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 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 RGB 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 x. The network accepts
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 \(f\): the vector \(x\) is represented via a scalar \(z\) such that \(x = zf\). Then, if the \(i^{th}\) intersection along the ray is \(s_i \overrightarrow{F}\) for \(s_i \in \mathbb{R}\), the URDF is defined as the distance to the nearest intersection along that ray, or \(d_{UR}(z) = \min_{i=1,\ldots,k} |z - s_i|\).

**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 \(d_{DR}(z) = \text{dir}(z)d_{UR}(z)\), 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 \(|d_{DR}(z)|\) and the nearest intersection on the ray is at \(z + d_{DR}(z)\).

### 4. Depth-Supervised DRDF Reconstruction

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 \(zf\), we aim to define a loss function \(L\) that evaluates a network’s prediction of the DRDF \(\hat{f}_\theta(zf, I)\).

Our key insight is that we can sometimes see this point \(zf\) in other views and these observations of \(zf\) 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 \(sf\) and ending at the point \(ef\) with the ending point \(ef\) closer to \(zf\) than \(sf\) is.

We can derive supervision on \(d_{DR}(z)\) from the segment of visible free space observable in the auxiliary view by using the fact that the nearest intersection at \(z\) is \(|d_{DR}(z)|\) 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 \(|d_{DR}(z)| \geq (e - z)\); 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 \(d_{DR}(z)\) 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 \(\bar{T}\).
and occlusion start/end events as 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 \( y = f_{\theta}(z \bar{r}, \ell) \) from our network and that the point at z along \( \bar{r} \) is in some segment of visible freespace from \( s \) to \( e \) units. Our goal is to penalize the prediction \( y \). Since there are two start/end event classes (I, O), there are four segment types: II, IO, OI, and OO. We show several of these segments in Fig. 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 inequities for occlusions.

\( L_{II} \): 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 \( \ell_1 \) error between the prediction \( y \) and the known DRDF, or \( L_{II}(y) = |y - l_s| \) if \( z < \frac{s + e}{2} \) and \( |y - l_e| \) otherwise. This penalty is zero only when \( y \) is equal to the known DRDF.

\( L_{OO} \): 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 \( |d_{DR}(z)| \) is the distance to the nearest intersection, \( d_{DR}(z) \) 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 \( \ell_1 \) penalty: if we denote halfway between \( l_s \) and \( l_e \) as \( h \), then this can be done as \( L_{OO}(y) = \max(0, l_e - h - |y - h|) \). The resulting penalty is zero if \( y \leq l_s \) or if \( y \geq l_e \).

\( L_{IO} \): IO segment. When the segment is bounded by an intersection event followed by an occlusion event, the situation is more complex and we define \( L_{IO} \) piecewisely. In the first half of the segment \( z < \frac{s + e}{2} \), \( d_{DR} \) is exactly known, and so we can use an \( \ell_1 \) penalty like the II case, so \( L_{IO}(y) = |y - l_s| \). In the second half, there are two options. If the nearest intersection is \( s \), then \( d_{DR}(z) = l_s \). Otherwise, the nearest intersection is unknown but after \( e \) and so \( d_{DR}(z) > l_e \) must hold (since \( d_{DR}(z) \leq l_e \) would imply the existence of an intersection before \( e \)). We take the minimum of errors for the two cases: \( \ell_1 \) distance to \( l_s \) and a \( \ell_1 \) penalty function \( \max(0, l_e - y) \), resulting in \( L_{IO}(y) = \min(\max(0, l_e - y), |y - l_s|) \). This part of the penalty is zero if either \( y = l_s \) or if \( y > l_e \).

\( L_{OI} \): OI segment. The \( L_{OI} \) loss is defined symmetrically to \( L_{IO} \), 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: \( L_{sep} \) represents a prior that surfaces tend to be separated by distances and \( L_{ent} \) captures a property of the DRDF that is true in the limit if our observations are randomly chosen.

\( L_{sep} \): 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 \( L_{sep} \) 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 \( L_{sep}(y) = |y - c| \), so long as there is no conflicting free space evidence from another view.

\( L_{ent} \): Sign Entropy Loss. The occlusion-based constraints
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 $OO$ segment that is contained within another depthmap’s $IT$ segment, then the $OO$ segment can be safely dropped since $L_{OO}(y) \leq L_{IT}(y)$. When segments disagree (e.g., due to inaccurate poses), we keep the segment with more auxiliary views in agreement. This approach handles merging $L_{sep}$: $L_{sep}$ 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 $L_{IT}$ and $L_{sep}$. We then train for 100 epochs with auxiliary losses, minimizing a sum of the segment losses $L_{IT}, L_{OO}, L_{OI}, L_{OO}$, and $L_{sep}$ as well as $\lambda L_{ent}$ with $\lambda$ 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 $3 \times 10^{-4}$. 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

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). Our method trained with only RGBD data recovers occluded empty floors, kitchen cabinets (row 1) and sides of kitchen island (row 3). Our method trained with only RGBD data recovers occluded empty floors, kitchen cabinets (row 1) and sides of kitchen island (row 3). Our method trained with only RGBD data recovers occluded empty floors, kitchen cabinets (row 1) and sides of kitchen island (row 3).
During training, we observed that 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.

**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).

<table>
<thead>
<tr>
<th>ODS</th>
<th>Scene F1</th>
<th>Ray Occ F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SPR</td>
<td>ODS</td>
</tr>
<tr>
<td>100</td>
<td>71.9</td>
<td>71.9</td>
</tr>
<tr>
<td>50</td>
<td>55.6</td>
<td>68.4</td>
</tr>
<tr>
<td>25</td>
<td>56.8</td>
<td>66.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ODS</th>
<th>Scene F1</th>
<th>Ray Occ F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SPR</td>
<td>ODS</td>
</tr>
<tr>
<td>25</td>
<td>86.3</td>
<td>77.2</td>
</tr>
<tr>
<td>12.5</td>
<td>72.8</td>
<td>75.3</td>
</tr>
<tr>
<td>6.3</td>
<td>40.9</td>
<td>73.4</td>
</tr>
<tr>
<td>3</td>
<td>42.9</td>
<td>69.8</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>ODS</th>
<th>Scene F1</th>
<th>Ray Occ F1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SPR</td>
<td>ODS</td>
</tr>
<tr>
<td>25</td>
<td>86.3</td>
<td>77.2</td>
</tr>
<tr>
<td>12.5</td>
<td>72.8</td>
<td>75.3</td>
</tr>
<tr>
<td>6.3</td>
<td>40.9</td>
<td>73.4</td>
</tr>
<tr>
<td>3</td>
<td>42.9</td>
<td>69.8</td>
</tr>
</tbody>
</table>

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 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 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%.

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 Bani, and Chris Rockwell for feedback. Shengyi Qian for the help with ViewSeg code used to implement the PixelNeRF baseline.
References


[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, Bingqiang 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, Bingqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In ICCV, 2021. 2


[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


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, r̃ for every point, x, we compute the z coordinate in the view frame of the auxiliary view. We also record the depth map value at the projection π(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 OI that is subsumed by an 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 $OI$, $OO$, $II$ in order. The $O$ events provide a weaker constraints as compared to the $I$. Throughout whenever there is an $I$ it leads to equality constraints in points on the segment closer to the event, while an $O$ 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 $I$ events lead to a singular solution (shown as a single white line) while $O$ 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 $I$ while vary largely for points close to the $O$. 

Figure 10. Under constrained penalty segments. We consider a ray that has three segments $OI, OC, TT$ 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 (OO). All DRDFs (1-5) match exactly in regions close to an intersection where we have equality constraints (e.g. TT segment). For regions along the ray not bound by segments DRDF is unconstrained. Penalty legend $0\rightarrow 1.7$.

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 ($L_{\text{ent}}$) 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 OC 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 $y \in \mathbb{R}^n$ 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 $y$ 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

$$p \log(p) + (1-p) \log(1-p) \text{ with } p = \frac{1}{N} \sum_{i=1}^{N} H(y_i), \quad (1)$$

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 $y$ 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

$$p \log(p) + (1-p) \log(1-p) \text{ with } p = \frac{1}{N} \sum_{i=1}^{N} \sigma(y_i), \quad (2)$$

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 $y$. 
One minimum in \( y \) is created by generating symmetric values, where each component in \( y \) 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 \( y \) such that \( y_i = -y_{n/2+i} \) for all \( 1 \leq i < \frac{n}{2} \). Then \( \sigma(y_1) + \sigma(y_{n/2+i}) = 1 \), and so \( \delta = \frac{1}{2} \sum_{i=1}^{n/2} (\sigma(y_i) + \sigma(y_{n/2+i})) = \frac{1}{2} \). This setting would happen in the limit if the components of \( y \) were symmetrically distributed over an interval \([-a, a] \) for \( a \in \mathbb{R}^+ \).

Of course, the surrogate function we minimize permits other solutions that balance out the right way. For instance, given on minimizer \( y \) one can generate another minimizer by adding a \( \delta_i \) in one component and adding an appropriate \( \delta_i \) in another (i.e., \( y'_i = y_i + \delta_i \)). This entails picking \( \delta_i \) and \( \delta_2 \) such that
\[
\sigma(y_1 + \delta_1) - \sigma(y_1) = -\sigma(y_2 + \delta_2) - \sigma(y_2),
\]
or that sum remains unmodified. Given a chosen a chosen \( \delta_1 \), some algebra reveals that
\[
\delta_2 = \sigma^{-1}(\sigma(y_1) - \sigma(y_1 + \delta_1) + \sigma(y_2)) - y_2.
\]
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 6 on Omnida.

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.

<table>
<thead>
<tr>
<th>Im. %</th>
<th>M %</th>
<th>Acc Cmp F1</th>
<th>Acc Cmp F1</th>
<th>Acc Cmp F1</th>
<th>Acc Cmp F1</th>
<th>Acc Cmp F1</th>
<th>Acc Cmp F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100</td>
<td>75.4 72</td>
<td>71.9 73.7 73.5 72.1</td>
<td>28.4 30.7 27.3 28.2 22.6 25.1</td>
<td>16.1 23.7 21.2 21.4 29.7 24.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>56</td>
<td>73.0 67.9 68.4 66.6 76.9 70.0</td>
<td>28.4 20.2 23.6 22.4 26.8 24.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>43</td>
<td>72.1 65.6 66.8 64.1 80.5 70.0</td>
<td>27.3 17.3 21.2 21.4 29.7 24.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Im. %</th>
<th>M %</th>
<th>SPR DRDF</th>
<th>SPR DRDF</th>
<th>D2-DRDF</th>
<th>D2-DRDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene</td>
<td>Ray</td>
<td>Scene</td>
<td>Ray</td>
<td>Scene</td>
<td>Ray</td>
</tr>
<tr>
<td>Im. %</td>
<td>M %</td>
<td>Acc Cmp F1</td>
<td>Acc Cmp F1</td>
<td>Acc Cmp F1</td>
<td>Acc Cmp F1</td>
</tr>
<tr>
<td>100</td>
<td>100</td>
<td>75.4 72</td>
<td>71.9 73.7 73.5 72.1</td>
<td>28.4 30.7 27.3 28.2 22.6 25.1</td>
<td>16.1 23.7 21.2 21.4 29.7 24.9</td>
</tr>
<tr>
<td>50</td>
<td>56</td>
<td>73.0 67.9 68.4 66.6 76.9 70.0</td>
<td>28.4 20.2 23.6 22.4 26.8 24.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>43</td>
<td>72.1 65.6 66.8 64.1 80.5 70.0</td>
<td>27.3 17.3 21.2 21.4 29.7 24.9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Im. %</th>
<th>M %</th>
<th>SPR DRDF</th>
<th>SPR DRDF</th>
<th>D2-DRDF</th>
<th>D2-DRDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene</td>
<td>Ray</td>
<td>Scene</td>
<td>Ray</td>
<td>Scene</td>
<td>Ray</td>
</tr>
<tr>
<td>Im. %</td>
<td>M %</td>
<td>Acc Cmp F1</td>
<td>Acc Cmp F1</td>
<td>Acc Cmp F1</td>
<td>Acc Cmp F1</td>
</tr>
<tr>
<td>25</td>
<td>86</td>
<td>82.9 74.1 77.2 69.2 79.1 72.8</td>
<td>43.9 37.3 40.3 29.3 35.6 32.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.5</td>
<td>78</td>
<td>81.4 72.3 75.3 65.9 79.2 70.9</td>
<td>41.8 33.4 37.1 26.3 33.0 29.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.3</td>
<td>69</td>
<td>80.5 69.5 73.4 68.8 77.5 71.8</td>
<td>40.0 27.5 32.6 27.4 28.8 28.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>51</td>
<td>80.5 63.7 69.8 63.7 81.1 70.4</td>
<td>37.1 14.0 20.3 24.4 29.5 26.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Im. %</th>
<th>M %</th>
<th>SPR DRDF</th>
<th>SPR DRDF</th>
<th>D2-DRDF</th>
<th>D2-DRDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scene</td>
<td>Ray</td>
<td>Scene</td>
<td>Ray</td>
<td>Scene</td>
<td>Ray</td>
</tr>
<tr>
<td>Im. %</td>
<td>M %</td>
<td>Acc Cmp F1</td>
<td>Acc Cmp F1</td>
<td>Acc Cmp F1</td>
<td>Acc Cmp F1</td>
</tr>
<tr>
<td>25</td>
<td>65.9 64.4 63.9 69.2 79.1 72.8</td>
<td>23.7 29.3 26.2 29.3 35.6 32.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.5</td>
<td>66.1 62.3 62.8 65.9 79.2 70.9</td>
<td>24.0 28.5 26.1 26.3 33.0 29.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.3</td>
<td>45.9 39.2 40.9 68.8 77.5 71.8</td>
<td>16.4 3.5 5.7 27.4 28.8 28.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>47.6 41.5 42.9 63.7 81.1 70.4</td>
<td>18.2 3.8 24.4 29.5 26.7</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 \(512 \times 512\) 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 performance 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 reconstruction 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 Matterport3D 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.
<table>
<thead>
<tr>
<th>Image</th>
<th>Density Field</th>
<th>Mesh DRDF</th>
<th>D2-DRDF</th>
<th>Ground Truth</th>
</tr>
</thead>
</table>

**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.