Closing the Loop: Single-Image to Camera Pose with Inverse NeRF and PoseCNN

Sibo Wang
Robotics
University of Michigan
Ann Arbor, United States
sibo@umich.edu

Yuxi Zhang
Robotics
University of Michigan
Ann Arbor, United States
yuxii@umich.edu

Yulun Zhuang
Robotics
University of Michigan
Ann Arbor, United States
yulunz@umich.edu

***Abstract*—In this paper, we present an efficient and robust system for view synthesis and pose estimation, integrating PoseCNN and iNeRF to achieve faster convergence and better camera pose estimation. By employing PoseCNN to provide an initial guess for the camera pose in iNeRF and generate object labels to mask out unrelated objects in the scene, we accelerate the optimization process in iNeRF. We demonstrate the effectiveness of our method through extensive experiments on the PROPS dataset, reducing translation and rotation errors by 44.1% and 44.2%, respectively, while boosting the convergence time by 104%. Our work has the potential to enhance the adaptability of iNeRF to a wide range of scenarios and applications.**

I. Introduction

The estimation of six degree of freedom (6DoF) camera pose is a critical task in robotics, augmented reality, and computer vision [1], [2]. Recent advances in deep learning and differentiable rendering have led to new techniques for pose estimation based on analysis-by-synthesis [3], [4]. However, most of these methods require accurate 3D models of objects or scenes, which can be challenging and time-consuming to obtain.

Neural Radiance Fields (NeRF) is a recent breakthrough in generative modeling that enables the capture of complex 3D structures and optical properties from just a few RGB images [5], [6]. The NeRF model can be used for rendering novel views of the scene or object. iNeRF is then proposed, which inverts a NeRF model for 6 DoF pose estimation [7]. iNeRF takes three inputs: an observed image, an initial estimate of the pose, and a NeRF model. It computes the appearance differences between the pixels rendered from the NeRF model and the pixels from the observed image. The gradients for the estimated pose are the obtained via backpropagation. As illustrated in Figure 1, this procedure is repeated iteratively until the rendered and observed images are aligned, thereby yielding an accurate pose estimate.

Despite its compelling result, a major limitation of iNeRF is the constraints on initial estimate of the pose. For complex scenes, iNeRF needs an informative prior pose fro the model to converge. Specifically, the rotational uncertainty needs to be within 40 degrees along an axis from the unit sphere. The translational uncertainty needs to be within 0.2 meters [7]. To address this issue, we propose to enhance the performance of iNeRF by integrating the pose and label of objects from PoseCNN [8]. Specifically, we assume a known relationship between the pose of an object and the frame of NeRF, which allows us to derive a more accurate initial camera pose based on object pose and frame transformations. Such initial estimate also relaxes the above-mentioned constraints. Additionally, we incorporate pixel-level labels in the key-point sampling process to replace SIFT.

To summarize, by integrating iNeRF with PoseCNN, we (i) obtain more accurate initial camera pose based on more relaxed and non-informative prior. (ii) introduce masked region sampling based on pixel labels to remove the use of SIFT. (iii) increase the overall convergence rate of iNeRF.

II. Related Work

In our project, we integrate PoseCNN and iNeRF for pose estimation and neural 3D shape representation, respectively, to enable faster convergence of the rendering process and better pose estimation.

A. *Pose Estimation from RGB Images*

Classical methods for object pose estimation detect and match keypoints with known 3D models, but they necessitate objects with rich textures to identify feature points. In contrast, recent deep-learning approaches propose CNN-based architectures for directly estimating object poses. For instance, PoseCNN estimates the 6D pose of objects using single RGB images and convolutional neural networks [8]. The network takes an RGB image as input and outputs a 6D pose comprising three translational and three rotational parameters.

By decomposing the pose estimation task into distinct components, PoseCNN explicitly models their dependencies and independencies. Initially, the network predicts an object label for each pixel in the input image. Subsequently, it calculates the unit vector from each pixel toward the center to ascertain the 2D coordinates of the object center, while also estimating the distance to the center. With these estimations, the model recovers the 3D translation (T). To determine the quaternion representation of rotation (R), the network regresses convolutional features within the object's bounding

This is the final project for course DeepRob. The open-source code of the project can be found in https://github.com/silvery107/fast-inerf

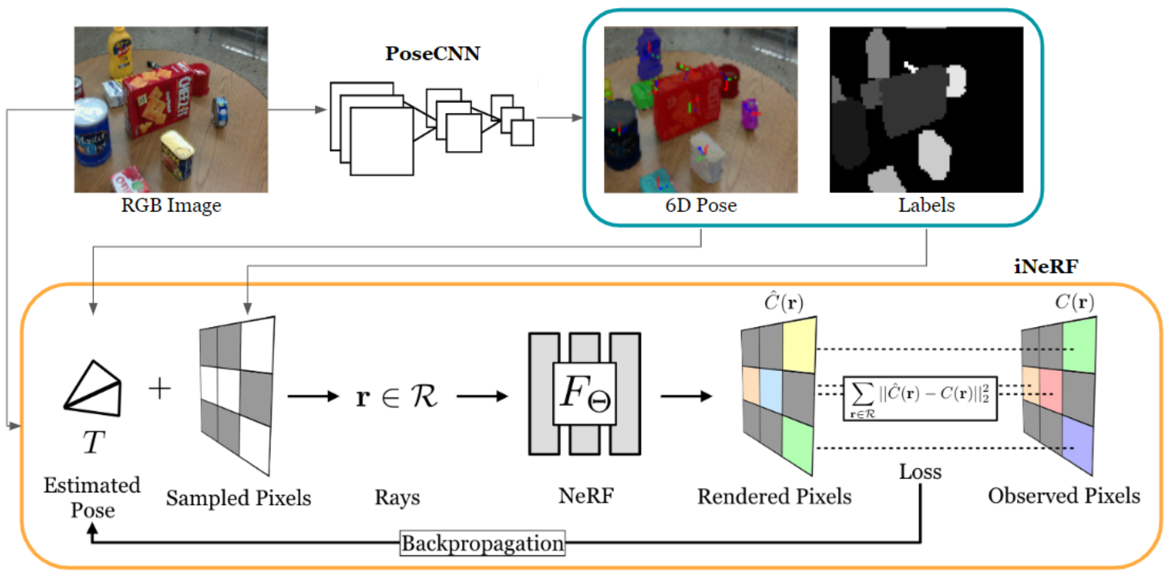


Fig. 1: System architecture overview

box. PoseCNN is trained to minimize the discrepancy between ground truth and predicted poses using a novel loss function that accommodates symmetric objects.

In our work, we employ PoseCNN to estimate object pose as an initial guess for the camera pose in iNeRF. Furthermore, the object labels generated by PoseCNN serve to mask out unrelated objects in the scene, accelerating the rendering process in iNeRF.

B. Neural 3D shape representations

NeRF optimizes continuous volumetric scene functions using sparse input views to synthesize novel perspectives of intricate environments [5]. By representing these scenes as 5D neural radiance fields, NeRF can capture complex geometry and materials through a basic MLP network. The input to this network consists of a 3D location and viewing direction, which are combined to predict volume density and view-dependent radiance at specific spatial points. The model then generates views by querying 5D coordinates along camera rays and projects colors and densities into images using volume rendering techniques.

NeRF excel in capturing high-frequency scene content, and when combined with volume rendering techniques, they enable a learned neural network to produce photo-realistic view synthesis. PixelNeRF is an extension that addresses NeRF's slow training process, predicting NeRF models from input images to generalize across various scenes or objects [9].

Unlike NeRF and its variants that learn a scene's structure from posed RGB images, iNeRF tackles the inverse problem of localizing new observations with unknown camera poses [7]. iNeRF estimates the camera pose from an input image by inverting the neural radiance field. It minimizes the residual between pixels rendered from a NeRF and those in an observed image, optimizing the camera pose. Leveraging NeRF's differentiable rendering nature, iNeRF uses gradient-based optimization methods to determine the optimal camera pose.

Building on the foundation of NeRF, PixelNeRF, and iNeRF, our work aims to develop a more efficient and robust system for view synthesis and pose estimation. By integrating PoseCNN into the iNeRF framework, we achieve faster convergence and eliminate the need for an initial pose assumption, enhancing the model's adaptability to a wide range of scenarios.

III. Algorithmic Extension

As shown in Figure 1, we propose to enhance the performance of iNeRF by integrating the pose and label of objects from PoseCNN. Specifically, if we assume known relationship between the pose of an object and the frame of NeRF, then a more accurate initial camera pose can be derived based on object pose and frame transformations. Meanwhile, the pixel-level labels can serve as key-point sampling in the iNeRF pipeline to replace SIFT.

A. Pose initialization

In vanilla iNeRF, the authors have noted some restrictions on the initial pose estimate [7]. The rotation error between the true pose and the initial estimate should be on "an axis along the unit sphere" and within $[-40, \ 40]$ degrees. The translation error should be within $[-0.2, \ 0.2]$ meters for simplistic scenes and $[-0.1, \ 0.1]$ meters for complex scenes. We propose to use pose estimate from PoseCNN to remove or relax these restrictions.

To utilize the object pose for camera pose initialization, we must assume a relationship between the 6D pose of the center object in the scene and the reference frame of the scene's NeRF model. Assumptions can take the forms of

1) The *6D pose* of an object is aligned with the NeRF frame, or the transformation between them is known.
2) The *position* of an object is at the origin of the NeRF frame, or the translation between them is known.

The first assumption relies on careful construction or extensive knowledge of the NeRF model, as we need to determine the rotational relationship between object poses and the NeRF frame. Let the known rotation and translation of the object in Nerf frame be $\mathbf{R_n}$ and $\mathbf{p}$. Further, let the rotation and and translation of the object in PoseCNN camera frame be $\mathbf{R_c}$ and $\mathbf{r}$, then based on the geometric relationship shown in Figure 2, the rotation of the camera in NeRF frame can be found by

$$\mathbf{R} = \mathbf{R_n}\mathbf{R_c}^\top \tag{1}$$

the translation of the camera in NeRF frame can be found by

$$\mathbf{t} = \mathbf{R}(\mathbf{r} - \mathbf{p}) \tag{2}$$

Thus, if the first assumption is satisfied, then the camera pose can be estimated without any restrictions.

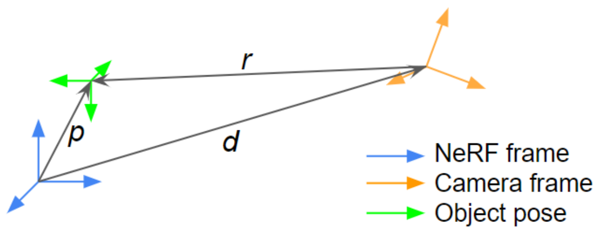


Fig. 2: Relationship between frames and poses

Comparatively. the second assumption is more relaxed and acceptable in common scenes. As the origin of NeRF is at the center of 3D scene box, in general, the translation between the origin and the position of an object can be determined. For instance, for the scene of PROPS Pose Dataset, we assume the position of the red cracker box is aligned with the NeRF frame's origin.

Under this assumption, we are able to relax the camera translation assumptions in vanilla iNeRF. Assuming the axis of translation is given, we can accurately estimating the magnitude of the translation using $\|\mathbf{d} - \mathbf{p}\| = \|\mathbf{r}\|$. Without loosing generality, let the axis of translation be d_1, then based on Figure 2,

$$\begin{aligned} (d_1 - p_1)^2 + (d_2 - p_2)^2 + (d_3 - p_3)^2 &= r_1^2 + r_3^2 + r_2^2 \\ \Rightarrow (d_1 - p_1)^2 &= r_1^2 + r_3^2 + r_2^2 - (d_2 - p_2)^2 - (d_3 - p_3)^2 \end{aligned}$$

where the quadratic can then be solve analytically. In other words, since we can accurately estimate the translation, we can remove the restrictive bounds. For the scene of PROPS Dataset, we assume $\mathbf{p} = \mathbf{0}$ and thus

$$d_1 = \sqrt{r_1^2 + r_3^2 + r_2^2 - d_2^2 - d_3^2} \tag{3}$$

B. Mask region sampling

The output from PoseCNN can be further utilized in the points sampling stage in iNeRF. We first briefly describe the rendering process and the importance of key-point sampling strategy. Then, we introduce and explain our novel *masked region sampling*, whic is based on PoseCNN.

For each iteration of estimated camera pose, iNeRF needs to render the pixels with NeRF and compare the error with the objective image. To render each pixel in NeRF, we need to compute the integral for the corresponding projected ray,

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt \tag{4}$$

where $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$ denotes probability that the ray travels from t_n to t_f without hitting any other particle [5]. NeRF discretizes the integral and solves weighted values for n sampled points. Thus, $\mathcal{O}(nHW)$ froward/backward passes are required, where H, W are the image height and width. The computation is is both time and memory consuming. Specifically, the memory required for back-propagation of all pixels is not affordable on any commercial GPU [7].

Thus, iNeRf introduced *interest region sampling*. The strategy uses an interest point detector, such as SIFT, to localize a set of candidate pixel locations in the observed image. The pixel are then dilated to form interest regions, from which M points are randomly sampled and used in the rendering-and-compare pipeline. The dilation is to prevent the method from only considering interest points on the observed image instead of interest points from both the observed and rendered images.

In our proposed model, naturally, the pixel-level object labels can be used as a mask for sampling points. Thus, we propose a new key-point sampling strategy, *masked region sampling*. First, we create a mask of all pixels that are labeled as objects (excluding background). Then, we randomly sample N points from the masked region, where N is a hyper-parameter. Compared to *interest region sampling*, *mask region sampling* removes the use of external feature extraction algorithm (SIFT), while achieving the similar end goal, which is to sample in regions where objects are likely to reside.

IV. Experiments and Results

In this section, we describe the experimental design and associated results that are used to analyze our method.

A. Experimental Setup

We utilize the PROPS Pose dataset [10] to train a NeRF representation of the scene of a set of daily life objects. To get a complete representation of the scene, we combine both training and validation sets of RGB images (add up to 1000) and uniformly sample 50, 100 and 150 images to form the primitive source images, since the NeRF model trains best with between 50-150 images [6] which exhibit minimal scene movement, motion blur or other blurring artifacts. We use 100 images for final NeRF training. The peak signal-to-noise ratio (PSNR) of each dataset setup is reported on Table I. Even

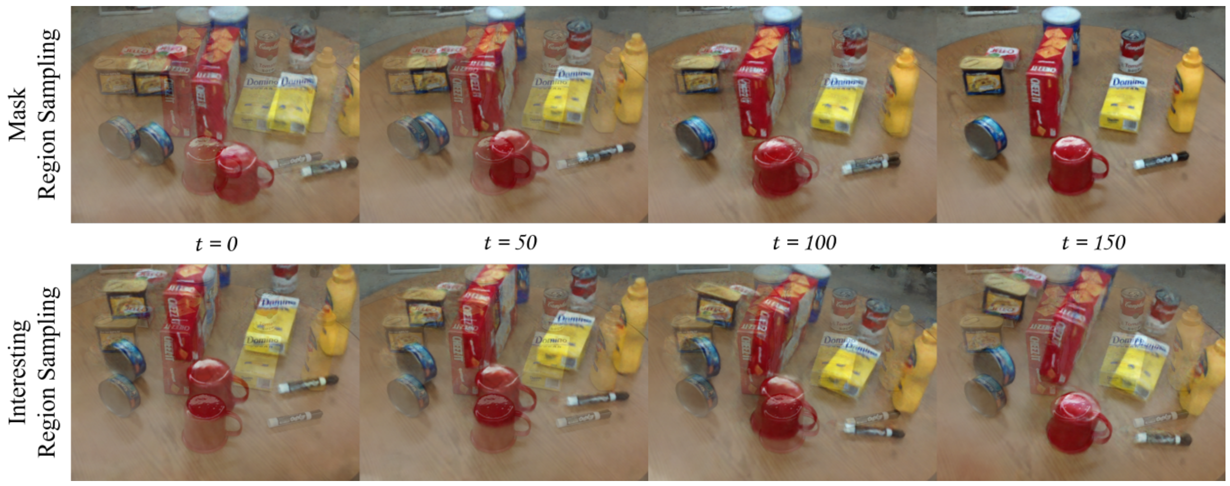


Fig. 3: Visualization of different sampling methods. We visualize the average of rendered images based on the estimated pose at time t and the test image. Adopting Mask Region Sampling helps our method to fast recover camera poses that align the rendered and test image to fine details. Interesting Region Sampling aligns the red mug, but fails to align the rest scene.

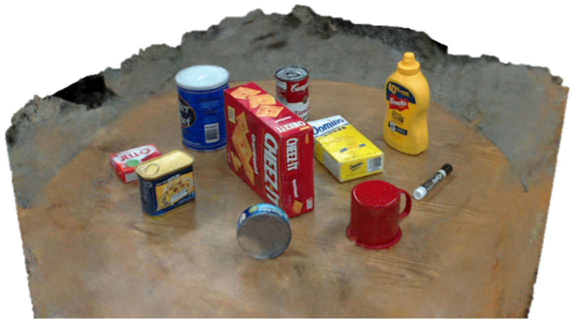


Fig. 4: NeRF representation of the PROPS pose dataset

though the camera extrinsic to each object in each image are given, we still need to estimate the camera extrinsic to the scene, so we take the advantage of the existing open-source software COLMAP [11] to extract the necessary camera data using structure from motion (SfM) and store them in a format compatible with the original NeRF codebase.

The PROPS NeRF model is trained using a PyTorch implementation of the original NeRF codebase for 100000 steps to get the best PSNR performance (Table I) and the training time is about 6 hours on a single Nvidia RTX 3060 GPU. The rendering view of its reconstruction is shown in Figure 4. The PoseCNN model is pre-trained on the PROPS Pose dataset and has 51% 6D pose prediction errors within 5° rotation error and 5 cm translation error.

In evaluation, we randomly sampled a view of the scene as the input image to our system and extract 6D pose predictions and segmentations of each object in that view. We apply Pose Initialization using the pose of the red creaker box as the closest estimation of NeRF origin to get a initial guess of the camera pose and then apply Mask Region Sampling to generate regions containing interesting pixels.

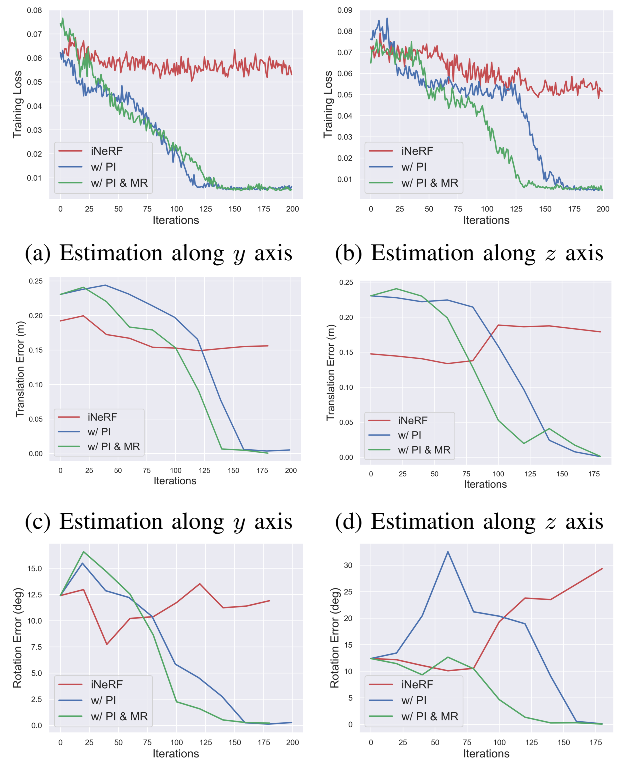
The estimation of camera pose is optimized by comparing the mean square error (MSE) between rendered image and the target image only on a random sampled batch of interesting pixels. We choose the batch size N to be 1024 [7] to have a good balance between convergence speed and computational efficiency as demonstrated in the original iNeRF paper. The gradients from these residuals are then backpropagated through the NeRF model to produce the gradients for the estimated pose. As illustrated in Figure 1, this procedure is repeated iteratively until the rendered and observed images are aligned.

To evaluate our method, we estimate d_1 along x, y and z axis separately and measure the analytical loss (Figure 5a and 5b), translation error (Figure 5c and 5d) and rotation error (Figure 5e and 5f) with respect to optimization iterations. We only plot loss and error curves for y and z axis for trending analysis but quantitative results are given on Table II by averaging 3 experiment runs.

B. Results

In Table I, we compare the effect of the number of images used for NeRF model training and also the convergence over training steps of each model. We found that PROPS NeRF performs well with fewer than 100 images but too few images will lead to blurry rendering. However, the NeRF model will struggle and may hallucinate ”floaters” at the boundaries of the box if too many images are given. We also found that PROPS NeRF model usually converges around 100000 training steps, and larger training steps may lead to overfit and decrease the PSNR. Thus, we prepare 100 images from PROPS Pose dataset to form the PROPS NeRF dataset and train it for 100000 steps.

Table II shows the performance of our algorithm extension based on the original iNeRF implementation. With Pose Initialization and Mask Region Sampling techniques, our method can achieve 44.1% and 44.2% improvements on mean translation errors and mean rotation errors respectively, and also 2.04 times faster to converge to the target camera pose. In ablation



(a) Estimation along y axis (b) Estimation along z axis

(c) Estimation along y axis (d) Estimation along z axis

(e) Estimation along y axis (f) Estimation along z axis

Fig. 5: Training loss, translation errors and rotation errors w.r.t. optimization iterations. PI stands for Pose Initialization and MR stands for Mask Region Sampling.

TABLE I: Comparisons between number of images and training steps for PROPS NeRF model

Number of Images	PSNR		
	50k	100k	150k
50	22.84	24.13	24.54
100	22.26	**26.08**	25.01
150	21.79	25.01	24.22

study, we only apply Pose Initialization for pose estimation. The results shows a 9% decrease on translation errors and 13.6% decrease on rotation errors, and also requires 33 more steps optimization iterations to convergence. The plot of one experiment run with respect to optimization iterations for y and z axes is shown in Figure 5.

TABLE II: Comparisons of pose estimation performance between original iNeRF and iNeRF enhanced with Pose Initialization and Mask Region Sampling

	w/ PI & MR	w/ PI	iNeRF [7]
Mean Tanslation Error < 5cm	**0.98**	0.89	0.68
Mean Rotation Error $< 5°$	**0.88**	0.76	0.61
Iterations to Convergence	**122**	155	250

Intuitively, we visualize the average of rendered images based on the estimated pose at time t and the test image for different sampling methods in Figure 3. Adopting Mask Region Sampling helps our method to fast recover camera poses that align the rendered and test image to fine details. Interesting Region Sampling aligns the red mug, but fails to align the rest scene.

V. Conclusions

In this paper, we have presented an efficient and robust system for view synthesis and pose estimation by integrating PoseCNN and iNeRF. Our method leverages the pose and object label predictions from PoseCNN to improve the initial camera pose estimation and accelerate the optimization process of camera pose estimation in iNeRF. We have demonstrated the effectiveness of our approach through extensive experiments on the PROPS dataset, achieving significant improvements in both translation and rotation errors while reducing convergence time.

Our work contributes to the ongoing efforts to develop more efficient and robust systems for view synthesis and pose estimation, enhancing the adaptability of iNeRF to a wide range of scenarios and applications. In future work, we plan to explore the integration of additional deep-learning approaches for pose estimation and investigate the potential of our method in real-world applications, such as robotics, augmented reality, and virtual reality.

References

[1] L. Manuelli, W. Gao, P. Florence, and R. Tedrake, “KPAM: KeyPoint affordances for category-level robotic manipulation,” in *Springer Proceedings in Advanced Robotics*. Springer International Publishing, 2022, pp. 132–157.

[2] P. Marion, P. R. Florence, L. Manuelli, and R. Tedrake, “Label fusion: A pipeline for generating ground truth labels for real RGBD data of cluttered scenes,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2018.

[3] X. Chen, Z. Dong, J. Song, A. Geiger, and O. Hilliges, “Category level object pose estimation via neural analysis-by-synthesis,” in *Computer Vision – ECCV 2020*. Springer International Publishing, 2020, pp. 139–156.

[4] K. Park, A. Mousavian, Y. Xiang, and D. Fox, “Latentfusion: End-to-end differentiable reconstruction and rendering for unseen object pose estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[5] 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,” *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[6] T. Müller, A. Evans, C. Schied, and A. Keller, “Instant neural graphics primitives with a multiresolution hash encoding,” *ACM Transactions on Graphics (ToG)*, vol. 41, no. 4, pp. 1–15, 2022.

[7] L. Yen-Chen, P. Florence, J. T. Barron, A. Rodriguez, P. Isola, and T.-Y. Lin, “inerf: Inverting neural radiance fields for pose estimation,” in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1323–1330.

[8] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, “Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes,” *arXiv preprint arXiv:1711.00199*, 2017.

[9] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, “pixelnerf: Neural radiance fields from one or few images,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4578–4587.

[10] X. Chen, H. Zhang, Z. Yu, S. Lewis, and O. C. Jenkins, “Progresslabeller: visual data stream annotation for training object-centric 3d perception,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 13 066–13 073.

[11] J. L. Schonberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4104–4113.