NeRF-GTO: Using a Neural Radiance Field to Grasp Transparent Objects

Anonymous Author(s)
Affiliation
Address
email

Abstract: The ability to grasp and manipulate transparent objects is a major challenge for robots. Existing depth cameras have difficulty detecting, localizing, and inferring the geometry of such objects. We propose using neural radiance fields (NeRF) to detect, localize, and infer the geometry of transparent objects with sufficient accuracy to find and grasp them securely. We leverage NeRF's view-independent learned density, place lights to increase specular reflections, and perform a transparency-aware depth-rendering that we feed into the Dex-Net grasp planner. We show how additional lights create specular reflections that improve the quality of the depth map, and test a setup for a robot workcell equipped with an array of cameras to perform transparent object manipulation. We also create synthetic and real datasets of transparent objects in real-world settings, including singulated objects, cluttered tables, and the top rack of a dishwasher. In each setting we show that NeRF and Dex-Net are able to reliably compute robust grasps on transparent objects, achieving 90 % and 100 % grasp success rates in physical experiments on an ABB YuMi, on objects where baseline methods fail. See supplementary material and https://sites.google.com/view/nerf-gto/home for code, video, and datasets.

1 Introduction

Transparent objects are common in homes, restaurants, retail packaging, labs, gift shops, hospitals, and industrial warehouses. Effectively automating robotic manipulation of transparent objects could have a broad impact, from helping in everyday tasks and performing tasks in hazardous environments. Existing depth cameras assume that surfaces of observed objects reflect light uniformly in all directions, but this assumption does not hold for transparent objects as their appearance varies significantly under different view directions and illumination conditions due to reflection and refraction. In this paper, we propose and demonstrate *NeRF-GTO*, a new method to sense the geometry of transparent objects and allow for robots to interact with them—potentially enabling automation of new tasks.

NeRF-GTO uses a Neural Radiance Fields (NeRF) as part of a pipeline (Fig. 1, right) to compute and execute robots grasps on transparent objects. While NeRF was originally proposed as an alternative for explicit volumetric representations to render novel views of complex scenes [1], it can also reconstruct the geometry of the scene. In particular, due to the view-dependent nature of the NeRF model, it can learn to accurately represent the geometry associated with transparency. The only input requirement to train a NeRF model is a set of images taken from a camera with known intrinsics (e.g., focal length, distortion) and extrinsics (position and orientation in the world). While the intrinsics can be determined from calibration techniques, and/or from the camera itself, determining the extrinsics is often a challenge [2, 3]; however, robots in a fixed workcell, or using a camera attached to a manipulator arm with accurate encoders, can readily determine camera extrinsics. This makes NeRF a particularly good match for robot manipulators.

In experiments, we show qualitatively and quantitatively that NeRF-based grasp-planning can achieve high accuracy on NeRF models trained from photo-realistic synthetic images and from real images, and achieve 90 % or better grasp success rates on real objects.

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

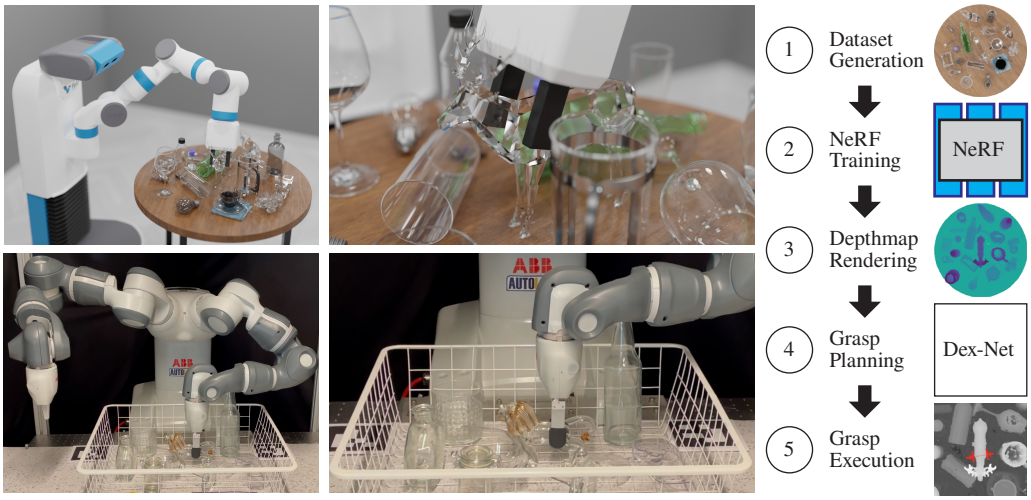


Figure 1: **Using NeRF to grasp transparent objects** Given a scene with transparent objects (left column), we the pipeline on the right to compute grasps (middle column). The top row shows NeRF-GTO working in a simulated scene while the bottom row shows it working in a physical scene.

The contributions of this paper are: (1) Integration of NeRF with robot grasp planning, (2) A transparency-aware depth rendering method for NeRF, (3) Experiments on synthetic and real images showing NeRF with Dex-Net generates high-quality grasps, (4) Synthetic and real image datasets with transparent objects for training NeRF models.

2 Related Work

Detecting Transparent Objects For robots to interact with transparent objects, they must first be able to detect them. The most recent approaches detecting and recognizing transparent objects are data-driven. Lai et al. [4], then Khaing et al. [5], proposed using a Convolutional Neural Network (CNN) to detect transparent objects in RGB images. Recently, Xie et al. [6] developed a transformer-based pipeline [7] used for transparent object segmentation. Other methods rely on deep-learning models to predict the object pose. Phillips et al. [8] trained a random forest to detect the contours of transparent objects for the purpose of pose estimation and shape recovery. Xu et al. [9] proposed a two-stage method for estimating the 6-degrees-of-freedom (DOF) pose of a transparent object with a single RGBD image by replacing the noisy depth values with estimated values and training a DenseFusion-like network structure [10] to predict the object's 6-DOF pose. Sajjan et al. [11] extend this and incorporate a neural network trained for 3D pose estimation of transparent objects in a robotic picking pipeline, while Zhou et al. [12, 13] train a grasp planner directly on raw images from a light-field camera. Zhu et al. [14] used an implicit function to complete missing depth given noisy RGBD observation of transparent objects. However, these data-driven methods rely on large annotated datasets that are hard to curate, whereas NeRF-GTO does not require any prior dataset.

Neural Radiance Fields Recently, implicit neural representations have led to significant progress in 3D object shape representation [15, 16, 17] and encoding the geometry and appearance of 3D scenes [18, 1]. Mildenhall et al. [1] presented Neural Radiance Fields (NeRF), a neural network whose input is a 3D coordinate with an associated view direction, and output is the volume density and view-dependent emitted radiance at that coordinate. Due to its view-dependent emitted radiance prediction, NeRF can be used to represent non-Lambertian effects such as specularities and reflections, and therefore capture the geometry of transparent objects. However, NeRF is slow to train and has low data efficiency. Yu et al. [19] proposed *Plenoctrees*, mapping coordinates to spherical harmonic coefficients, shifting the view-dependency from the input to the output. In addition, Plenoctrees pre-samples the model into a sparse octree structure, achieving a significant speedup in training over NeRF. Deng et al. [20] proposed JaxNeRF, an efficient JAX implementation of NeRF that was able to reduce the training time of a NeRF model from over a day to several hours. Deng et al. [21] add depth supervision to train NeRF 2 to 6$\times$ faster given fewer training views. In this work, we propose to use NeRF to recover the geometry of transparent objects for the purpose of robotic manipulation.

Robotic Grasping Traditional robot grasping methods analyze the object shape to identify successful grasp poses [22, 23, 24]. Data-driven approaches learn a prior using labeled data [25, 26] or through self-supervision over many trials in a simulated or physical environment [27, 28] and generalize to grasping novel objects with unknown geometry. Both approaches rely on RGB and depth sensors to generate a sufficiently accurate observation of the target object surface, such as depth maps [29, 30, 31], point clouds [32, 33, 34, 9], octrees [35], or a truncated signed distance function (TSDF) [36, 37] from which it can compute the grasp pose. While various grasp-planning methods use different input geometry to compute grasps, in this paper we propose a method to render a high-quality depth map from a NeRF model to then pass to Dex-Net [29] to compute a grasp. While standard depth cameras have gaps in their depth information that needs to be processed out with hole-filling techniques, the depth map rendering from NeRF is directly usable. It is possible that other grasp-planning techniques may be able to plan grasps from NeRF models.

3 Problem Statement

We assume an environment that has an array of cameras at fixed known locations, or that the robot can manipulate a camera (e.g., wrist-mounted) to capture multiple images of the scene. Given the environment contains rigid transparent objects, compute a frame for a robot gripper that will result in a stable grasp of a transparent object.

4 Method

In this section, we provide a brief background on NeRF, then describe recovering geometry of transparent objects, integrating with grasp analysis, and improving performance with additional lights.

4.1 Preliminary: Training NeRF

NeRF [1] learns a neural scene representation that maps a 5D coordinate containing a spatial location (x, y, z) and viewing direction (θ, ϕ) to the volume density σ and RGB color $\mathbf{c}$. Training NeRF's multilayer perceptron (MLP) requires multi-view RGB images of a static scene with their corresponding camera poses and intrinsic parameters. The expected color $C(\mathbf{r})$ of the camera ray $\mathbf{r} = \mathbf{o} + t\mathbf{d}$ between near and far scene bounds t_n and t_f is:

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt, \tag{1}$$

where $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$ is the probability that the camera ray travels from near bound t_n to point t without hitting any surface. NeRF approximates the expected color $\hat{C}(\mathbf{r})$ as:

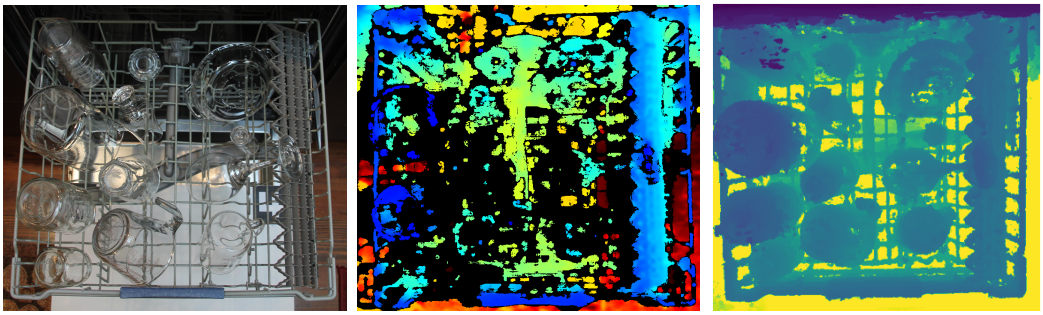
$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i, \tag{2}$$

where $T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right)$ and $\delta_i = t_{i+1} - t_i$ is the distance between consecutive samples on the ray $\mathbf{r}$. During training, NeRF minimizes the error between the rendered and ground truth rays' colors using gradient descent.

4.2 Recovering Geometry of Transparent Objects

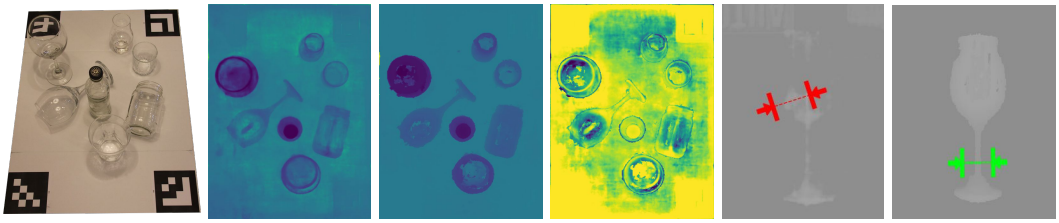
We observe that NeRF does not directly support transparent object effects—it casts a single ray per source image pixel without reflection, splitting, or bouncing. The incorporation of the viewing-direction in its regression and supervising with view-dependent emitted radiance allows recovery of non-Lambertian effects such as reflections from a specular surface. However, while RGB color $\mathbf{c}$ is view-dependent, the volume density σ is not—meaning NeRF has to learn a non-zero σ to represent any color at that spatial location. Visually, the usual result is that the transparent object shows up as a "ghostly" or "blurry" version of the original object.

When training, a NeRF model learns a density σ of each spatial location. This density corresponds to the transparency of the point, and serves to help learn how much a spatial location contributes to



Real Image | RealSense Depth | Depth (NeRF-GTO)

Figure 2: **Comparison to RealSense Depth Camera**. We compare the results of the proposed pipeline in a real-world setting against the depth map produced by an Intel RealSense camera. In the left image is the real-world scene, the middle shows the depth image from the RealSense, and the right shows the result of our pipeline. The color scheme in the RealSense image is provided by the RealSense SDK, while the color scheme in the right column is from MatPlotLib. We observe that the RealSense depth camera is unable to recover depth from a large portion of the scene, shown in black. On the other hand, the proposed pipeline, while having a few holes, can recover depth for most of the scene.



(A) Real Image | (B) Depth Map Vanilla NeRF | (C) Depth Map NeRF-GTO | (D) Depth Difference | (E) Dex-Net Grasp Vanilla NeRF | (F) Dex-Net Grasp NeRF-GTO

Figure 3: **Using NeRF to render depth for grasping transparent objects**. NeRF-GTO uses a transparency-aware depth rendering to render depth maps that can be used for grasp planning. In contrast, Vanilla-NeRF's depth maps are filled with holes and result in poor grasp predictions.

the color of a ray cast through it. Although NeRF converts each σ_i to an occupancy probability $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$, where δ_i is the distance between integration times along the ray, thus implicitly giving α_i an upper bound of 1, it does not place a bound on the raw σ value. We use the raw value of σ to determine if a point in space is occupied.

4.3 Rendering Depth for Grasp Analysis

We propose using Dex-Net to compute grasps on transparent objects. Dex-Net computes candidate grasp poses given a depth image of a scene. To generate a depth image, we consider two candidate reconstructions of depth. First, we use the same rendering of sampled points along a camera ray that NeRF uses. This *Vanilla NeRF* reconstruction first converts σ_i to an occupancy probability α_i. It then applies the transformation $w_i = \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j)$. To render depth at pixel coordinate $[u, v]$, it computes the sum of sample distances from the camera weighted by the termination probability $D[u, v] = \sum_{i=1}^{N} w_i \delta_i$. When applied on transparent objects, however, this results in noisy depth maps, as shown in Fig. 3.

Instead, we consider a second, transparency-aware method that searches for the first sample along the ray for which $\sigma_i > m$ where m is a fixed threshold. The depth is then set to the distance of that sample δ_i. We explore different values for m, and observe that low values result in a noisy depth map while high values create holes in the depth map. In our experiments we set $m = 15$ (see Fig. 8).

4.4 Improving Reconstruction with Light Placement

For NeRF to learn the geometry of a transparent object, it must be able to "see" it from multiple camera views. If the transparent object is not visible from any views, then it will have no effect on the loss function used in training, and thus not be learned. We thus look for a way to improve visibility of transparent objects to NeRF.

One property that transparent objects share (e.g., glass, clear plastic) is that they are glossy and thus produce specular reflections when the camera view direction is opposite to the surface normal of the incident direction of light. To the NeRF model, a specular reflection viewed from multiple angles will appear as a white point on a solid surface—i.e., $\mathbf{c} = [1, 1, 1]^T$ and $\sigma > 0$, while from other angles it will appear as $\sigma \leq 0$. As σ is view-independent, NeRF learns a σ between fully opaque and fully transparent for such points.

By placing additional lights in the scene, we create more angles from which cameras will see specular reflections from transparent objects—this results in NeRF learning a model that fills holes in the scene. While the number and placement of lights for optimal training is dependent on both the expected object distribution and camera placement, in experiments (Sec. 5.5) we show that increasing from 1 light to a 5x5 array of lights improves the quality of the learned geometry.

5 Experiments

We experiment in both simulation and on a physical ABB YuMi robot. We generate multiple datasets, where each dataset consists of images and associated camera transforms of one static scene containing one or more transparent objects. We train NeRF models using a modified JaxNeRF [20] implementation on 4 Nvidia V100 GPUs. We use an existing pre-trained Dex-Net model for grasp planning without modification or fine-tuning. We can do this since NeRF models can be rendered to depth maps from arbitrary camera intrinsics and extrinsics, thus we match our NeRF rendering to the Dex-Net model instead of training a new one.

5.1 Datasets

As existing NeRF datasets do not include transparent objects, and existing transparent-object-grasping datasets do not include multiple camera angles, we generate new datasets using 3 different methods: synthetic, Cannon EOS 60D camera with a Tamron Di II lens with a locked focal length, and an Intel RealSense.

For synthetic datasets, we use Blender 2.92's physically-based Cycles renderer with path tracing set to 10240 samples per pixel, and max light path bounces set to 1024. We chose theses settings by increasing them until renderings were indistinguishable from the previous setting—finding that lower settings lead to dark regions and smaller specular reflections. For glass materials, we set the index of refraction to 1.45 to match physical glass. We include 8 synthetic datasets of transparent objects: 2 scenes with clutter: light array and single light; 4 singulated objects from Dex-Net: Pipe Connector, Pawn, Turbine Housing, Mount; and 2 household objects: Wineglass upright and Wineglass on side. As these computationally demanding to render due to the high quality settings, we distribute these as part of the contribution.

For the Cannon EOS and RealSense real-world datasets, we place ArUco markers in the scene to aid in camera pose recovery and take photos around the objects using a fixed ISO, f-stop, and focal length. We use bundle adjustment from COLMAP [2, 3] to refine the camera poses and intrinsics to high accuracy. We include 8 physical datasets of transparent objects with a variety of camera poses: table with clutter, Dishwasher, Tape Dispenser, Wineglass on side, Flask, Safety Glasses, Bottle upright, Lion Figurine in clutter. The main difficulty in generating these datasets is calibration and computing high-precision camera poses.

The datasets, available at `https://sites.google.com/view/nerf-gto/home` differ from other datasets in their focus on scenes with transparent objects in a graspable setting, with over 70 camera poses each.

5.2 Synthetic Grasping Experiments

We also test the ability of NeRF-GTO to generate grasps on the synthetic singulated transparent Dex-Net object datasets. For each dataset, we evaluate the grasp in simulation using a wrench resistance metric measuring the ability of the grasp to resist gravity [38]. Fig. 4 shows images of the synthetic objects, NeRF-GTO-generated depth map, and an example sampled grasp for each. To measure the effect of training time on grasp-success rate, we simulate and record grasps over the course of training. In Fig. 5, we observe that grasp success rate improves with training time, but plateaus between 80 % and 98 % success rate at around 50k to 60k iterations. This suggests that there may be a practical fixed iteration limit to obtain high grasp success rates.

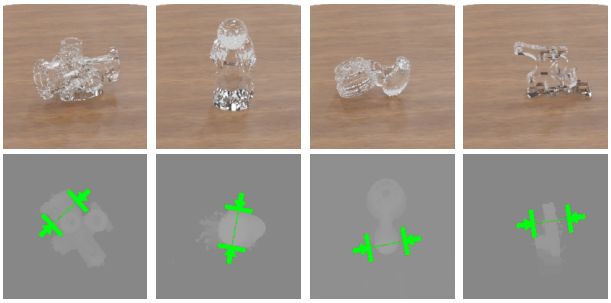


Figure 4: **Synthetic singulated objects** used in simulation experiments. **Top row:** image of the object in the training data. **Bottom row:** computed depth map and candidate grasp.

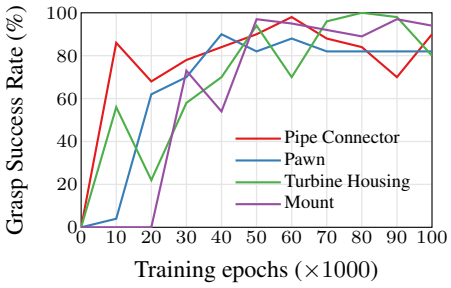


Figure 5: **Grasp success rate vs training epochs.** As opposed to view-synthesis, which requires over 200k epochs, we observe high grasp success rates after 50k to 60k epochs.

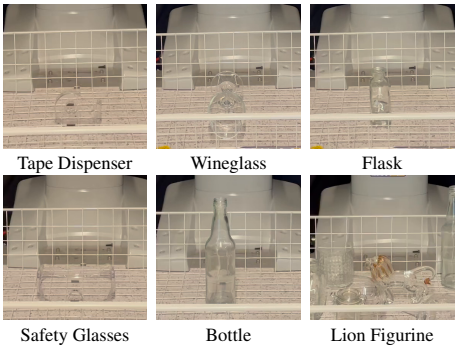


Figure 6: **Physical grasps objects.** In the background is the base of the YuMi robot.

Object	PhoXi	Vanilla NeRF	NeRF-GTO
Tape Dispenser	0/10	0/10	**10/10**
Wineglass	0/10	0/10	**9/10**
Flask	0/10	1/10	**9/10**
Safety Glasses	0/10	0/10	**10/10**
Bottle	0/10	**10/10**	**10/10**
Lion Figurine	0/10	3/10	**10/10**

Table 1: **Physical grasp success rate.** For each object, we compute a depth map using a PhoXi camera, unmodified NeRF, and the proposed method for grasping transparent objects. From the depth map, Dex-Net computes a 10 different grasps, and an ABB YuMi attemps the grasp. Successful grasps lift the object.

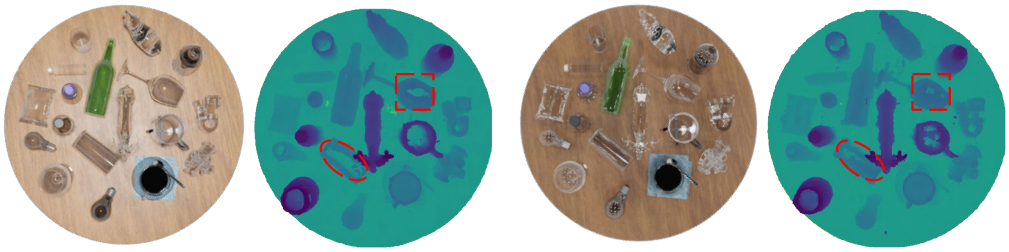
We test NeRF-GTO on a scene of a tabletop cluttered with transparent objects. In this experiment, the goal is to grasp a transparent object placed in a stable pose in close proximity to other transparent objects. The challenge is twofold: the depth rendering quality should be sufficient for both grasp planning and collision avoidance. Fig. 1 shows the robot and scene in the upper left, and the overhead image, depth, and computed grasp inline in the pipeline, and the final computed grasp with simulated execution is in the upper middle image. The final grasp contact point was accurate to a 2 mm tolerance, suggesting that NeRF-GTO with sufficient images taken from precisely-known camera locations may be practical in highly clutter environments.

5.3 Physical Grasping Experiments

To test the NeRF-GTO in a physical setup, we place transparent singulated objects in front of an ABB YuMi robot, and have the robot perform the computed grasps. We compare to 2 baselines: (1) *PhoXi*, in which a PhoXi camera provides the depth map; and (2) *Vanilla NeRF*, in which we use the original depth rendering from NeRF. The PhoXi camera is normally able to generate high-precision depth maps for non-transparent objects. All methods use the same pre-trained Dex-Net model, and both Vanilla NeRF and NeRF-GTO use the same NeRF model—the only difference is the depth rendering. We test with 6 objects (Fig. 6), and compute and execute 10 different grasps for each and record the success rate. A grasp is successful if the robot lifts the object. In Table 1, we see that NeRF-GTO gets 90 % and 100 % success rates for all objects, while the baselines get few successful grasps. The PhoXi camera is unable to recover any meaningful geometry which causes Dex-Net predictions to fail. The Vanilla NeRF depth maps often have unpredictable protrusions that result in Dex-Net generating unreliable grasps.

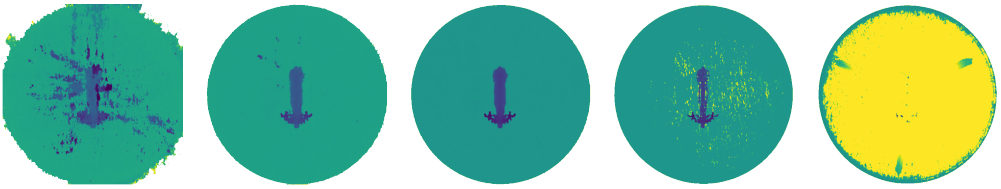
5.4 Comparison to RealSense Depth

We qualitatively compare the the rendered depth map of the proposed pipeline against a readily-available depth sensor on scenes with transparent objects in real-world settings. We select the Intel



(a) RGB Scene Single Light Source | (b) Depth Rendering Single Light Source | (c) RGB Scene Multiple Light Sources | (d) Depth Rendering Multiple Light Sources

Figure 7: More lights mean more specular reflections, and result in better NeRF depth estimation of transparent surfaces. In (a) and (b), we show a scene lit by a single overhead high-intensity light. In (c) and (d) we show the same scene lit by an overhead 5x5 array of lights. The combined light wattage is equal in both scenes. Images (a) and (c) are views of the scene, and (b) and (d) are the corresponding depth images obtained from the pipeline. Two glasses on their sides are missing top surfaces (outlined in dashed red) in (b), while the effect is reduced in (d) due to the additional light sources.



$\sigma = 1$ | $\sigma = 5$ | $\sigma = 15$ | $\sigma = 150$ | $\sigma = 500$

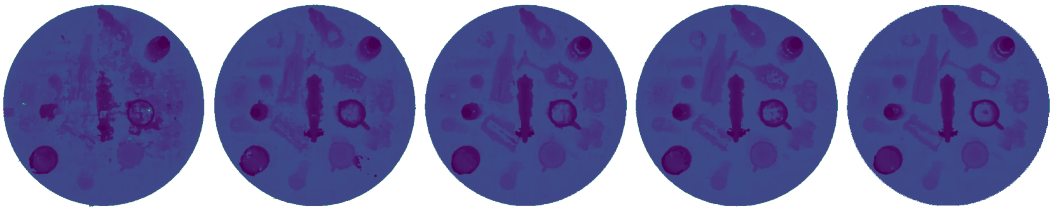
Figure 8: **depth rendering using NeRF with different thresholds** Here we show the effect of the threshold value on the depth rendering on an isolated deer figurine. Values too low result in excess noise, while values too high cause parts of the scene to disappear.

RealSense as it is common to robotics applications, readily available, and high-performance. The RealSense, like most stereo depth cameras, struggles with transparent objects as they are unable to compute a stereo disparity between pixels from different cameras when the pixels are specular reflections or the color of the object behind the transparent object. The RealSense optionally projects a structured light pattern on the the scene to aid in computing depth from textureless surfaces, however, in experiments we observed no qualitative difference with and without the light pattern emitter enabled. To run this experiment, we use a Canon EOS as described in the main text, and use a RealSense to take a depth picture of the scene. In this experiment, we observe that the RealSense is unable to compute the depth of most transparent objects, and often produces regions of unknown depth (shown in black) where transparent objects are. On the other hand, the proposed pipeline produces high-quality depth maps with only a few areas of noise. The results of these experiments is shown in Fig. 2.

5.5 One vs Many Lights

We experiment with different light setups to test the effect of specular reflections on the ability of NeRF to recover geometry of transparent objects. We create two scenes (Fig. 7), one with a single bright light source directly above the work surface, and another with an array of 5x5 (25) lights above the work surface. We set the total wattage of the lights in each scene to be the same. Since most lights in the multiple light scene are further away from the work surface than the single light source, the scene appears darker, though more evenly illuminated. The effect of the specular reflections is prominent on the lightbulb in the lower part of the image. In the single light source, there is a single specular reflection, while in the multiple light scene, the reflection of the array of lights is visible.

With the same camera setup for both scenes, we train NeRF models with the same number of iterations. We show the depth rendering in Fig. 7 and circle a glass and a wineglass on their side. In the single-light source image, the closer surfaces of the glasses are missing, while in the multiple-light source depth image, the glasses are nearly fully recovered. This suggests that additional lights in the scene can help NeRF recover the geometry of transparent objects better.



9 Cameras 16 Cameras 25 Cameras 36 Cameras 49 Cameras

Figure 9: **Depth rendering using a grid of overhead cameras.** Using increasing amounts of overhead cameras improves the quality of the depth map and its utility in grasping, however, beyond a certain number of cameras there is a diminishing return.

5.6 Workcell Setup

We experiment with a potential setup for a robot workcell in which a grid of overhead cameras captures views of the cluttered scene so that a robot manipulator arm can then perform tasks with transparent objects in the workcell. We propose that a grid of overhead cameras would be practical to setup and would not obstruct manipulator tasks nor operator interventions. The objective is to determine how many overhead cameras would be needed to recover a depth map of sufficient accuracy to perform maniplation tasks.

We place a 2 m by 2 m grid of cameras 1 m above the work surface, and have them all point at the center of the work surface. Each camera has the same intrinsics, and are evenly spaced along the grid. We experiment with grids having 4, 9, 16, 25, 36, and 49 cameras. The environment has the same 5x5 grid of lights as before. For each camera grid, we train JaxNeRF for 50k iterations and compare performance.

After training, we observe increasing peak signal to noise ratios (PSNR) and structural similarity (SSIM) scores with increasing number of cameras. The 2x2 grid of cameras produces a high train-to-test ratio for PSNR, likely indicating overfitting to training data, and results in a depth map without apparent geometry. This ratio decreases with additional cameras. The minimum number of cameras for this proposed setup appears to be around 9 (3x3) as its depth map is usable for grasp planning, while the 5x5 grid shows better PSNR and SSIM and ratio between train and test PSNR, and the 7x7 grid is the best. See Fig. 9 for a visual comparison. Additionally, we trained 9x9, 11x11, and 13x13 grids, observing no statistically significant improvement beyond the 7x7 grid.

6 Conclusion

In this work, we showed that NeRF can recover the geometry of transparent objects with sufficient accuracy to compute grasps for robot manipulation tasks. NeRF learns the density of all points in space, which corresponds to how much the view-dependent color of each point contributes to rays passing through it. With the key observation that specular reflections on transparent objects cause NeRF to learn a non-zero density, we recover the geometry of transparent objects through a combination of additional lights to create specular reflections and thresholding to find transparent points that are visible from some view directions. With the geometry recovered, we pass it to a grasp planner, and show that the recovered geometry is sufficient to compute a grasp, and accurate enough to achieve 90 % and 100 % grasp success rates in physical experiments on an ABB YuMi robot. We created synthetic and real datasets for experiments in transparent geometry recovery, but we believe these datasets may be of interest to researchers interested in extending NeRF capabilities in other ways and thus contribute them as part of this research project. Finally, to test if NeRF could be used in a robot workcell, we experimented with grids of cameras facing a worksurface and their ability to recover geometry in potential setup, and showed the increased capabilities and point of diminishing return for additional cameras.

In future work, we hope to address one of the main drawbacks of NeRF—the long training time required to obtain a NeRF model. Many research groups have sped up training time through improved implementations, new algorithms, new network architectures, pre-conditioned network weights, focused sampling, and more. While these approaches apply to general NeRF training, we plan to exploit features specific to robot scenerios to speed up training, including using depth camera data as additional training data, manipulator-arm-mounted cameras to inspect regions of interest, and visio-spatial foresight to adapt to changes in the environment.

References

[1] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020.

[2] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[3] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[4] P.-J. Lai and C.-S. Fuh. Transparent object detection using regions with convolutional neural network. In *IPPR Conference on Computer Vision, Graphics, and Image Processing*, volume 2, 2015.

[5] M. P. Khaing and M. Masayuki. Transparent object detection using convolutional neural network. In *International Conference on Big Data Analysis and Deep Learning Applications*, pages 86–93. Springer, 2018.

[6] E. Xie, W. Wang, W. Wang, P. Sun, H. Xu, D. Liang, and P. Luo. Segmenting transparent object in the wild with transformer. *arXiv preprint arXiv:2101.08461*, 2021.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[8] C. J. Phillips, M. Lecce, and K. Daniilidis. Seeing glassware: from edge detection to pose estimation and shape recovery. In *Robotics: Science and Systems*, volume 3, 2016.

[9] C. Xu, J. Chen, M. Yao, J. Zhou, L. Zhang, and Y. Liu. 6dof pose estimation of transparent object from a single rgb-d image. *Sensors*, 20(23):6790, 2020.

[10] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3343–3352, 2019.

[11] S. Sajjan, M. Moore, M. Pan, G. Nagaraja, J. Lee, A. Zeng, and S. Song. Clear grasp: 3d shape estimation of transparent objects for manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3634–3642. IEEE, 2020.

[12] Z. Zhou, T. Pan, S. Wu, H. Chang, and O. C. Jenkins. Glassloc: Plenoptic grasp pose detection in transparent clutter. *arXiv preprint arXiv:1909.04269*, 2019.

[13] Z. Zhou, X. Chen, and O. C. Jenkins. Lit: Light-field inference of transparency for refractive object localization. *IEEE Robotics and Automation Letters*, 5(3):4548–4555, 2020.

[14] L. Zhu, A. Mousavian, Y. Xiang, H. Mazhar, J. van Eenbergen, S. Debnath, and D. Fox. Rgb-d local implicit function for depth completion of transparent objects. *arXiv preprint arXiv:2104.00622*, 2021.

[15] L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[16] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[17] Z. Chen and H. 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.

[18] V. Sitzmann, M. Zollhöfer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. *arXiv preprint arXiv:1906.01618*, 2019.

[19] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. *arXiv preprint arXiv:2103.14024*, 2021.

[20] J. T. B. Boyang Deng and P. P. Srinivasan. Jaxnerf: an efficient jax implementation of nerf, 2020, 2020. URL http://github.com/google-research/google-research/tree/master/jaxnerf.

[21] K. Deng, A. Liu, J.-Y. Zhu, and D. Ramanan. Depth-supervised nerf: Fewer views and faster training for free. *arXiv e-prints*, pages arXiv–2107, 2021.

[22] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber. A survey on learning-based robotic grasping. *Current Robotics Reports*, pages 1–11, 2020.

[23] A. Bicchi and V. Kumar. Robotic grasping and contact: A review. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 348–353. IEEE, 2000.

[24] R. M. Murray, Z. Li, and S. S. Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[25] D. Kappler, J. Bohg, and S. Schaal. Leveraging big data for grasp planning. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4304–4311. IEEE, 2015.

[26] D. Prattichizzo, J. C. Trinkle, B. Siciliano, and O. Khatib. Springer handbook of robotics. *Grasping; Springer: Berlin/Heidelberg, Germany*, pages 671–700, 2008.

[27] E. Jang, C. Devin, V. Vanhoucke, and S. Levine. Grasp2vec: Learning object representations from self-supervised grasping. *arXiv preprint arXiv:1811.06964*, 2018.

[28] G. Peng, Z. Ren, H. Wang, and X. Li. A self-supervised learning-based 6-dof grasp planning method for manipulator. *arXiv preprint arXiv:2102.00205*, 2021.

[29] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. *arXiv preprint arXiv:1703.09312*, 2017.

[30] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[31] J. Redmon and A. Angelova. Real-time grasp detection using convolutional neural networks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1316–1322. IEEE, 2015.

[32] A. Mousavian, C. Eppner, and D. Fox. 6-dof graspnet: Variational grasp generation for object manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2901–2910, 2019.

[33] Y. Qin, R. Chen, H. Zhu, M. Song, J. Xu, and H. Su. S4g: Amodal single-view single-shot se (3) grasp detection in cluttered scenes. In *Conference on robot learning*, pages 53–65. PMLR, 2020.

[34] M. Sundermeyer, A. Mousavian, R. Triebel, and D. Fox. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. *arXiv preprint arXiv:2103.14127*, 2021.

[35] Y. Avigal, V. Satish, Z. Tam, H. Huang, H. Zhang, M. Danielczuk, J. Ichnowski, and K. Goldberg. Avplug: Approach vector planning for unicontact grasping amid clutter. 2021.

[36] M. Breyer, J. J. Chung, L. Ott, R. Siegwart, and J. Nieto. Volumetric grasping network: Real-time 6 dof grasp detection in clutter. *arXiv preprint arXiv:2101.01132*, 2021.

[37] S. Song, A. Zeng, J. Lee, and T. Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3): 4978–4985, 2020.

[38] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg. Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning. In *2018 IEEE International Conference on robotics and automation (ICRA)*, pages 5620–5627. IEEE, 2018.