RGB-D Local Implicit Function for Depth Completion of Transparent Objects

Luyang Zhu[1,2]*   Arsalan Mousavian[2]   Yu Xiang[2]   Hammad Mazhar[2]

Jozef van Eenbergen[2]   Shoubhik Debnath[2]   Dieter Fox[1,2]

[1]University of Washington   [2]NVIDIA

Abstract

Majority of the perception methods in robotics require depth information provided by RGB-D cameras. However, standard 3D sensors fail to capture depth of transparent objects due to refraction and absorption of light. In this paper, we introduce a new approach for depth completion of transparent objects from a single RGB-D image. Key to our approach is a local implicit neural representation built on ray-voxel pairs that allows our method to generalize to unseen objects and achieve fast inference speed. Based on this representation, we present a novel framework that can complete missing depth given noisy RGB-D input. We further improve the depth estimation iteratively using a self-correcting refinement model. To train the whole pipeline, we build a large scale synthetic dataset with transparent objects. Experiments demonstrate that our method performs significantly better than the current state-of-the-art methods on both synthetic and real world data. In addition, our approach improves the inference speed by a factor of 20 compared to the previous best method, ClearGrasp [43]. Code will be released at https://research.nvidia.com/publication/2021-03_RGB-D-Local-Implicit.

1. Introduction

Depth data captured from RGB-D cameras has been widely used in many applications such as augmented reality and robot manipulation. Despite their popularity, commodity-level depth sensors, such as structured-light cameras and time-of-flight cameras, fail to produce correct depth for transparent objects due to the lack of light reflection. As a result, many algorithms utilizing RGB-D data cannot be directly applied to recognize transparent objects which are very common in household scenarios.

Previous works on estimating geometry of transparent objects are often studied under controlled settings [47, 51]. Recently, Li *et al.* [26] proposed a physically-based neu-
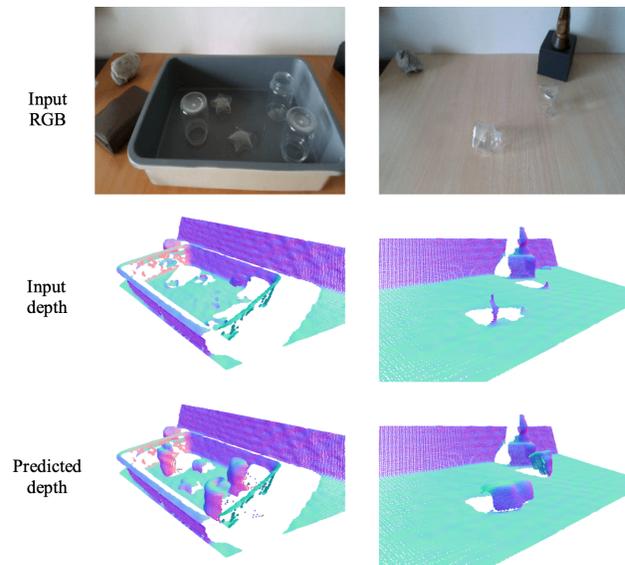

Figure 1. Our method can predict the depth of unseen transparent objects from a noisy RGB-D image. We back-project the depth map into the point cloud and render it in a novel viewpoint to better visualize the 3D shape. Zoom in to see details.

ral network to reconstruct 3D shape of transparent objects from multi-view images. Although their method is less restricted compared to previous ones, it still requires the environment map and the refractive index of transparent objects. ClearGrasp [43] achieves impressive results on depth completion of transparent objects. It first predicts masks, occlusion boundaries and surface normals from RGB images using deep networks, and then optimizes initial depth based on the network predictions. However, the optimization requires transparent objects to have contact edges with other non-transparent objects. Otherwise, the depth in transparent area becomes undetermined and can be assigned random value. In addition, it can not be deployed in real time applications due to the expensive optimization process.

In this paper, to overcome the limitations of existing works, we present a fast end-to-end framework for depth completion of transparent objects from a single RGB-D image. The core to our approach is a Local Implicit Depth Function (LIDF) defined on ray-voxel pairs consisting of

---

[1]Work done while author was an intern at NVIDIA.

camera rays and their intersecting voxels. The motivations for LIDF are: 1) The depth of a transparent object can be inferred from its color and the depth of its non-transparent neighborhood. In particular, color can provide useful visual cues for the 3D shape and curvature while local depth helps to reason about the spatial arrangement and location of transparent objects. 2) Directly regressing the complete depth map using a deep network can easily overfit to the objects and scenes in the training data. By learning at the local scale (a voxel in our case) instead of the whole scene, LIDF can generalize to unseen objects because different objects may share similar local structures. 3) Voxel grids provide a natural partition of the 3D space. By defining implicit function on ray-voxel pairs, we can significantly reduce the inference time as the model only needs to consider occupied voxels intersected by the camera ray. Based on these motivations, we present a model to estimate the depth of a pixel by learning the relationship between the camera ray and its intersecting voxels given the color and local depth information. To further utilize the geometry of transparent object itself, we propose a depth refinement model to update the prediction iteratively by combining the input RGB, input depth points and the predicted depth from LIDF. To train the whole pipeline, we create a large scale synthetic dataset, Omiverse Object dataset, using the NVIDIA Omniverse platform [1]. Our dataset provides over 60,000 images including both transparent and opaque objects in different scenes. The dataset is generated with diverse object models and poses, lighting conditions, camera viewpoints and background textures to close the sim-to-real gap. Experiments show that training on the Omniverse Object dataset can boost the performance for both our approach and competing methods in real-world testing cases.

Our approach is inspired by recent advances in neural radience field [34, 44, 27]. NeRF [34] can learn a continuous function of the 3D geometry and appearance of a scene, thus achieving accurate reconstruction and rendering results. However, NeRF has slow inference speed due to inefficient 3D points sampling. Our approach tackles this problem by querying intersecting voxels instead of 3D points along the ray. NSVF [27] also utilizes a sparse voxel grid to reduce the inference time, but it still needs to sample 3D points inside the voxel while our method directly learns the offset of a ray-voxel pair to obtain the possible terminating 3D location of the ray. Experiments demonstrate that learning offsets can produce better depth than sampling 3D points based on heuristic strategies. In addition, NeRF-based methods needs to train a network for every new scene to model the complex geometry and appearance whereas our method generalizes to unseen objects and scenes in depth completion of transparent objects.

Our contributions are summarized as follows: 1) We propose LIDF, a novel implicit representation defined on ray-voxel pairs, leading to fast inference speed and good generality. 2) We present a two-stage system, including networks to learn LIDF and a self-correcting refinement model, for depth completion of transparent objects. 3) We build a large scale synthetic dataset proved to be useful to transparent objects learning. 4) Our full pipeline is evaluated qualitatively and quantitatively, and outperform the current state-of-the-art in terms of accuracy and speed.

2. Related Work

Depth estimation. Depth estimation can be classified into three categories based on the input. Several methods have been proposed to directly regress the depth map from the color image using convolutional neural networks [12, 41, 25, 7, 14, 15, 17]. Most of them are trained on large scale datasets generated from RGB-D cameras, thus they can only reproduce the raw depth scan. Our method, on the contrary, focuses on the depth estimation for transparent objects where depth sensor typically fails. Another line of related work explores the task of depth completion given RGB images and sparse sets of depth measurements [31, 8, 40, 50, 10, 35]. These works improve the depth estimation over color-only methods, but they still produce low quality results because of limited information provided by sparse depth. Our method falls into the third category which tries to complete depth maps given noisy RGB-D images. Barron and Malik [3] propose a joint optimization for intrinsic images. Firman *et al.* [13] predict unobserved voxels from a single depth image using the voxlet representation. Matsuo and Aoki [32] reconstruct depths by ray-tracing to estimated local tangents. Recent works [53, 43] estimate surface normals and occlusion boundaries only from color images using deep networks and solve a global optimization based on those predictions as well as observed depths. The optimization is very slow and produces bad results if the network predictions are not accurate. We address these limitations by learning a implicit function using color and local depth jointly. Experiment shows that our method can achieve better results and $20\times$ speedup compared to [43].

Transparent objects. Transparent objects have been studied in various computer vision tasks, including object pose estimation [24, 30, 29, 38, 28], 3D shape reconstruction [2, 18, 39, 26, 43] and segmentation [21]. However, most of these works assume known background patterns [18, 39], known object 3D models [24, 30, 38], or multi view/stereo input [28, 26]. Our approach does not require any priors and can estimate the depth of transparent objects from a single view RGB-D image. Sajjan *et al.* [43] is the closest work to ours. However, they pretrain their networks on out-of-domain real datasets while our method is trained purely on synthetic data but achieves better performance.
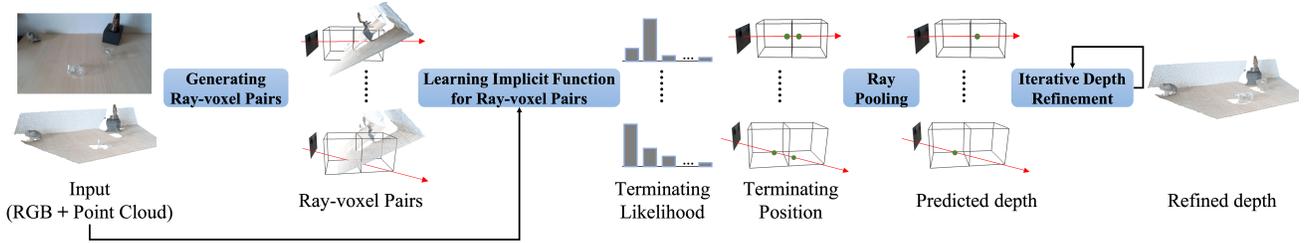
Figure 2. Pipeline overview. Our method takes as input RGB-D images with missing depth for transparent objects, and predicts the full depth map. We first generate ray-voxel pairs by finding all occupied voxels intersected by a ray. Then we learn a local implicit function for each pair to estimate the terminating probability and position of the ray inside the voxel. The prediction of all pairs along the ray are accumulated by our ray pooling module to get the initial depth prediction. Finally, we improve the result by iterative depth refinement.

Implicit function learning. Our method takes inspiration from the implicit neural models [9, 33, 36, 16, 42, 49, 11], which are not restricted by topology and can represent the 3D surface continuously. Most of these methods [9, 33, 36] can not scale to scene level as they only use a single latent vector to encode the shape. Voxel-based implicit representations [5, 20, 37] address this limitation by learning local shape priors. However, they require coarse voxel grids or sparse point clouds as input, which already provide some information about 3D shapes. Our task is more challenging as the point cloud of transparent objects is totally missing.

Recently, Mildenhall *et al.* [34] propose to represent scenes as neural radiance fields, achieving impressive results on novel view synthesis. However, their method needs to be retrained for every new scene and takes a long time to render one image. Schwarz *et al.* [44] solves the first drawback by learning a generative radiance field, but their results are limited to single object or human face. Liu *et al.* [27] tackles the second problem by introducing sparse voxel grids, but they still need to sample points based on heuristic strategies while our method learns to estimate the possible terminating position for ray-voxel pairs.

3. Method

Figure 2 provides the overview of our proposed end-to-end system for depth completion of transparent objects. Our approach takes as input a RGB image $\mathcal{I} \in \mathbb{R}^{H \times W \times 3}$, an incomplete depth image $\mathcal{D} \in \mathbb{R}^{H \times W}$ and camera intrinsics $K \in \mathbb{R}^{3 \times 3}$, and predicts the full depth map $\hat{\mathcal{D}} \in \mathbb{R}^{H \times W}$, where H and W are height and width of the input image.

Our method consists of two stages: The first stage learns LIDF defined on ray-voxel pairs to represent the local geometry of the scene, which is done in three steps. First, ray-voxel pairs are generated by marching a camera ray through the pixel to find occupied voxels intersected by the ray (Section 3.1). Then networks are trained end-to-end to learn the terminating probability and position of the ray inside the voxel given RGB embedding, voxel embedding and ray embedding (Section 3.2). Finally, predictions of all pairs along the ray are accumulated by a ray pooling module to get the

depth estimation (Section 3.3). Voxel embedding in the first stage does not encode the geometry of transparent objects due to the missing depth. The second stage of our method (Section 3.4) presents a refinement model to solve this problem. The refinement model first recomputes the voxel and ray embeddings based on input and predicted depth points. After that, it refines the depth estimation by combining the new voxel and ray embeddings with the original RGB embedding. The formulation of the refinement model allows for iteratively applying fine adjustments that preserve common 3D structures such as shapes of the objects, how the objects are lying on the table, and objects following the direction of gravity. Experiments demonstrate the generalization power of the depth refinement model on unseen objects in the real world.

3.1. Generating Ray-voxel Pairs

To predict the missing depth of a pixel, we first need to find all the occupied voxels intersected by its camera ray. Figure 3 provides different steps for the ray-voxel pair generation. We start by dividing a fixed workspace into an axis-aligned voxel grid $\mathbf{V}$ in camera coordinates, where our system only completes the depth for any object inside. After that, the input depth map $\mathcal{D}$ is back-projected into an "organized" point cloud $P \in \mathbb{R}^{H \times W \times 3}$ using the camera intrinsics K. We filter out points with zero depth to get the valid point cloud $P^{\mathrm{valid}} \subset P$. A voxel is considered as occupied if it has at least one valid depth point. Based on this definition, the set of occupied voxels $\mathbf{V}^{\mathrm{occ}}$ can be computed by doing a boundary test between P^{valid} and all voxel cells $v \in \mathbf{V}$. Finally, we shoot a camera ray r_i through the i-th pixel $\mathcal{I}_i$ and apply the ray-AABB intersection test [22] between r_i and $\mathbf{V}^{\mathrm{occ}}$ using a customized CUDA kernel which allows for real time computation.

The generated ray-voxel pairs (Figure 3 (d)) are denoted by $\Phi_{ij} = (r_i, v_j^{\mathrm{occ}}, d_{ij}^{\mathrm{in}}, d_{ij}^{\mathrm{out}})$, where r_i is the ray direction for the i-th pixel $\mathcal{I}_i$, v_j^{occ} is the voxel intersected by the ray r_i with index j in $\mathbf{V}^{\mathrm{occ}}$, d_{ij}^{in} and d_{ij}^{out} are the position of the entering and leaving intersection point respectively.

(a) Full voxel grid   (b) Occupied voxels

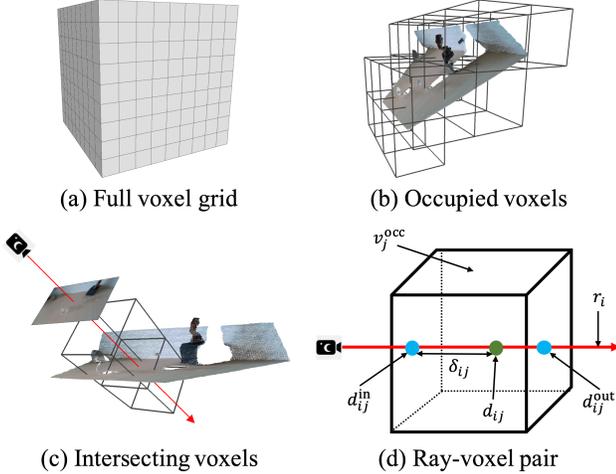(c) Intersecting voxels   (d) Ray-voxel pair

Figure 3. Ray-voxel pair generation. (a) Predefined voxel grid. (b) Occupied voxels. (c) Intersecting voxels along the ray. (d) Variables defined for the ray-voxel pair.

3.2. Learning LIDF for Ray-voxel Pairs

LIDF is defined on ray-voxel pairs to reason about the relationship between camera rays and their intersecting voxels. Specifically, we want to estimate the terminating probability and position of the camera ray inside the intersecting voxel given the following inputs: 1) RGB embedding corresponding to the pixel of the camera ray; 2) Voxel embedding computed from valid points inside the intersecting voxel; 3) Ray embedding which encodes the high frequency information of the ray direction and ray-voxel intersection points. As illustrated in Figure 4, the learning process of LIDF for the ray-voxel pair Φ_{ij} can be formulated as:

$$p_{ij}^{\text{end}} = \mathcal{F}_{\text{prob}}(\mathcal{H}_{ij}) \tag{1}$$

$$d_{ij} = d_{ij}^{\text{in}} + \delta_{ij} \cdot r_i, \delta_{ij} = \mathcal{F}_{\text{pos}}(\mathcal{H}_{ij}) \tag{2}$$

$$\mathcal{H}_{ij} = \mathcal{H}_i^{\text{rgb}} \oplus \mathcal{H}_j^{\text{vox}} \oplus \gamma(r_i) \oplus \gamma(d_{ij}^{in}) \oplus \gamma(d_{ij}^{out}), \tag{3}$$

where p_{ij}^{end} is the probability of the ray r_i terminating in the voxel v_j^{occ}; d_{ij} is the terminating position inside v_j^{occ} for r_i; δ_{ij} is the distance from d_{ij}^{in} to d_{ij} along the ray; $\mathcal{H}_{ij}$ is the input feature embedding for Φ_{ij}; $\mathcal{H}_i^{\text{rgb}}$ is the RGB embedding for pixel $\mathcal{I}_i$; $\mathcal{H}_j^{\text{vox}}$ is the voxel embedding for v_j^{occ}; $\gamma(\cdot)$ is the high frequency positional encoding function [34]; $\oplus$ is the concatenation operator for feature vectors. $\mathcal{F}_{\text{prob}}$ is parametrized as a MLP while $\mathcal{F}_{\text{pos}}$ is parametrized as a separate MLP with iterative error feedback [4].

RGB Embedding. The RGB embedding $\mathcal{H}^{\text{rgb}}$ encodes the color information of all pixels. Given the input image $\mathcal{I}$, we first apply ResNet34-8s network [48] Ψ_{rgb} to get a dense feature map $\Psi_{\text{rgb}}(\mathcal{I}) \in \mathbb{R}^{H \times W \times C_r}$, where H is the image height, W is the image width and C_r is the dimension of the dense feature map. To compute the RGB Embedding
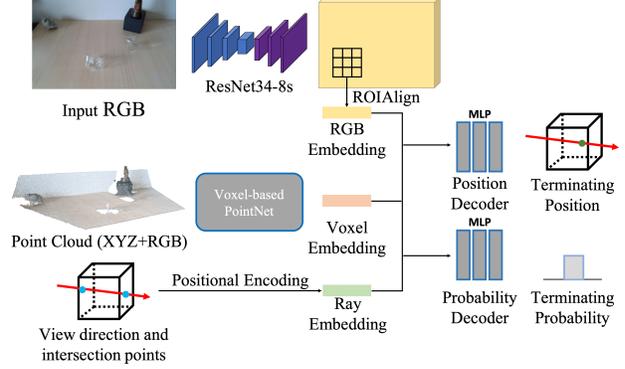

Figure 4. LIDF for each ray-voxel pair. Our proposed implicit function queries the termination probability and position given the RGB, voxel and ray embeddings.

$\mathcal{H}_i^{\text{rgb}}$ for the i-th pixel $\mathcal{I}_i$, we use ROIAlign [19] to pool features from a local patch of $\Psi_{\text{rgb}}(\mathcal{I})$ centered at $\mathcal{I}_i$. We set $C_r = 32$ in our experiments. For ROIAlign, the size of input ROI is 8×8 and the size of output feature map is 2×2. We directly flatten the output feature map of ROIAlign to get the final RGB embedding.

Voxel Embedding. The voxel embedding $\mathcal{H}^{\text{vox}}$ encodes the geometry information for occupied voxels $\mathbf{V}^{\text{occ}}$. Inspired by [52], we propose a two stage voxel-based PointNet encoder, which can better fuse the global and local geometry inside a voxel. Unlike [52] who applies maxpooling to all points to get a global embedding, our network computes per-voxel embeddings by maxpooling points inside each voxel. Please see supplementary for details.

3.3. Ray Pooling

Given the predicted terminating probability and position for each ray-voxel pair, we need to estimate the depth of ray r_i. We achieve this by applying a pooling operation to all pairs containing r_i. There are two natural choices for the pooling: argmax and weighted sum. The argmax operator computes the depth of r_i as the depth estimation of the ray-voxel pair with the maximal ray terminating probability:

$$d_i = d_{ij'}, j' = \underset{j:v_j^{\text{occ}} \in \mathbf{V}_{r_i}^{\text{occ}}}{\operatorname{argmax}} p_{ij}^{\text{end}}, \tag{4}$$

where d_i is the predicted terminating position of ray r_i; j' is the index of the occupied voxel that has the largest terminating probability; $\mathbf{V}_{r_i}^{\text{occ}}$ represents all the occupied voxels intersected by ray r_i.

The weighted sum operator computes the depth of r_i as the sum of depth estimation of all pairs along the ray weighted by their terminating probabilities:

$$d_i = \sum_{j:v_j^{\text{occ}} \in \mathbf{V}_{r_i}^{\text{occ}}} d_{ij} p_{ij}^{\text{end}}. \tag{5}$$
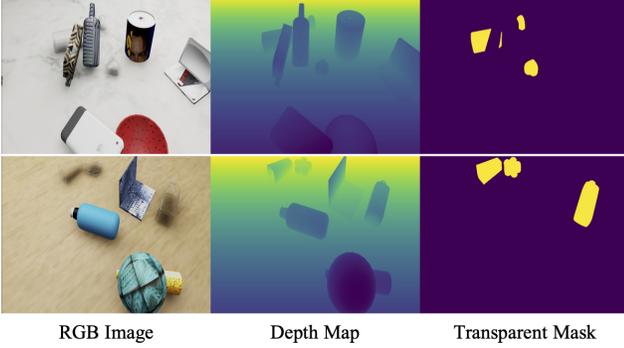
RGB Image   Depth Map   Transparent Mask

Figure 5. Examples from our Omniverse Object Dataset.

Our experiments, in Section 4.2, show that argmax pooling leads to better accuracy in depth prediction because weighted sum can introduce noise brought by voxels that are far away from the ground truth even if their terminating probabilities are low.

3.4. Depth Refinement Model

Since the real depth scan around transparent objects can be very noisy or totally missing depth values, the voxel embedding in those areas can encode neither the geometry of transparent objects nor the spatial arrangement between transparent objects and non-transparent ones. To overcome this issue, we propose a self-correcting model to refine the depth estimation progressively. For each iteration, the refinement model recomputes the voxel embedding using input valid points as well as predicted missing points from last iteration, and estimates the correction offset. The forward pass for the k-th iteration ($k \geq 1$) can be formulated as:

$$\hat{d}_i[k] = \hat{d}_i[k-1] + \hat{\delta}_i[k] \cdot r_i \tag{6}$$

$$\hat{\delta}_i[k] = \hat{\mathcal{F}}_{\text{pos}}(\hat{\mathcal{H}}_i[k]) \tag{7}$$

$$\hat{\mathcal{H}}_i[k] = \mathcal{H}_i^{\text{rgb}} \oplus \hat{\mathcal{H}}_{j_{k-1}}^{\text{vox}}[k] \oplus \gamma(r_i) \oplus \gamma(\hat{d}_i[k-1]), \tag{8}$$

where $\hat{d}_i[k]$ is the terminating position of ray r_i under k-th iteration and we set $\hat{d}_i[0] = d_i$; $\hat{\delta}_i[k]$ is the offset along the ray from $\hat{d}_i[k-1]$ to $\hat{d}_i[k]$; $\hat{\mathcal{H}}_i[k]$ is the input feature embedding for ray r_i under k-th iteration; j_{k-1} is the index for the occupied voxel containing position $\hat{d}_i[k-1]$; $\hat{\mathcal{H}}_{j_{k-1}}^{\text{vox}}[k]$ is the voxel embedding under k-th iteration for voxel $v_{j_{k-1}}^{\text{occ}}$. We compute $\hat{\mathcal{H}}_{j_{k-1}}^{\text{vox}}[k]$ by sending both input valid points and predicted missing points inside $v_{j_{k-1}}^{\text{occ}}$ into our proposed PointNet encoder. $\hat{\mathcal{F}}_{\text{pos}}$ is parametrized as a MLP with iterative error feedback. It is worth noting that all networks in the refinement model do not share weights with the first stage networks. Refinement networks can be applied iteratively to improve the quality of the depth. We apply refinement networks for 2 iterations and refining more than 2 iterations results in marginal improvement of depth quality.

3.5. Loss Functions

Our system is trained using the following loss:

$$\mathcal{L} = \omega_{\text{pos}}\mathcal{L}_{\text{pos}} + \omega_{\text{prob}}\mathcal{L}_{\text{prob}} + \omega_{\text{sn}}\mathcal{L}_{\text{sn}}, \tag{9}$$

where $\mathcal{L}_{\text{pos}}$ is the L1 loss between the predicted and groundtruth depth; $\mathcal{L}_{\text{prob}}$ is the cross entropy loss for terminating probabilities of all intersecting voxels along the ray; $\mathcal{L}_{\text{sn}}$ is the cosine distance of surface normals computed from predicted and groundtruth point cloud respectively, which can regularize the network to learn meaningful object shapes; ω_{pos}, ω_{prob} and ω_{sn} are weights for different losses.

3.6. Implementation Details

We set image resolution to 320×240 in all experiments. To simulate the noise pattern of real depth scans on our synthetic training data, we remove all depth values for transparent objects, part of the depth values for opaque objects, and create some random holes in the depth map for background. We also augment the color input by adding pixel noise, motion blur and random noise in HSV space. For training, our method only predicts depth for corrupted pixels mentioned above. For testing, depth values of all pixels are estimated so that our method does not rely on the segmentation of transparent objects. Our two stage networks are trained separately. The first stage networks are trained for 60 epochs using Adam optimizer [23] with a fixed learning rate of 0.001. After that, the first stage networks are frozen and we train the refinement networks for another 60 epochs using Adam optimizer. The first 30 epochs are trained with learning rate 0.001 using the proposed loss function. The later 30 epochs are trained with learning rate 0.0001 using hard negative mining, where only top 10 percent of pixels with largest errors are considered. We set $\omega_{\text{pos}} = 100$ and $\omega_{\text{sn}} = 10$ for the first stage and first 30 epochs of the second stage. For the later 30 epochs of second stage, we set $\omega_{\text{pos}} = 20$ and $\omega_{\text{sn}} = 2$. ω_{prob} is set to 0.5 for the first stage and 0 for the depth refinement model.

4. Experiments

Datasets Our full pipeline is trained on the ClearGrasp dataset [43] and a new dataset we generated using the Omniverse Platform [1], which we call Omniverse Object dataset. The dataset provides various supervisions for transparent and opaque objects in cluttered scenes. Figure 5 visualizes some examples of the dataset. 3D object models in Omniverse Object dataset are collected from ClearGrasp and ShapeNet [6]. To get natural poses of objects, we use NVIDIA PhysX engine to simulate objects falling to the ground. Then we randomly select some objects and set their materials to the glass. We also augment the data by changing textures for the ground and opaque objects, lighting conditions and camera views. See supplementary for

more details about Omniverse Object dataset. The evaluation is done on the ClearGrasp dataset [43]. It has 4 types of different testing data: Synthetic images of 5 training objects (Syn-known); Synthetic images of 4 novel objects (Syn-novel); Real world images of 5 training objects (Real-known); Real world images of 5 novel objects (Real-novel), 3 of them are not present in synthetic data.

Metrics We closely follow the evaluation protocal of [43] for the depth estimation. The prediction and groundtruth are first resized to 144×256 resolution, then we compute errors in the transparent objects area using the following metrics:

Root Mean Squared Error (RMSE): $\sqrt{\frac{1}{|\hat{\mathcal{D}}|} \sum_{d \in \hat{\mathcal{D}}} ||d - d^*||^2}$

Absolute Relative Difference (REL): $\frac{1}{|\hat{\mathcal{D}}|} \sum_{d \in \hat{\mathcal{D}}} |d - d^*|/d^*$

Mean Absolute Error (MAE): $\frac{1}{|\hat{\mathcal{D}}|} \sum_{d \in \hat{\mathcal{D}}} |d - d^*|$

Threshold: % of d_i satisfying $\max(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}) < \delta$

The first 3 metrics are computed in meters. For the threshold, δ is set to 1.05, 1.10 and 1.25.

4.1. Comparison to State-of-the-art Methods

We compare our approach to several state-of-the-art methods in Table 1. For fair comparison, we evaluate all related works using their released checkpoints (denoted by method name) as well as retraining on our data (denoted by method name with subscript ours. All baselines are trained on both datasets together which is the same setting as our proposed method). RGBD-FCN$_{ours}$ is a strong baseline proposed by ourselves. It directly regresses depth maps using fully convolutional networks from RGB-D images. We use Resnet34-8s [48] as the network architecture and train the network on our data. NLSPN [35] is the state-of-the-art method for depth completion on NYUV2 [45] and KITTI [46] dataset. Cleargrasp [43] is the state-of-the-art method for depth completion of transparent objects. For our approach, we use the best model: LIDF plus the depth refinement model. Our method achieves the best result on all datasets even when baseline methods are trained on the same data. It also shows that training on Omniverse Object dataset can boost the performance of baseline methods. In Figure 6, we provide qualitative comparison by rendering point cloud in a novel view. Our approach can generate more meaningful depth than baseline methods.

Inference speed is an important factor for deploying the model in real applications. We compare the inference time of our method to the best performing baseline Clear-Grasp [43]. While our method provides an end-to-end solution to complete the depth map of the scene, ClearGrasp uses a two-stage approach where the first stage uses CNN to predict some intermediate observations and the second stage solves an expensive optimization to recover the missing depths. In order to compare the runtime fairly, we run both methods on a machine equipped with a single P100 GPU and compute the average time over the whole testing

set. Our method takes **0.09** second for one image while ClearGrasp needs to spend **1.82** second, showing that our method is **20x** faster.

Methods	RMSE↓	REL↓	MAE↓	$\delta_{1.05}$ ↑	$\delta_{1.10}$ ↑	$\delta_{1.25}$ ↑
	Cleargrasp Syn-known					
RGBD-FCN$_{ours}$	0.028	0.039	0.021	76.53	91.82	99.00
NLSPN [35]	0.136	0.231	0.113	19.02	35.95	70.43
NLSPN$_{ours}$ [35]	0.026	0.041	0.021	74.89	89.95	98.59
CG [43]	0.041	0.055	0.031	69.43	89.17	96.74
CG$_{ours}$ [43]	0.034	0.045	0.026	73.53	92.68	98.25
Ours	**0.012**	**0.017**	**0.009**	**94.79**	**98.52**	**99.67**
	Cleargrasp Syn-novel					
RGBD-FCN$_{ours}$	0.033	0.058	0.028	52.40	85.64	98.94
NLSPN [35]	0.132	0.239	0.106	16.25	32.13	64.78
NLSPN$_{ours}$ [35]	0.029	0.049	0.024	64.83	88.20	98.57
CG [43]	0.044	0.074	0.038	41.37	79.20	97.29
CG$_{ours}$ [43]	0.037	0.062	0.032	50.27	84.00	98.39
Ours	**0.028**	**0.045**	**0.023**	**68.62**	**89.10**	**99.20**
	Cleargrasp Real-known					
RGBD-FCN$_{ours}$	0.054	0.087	0.048	36.32	67.11	96.26
NLSPN [35]	0.149	0.228	0.127	14.04	26.67	54.32
NLSPN$_{ours}$ [35]	0.056	0.086	0.048	40.60	67.68	96.25
CG [43]	0.039	0.051	0.029	72.62	86.96	95.58
CG$_{ours}$ [43]	0.032	0.042	0.024	74.63	90.69	98.33
Ours	**0.028**	**0.033**	**0.020**	**82.37**	**92.98**	**98.63**
	Cleargrasp Real-novel					
RGBD-FCN$_{ours}$	0.042	0.070	0.037	42.45	75.68	99.02
NLSPN [35]	0.145	0.240	0.123	13.77	25.81	51.59
NLSPN$_{ours}$ [35]	0.036	0.059	0.030	51.97	84.82	99.52
CG [43]	0.034	0.045	0.025	76.72	91.00	97.63
CG$_{ours}$ [43]	0.027	0.039	0.022	**79.5**	93.00	99.28
Ours	**0.025**	**0.036**	**0.020**	76.21	**94.01**	**99.35**

Table 1. Quantitative Comparison to state-of-the-art methods. ↓ means lower is better, ↑ means higher is better. Please refer to the text for more details.

4.2. Ablation Studies

In this section, we first evaluate the effect of the depth refinement model. After that, we compare several configurations for our first stage networks. To focus on the generalization ability, we only report quantitative results on Clear-Grasp Real-novel dataset. Please refer to the supplementary for results on other testing data.

Depth refinement model. To demonstrate the effectiveness of the depth refinement model, we run two experiments with and without refinement. Table 2 shows that adding depth refinement can boost the performance on real unseen objects. In Figure 7, we can see that refinement model can correct the tilted error from the non-refinement prediction.

Input Modalities. Our method uses combination of RGB and depth information to complete the depth map. To evaluate the contribution of each input modality, we retrained
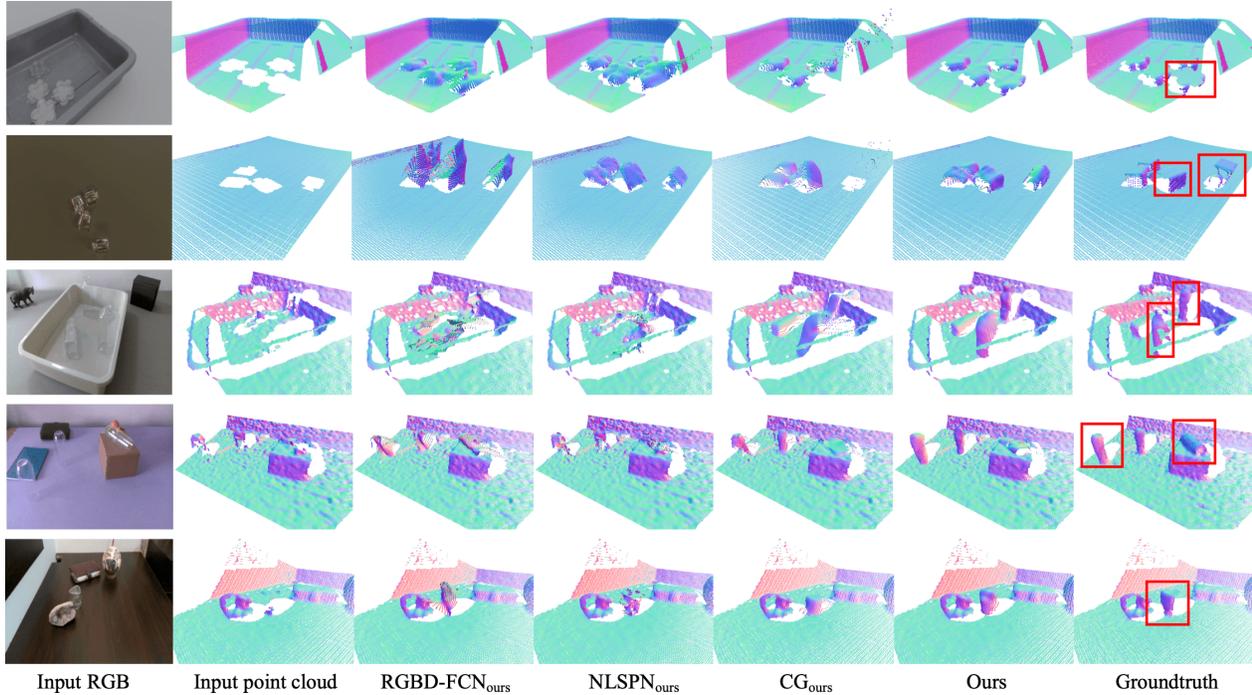
Figure 6. Qualitative comparison to state-of-the-art methods. For all the baselines, we provide visualizations of the version retrained on our data for fair comparison. The point clouds are colored by surface normal and rendered in a novel viewpoint to better visualize the 3D shape. Red boxes in the groundtruth highlight areas where our method performs much better than all baselines. Zoom in to see details.

Refinement	RMSE↓	REL↓	MAE↓	$\delta_{1.05}$ ↑	$\delta_{1.10}$ ↑	$\delta_{1.25}$ ↑
×	0.028	0.043	0.023	65.17	92.23	99.30
✓	**0.025**	**0.036**	**0.020**	**76.21**	**94.01**	**99.35**

Table 2. Depth refinement model. × denotes prediction from first stage while ✓ denotes prediction from refinement model.

RGB	Depth	RMSE↓	REL↓	MAE↓	$\delta_{1.05}$ ↑	$\delta_{1.10}$ ↑	$\delta_{1.25}$ ↑
✓	✓	**0.028**	**0.043**	**0.023**	**65.17**	**92.23**	**99.30**
	✓	0.052	0.084	0.043	39.14	66.49	97.03
✓		0.062	0.096	0.054	35.02	55.98	93.69

Table 3. Ablation studies for effect of different modalities

the first stage model using only one of modalities. Table 3 shows how much each modality contributes to the accuracy of the predicted depth. RGB alone becomes similar to depth estimation from single image and it cannot predict the depth accurately. In addition, depth alone is not able to predict depth for transparent objects well because transparent objects are not observed in depth and without RGB information the model is agnostic of transparent objects.

Ray Info	Pos. Enc	# Vox	RMSE↓	REL↓	MAE↓	$\delta_{1.05}$ ↑	$\delta_{1.10}$ ↑	$\delta_{1.25}$ ↑
✓	✓	8^3	**0.028**	**0.043**	**0.023**	**65.17**	**92.23**	99.30
	N/A	8^3	0.050	0.064	0.035	57.88	80.29	95.87
✓		8^3	0.032	0.050	0.026	60.96	83.86	99.21
✓	✓	4^3	0.030	0.046	0.025	64.00	88.96	**99.66**
✓	✓	16^3	0.038	0.058	0.031	56.78	81.49	96.21

Table 4. Ablation Study for different design choices such as including ray information in the embedding, applying positional encoding, and the number of voxels on the accuracy

Ray Information. To further highlight the effectiveness of our ray-voxel formulation, we compared our method with the variation where RGB embedding and voxel embedding are concatenated with each other and there is no information about the ray (ray-voxel intersection points and ray direction) provided to the model. Table 4 shows that ray information provides crucial guidance to the model to predict the depth effectively.

Positional encoding. Positional encoding was shown to be quite important in [34] to capture high frequency details. We also conducted an experiment by disabling the positional encoding. Table 4 shows that positional encoding is making significant changes in $\delta_{1.05}$ and $\delta_{1.10}$ which reflect the ability of the model to capture fine details.

Number of Voxels. We explore the sensitivity of our method to the number of voxels. Smaller number of voxels makes ray termination classification easier but regression for the offset more challenging while larger number of voxels leads to the opposite. Table 4 shows that if we make the number of voxels too large the performance drops more significantly compared to the smaller number of voxels.

Training Data. We analyze the effects of training data in Table 5. We find training our method purely on ClearGrasp or Omniverse leads to similar results, but training on both datasets can improve the performance a lot. This indicates that Omniverse dataset can be a good complementary to
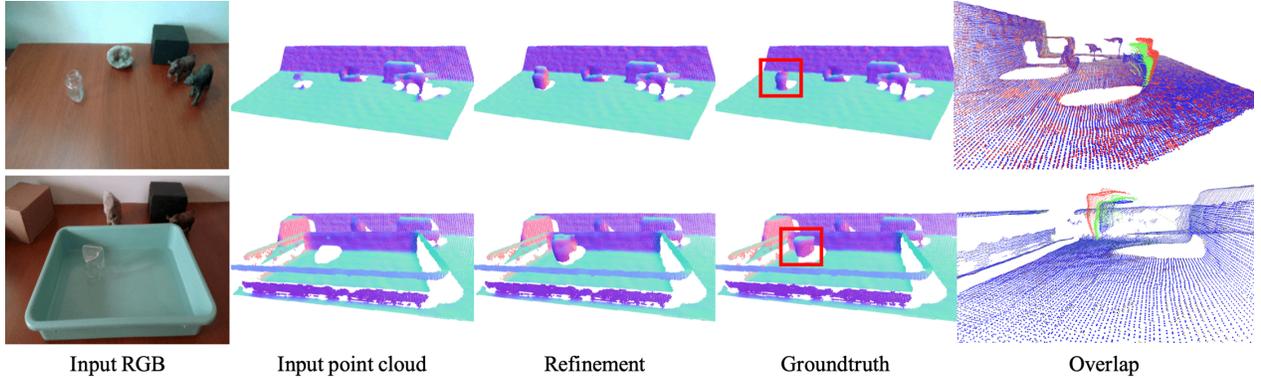
Figure 7. Examples of the depth refinement model. The point clouds (column 2-4) are colored by surface normal and rendered in a novel viewpoint to better visualize the 3D shape. The red box in the Groundtruth highlights the interest area. The last column visualizes the overlap of point clouds for non-refinement (red), refinement (green) and groundtruth (blue). Please zoom in to see details.

Data	RMSE↓	REL↓	MAE↓	$\delta_{1.05}$ ↑	$\delta_{1.10}$ ↑	$\delta_{1.25}$ ↑
CG+Omni	**0.028**	**0.043**	**0.023**	**65.17**	**92.23**	99.30
Omni	0.037	0.057	0.030	52.82	85.16	**99.43**
CG	0.038	0.057	0.031	58.95	80.13	96.87

Table 5. Quantitative effect of training data on the generalization to real novel objects.

ClearGrasp for transparent objects learning.

Ray Pooling. Table 6 provides the comparison for two types of pooling operators described in section 3.3. The argmax operator performs better than weighted sum operator on real-novel objects. We argue it is mainly because weighted sum operator suffers from the noise introduced by voxels far away from the ground truth. Figure 8 provides the accuracy histogram of the terminating voxel classification for the argmax operator. We can see that 84.77% of pixels are correctly classified and 97.71% of pixels are classified within ±1 voxel. We further compute the distance from the ground truth to the misclassfied ±1 voxels. The mean distance from the ground truth position to the leaving (entering) voxel position along the ray for −1 (+1) voxel is 0.064 (0.048) meter, indicating that groundtruth is very close to the misclassified −1 (+1) voxel. This implies that erroneous ending voxel classification will not hurt the performance much for the argmax operator.

Ray Pooling	RMSE↓	REL↓	MAE↓	$\delta_{1.05}$ ↑	$\delta_{1.10}$ ↑	$\delta_{1.25}$ ↑
Argmax	**0.028**	**0.043**	**0.023**	**65.17**	**92.23**	**99.30**
WeightedSum	0.032	0.051	0.027	59.31	84.25	98.34

Table 6. Ablation study for different ray pooling strategies.

Candidate points selection. To get the depth of the camera ray, our method learns the offset inside each voxel and apply argmax ray pooling while NeRF [34] tries to densely sample points and apply volume rendering equation. We slightly modify NeRF by sampling points only inside intersecting voxels. Table 7 shows that learning the position of candidate points is better than heuristic sampling strategy.
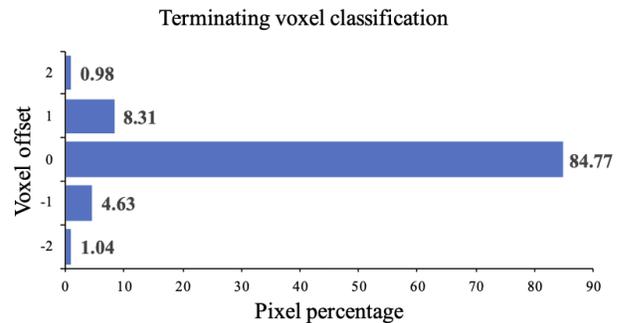

Figure 8. Distribution of the terminating voxel classification. Vertical axis represents the offset from the groundtruth to the predicted terminating voxel along the ray. For example, +1(-1) represents the predicted terminating voxel is 1 voxel behind (before) the groundtruth along the ray. Horizontal axis represents the percentage of pixels falls into that category.

Candidates	RMSE↓	REL↓	MAE↓	$\delta_{1.05}$ ↑	$\delta_{1.10}$ ↑	$\delta_{1.25}$ ↑
Learned offset	**0.028**	**0.043**	**0.023**	**65.17**	**92.23**	**99.30**
Sample points	0.038	0.058	0.031	54.09	79.48	98.28

Table 7. Ablation study for candidate points selection.

5. Conclusion

We have presented a novel framework for depth completion of transparent objects. Our method consists of a local implicit depth function defined on ray-voxel pairs and an iterative depth refinement model. We also introduce a large scale synthetic dataset for transparent objects learning, which can boost the performance for both our approach and other competing methods. Our pipeline is only trained on synthetic datasets but can generalize well to real world scenarios. We thoroughly evaluated our method compared to prior art and ablation baselines. Both quantitative and qualitative results demonstrate substantial improvements over the state-of-the-art in terms of accuracy and speed.

References

[1] NVIDIA Omniverse Platform. https://developer.nvidia.com/nvidia-omniverse-platform.

[2] Sven Albrecht and Stephen Marsland. Seeing the unseen: Simple reconstruction of transparent objects from point cloud data. In *Robotics: Science and Systems*, 2013.

[3] Jonathan T Barron and Jitendra Malik. Intrinsic scene properties from a single rgb-d image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17–24, 2013.

[4] Joao Carreira, Pulkit Agrawal, Katerina Fragkiadaki, and Jitendra Malik. Human pose estimation with iterative error feedback. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4733–4742, 2016.

[5] Rohan Chabra, Jan Eric Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3D reconstruction. In *European Conference on Computer Vision (ECCV)*, 2020.

[6] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[7] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 730–738, 2016.

[8] Yun Chen, Bin Yang, Ming Liang, and Raquel Urtasun. Learning joint 2d-3d representations for depth completion. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 10023–10032, 2019.

[9] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5939–5948, 2019.

[10] Xinjing Cheng, Peng Wang, and Ruigang Yang. Depth estimation via affinity learned with convolutional spatial propagation network. In *European Conference on Computer Vision (ECCV)*, pages 103–119, 2018.

[11] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6970–6981, 2020.

[12] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in NeuralInformation Processing Systems (NeurIPS)*, pages 2366–2374, 2014.

[13] Michael Firman, Oisin Mac Aodha, Simon Julier, and Gabriel J Brostow. Structured prediction of unobserved voxels from a single depth image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5431–5440, 2016.

[14] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2002–2011, 2018.

[15] Ravi Garg, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision (ECCV)*, pages 740–756, 2016.

[16] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *IEEE International Conference on Computer Vision (ICCV)*, pages 7154–7164, 2019.

[17] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 270–279, 2017.

[18] Kai Han, Kwan-Yee K Wong, and Miaomiao Liu. A fixed viewpoint approach for dense reconstruction of transparent objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4001–4008, 2015.

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.

[20] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, and Thomas Funkhouser. Local implicit grid representations for 3d scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[21] Agastya Kalra, Vage Taamazyan, Supreeth Krishna Rao, Kartik Venkataraman, Ramesh Raskar, and Achuta Kadambi. Deep polarization cues for transparent object segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[22] Timothy L Kay and James T Kajiya. Ray tracing complex scenes. *ACM SIGGRAPH computer graphics*, 20(4):269–278, 1986.

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

[24] Ulrich Klank, Daniel Carton, and Michael Beetz. Transparent object detection and reconstruction on a mobile platform. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5971–5978, 2011.

[25] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *International Conference on 3D Vision (3DV)*, pages 239–248, 2016.

[26] Zhengqin Li, Yu-Ying Yeh, and Manmohan Chandraker. Through the looking glass: Neural 3d reconstruction of transparent shapes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[27] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *Advances in NeuralInformation Processing Systems (NeurIPS)*, 33, 2020.

[28] Xingyu Liu, Rico Jonschkowski, Anelia Angelova, and Kurt Konolige. Keypose: Multi-view 3d labeling and keypoint

estimation for transparent objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[29] Ilya Lysenkov, Victor Eruhimov, and Gary Bradski. Recognition and pose estimation of rigid transparent objects with a kinect sensor. *Robotics*, 273:273–280, 2013.

[30] Ilya Lysenkov and Vincent Rabaud. Pose estimation of rigid transparent objects in transparent clutter. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 162–169, 2013.

[31] Fangchang Ma and Sertac Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2018.

[32] Kiyoshi Matsuo and Yoshimitsu Aoki. Depth image enhancement using local tangent plane approximations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3574–3583, 2015.

[33] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4460–4470, 2019.

[34] 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. In *European Conference on Computer Vision (ECCV)*, 2020.

[35] Jinsun Park, Kyungdon Joo, Zhe Hu, Chi-Kuei Liu, and In So Kweon. Non-local spatial propagation network for depth completion. In *European Conference on Computer Vision (ECCV)*, 2020.

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

[37] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision (ECCV)*, 2020.

[38] Cody J Phillips, Matthieu Lecce, and Kostas Daniilidis. Seeing glassware: from edge detection to pose estimation and shape recovery. In *Robotics: Science and Systems (RSS)*, volume 3, 2016.

[39] Yiming Qian, Minglun Gong, and Yee Hong Yang. 3d reconstruction of transparent objects with position-normal consistency. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4369–4377, 2016.

[40] Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc Pollefeys. Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3313–3322, 2019.

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

[42] 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 *IEEE International Conference on Computer Vision (ICCV)*, pages 2304–2314, 2019.

[43] Shreeyak Sajjan, Matthew Moore, Mike Pan, Ganesh Nagaraja, Johnny Lee, Andy Zeng, and Shuran Song. Clear grasp: 3d shape estimation of transparent objects for manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3634–3642, 2020.

[44] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 2020.

[45] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision (ECCV)*, 2012.

[46] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *International Conference on 3D Vision (3DV)*, 2017.

[47] Bojian Wu, Yang Zhou, Yiming Qian, Minglun Gong, and Hui Huang. Full 3d reconstruction of transparent objects. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 37(4):103:1–103:11, 2018.

[48] Yu Xiang, Christopher Xie, Arsalan Mousavian, and Dieter Fox. Learning rgb-d feature embeddings for unseen object instance segmentation. In *Conference on Robot Learning (CoRL)*, 2020.

[49] 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 (NeurIPS)*, pages 492–502, 2019.

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

[51] Sai-Kit Yeung, Chi-Keung Tang, Michael S Brown, and Sing Bing Kang. Matting and compositing of transparent and refractive objects. *ACM Transactions on Graphics (TOG)*, 30(1):1–13, 2011.

[52] Wentao Yuan, Tejas Khot, David Held, Christoph Mertz, and Martial Hebert. Pcn: Point completion network. In *International Conference on 3D Vision (3DV)*, pages 728–737, 2018.

[53] Yinda Zhang and Thomas Funkhouser. Deep depth completion of a single rgb-d image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 175–185, 2018.