What's Behind the Couch? Directed Ray Distance Functions (DRDF) for 3D Scene Reconstruction

Nilesh Kulkarni Justin Johnson David F. Fouhey
University of Michigan
{nileshk, justincj, fouhey}@umich.edu

Abstract

We present an approach for scene-level 3D reconstruction, including occluded regions, from an unseen RGB image. Our approach is trained on real 3D scans and images. This problem has proved difficult for multiple reasons; Real scans are not watertight, precluding many methods; distances in scenes require reasoning across objects (making it even harder); and, as we show, uncertainty about surface locations motivates networks to produce outputs that lack basic distance function properties. We propose a new distance-like function that can be computed on unstructured scans and has good behavior under uncertainty about surface location. Computing this function over rays reduces the complexity further. We train a deep network to predict this function and show it outperforms other methods on Matterport3D, 3D Front, and ScanNet.

1. Introduction

Consider the top-left of Figure 1. What happens if you step to the right and look behind the chair? To you, this single image represents a rich 3D world in which the cabinet continues behind the chairs. This work aims to learn a mapping from a single image to a 3D reconstruction of a scene, including visible *and* occluded surfaces, while learning from real, unstructured scans like Matterport3D [5] or ScanNet [10]. These non-watertight scans are currently one of the richest sources of real-world 3D ground truth, and as more devices integrate LIDAR scanners, their importance will only grow.

Learning from these scans poses challenges to current computer vision methods. Representations based on voxels [9, 18] scale poorly with size, and meshes [54] struggle with widely varying topology. Implicit functions [35, 40] have shown promise at overcoming the size and topology issues of meshes and voxels. However, most work has focused on watertight meshes [7, 35, 38, 40, 45] where there is a well-defined inside and outside. Watertightness

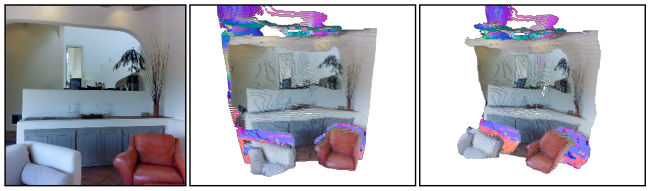


Figure 1. Our approach generates 3D from a single unseen image (left) while training on real 3D scans. We show the two rendered novel views of our method's output (middle, right). Visible surfaces are colored by the image and occluded ones are colored with surface normals: pink is upwards and lavender faces the camera. We can observe the occluded cabinet and floors behind the couch.

enables using *signed distance functions* (SDF) or occupancy functions, but limits methods to watertight data like ShapeNet [6], humans [40], or memorizing single watertight scenes [45]. Real 3D scans (e.g., [5, 10]), on the other hand are off-limits. Exceptions include [8], which fits an *unsigned distance function* (UDF) to instance-specific models, and SAL [1, 2] which learns SDFs on objects with well-defined insides and outsides that also have holes.

We believe the lack of success on learning to predict implicit functions on datasets like Matterport3D [5] stems from three key challenges. First, the unstructured, scene-level nature of the 3D data precludes functions like the SDF or occupancy: even if one starts with watertight surfaces, clipping with the view frustum can make the SDF poorly defined. Second, the distance function to a scene can depend on large parts of the image: as a ray goes through the scene, the nearest point often wanders around, producing a complex distance function. Finally, predicted distance function from models in our settings are *not* characteristics of real distance functions. Instead, we show that the uncertainty about a surface location creates incentives for a model's output to violate basic properties of distance functions, hampering surface extraction.

We propose an approach (§3) for this task, including a new distance-like function, named the *Directed Ray Distance Function* (DRDF). The DRDF can be computed on unstructured scans, does not assume the underlying mesh

is watertight, and depends only on points on a ray through a pixel. The DRDF's behavior near an intersection is relatively stable under uncertainty (§3.3), and a simple zero-crossing (plus a sign check) can find surfaces.

We learn to predict the DRDF with a PixelNerf [56]-style architecture. We compare our proposed approach, DRDF, with other distance functions and ways of obtaining a reconstruction of the scene, such as Layered Depth Images (LDI)[44]. Our experiments (§4) compare the approaches on Matterport3D [5], 3DFront [17], and ScanNet [10]. Our evaluations show that the DRDF is substantially better at recovering the full 3D scene across a variety of metrics, including both visible and invisible regions.

2. Related Work

Our approach aims to infer the full 3D structure of a scene from a single image using implicit functions, which connects with many tasks in 3D computer vision.

Scenes from a Single Image. Reconstructing the 3D scene from image cues is a long-term goal of computer vision. Most early learning-based work focuses on 2.5D properties [4] that are visible in the image, such as qualitative geometry [15, 24], depth [42] and normals [16]. Our work instead aims to infer the full 3D of the scene, including invisible parts. Most work on invisible surfaces focuses on single objects with voxels [9, 18, 21], point-clouds [14, 32], CAD models [26] and meshes [19, 20]. These approaches are usually trained with synthetic data, e.g., ShapeNet [6] or images that have been aligned with synthetic ground-truth 3D [48]. The existing scene-level work, e.g., [30, 31, 37, 51] trains on synthetic datasets containing pre-individuated, watertight objects like SunCG [46]. Our work instead can be learned on real 3D like Matterport3D [5]. On a different front, our work is part of a larger trend in aiming to understand the interplay between 3D, uncertainty, and learning [3, 28, 39]; this work has largely been applied in the depth-map space.

Implicit Functions for 3D reconstruction. We approach the problem with using implicit functions [7, 35, 38], which have shown promise in handling problems of scale and varying topology. These implicit functions have also been used in view synthesis [34, 36, 56, 57], which differs from our work in goals. In reconstruction, implicit functions have shown impressive results on two styles of task: fitting to a single model/scene (e.g., SIREN [45]) and predicting new single objects (e.g., PiFU [40, 55]). Our work falls in the latter category as it predicts previously unseen scenes. While implicit functions have shown impressive results on humans [40, 41] and ShapeNet objects [55], most existing work relies on watertight meshes. Our non-watertight setting is more challenging, and two solutions have been proposed: assuming the existence of the SDF and supervising it indirectly (SAL: [1, 2]), and training with an unsigned distance function (UDF) [8] (although we stress that [8] does not predict scene-level distance functions from RGB images). The DRDF can also be trained on non-watertight meshes, but outperforms these two approaches: the DRDF works even if the SDF is not defined, and is relatively stable under uncertainty.

Recovering Occluded Surfaces. Our system produces the full 3D structure of a scene, including occluded parts from a single image. This topic has been of great interest to the community (in addition to previously mentioned work on volumetric 3D e.g., [9, 18]). Early work often used vanishing-point-aligned box [12, 23] trained on annotated data. While our approach predicts floors, this is learned, not explicitly baked in, unlike modern inheritors that have explicit object and layout components [27, 50] or the ability to query for two layers [25]. One alternate approach is layered depth images (LDI) [13, 44] or multi-plane depthmaps. LDIs can be learned without direct supervision [52], but when LDIs are trained directly, they fare worse than DRDF.

3. Method

We aim to reconstruct the full 3D extent of a previously unseen scene from a single RGB image, including occluded portions and while training on real scene captures [5]. This setting is challenging because: the meshes are not watertight; scene distance functions are hard to model; and uncertainty about the surface rewards the network for distorted outputs that lack *critical* distance function properties.

We propose a new distance-like function, the *Directed Ray Distance Function (DRDF)*, that performs well in our setting and couple it with a deep network to predict it. Our method starts with a network that predicts distance functions using pixel-aligned features (§3.1). Our network is generic and can be paired with a distance function (e.g., SDF) and a decoding strategy (e.g., find zero-crossings). The choice of distance function and decoding is critical: our DRDF (§3.2) can be computed on non-watertight objects and has better properties under uncertainty (§3.3).

3.1. Setting and Base Architecture

Our goal is to learn an image-conditioned mapping from a 3D point $\mathbf{x}$ to a (potentially signed) distance to the scene surfaces, or $f_\theta(\mathbf{x}; I) : \mathbb{R}^3 \rightarrow \mathbb{R}$. We shown an overview of approach in Fig. 2. We use a PixelNerf [56]-style architecture consisting of a backbone $b_\theta(\cdot)$ and MLP regressor $h(\cdot)$ that are jointly trained. The backbone produces a feature map $\mathbf{B}$, and the MLP uses $\mathbf{B}$ along with a query point $\mathbf{x} \in \mathbb{R}^3$ to predict a distance function. We extract a feature vector $v(\mathbf{x}, \mathbf{B})$ from $\mathbf{B}$ via bi-linear interpolation. This feature consists of the image feature at the projection of $\mathbf{x}$ using the camera π and is concatenated with positional em-

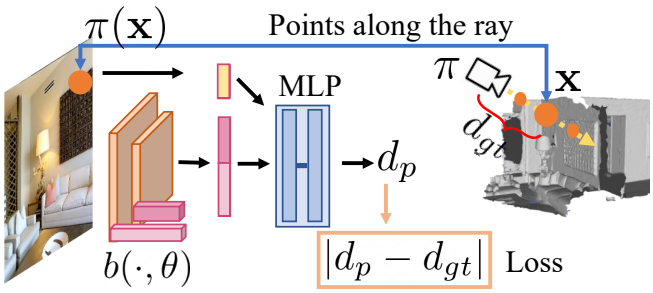


Figure 2. Approach. At training time, our model takes an image and set of 3D points along rays and is supervised on the distance to the nearest intersection via the real 3D geometry. For inference, our model predicts the distance for each point in a 3D grid, which is then decoded to intersections. More details under §3.1

bedding [36] of $\mathbf{x}$. The final prediction d_p is made by a MLP regressor head $h_\theta(v(\mathbf{x}, B)$, making the full output of $f_\theta(\mathbf{x}; I)$ equal to $h_\theta(v(\mathbf{x}, b_\theta(I)))$.

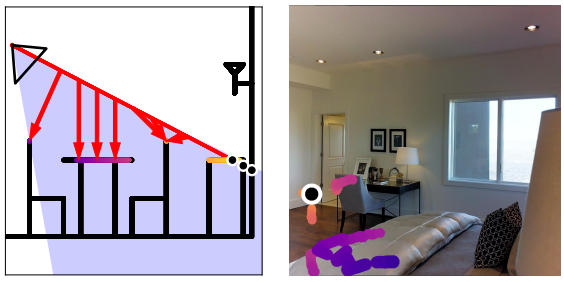
At training time, we are given n samples $\{(\mathbf{x}_i, I_i, d_i)\}_{i=1}^n$, each consisting of an image I_i, point $\mathbf{x}_i$, and target distance d_i. We find the minimizer θ^* of the empirical risk $\sum_{i=1}^n \mathcal{L}(f_\theta(\mathbf{x}_i, I_i), d_i)$ for a loss $\mathcal{L}$. By changing the targets d_i, one can change distance functions.

At test time, one predicts the distance at a set of points in 3D, which in turn must be *decoded* into a set of surface locations. Given a ground truth distance function, this decoding is usually a trivial post-processing and one can use any number of properties of distance functions (e.g., the presence of a zero-crossing in value or its derivative) to obtain a surface. However, in our setting with uncertainty, we find the choice of distance function is important. Our proposed approach is well-behaved under uncertainty, and so its decoding is trivial and hyper-parameter free, but we extensively optimize the decoding strategies of baseline methods.

3.2. Distance Functions

It is critical to distinguish between *scene distance functions* and *ray distance functions*. Given a point $\mathbf{x}$, the scene distance considers the distance between $\mathbf{x}$ and all points in the scene. For example, given a set of 3D points $\mathcal{P}$, the *unsigned distance function* (UDF) is the distance to the nearest point in the scene, $d_{\text{U}}(\mathbf{x}) = \min_{\mathbf{x}' \in \mathcal{P}} ||\mathbf{x} - \mathbf{x}'||_2$. A ray distance, on the other hand, only considers the points in $\mathcal{P}$ along the ray joining $\mathbf{x}$ and the camera center (assumed to be $\mathbf{0}$), or $\{s\mathbf{x} : s \in \mathbb{R}^+, s\mathbf{x} \in \mathcal{P}\}$. Ray distance functions can be constructed by applying a standard distance function to the 1D space along the ray. Suppose $\mathcal{D}$ is set of *distances* to surfaces the ray through $\mathbf{x}$ crosses, or $\mathcal{D} = \{s : s \in \mathbb{R}^+, s\mathbf{x} \in \mathcal{P}\}$. Then, given a distance z along the ray, the *unsigned ray distance function* (URDF) $d_{\text{UR}}(z)$ is defined as $\min_{s \in D} |z - s|$. The URDF is precisely the 1D case of the UDF applied to distances along the ray.

Considering only the ray through a pixel has consequences for learning to predict distances. By definition, the only surfaces in the scene that define a ray distance function are also on the ray to $\mathbf{x}$; these surfaces thus also project to the same location as $\mathbf{x}$. In contrast, the set of surfaces that defines a scene distance can fall all over the the scene. We illustrate this in Fig. 3. Given the set of points along ray through $\mathbf{x}$, or $\{s\mathbf{x} : s \in \mathbb{R}^+\}$, we can compute the set of scene points that are nearest to a point on the ray, or $\{\arg\min_{x' \in \mathcal{P}} ||s\mathbf{x} - \mathbf{x}'||_2 : s \in \mathbb{R}^+\}$. The projection of this is usually large. We can compute the set of scene points that are nearest to a point on the ray or .... This set of points usually projects to all over the image. We quantify this by measuring how far the projection of all the nearest points is from the ray center. The largest distance gives the minimum distance the network needs to look. The average maximum distance from ray center is 96px on a 256^2px image (averaged over 50K rays on Matterport3D [5]). The network might also need to look and examine all the points within 1m of the ray. This requires looking 145px away from the ray on an average. Thus, accurately inferring a UDF often requires a network to integrate information from much more of the image compared to a ray distance function.



(a) 2D Cross Section (b) 3D Example

Figure 3. Scene vs. ray distances. (a) A 2D cross-section of a simplified kitchen. The red ray intersects the scene at the black and white points along the ray. These points define the *ray distance*. However, the nearest points on the ray in the scene (colored by distance Min Max) cover the nearly the full scene. These far-flung points define the *scene distance*. We shade this area in light blue. (b) A real 3D example of a different scene. The colored points project all over the scene.

Focusing on rays, we can define more ray distance functions. For watertight meshes, one can have a predicate *inside*(z) that is 1 when z is *inside* an object; then the *Signed Ray Distance Function* (SRDF) is $d_{\text{SR}}(z) = -\textit{inside}(z) \min_{s \in \mathcal{D}} |z - s|$. Since our setting is non-watertight, the SRDF is impossible, as is using *inside* to define occupancy. However, one can define a proximity-based occupancy ray function (ORF) that is 1 if z is within r of an intersection $d_{\text{ORF}}(z) = \left(\min_{s \in \mathcal{D}} |z - s|\right) < r$.

We introduce a new ray distance-like function, called the *Directed Ray Distance Function*. Given a set of intersections $\mathcal{D}$ along the ray and distance z, the DRDF is $d_{\text{DRDF}}(z) = (\arg\min_{s \in \mathcal{D}} |s - z|) - z$. Like the URDF, there

is no notion of inside, so the DRDF can be used with unstructured scans. Near an intersection, the DRDF behaves like the SDF and crosses zero. In exchange, there is a sharp discontinuity halfway between intersections.

3.3. Distance Functions Under Uncertainty

When a network is trained on one unambiguous scene, its optimal output is to reproduce the scene perfectly. We now ask what happens with uncertainty, e.g., if a network predicts an new scene from an RGB image. We analyze the uncertainty of a single intersection, which we now assume is a random variable S rather than a fixed location. In this case, the ray distance function depends on what value s the random variable S takes on. We denote the distance function at z if the intersection is at s as $d(z; s)$.

Under uncertainty, a network trained to minimize the MSE is optimal when it produces the expected distance under S, or $\mathrm{E}_S[d(z;s)] = \int_{\mathbb{R}} d(z;s)p(s)ds$ where $p(s)$ is the density of S. This expectation minimizes other losses: if $d(z; s)$ is an indicator, $\mathrm{E}_S[d(z;s)]$ minimizes the cross-entropy loss; and if the distribution over distances to S is symmetric, then $\mathrm{E}_S[d(z;s)]$ also minimizes the L1 loss.

We can understand optimal behavior under uncertainty for a given form of S by calculating $\mathrm{E}_S[d(z;s)]$ for different ray distance functions. We derive results for when S is normally distributed with its mean μ at the true intersection, standard deviation σ, and CDF $\Phi(s)$. Since distance functions also depend on the next intersection, we assume the next intersection is at $S + n$ for a constant $n \in \mathbb{R}^+$.

We summarize some salient results here, and a detailed analysis appears in the supplement. No expected function perfectly matches its true function, but each varies in where the distortion occurs. Figure 4 shows $\mathrm{E}_S[d(z;s)]$ for three ray distance functions (for $n = 1, \sigma = 0.2$). At the intersection, the expected SRDF and DRDF closely match the true function while the expected URDF is grossly distorted. Full derivations appear in the supplemental, but the expected URDF has a minimum value of $\approx\sigma\sqrt{2/\pi}$ rather than 0. Similarly, its previously sharp derivative is now now $\approx 2\Phi(z) - 1$, which is close to ± 1 *only* when z is far from the intersection. In contrast, the expected DRDF's distortion occurs at $\mu + \frac{n}{2}$, and its derivative $(np(z - \frac{n}{2}) - 1)$ is close to -1, except when z is close to $\mu + \frac{n}{2}$.

These distortions disrupt decoding of distance functions to surfaces. An actual URDF can be converted to a surface by thresholding, but the expected URDF has minimum value $\approx\sigma\sqrt{2/\pi}$, not 0. In 3D estimation, the errors are usually heteroscedastic, e.g., a nearby intersection often has less uncertainty than a far intersection (even just because the error is relative). So a threshold that handles certain (e.g., near) intersections may miss a uncertain (e.g., far) intersections; getting these far intersections, conversely, will dilate the near intersection. The expected URDF retains

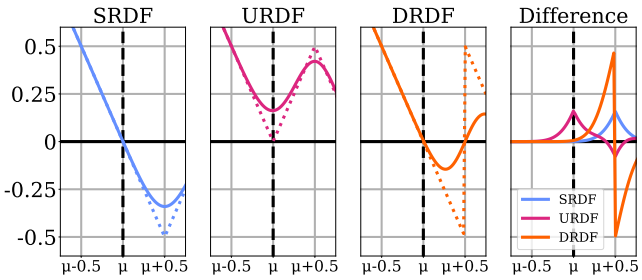


Figure 4. Suppose the surface's location is normally distributed with mean μ at its true location and σ=0.2, and the next surface 1 is unit away. We plot the expected (solid) and true (dashed) distance functions for the SRDF, URDF, and DRDF and their difference (expected - true). The SRDF and DRDF closely match the true distance near the surface; the URDF does not.

the zero-crossing of the derivative, but the discontinuity is blunted and our empirical results suggest that finding the zero-crossing is less effective than other schemes. Similar effects hamper proximity based occupancy: the optimal output at z is the chance a surface is within r, which squashes the network output in an uncertainty-dependent fashion.

The DRDF is more stable under uncertainty. Finding a zero-crossing requires no tuning, and the zero-crossing at the intersection is preserved except when σ is large (e.g., σ=$\frac{n}{3}$), at which point other functions have also broken down. This is because its uncertainty-driven distortions occur halfway to the other intersection. The only nuance is a second zero-crossing after the intersection, which can be ignored by filtering based on the direction of the crossing.

Limitations. We briefly discuss some limitations of our analysis, with full details in the supplement. *L1-vs-MSE:* our results are for $\mathrm{E}_S[d(z;s)]$ (which optimizes the MSE) rather than the median (which optimizes the L1 loss); empirically, we find these values to be usually nearly identical, due to the symmetry of our noise. *Rays-vs-Scenes:* Our analysis focuses on rays. Our analysis of UDRFs directly extends to a UDF to a plane under isotropic Gaussian noise, which has the same form as the URDF but with point-plane distance. In general, the expected UDF with non-trivial noise is non-zero since it is a convex combination of 0 and positive values. *Training Effects:* Our analysis describes the optimal behavior of the network; while our networks may not achieve this, we find good qualitative agreement.

3.4. Implementation Details

We now describe the implementation of our network. We use this network to estimate a variety of distance functions, and so our presentation is generic until the decoding step.

Training. Given samples $\{\mathbf{x}_i, I_i, d_i\}_{i=1}^n$ consisting of 3D points, images, and target distance functions respectively, we train our network to minimize the L1 loss, $\sum_{i=1}^n |d_i - h_\theta(v(\mathbf{x}_i, b_\theta(I_i)))|$, where the predictions are in log-space

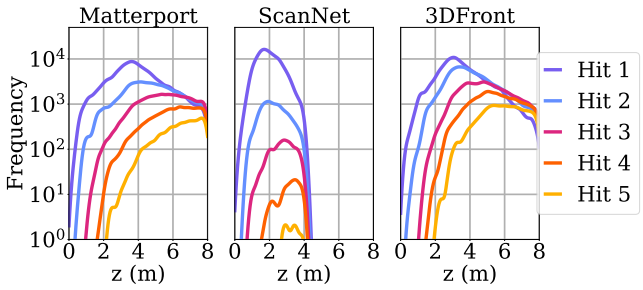


Figure 5. Ray Hit count distribution. We compare the distribution over surface hit locations for first 4 hits over 1M rays and observe that ScanNet has $\leq$ 1% rays as compared to Matterport and 3DFront which have $\geq$ 25% rays with more than 2 hits

truncated at 1m following other methods that predict TSDFs [11, 47]. We optimize using AdamW [29, 33] using a learning rate of 10^{-4}. We sample points $\mathbf{x}_i$ for each scene two ways: for each intersection at l, we sample 512 points from $\mathcal{N}(l, 0.1)$; for each ray we sample 512 points uniformly from 0 to a maximum prediction depth. We do this for 20 intersections and 20 rays per scene in a batch. We train with a batch size of 10 for 250K iterations and freeze batch norm layers after $\frac{1}{4}$th of the iterations.

Inference. At inference time, we extract backbone features at a regular $N \times N$ grid (N=128) in one forward pass of b. For each grid point, we predict the distance function for $K = 128$ points linearly spaced on each ray from 0 to a maximum depth. This entails making N^2K predictions with the regression head, which is parallelizable, yielding a frustum-shaped volume of locations with predictions. Once a distance volume is predicted, methods vary in how they decode the distances to surfaces. The DRDF requires finding positive zero-crossings; we explain others in §4.1.

4. Experiments

We evaluate DRDF on real images of scenes and compare to alternate choices of distance functions and formulations of the scene prediction problem such as Layered Depth Images[44]. We also try extensive variants of decoding schemes for our baseline methods. Our experiments evaluate both the methods ability to predict the visible and occluded parts of the scene using standard metrics and new metric that evaluates along rays.

Metrics. We use three metrics. No one metric properly quantifies reconstruction as each captures a different aspect of the task [49]. The first is scene chamfer errors at multiple thresholds. The others are accuracy/completeness [43] and their harmonic mean, F1-score [49], for scenes and rays (focusing on occluded points).

Chamfer L1. We compute Symmetric Chamfer L1 error for each scene with 30K points sampled from the ground truth and the prediction. We plot the fraction of scenes with Symmetric Chamfer L1 errors that are less than t for $t \in [0, 1]$m. This is more informative than just the mean across the dataset and compares performance over the curve.

Scene (Acc/Cmp/F1). Like [43, 49], we report accuracy/Acc (% of predicted points within t of the ground-truth), completeness/Cmp (% of ground-truth points within t of the prediction), and their harmonic mean, F1-score. This gives a summary of overall scene-level accuracy.

Rays (Acc/Cmp/F1), Occluded Points. We additionally evaluate each ray independently, measuring Acc/Cmp/F1 on each ray and reporting the mean. In the paper, we report results for occluded points (defined as all surfaces past the first for both ground-truth and predicted). The supplement contains full results. Evaluating each ray independently applies a more stringent test for occluded surfaces compared to scenes: with scene-level evaluation on a high resolution image, a prediction can miss a hidden surface (e.g., the 2nd surface) on every other pixel since the missing predictions will be covered for by hidden surfaces in adjacent rays. Ray-based evaluation, however, requires each pixel to have all hidden surfaces present to receive full credit.

Datasets. We desire three key properties in datasets: the images should be real to prevent the abuse of rendering artifacts; the mesh should be a real capture since mimicking capture holes is itself a research problem; and the data ought to have lots of occluded geometry. Our primary dataset is Matterport3D [5], which satisfies all properties. We also evaluate on 3DFront [17] and ScanNet [10]. While 3DFront does not have capture holes, cutting it with a view frustum creates holes. ScanNet [10] is a popular dataset in 3D reconstruction, but has far less occluded geometry compared to the other datasets. A full description of the datasets appears in the supplement.

Matterport3D [5]. We use the *raw images* captured with the Matterport camera. We split the 90 scenes into train/val/test (60/15/15) and remove images that are too close to the mesh ($\geq$60% of image within 1m) or are $>$20° away from level. We then sample 13K/1K/1K images for train/val/test set.

3DFront [17]. This is a synthetic dataset of houses created by artists with underling hole-free 3D geometry. We collect 4K scenes from 3DFront [17] after removing scenes with missing annotations. We select 20 camera poses and filter for bad camera poses similar in Matterport3D [5]. Our train set has 3K scenes with approximately 47K images. Our val and test sets have 500 scenes with 1K images each.

ScanNet [10]. We use splits from [10] (1045/156/312 train/val/test) and randomly select 5 images per scene for train set, and 10 images per scene for val/test set. We then sample to a set of 33K/1K/1K images per train/val/test.

Scene Statistics. To give a sense of scene statistics, we plot the frequency of the locations of the first 5 hits for each dataset (computed on 1M rays each) in Figure 5. 99% of ScanNet rays have 1 or 2 hits, while $\geq$24% of Matterport3D [5] and 3DFront [17] rays have more than 2 hits.

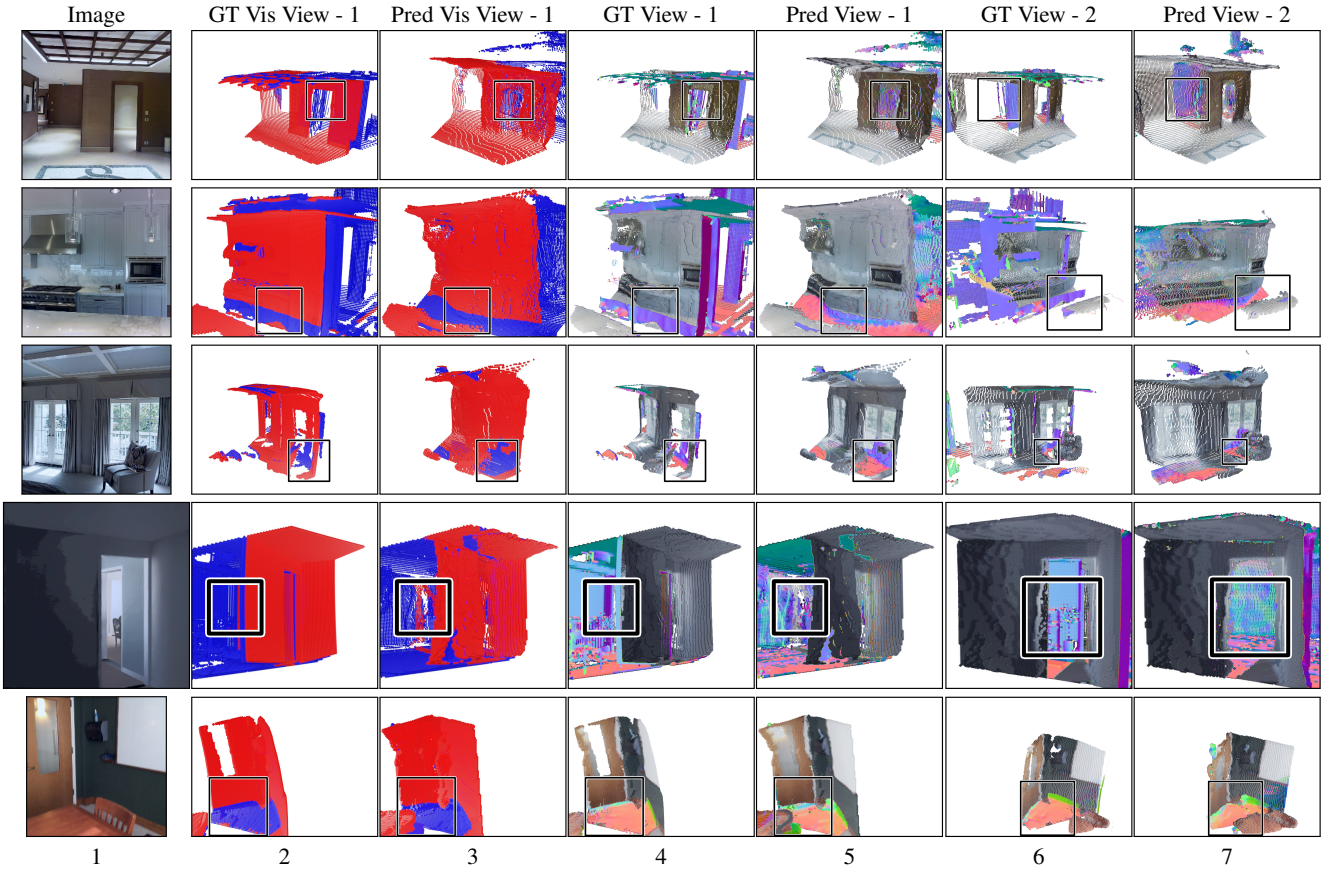


Figure 6. Novel views from DRDF We render outputs of our method and the ground-truth from new viewpoints with $\geq 45^\circ$ view change. Columns 2 & 3 show new views where visible points are red and occluded points are blue. In other columns, we color the visible regions with the image and occluded regions with computed surface normals (, scheme from camera inside a cube). DRDF recovers occluded regions, such as a room behind the door (row 1 & 4), a hidden floor behind the kitchen counter (row 2), and a wall and floor behind the chair/couch (row 3 & 5). Rows 1-3 are Matterport3D [5]; Row4 is 3DFront [17]; Row 5 is ScanNet [10].

4.1. Baselines

We compare against baselines to test our contributions. For fair comparison, all approaches use the same ResNet-34 [22] backbone and extract features from multiple layers via bilinear interpolation [56]. Thus, different distance functions are trained identically by replacing the target distance. Each method's description consists of two parts: a prediction space parameterization and a decoding strategy to convert the inferred distances to surfaces.

Picking decoding strategies. Most baselines predict a distance function rather than a set of intersections and thus need a decoding strategy to convert distances to a set of surface locations. Some baselines have trivial strategies: LDI directly outputs intersections; SAL and DRDF use zero-crossings and have no parameters. However, UDF, URDF, and ORF are sensitive to decoding strategy. We tried multiple strategies for each based on past work and theoretical analysis of their behavior and report the best strategy (measured with Scene F1 on Matterport3D [5]) as well as alternates. When decoding strategies have detection parameters, we tune to ensure similar completeness to our method. Accuracy and completeness trade-off, and having one roughly fixed ensures that methods are being compared at similar points on the operating curve, making F1 score meaningful.

Layered Depth Images (LDI). To test the value of framing the problem as implicit function prediction, we train a method to predict a k-channel depthmap where the i^{th} output is trained to predict the i^{th} intersection along the ray through the pixel. We use a L1 loss per pixel and intersection. We set $k = 4$, the same number of intersections the proposed approach uses.
Decoding. The LDI directly predicts surface locations.

Layered Depth Images with Confidence (LDI + Conf.). We augment the the LDI baseline with k additional channels that represent the likelihood the i^{th} intersection exists. These added channels are trained with binary cross-entropy.
Decoding. For each pixel, we accept or ignore each layer based on the whether the predicted probability exceeds 0.5.

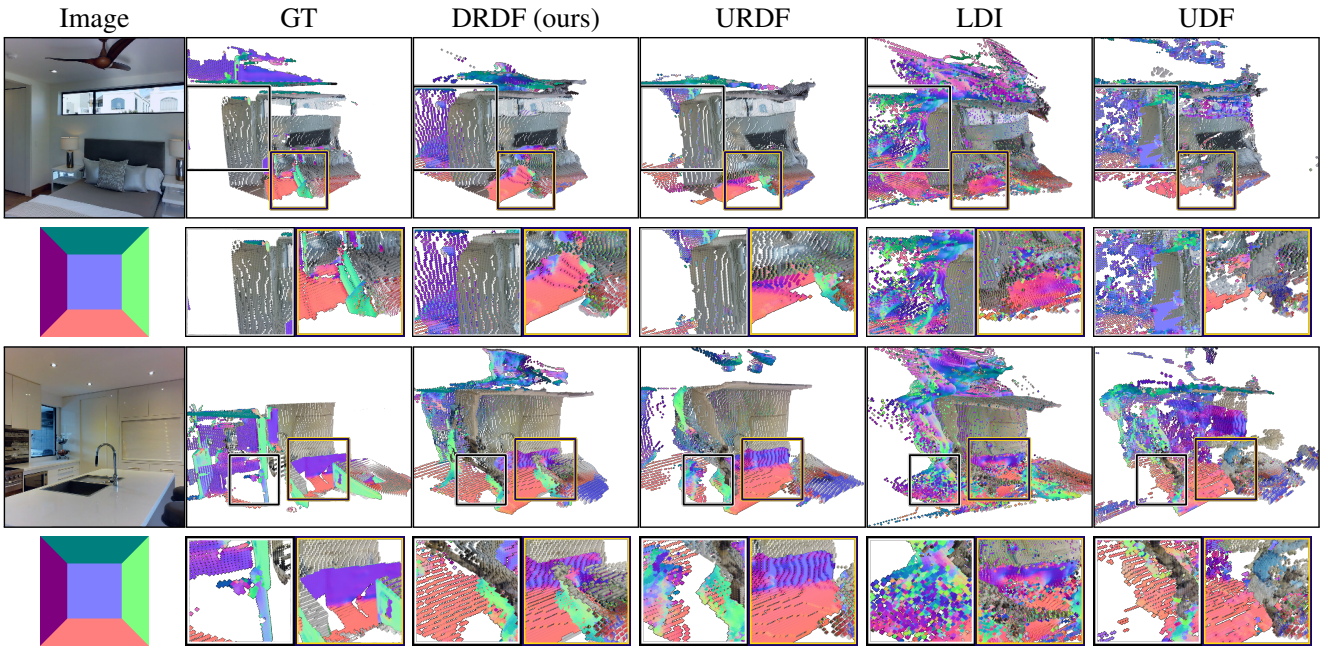


Figure 7. Qualitative Comparison with Baselines We render the generated 3D outputs in a new view (rotate left ←). We show two crops on the next row for better visual comparision. Visible regions are colored using the image and occluded regions are colored with surface normals (color scheme with a camera inside the cube). DRDF produces higher quality results compared to LDI and UDF (row 1, 2, more consistency, smoother surface, no blobs). In row 1 (crop 2), URDF misses parts of floor; in row 2 (crop 2), URDF misses the green colored side of the kitchen counter. See supp. for more results.

Unsigned Distance Function (UDF) [8]. To test the value of predicting distances on rays, we learn to predict the UDF to the scene rather than the ray.
Decoding. We use `scipy.argrelextrema` [53] to find local extrema. We find local minimas of the distance function within a 1m window along the ray. We found this works better than absolute thresholding (by 14.7 on F1). Sphere tracing and gradient-based optimization as proposed by [8] performs substantially worse (25.7 on F1), likely since it assumes the predicted UDF behaves similar to a GT UDF.

Unsigned Ray Distance Function (URDF). For direct comparison with ray-vs-scene held constant, we compare with the URDF.
Decoding. Decoding a URDF is challenging. We perform non-maximum suppression on the thresholded data by finding connected components of the ray that have predicted distance below a tuned-constant τ, and keep the first prediction as the surface location. We found this to outperform: thresholding (by 5.3 on F1); finding zero-crossings of the numerical gradient (by 11 on F1); and using sphere tracing and gradient-based optimization as in [8] (by 6.6 on F1).

Ray Sign-Agnostic Learning Loss (SAL) [1]. Traditional SDF learning is impossible due to the non-watertightness of the data and so we use the sign agnostic approach proposed by [1]. We initialize our architecture with the SAL initialization and train with the SAL loss. The SAL approach assumes that while the data may not be watertight due to noisy capture, the underlying model is watertight. In this case, rays start and end *outside* objects (and thus the number of hits along each ray is even). This is not necessarily the case on Matterport3D [5] and 3DFront [17].
Decoding. Following [1], we find surfaces as zero-crossings of the predicted distance function along the ray.

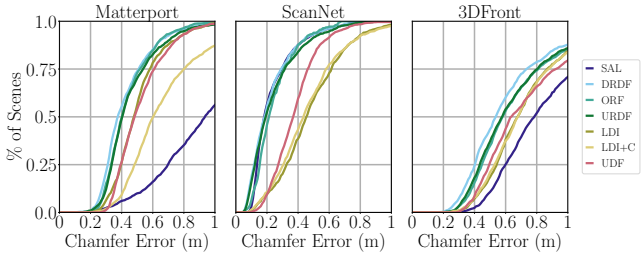


Figure 8. Chamfer L1 distances. We plot % of scenes on Y-axis as function of symmeteric Chamfer L1 error below a threshold t on X-axis. On Matterport and 3DFront datasets DRDF is better and comparable to the best alternate method on ScanNet.

Ray Occupancy (ORF). Traditional interior/exterior occupancy is impossible on non-watertight data, but one can predict whether a point is within r of a surface as a classification problem. This tests the value of predicting distances, and not just occupancy. We tried several values of r ([0.1, 0.25, 0.5, 1]m) and report the best-performing version.

Table 1. Scene Acc/Comp/F1Score. Thresholds: 0.5m (MP3D [5], 3DFront [17]), 0.2m (ScanNet [10]). **Bold is best**, underline is 2nd best per column. DRDF is best in F1 and accuracy, and always comparable to the best in completeness.

Method	MP3D [5]			3DFront [17]			ScanNet [10]		
	Acc	Cmp	F1	Acc	Cmp	F1	Acc	Cmp	F1
LDI	66.2	72.4	67.4	68.6	46.5	52.7	19.3	28.6	21.5
LDI +C	64.8	55.1	57.7	70.8	45.1	52.4	19.9	32.0	23.3
SAL [1]	66.1	25.5	34.3	80.7	28.5	39.5	51.2	**70.0**	57.7
UDF	58.7	**76.0**	64.7	70.1	51.9	57.4	44.4	62.6	50.8
ORF	73.4	69.4	69.6	86.4	48.1	59.6	51.5	58.5	53.7
URDF	74.5	67.1	68.7	85.0	47.7	58.7	61.0	57.8	58.2
DRDF (ours)	**75.4**	72.0	**71.9**	**87.3**	**52.6**	**63.4**	**62.0**	62.7	**60.9**

Table 2. Ray Acc/Comp/F1Score on Occluded Points. Thresholds: 0.5m (MP3D [5], 3DFront [17]), 0.2m (ScanNet [10]). DRDF is best on F1 and Acc, and is occasionally 2nd best on Cmp. Gains on occluded points are even larger than the full scene.

Method	MP3D [5]			3DFront [17]			ScanNet [10]		
	Acc	Cmp	F1	Acc	Cmp	F1	Acc	Cmp	F1
LDI	13.9	**42.8**	19.3	17.8	35.8	22.2	0.5	9.0	2.4
LDI +C	18.7	21.7	19.3	17.7	22.6	19.9	1.1	2.4	3.5
SAL [1]	5.5	0.5	3.5	24.1	4.3	11.4	2.4	**38.7**	5.6
UDF	15.5	23.0	16.6	29.3	21.3	23.4	1.8	7.8	5.5
ORF	26.2	20.5	21.6	53.2	22.0	31.0	6.6	12.3	11.4
URDF	24.9	20.6	20.7	47.7	23.3	30.2	8.4	11.6	13.8
DRDF (ours)	**28.4**	30.0	**27.3**	**54.6**	**56.0**	**52.6**	**9.0**	20.4	**16.0**

Decoding. Each surface in the prediction is, in theory surrounded by two crossings each with probability 0.5 that are equidistant from the surface: an onset/rising crossing and offset/falling crossing. Finding 0.5-crossings leads to doubled predictions. Instead, we consider all adjacent and nearby pairs of offsets and onsets, and average them; unpaired crossings are kept on their own. We found this outperformed keeping just a single crossings (by 4.7 on F1).

4.2. Results

Qualitative Results. Qualitative results of our method appear throughout the paper (by itself in Fig. 6 and compared to baselines in Fig. 7). Our approach is is often able to generate parts of the occluded scene, such as a room behind a door, cabinets and floor behind kitchen counters, and occluded regions behind furniture. Sometimes the method completes holes in the ground-truth that are due to scanning error. On the other hand we see our method sometimes fails to characterize the missing parts as detailed occluded 3D.

Compared to baselines, our approach generally does qualitatively better. LDI and UDF often have floating blobs or extruded regions, due to either predicting too many layers (LDI) or having a distance function that is challenging to manage (UDF). URDF produces qualitatively better results, but often misses points, especially in occluded regions.

Quantitative Results. These results are borne out in the quantitative results. We report the Chamfer plot in Figure 8, the scene distance metrics in Tab. 1 and occluded surfaces metrics along rays in Tab. 2. DRDF consistently does at least as well, or substantially better than the baselines on Chamfer. In general, DRDF does substantially better than all baselines. In a few situations, a baseline outperforms DRDF in completeness at the cost of substantially worse accuracy. However, no one baseline is competitive with DRDF *across* datasets: SAL works well on ScanNet [10] and LDI works well on MP3D [5].

LDI performs worse than DRDF because it cannot vary its number of intersections; simply adding a second stack of outputs (LDI + C.) is insufficient. We hypothesize this is because DRDF can learn *where* things tend to be, while LDI-based methods also have to learn the order in which things occur (e.g., is the floor 2nd intersection or the 3rd intersection at this pixel?). SAL performs competitively on ScanNet, likely because of the relatively limited variability in numbers of intersections per ray; when tested on Matterport3D and 3DFront, its performance drops substantially.

The most straightforward way to learn on non-watertight data is to predict unsigned scene distances [8] which has been shown to work with memorizing 3D scenes. However, predicting it from a single image is a different problem entirely, and scene distances require integration of information over large areas. This leads to poor performance. Predicting distances on rays can alleviate this challenge, but recovering intersections remains hard even with multiple decoding strategies. Hence, DRDF outperforms URDF on scene metrics and even more so on occluded points in Tab. 2. Similarly ORF too requires decoding strategies and is sensitive to training parameter choices. In contrast, by accounting for the uncertainty in surface location, DRDF requires a simple decoding strategy and outperforms the other methods.

Conclusions. We have introduced a method for predicting full 3D from a single RGB image with implicit functions. Our contribution is a new distance-like function, the DRDF, and show it has more desirable theoretical properties under uncertainty compared to standard unsigned distance function. When we use DRDF in an implicit function method and train on 3D scan datasets, we show it obtains substantially better performance compared to existing approaches.

Our approach advances the state of the art in 3D from a single image, which can in turn be used for tasks as varied as content creation to AR/VR to autonomous systems. Many have positive applications, but some do not. Moreover, much of our data does not reflect most peoples' reality – MP3D, for instance, has many lavish houses – and so this may widen technological gaps. On the other hand, by handling non-watertight scans, our system may enable learning from scans collected by ordinary people rather than experts.

References

[1] Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2565–2574, 2020. 1, 2, 7, 8

[2] Matan Atzmon and Yaron Lipman. Sal++: Sign agnostic learning with derivatives. *arXiv preprint arXiv:2006.05400*, 2020. 1, 2

[3] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Estimating and exploiting the aleatoric uncertainty in surface normal estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13137–13146, 2021. 2

[4] Harry Barrow, J Tenenbaum, A Hanson, and E Riseman. Recovering intrinsic scene characteristics. *Comput. Vis. Syst*, 2(3-26):2, 1978. 2

[5] 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. *arXiv preprint arXiv:1709.06158*, 2017. 1, 2, 3, 5, 6, 7, 8

[6] 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. 1, 2

[7] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 1, 2

[8] Julian Chibane, Aymen Mir, and Gerard Pons-Moll. Neural unsigned distance fields for implicit function learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, December 2020. 1, 2, 7, 8

[9] 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 *European conference on computer vision*, pages 628–644. Springer, 2016. 1, 2

[10] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017. 1, 2, 5, 6, 8

[11] Angela Dai, Christian Diller, and Matthias Nießner. Sg-nn: Sparse generative neural networks for self-supervised scene completion of rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 849–858, 2020. 5

[12] Luca Del Pero, Joshua Bowdish, Bonnie Kermgard, Emily Hartley, and Kobus Barnard. Understanding bayesian rooms using composite 3d object models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013. 2

[13] Helisa Dhamo, Nassir Navab, and Federico Tombari. Object-driven multi-layer scene decomposition from a single image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5369–5378, 2019. 2

[14] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 2

[15] Sanja Fidler, Sven Dickinson, and Raquel Urtasun. 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In *Advances in neural information processing systems*, pages 611–619, 2012. 2

[16] David F Fouhey, Abhinav Gupta, and Martial Hebert. Data-driven 3d primitives for single image understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3392–3399, 2013. 2

[17] 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. *arXiv preprint arXiv:2011.09127*, 2020. 2, 5, 6, 7, 8

[18] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *European Conference on Computer Vision*, pages 484–499. Springer, 2016. 1, 2

[19] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9785–9795, 2019. 2

[20] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018. 2

[21] Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3d object reconstruction. In *2017 International Conference on 3D Vision (3DV)*, pages 412–420. IEEE, 2017. 2

[22] 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. 6

[23] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. In *2009 IEEE 12th international conference on computer vision*, pages 1849–1856. IEEE, 2009. 2

[24] Derek Hoiem, Alexei A Efros, and Martial Hebert. Geometric context from a single image. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 1, pages 654–661. IEEE, 2005. 2

[25] Theerasit Issaranon, Chuhang Zou, and David Forsyth. Counterfactual depth from a single rgb image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2

[26] Hamid Izadinia, Qi Shan, and Steven M Seitz. Im2cad. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5134–5143, 2017. 2

[27] Ziyu Jiang, Buyu Liu, Samuel Schulter, Zhangyang Wang, and Manmohan Chandraker. Peek-a-boo: Occlusion reasoning in indoor scenes with plane representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–121, 2020. 2

[28] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5580–5590, 2017. 2

[29] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[30] Nilesh Kulkarni, Ishan Misra, Shubham Tulsiani, and Abhinav Gupta. 3d-relnet: Joint object and relational network for 3d prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2212–2221, 2019. 2

[31] Lin Li, Salman Khan, and Nick Barnes. Silhouette-assisted 3d object instance reconstruction from a cluttered scene. In *ICCV Workshops*, 2019. 2

[32] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. *arXiv preprint arXiv:1706.07036*, 2017. 2

[33] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[34] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. *arXiv preprint arXiv:2008.02268*, 2020. 2

[35] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 1, 2

[36] 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, 3

[37] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 55–64, 2020. 2

[38] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 1, 2

[39] Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. On the uncertainty of self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3227–3237, 2020. 2

[40] 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 *Proceedings of the IEEE International Conference on Computer Vision*, pages 2304–2314, 2019. 1, 2

[41] Shunsuke Saito, Tomas Simon, Jason Saragih, and Hanbyul Joo. Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 84–93, 2020. 2

[42] Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):824–840, 2008. 2

[43] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 1, pages 519–528. IEEE, 2006. 5

[44] 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. 2, 5

[45] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33, 2020. 1, 2

[46] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1746–1754, 2017. 2

[47] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. Neuralrecon: Real-time coherent 3d reconstruction from monocular video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15598–15607, 2021. 5

[48] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2974–2983, 2018. 2

[49] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3405–3414, 2019. 5

[50] Shubham Tulsiani, Saurabh Gupta, David F Fouhey, Alexei A Efros, and Jitendra Malik. Factoring shape, pose, and layout from the 2d image of a 3d scene. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 302–310, 2018. 2

[51] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2635–2643, 2017. 2

[52] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3d scene inference via view synthesis. In *ECCV*, 2018. 2

[53] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, De-

nis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. 7

[54] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *ECCV*, 2018. 1

[55] Qiangeng Xu, Weiyue Wang, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. In *Advances in Neural Information Processing Systems*, pages 492–502, 2019. 2

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

[57] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 2