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

Anonymous Author(s)
Affiliation
Address
email

Abstract: Transparent objects are a bane for robot manipulation tasks. Existing depth sensors have difficulty detecting, localizing, and inferring their geometry, making grasping contacts with them precarious. We propose using neural radiance fields (NeRF) to detect, localize, and infer the geometry of transparent objects with sufficient accuracy to find and perform grasps. We leverage NeRF's view-independent learned density, combined with placing 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 possible 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 cluttered tables, laboratory setting, and the top rack of a dishwasher. In each setting we show that NeRF and Dex-Net are able to compute grasps on transparent objects. See supplementary material for code and datasets.

1 Introduction

Transparent objects can be found everywhere, from homes to industrial warehouses to minerals in nature. Effectively automating robotic manipulation of transparent objects could have a broad impact, from helping in everyday tasks such as doing dishes, to performing tasks in hazardous environments. While the transparency of an object poses little challenge for humans, to robots, even simple manipulation tasks are extremely challenging due to the limitations of existing depth cameras. These cameras, and the underlying methods they use, tend to assume that all surfaces of observed objects are Lambertian, reflecting light uniformly in all directions, however, this assumption does not hold for transparent objects for which the appearance varies significantly under different illumination conditions due to the reflection and refraction properties of transparent materials. In this paper, we propose and demonstrate that recently developed neural radiance field methods can be trained to sense the geometry of transparent objects and allow for robots to interact with them—potentially enabling automation of new tasks.

The sensing modality we propose to use is Neural Radiance Fields (NeRF). While NeRF was originally proposed as an alternative for explicit volumetric representations, and shown to realistically render novel views of complex scenes [1], we instead look to its ability to 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 non-Lambertian effects 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 found through online databases, 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.

Grasping a transparent object, once its geometry is known, is like grasping any object of known geometry. While various grasp-planning methods use different input geometry to compute grasps,

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

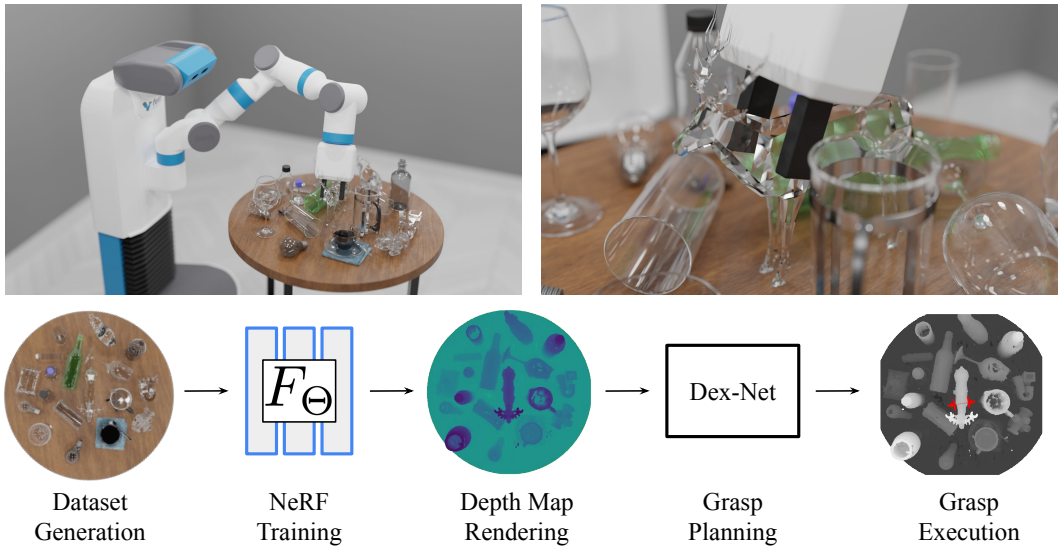


Figure 1: **Using NeRF to grasp transparent objects** Given a scene with transparent objects (top left), we (bottom row) train a NeRF model, compute a transparency-aware depth rendering, compute a grasp with Dex-Net, and then execute the grasp. The quality of the depth map is high enough that the grasp can be executed without unwanted collisions (top right).

e.g., depth maps [4], point clouds [5], and octrees [6], in this paper we propose a method to render a high-quality depth map from a NeRF model to then pass to Dex-Net [4] to compute a grasp. While standard depths 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. We observe that other grasp-planning techniques may similarly be able to render or generate geometry from NeRF models as well.

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.

The contributions of this paper are:

- Integration of NeRF with robot grasp planning.
- A transparency-aware depth rendering method for NeRF.
- Experiments on synthetic and real images showing NeRF with Dex-Net generates high-quality grasps.
- Synthetic and real image datasets with transparent objects for training NeRF models.

2 Related Work

2.1 Detecting Transparent Objects

For robots to interact with transparent objects, they must first be able to detect them. The most recent approaches to solve the problem of detecting and recognizing transparent objects are data-driven. Lai et al. [7] followed by Khaing et al. [8] proposed using a Convolutional Neural Network (CNN) to detect transparent objects in RGB images. Recently, Xie et al. [9] developed a transformer-based pipeline [10] used for transparent object segmentation. Other methods rely on deep learning models to predict the object pose. Phillips et al. [11] trained a random forest to detect the contours of transparent objects for the purpose of pose estimation and shape recovery. Xu et al. [5] 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 [12] to predict the object's 6-DOF pose. Sajjan et al. [13] take it a step further and incorporate a neural network trained for 3D pose estimation of transparent objects

in a robotic picking pipeline, while Zhou et al. [14, 15] train a grasp planner directly on raw images from a light-field camera. Zhu et al. [16] 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 the NeRF-based system we propose does not require any prior curated dataset.

2.2 Neural Radiance Fields

Recently, implicit neural representations have led to significant progress in 3D object shape representation [17, 18, 19] and encoding the geometry and appearance of 3D scenes [20, 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. [21] 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. Jain et al. [22] showed how adding an auxiliary semantic consistency loss to the training procedure can encourage realistic renderings at novel poses using significantly fewer views during training. Deng et al. [23] 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. In this work we propose to use NeRF to recover the shape and geometry of transparent objects for the purpose of robotic manipulation.

2.3 Robotic Grasping

Traditional robot grasping methods analyze the object shape by leveraging knowledge of the object geometry to identify successful grasp poses [24, 25, 26]. Data-driven approaches learn a prior using labeled data [27, 28] or through self-supervision over many trials in a simulated or physical environment [29, 30] 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 [4, 31, 32], point clouds [33, 34, 35] or a truncated signed distance function (TSDF) [36, 37] from which it can compute the grasp pose.

3 Problem Statement

Given an environment containing transparent objects, compute a frame for a robot gripper that will result in a stable grasp of a transparent object. We assume the environment 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. Second, we assume the stability of grasps is independent of the objects transparency—in other words, if a grasp planner is given the geometry of the scene, the quality of the grasps it generates does not depend on the transparency of the object. We make this second assumption, observing that transparent objects from example robot automation scenarios are made from glass or plastic and thus have rigidity consistent with the training models for grasp planners.

4 Method

4.1 Preliminary: Training NeRF

The NeRF model [1] learns a neural scene representation that maps a single continuous 5D coordinate containing a spatial location (x, y, z) and viewing direction (θ, ϕ) to the volume density σ and RGB color $\mathbf{c}$ at that spatial location. Training is performed using a multilayer perceptron (MLP) and requires multi-view RGB images of a static scene with their corresponding camera poses and intrinsic parameters, as well as the scene bounds. 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:

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

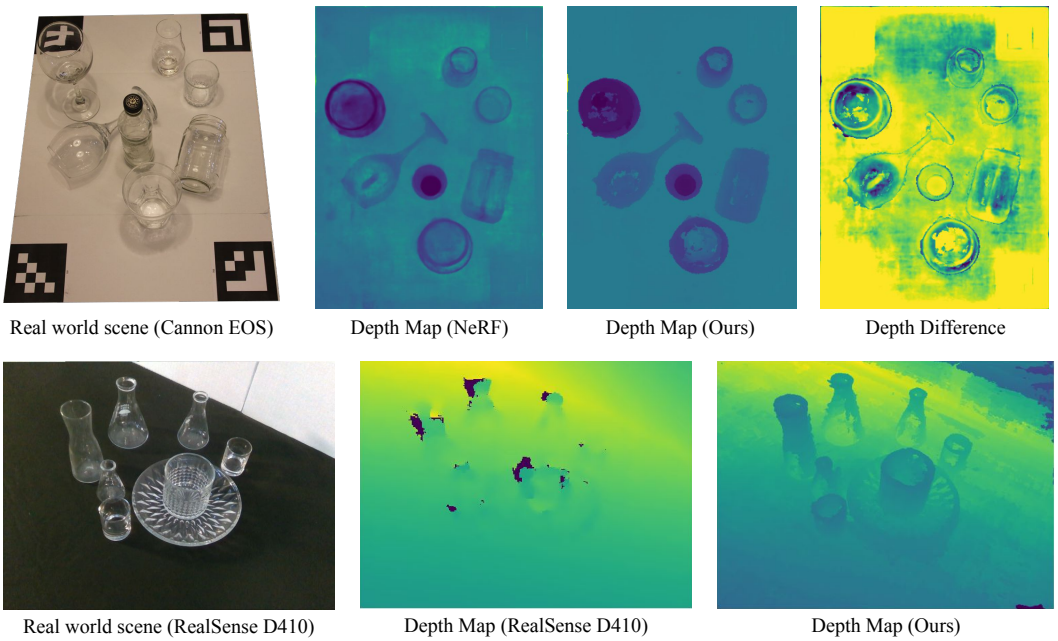


Figure 2: **Depth sensing in the real-world.** RGBD sensors, such as the RealSense use structured light and disparity to compute a depth image, and struggle with transparent objects. On the other hand, NeRF is able to learn the geometry of transparent objects, and with appropriate depth rendering can produce a high-quality depth map that includes transparent object surfaces. The top row shows a view from a Cannon EOS dataset, while the bottom row shows a view from a RealSense dataset (see Sec. 5 for description).

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.

From Eq. 1, the RGB color $\mathbf{c}(\mathbf{x}, \mathbf{d})$ is view-dependent while the volume density $\sigma(\mathbf{x})$ is view-independent. This allows NeRF to represent non-Lambertian effects, such as reflections and refractions, and at the same time account for the occupancy at a spatial location.

We exploit this property of NeRF to estimate the position of transparent surfaces that may be invisible from some view but visible from another, allowing NeRF to detect the transparent surface via the view-independent volume density.

4.2 Recovering Geometry of Transparent Objects

When training a NeRF model, it learns a density (σ) of a point in space. This density corresponds to the transparency of the point, and when training, the σ serves to help learn whether a point in space contributes to the color of a ray cast through it. We use the value of σ to determine if a point in space is occupied (and in a sense, are only using color to recover this value, an inversion of view-dependent rendering intent behind NeRF).

In the NeRF model, σ is view-independent, a natural choice for scenes that contain non-transparent objects, since, while color contributions are view-dependent, the occupancy of points in such a scene is not. When training NeRF for a scene that contains transparent objects, one may consider

updating σ to be view-dependent, under the notion that this would properly capture the effect of specular reflections—which are so bright as to obscure the visibility of anything behind them. While a view-dependent σ may help with view-dependent rendering of scenes with transparent objects, we hypothesize that a view-dependent σ, would make recovering the geometry of transparent objects more difficult. With a view-independent σ, we only have to query a single point in space to determine if it is non-transparent from some angle, whereas a view-dependent σ would have to check all view direction for that same point in space, as a specular reflection that reveals the occupancy may only be visible from a few view directions.

The downside of view-independent σ is that the RGB reconstruction of transparent objects is inaccurate, appearing smudged and ghostly. This appears to be the result of the network learning a σ value between transparent and opaque—transparent for views which see through the point, and opaque for views that see specular reflections at the point.

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 reconstruction first converts σ_i to an occupancy probability $\alpha_i = 1 - \exp(-\sigma_i \delta_i)$ by multiplying the distance between adjacent samples $\delta_i = t_{i+1} - t_i$. 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 sume 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 procedure computes noisy depth maps, as demonstrated in Fig. 2.

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. 5).

4.4 Improving Reconstruction with Light Placement

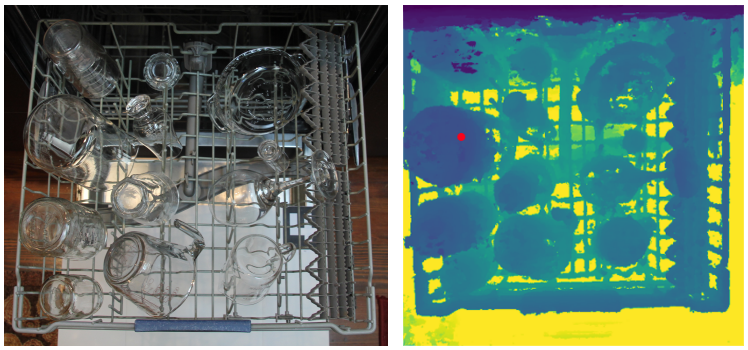
For NeRF to learn the geometry of a transparent object, it must be able to "see" it from multiple camera views. If the transparent object is not visible from any views, then it will have no effect on the loss function used in training, and thus not be learned. We thus look for a way to improve visibility 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. 5.3) we show that increasing from 1 light to a 5x5 array of lights improves the quality of the learned geometry.

5 Experiments

To experiment with grasping transparent objects using NeRF and Dex-Net, we generate multiple scenes in both simulation and real, and compute depth maps and grasps. We train NeRF models using the JaxNeRF [23] implementation on 4 Nvidia V100 GPUs. Since we can render depth maps with arbitrary intrinsics and extrinsics given a trained model, we use an existing Dex-Net model trained with the intrinsics for Phoxi camera position 0.6 m above the work surface.



Camera View Rendered Depth

Figure 3: **Unloading a dishwasher**. On the left is a cropped portion of an image of a real-world dishwasher top-rack loaded with glassware used to train the NeRF model. The right is a depth image rendered from NeRF model and according to the intrinsics expected by Dex-Net. A red dot shows where Dex-Net computes a suction grasp to lift up the french press. Several objects are difficult to see in the camera view, but show up clearly in the depth view.

As existing NeRF datasets do not include transparent objects 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 robot-mounted a Intel RealSense.

For synthetic datasets, we use Blender 2.92's Cycles renderer with path tracing set to 10240 samples per pixel, and max light path bounces set to 1024. For glass materials, we set the index of refraction to 1.45 to match physical glass. We chose the sampling and max-bounce settings by increasing them until renderings were indistinguishable from the previous setting—noting that lower settings would lead to smaller specular reflections and dark regions where path tracing internal reflections terminated early.

For the Cannon EOS 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.

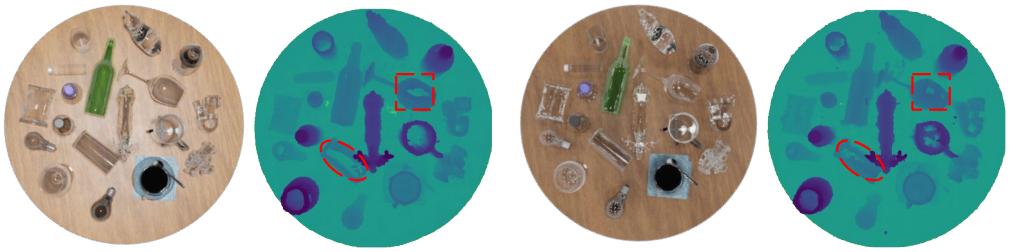
For the RealSense datasets, we mount the an Intel RealSense camera to a Fetch robot, using a 3D printed mount to keep the camera's pose relative to the arm fixed. We then program the robot to have the camera trace an arc at multiple elevations, while recording the pose of the camera through forward kinematics.

5.1 Synthetic Experiments

To test the pipeline in a controlled setting, we generate scene of tabletop cluttered with transparent objects, and run the pipeline shown in Fig. 1. In this experiment, the goal is to grasp a transparent object placed in a stable pose on a cluttered table, 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. To plan a parallel-jaw grasp from the rendered depth map we use a grasp planner based on Dex-Net 4.0 Grasp Quality CNN (GQ-CNN) [38] that samples grasp candidates from the image, evaluates each candidate and returns the grasp with the highest predicted quality along with its associated quality metric. The robot and scene are shown in the upper left of Fig. 1, the overhead image, depth, and computed grasp are shown inline in the pipeline, while the final computed grasp with simulated execution is show in the upper right. The final grasp contact point computed by Dex-Net was accurate to a 2 mm tolerance. The simulated environment provides precise camera poses, but conversely is made more difficult due to the transparent objects having no imperfections.

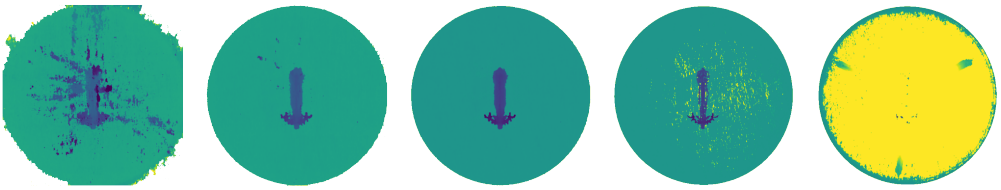
5.2 Grasping real-world Objects

We test in a real-world scenario of automated unloading the top-rack of a dishwasher loaded with glasses, jars, coffee presses, mug, and dishes. We collect a dataset of 68 images using the Cannon EOS setup, and generate a depth map and grasps on objects in the rack. See Fig. 3 for the real image



(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 4: 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 5: **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.

and depth image with an example associated suction grasp point computed by Dex-Net. From the figure, we can see that the depth rendering includes all glass dishes, even though many are hard to see in the real image. We also show a candidate suction grasp computed by Dex-Net 3.0 [39] that places a suction contact on bottom of the upside-down french press.

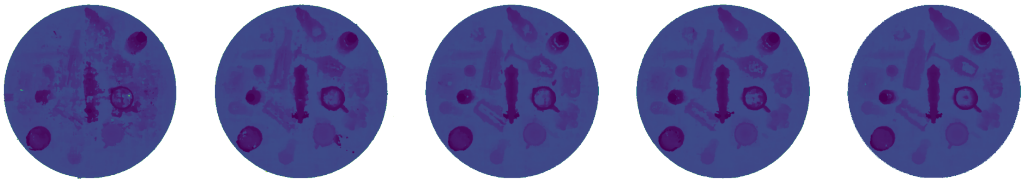
5.3 One vs Many Lights

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. 4), 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 futher 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. 4 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.

5.4 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



9 Cameras 16 Cameras 25 Cameras 36 Cameras 49 Cameras

Figure 6: **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.

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 an 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. 6 for a visual comparison. Additionally, we trained 9x9, 11x11, and 13x13 grids, observing that 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 a point, which corresponds to how much the view-dependent color of that 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 perform the grasp. 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. Already many research groups have sped up training time through improved implementations, new algorithms and 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] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

[22] A. Jain, M. Tancik, and P. Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. *arXiv preprint arXiv:2104.00677*, 2021.

[23] 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.

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

[25] 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.

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

[27] 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.

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

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[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, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26), 2019.

[39] 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.