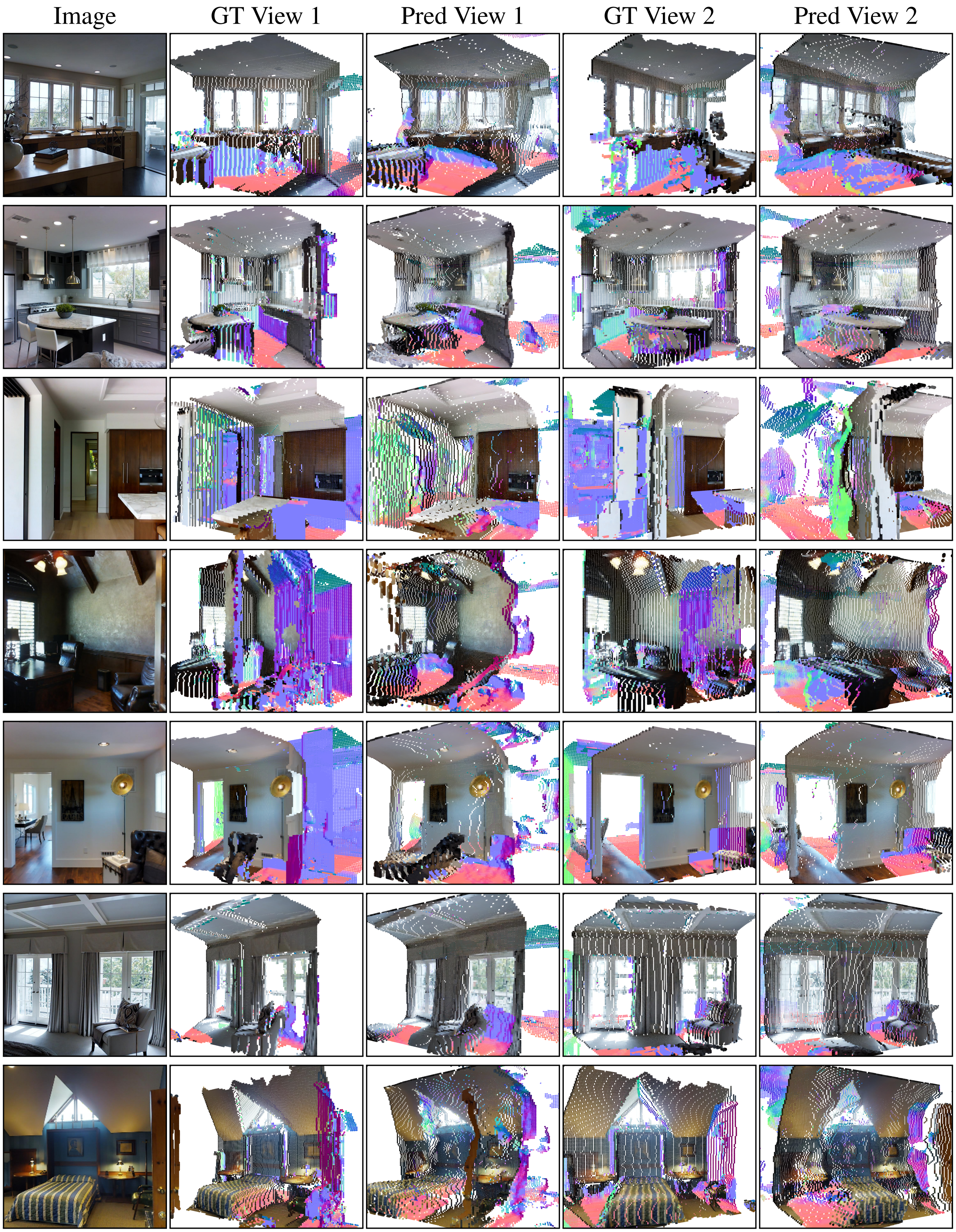


Figure 13. Matterport3D Novel Views. We show outputs of D2-DRDF from novel views on previously unseen input images. In column 2, 3 we show ground truth and prediction for view 1 and in 4,5 we show it for view 2. D2-DRDF is able to recover the inside of the kitchen island in *rows 1, 2*. Our model reconstructs the occluded wall, and empty floor in *rows 5,6*. Please see videos in `matterport_novel.mp4` for additional results

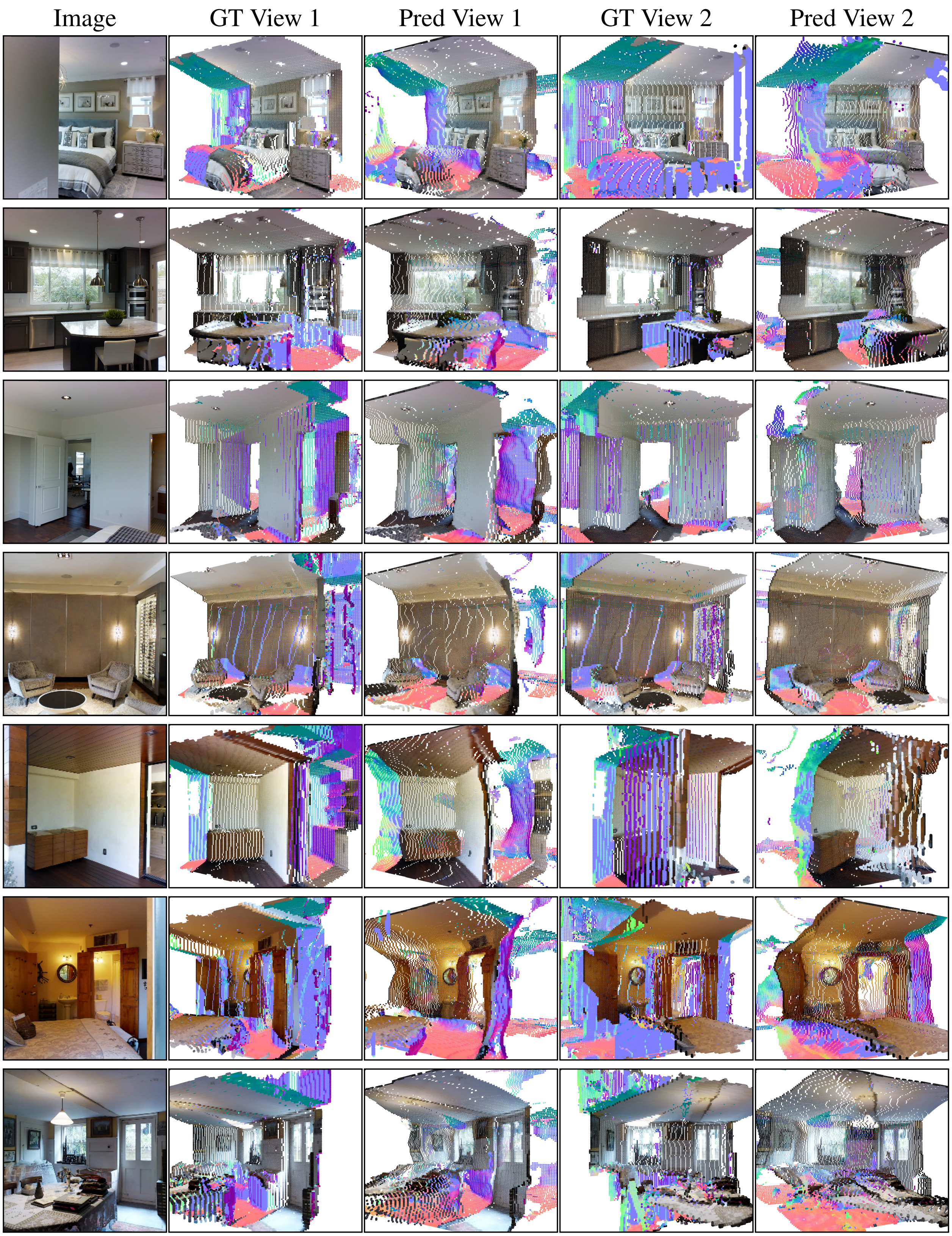


Figure 14. Matterport3D Novel Views. Additional results to Fig 13. Row 2 shows reconstruction of the occluded kitchen island; Row 3 shows the reconstruction of an occluded empty room.

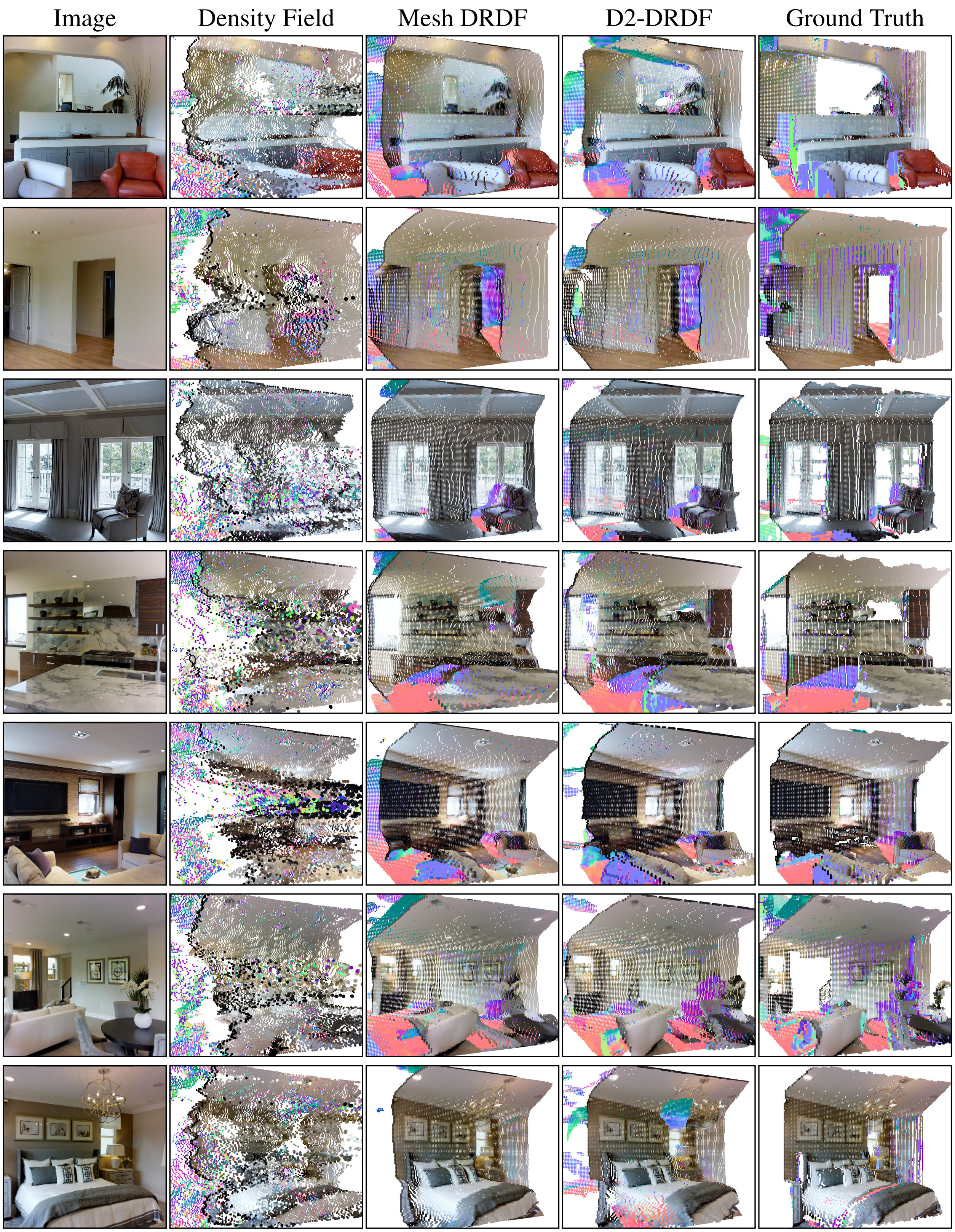


Figure 15. Matterport3D Baselines. Column 1 shows the input image for all the methods. Our method (column 4) shows comparable reconstruction results to the mesh supervised DRDF (column 3). The density fields baselines (column 2) fails to recover sharp occluded reconstruction while D2-DRDF get occluded parts of floors, kitchen islands, walls, kitchen cabinets. Please see video visualizations

Figure 16. OmniData [11] Novel Views. We follow the color scheme from Fig 12, 13 and show reconstruction results on unseen RGB images from OmniData. Please see the video for additional results

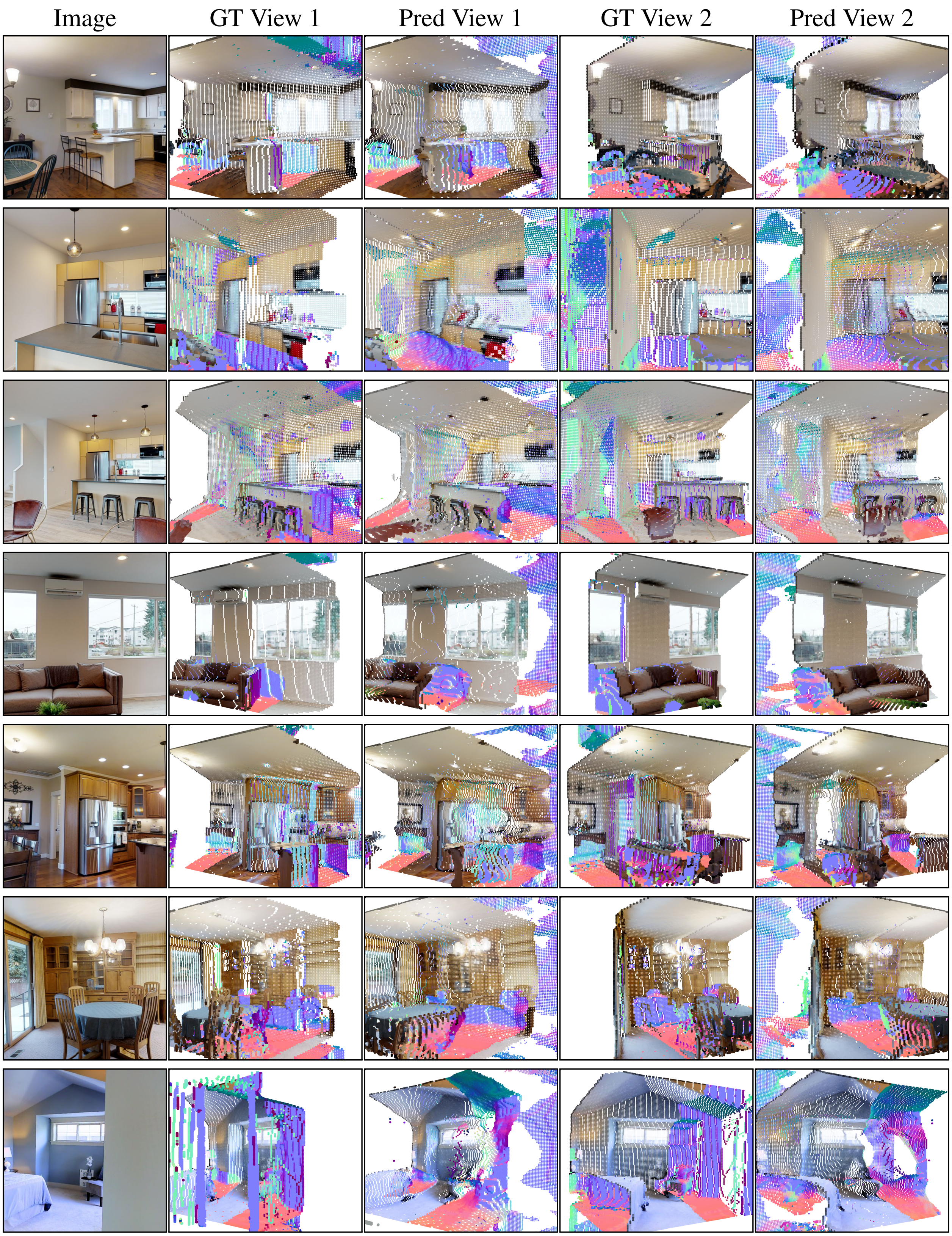


Figure 17. OmniData [11] Novel Views. We follow the color scheme from Fig 12, 13 and show reconstruction results on unseen RGB images from OmniData. Please see the video for additional results