SUP-NeRF: A Streamlined Unification of Pose Estimation and NeRF for Monocular 3D Object Reconstruction

Yuliang Guo[1⋆] Abhinav Kumar[1,2] Cheng Zhao[1]
Ruoyu Wang[1] Xinyu Huang[1] Liu Ren[1]

[1]Bosch Research North America & Bosch Center for Artificial Intelligence (BCAI)
[2]Michigan State University
[1][yuliang.guo2,cheng.zhao,ruoyu.wang,xingyu.huang,liu.ren]@us.bosch.com
[2]kumarab6@msu.edu
https://yuliangguo.github.io/supnerf

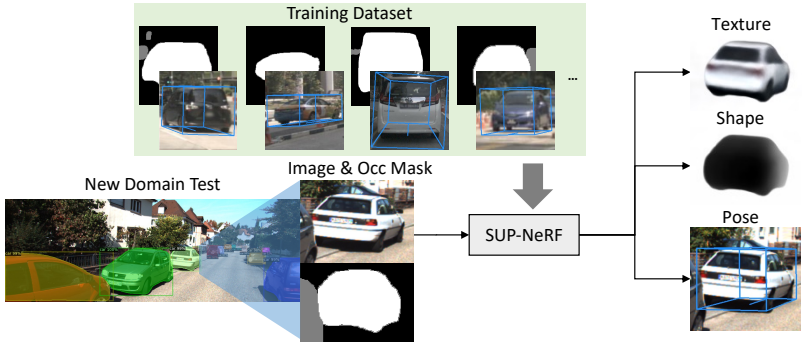


Fig. 1: Teaser. SUP-NeRF is a unified solution that predicts an object's pose, shape, and texture using a single network. SUP-NeRF is trained on real driving scenes with imprecise labels, and it adapts robustly to new cross-dataset scenarios.

Abstract. Monocular 3D reconstruction for categorical objects heavily relies on accurately perceiving each object's pose. While gradient-based optimization in a NeRF framework updates the initial pose, this paper highlights that scale-depth ambiguity in monocular object reconstruction causes failures when the initial pose deviates moderately from the true pose. Consequently, existing methods often depend on a third-party 3D object to provide an initial object pose, leading to increased complexity and generalization issues. To address these challenges, we present SUP-NeRF, a Streamlined Unification of object Pose estimation and NeRF-based object reconstruction. SUP-NeRF decouples the object's dimension estimation and pose refinement to resolve the scale-depth ambiguity, and introduces a camera-invariant projected-box representation that generalizes cross different domains. While using a dedicated pose estimator that smoothly integrates into an object-centric NeRF, SUP-NeRF is free from external 3D detectors. SUP-NeRF achieves state-of-the-art results in both reconstruction and pose estimation tasks on the nuScenes dataset. Furthermore, SUP-NeRF exhibits exceptional cross-dataset generalization on the KITTI and Waymo datasets, surpassing prior methods with up to 50% reduction in rotation and translation error.

⋆ Project Lead

1 Introduction

Monocular 3D object reconstruction [8,9,13,19,49] is a critical technology with broad applications in autonomous driving [18, 29, 49], AR/VR, robotics [43], and embodied AI [1,40,51]. Its impact is particularly significant in autonomous driving, where it plays a vital role in tasks like auto-labeling [49] and world-model construction [18,29]. This technology is essential for reconstructing 3D models from typical driving scenarios, where single-view observations are common. This paper focuses explicitly on the problem of recovering an object's pose, shape, and texture from a single image under real driving scenarios, see Fig. 1. Our aim is to devise an efficient solution, thereby unlocking the full potential of monocular reconstruction for active tasks such as planning and control.

Neural Radiance Field (NeRF) technology [27,30,48] has revolutionized the field of 3D reconstruction. It excels at capturing fine details in a scene and generating novel views from existing reconstructions. An exciting branch, Object-centric NeRF [11,12,18,29,34,45,46], takes things further, allowing for flexible data creation and tackling reconstruction challenges from limited number of views by incorporating category-specific prior knowledge. Earlier object-centric NeRF methods [8, 12, 13, 19, 46, 48, 49] require densely overlapping views and precise poses, limiting their use to synthetic data. Later advances [18,29] enable monocular training with imperfect masks.

Although promising, object NeRFs show limitations in correcting erroneous object poses. First, the well-known scale-depth ambiguity [16,25,36] also affects the object NeRF optimization. Fig. 2 shows the optimization failure of a monocular NeRF method under longitudinal translation error when it is given complete freedom in adjusting the scale within the normalized space. Secondly, NeRF also struggles with large rotation errors. Appendix A6.1 shows that NeRF typically requires the initial rotation error to be below 25°, and the pose optimization cannot go lower than 7°.

Recent papers [18,29] use an external monocular 3D detector to provide initial poses to NeRF. However, monocular 3D detectors suffer from generalization issues [16,17] and increased complexity, ultimately affecting the NeRF pipeline. BootInv [34], a more recent method, tackles pose estimation, shape, and appearance reconstruction in a single framework but requires additional pre-training of dense field generator and struggles with handling moderate occlusions.

To resolve the scale-depth ambiguity and achieve greater generalization, this paper proposes SUP-NeRF, a Streamlined Unification of Pose estimation and NeRF. SUP-NeRF uses an image encoder to output the object dimensions and then refines for object pose using an iterative process while keeping its size fixed. Besides decoupling of an object's dimension and pose to resolve the scale-depth ambiguity, the introduction of a novel projected-box representation in the iterative update refinement also enhances the generalization to novel data since the representation makes the deep pose refiner independent from camera intrinsic parameters. In addition, SUP-NeRF tackles the complexity challenge by sharing the NeRF image encoder with the pose estimation, making the pose estimation module a seamless fit for object NeRFs.

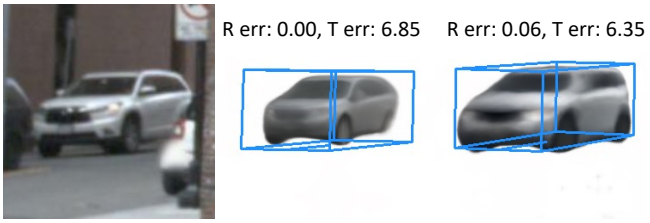


Fig. 2: Scale-depth ambiguity in NeRF. Given the input image (left), joint optimization of pose, shape, and texture in NeRF has full freedom to rescale the shape within the normalized shape space (blue box) or move the 3D box. Such phenomenon is observed from the evolution of the rendered objects from iteration 0 (middle) to iteration 50 (right).

In our experiments, we first comprehensively benchmark and conduct ablations on SUP-NeRF on the nuScenes [3] dataset. SUP-NeRF outperforms state-of-the-art (SoTA) methods and achieves SoTA object reconstruction and pose performance on this dataset. We next conduct cross-dataset object reconstruction and pose estimation experiments of the nuScenes models on the KITTI and Waymo datasets to test SUP-NeRF's generalization capability. SUP-NeRF shows more significant improvements over SoTA in such cross-dataset settings.

In summary, our main contributions include:

- We introduce SUP-NeRF, an streamlined method unifying object pose estimation and monocular object-centric neural reconstruction.
- We identify depth-scale ambiguities as the key barrier to object NeRF's pose optimization and craft a module to overcome this effectively, while also integrating seamlessly with existing object NeRF frameworks.
- We introduce a novel projected box representation for the pose estimation module, enabling the deep refiner to operate in a space invariant to camera intrinsic parameters, thereby enhancing SUP-NeRF's generalization capability.
- SUP-NeRF achieves SoTA object reconstruction and pose estimation results, outperforming other methods on the nuScenes dataset, and in challenging cross-dataset experiments on both KITTI and Waymo datasets.

2 Related Work

Object-centric NeRF. Object-centric NeRF [11, 12, 18, 29, 34, 45, 46] is a specialized NeRF variant for modeling specific object categories. Its pipeline feeds a NeRF decoder with shape and texture codes, generating occupancy and color values at 3D positions, followed by volumetric rendering to produce a rendered image. The entire process is differentiable, allowing back-propagation of the photometric loss to update shape codes, texture codes, network weights, and object poses. Early object-centric NeRFs [8, 12, 13, 19, 45, 46, 48, 49] required accurate object poses, full visibility, and dense multi-view observations to reconstruct a categorical model, making them inaccessible for real-world applications. Some approaches [12, 23] claim successful object pose recovery through gradient-based

Table 1: Literature Summary. SUP-NeRF sets a benchmark for minimal requirements, eliminating the need for external 3D detectors, CAD models, or dense NOCS generators. It is dedicated to the reconstruction of object shapes and pose estimation directly from a single image of driving scenes, employing an object-centric framework.

Methods	Indoor	Outdoor	Views	Scope	CAD	wt. Det3D	wt. NeRF	NOCS Gen.
6DoF Pose [21,26]	✓	×	single	object	✓	✓	×	×
3D Detector [16,42]	×	✓	single	image	×	✓	×	×
NeRF [27]	✓	✓	multi	scene	×	×	✓	×
Object-NeRF [18,29]	×	✓	single	object	×	×	✓	×
BootInv [34]	×	✓	single	object	×	✓	✓	✓
SUP-NeRF	×	✓	single	object	×	✓	✓	×

pose refinement from monocular images. However these successes are still limited to synthetic dataset, and oversimplified object/camera poses. Recent methods [18,29] investigate more practical pipelines demonstrating the object-NeRF's potential to train and test on driving scenes with limited views and occluded object data. However, both methods require relatively accurate poses from heavy third-party 3D detectors [32,36]. Other recent works [2,5] try freeing NeRF from external camera poses; however, solving camera poses from densely overlapping views is a different problem than recovering an object's pose from a single image. Compared to these methods, SUP-NeRF recovers object pose from a single image without a 3D detector in the pipeline. A more recent work [34] also unifies pose estimation and NeRF in a single network, and enables high-fidelity object generation. However, its training requires additional NOCS dense generator and it is highly sensitive to moderate occlusions compared to our method.

Monocular 3D Object Pose Estimation. Advancements in deep learning and the availability of 3D-labeled datasets [3, 6, 10] enable many monocular 3D object detection [16, 24, 25, 33, 36, 42, 47] and 6DoF pose estimation methods [14, 21, 22, 26, 28, 39, 44, 50]. Although estimating 3D information from a single image is ill-posed due to depth-scale ambiguity, leveraging constraints like CAD models or sizes enables accurate object pose recovery via PnP processes [35, 39, 41] or iterative optimization [14, 21]. These indoor 6DoF pose estimation mostly follow an object-centric design which aims to estimate highly accurate object pose within the 2D image patches out of a previous 2D object detector. Recent studies [4, 20] use the NeRF pipeline to reconstruct 3D models or to train model-based keypoint detectors to estimate object poses during inference. However, none of these model-based methods is ideal for outdoor driving scenarios, as pre-reconstruction of models is often impossible in new scenes. Fortunately, inferring the 3D dimensions of objects from an image is feasible when its appearance is tightly bounded to its fine-grained class [28], which keeps an outdoor 3D detector from being fully ill-posed. Nowadays, reliable outdoor 3D detectors usually favors a image-centric design to direct detect many 3D objects captured within the whole image, which usually requires a large backbone network [16, 42]. However, the performance of SoTA detectors drops significantly when applied to datasets with distribution different from the training data [16, 25, 42].

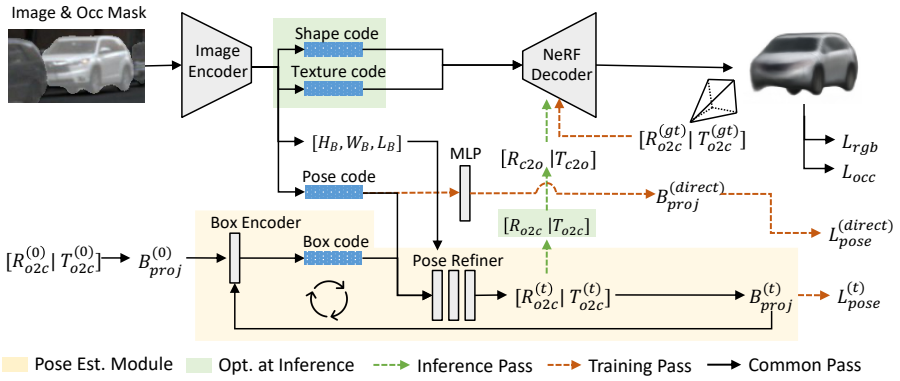


Fig. 3: SUP-NeRF Overview. SUP-NeRF unifies pose estimation and NeRF. The pose estimation module enables SUP-NeRF to work for objects in diverse poses **without** external 3D detectors. The increase of complexity only constitutes a few MLP layers.

Tab. 1 outline the differences in methodologies. Unlike existing approaches, SUP-NeRF unifies pose estimation and object NeRF in a streamlined manner, adopting a straightforward object-centric design akin to indoor techniques. Without the need for CAD models, SUP-NeRF is suitable for outdoor driving scenarios. Despite its focused scope on object-centric pose estimation, SUP-NeRF showcases enhanced cross-domain generalization capabilities compared to more complex image-centric 3D detectors within this realm.

3 SUP-NeRF

Our goal is to create a NeRF framework that bypasses NeRF's limitations without external 3D detectors. Fig. 3 shows the overall pipeline of SUP-NeRF, a streamlined unified network that predicts object pose, shape, and texture from a single image. SUP-NeRF consists of three modules: a Resnet-based encoder, a pose estimation module, and a NeRF decoder. SUP-NeRF processes an image and its occlusion mask, with the encoder turning the masked image into shape, texture, and pose codes and predicted object dimensions. The pose code, object dimensions and a box code, are passed to the pose estimation module to iteratively refine the object pose $[R_{o2c}|T_{o2c}]$. After a few iterations, the estimated pose is converted to the camera pose $[R_{c2o}|T_{c2o}]$ and passed to the NeRF decoder. In the NeRF phase, the decoder uses the shape, texture codes, and camera pose for volumetric rendering, creating both RGB and occupancy images.

Pose Estimation Module. The key component of SUP-NeRF is the pose estimation module, which we show in Fig. 4. It aims to provide reliable object poses for targets at all ranges and orientations. Our pose estimation module iterative updates the input pose, based on the visual difference between the input pose and the observed object in the image. Given predicted object dimensions $[H_B, W_B, L_B]$, camera intrinsic K, and current pose $R^{(t)}, T^{(t)}$, image projections

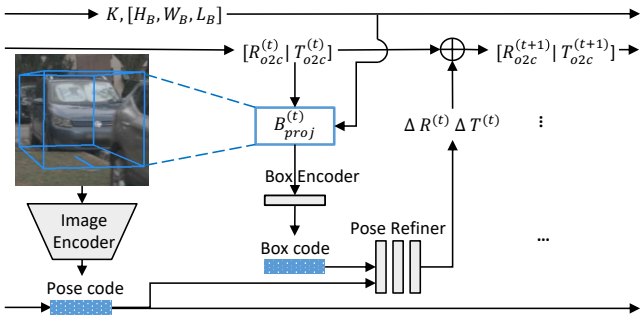


Fig. 4: SUP-NeRF Pose Estimation Module. The pose estimation module of SUP-NeRF iteratively updates the object's pose while preserving scale. It takes the projection of 3D box corners as a visual representation of the input pose and estimate the pose update via comparing it to observed image in a latent embedding space. These designs handle scale-depth ambiguity and make the deep refiner independent from camera intrinsic parameters for better cross-domain generalization.

of the 3D box corners $B_{proj}^{(t)}$ is computed, which can be interpolated as a visual representation of current pose. After encoding the 8 corners into a higher dimensional box code, SUP-NeRF feeds into the pose refiner along with a pose code from the image encoder to predict the desired pose update $\Delta R^{(t)}, \Delta T^{(t)}$. SUP-NeRF then composes the pose changes with current pose to compute the next pose state, $R^{(t+1)}, T^{(t+1)}$. This updating process starts from a fully random object pose within the viewing frustum of the image patch, and takes a few iterations such that the final pose approaches the true pose even from a random initial pose.

The pose estimation module incorporates the following unique designs:

- SUP-NeRF preserves the object dimensions to ensure unambiguous pose updates. Our implementation, using projected 3D boxes on the image plane, leads to a concise encoding of shape and pose, reducing computation in image rendering and encoding per iteration, as opposed to previous methods [21, 29] relying on full CAD models. Since this representation makes the deep pose refiner's task independent from camera intrinsic parameters, it leads to better generalization to different testing domains, as evidenced in Tabs. 3 and 4.
- SUP-NeRF updates the object pose iteratively rather than conduct a direct pose estimation using a PnP-based [14] because the later is sensitive to missed or swapped corners thus often produces far out-of-distribution poses. Our ablation studies in Tab. 6 confirm this difference.
- SUP-NeRF represents the rotation update in axis-angle representation [37]. During rotation updating process, we convert the axis-angle space 3D rotation $\Delta R^{(t)}$ back to $SO(3)$, compose it with the previous rotation to compute the new rotation state as $R_{o2c}^{(t+1)}$.
- SUP-NeRF represents the translation update in a relative space, indicated as $\Delta T^{(t)} : [v_x^{(t)}, v_y^{(t)}, \rho^{(t)}]$, where $v_x^{(t)}$ and $v_y^{(t)}$ indicate the location change in

image space, and $\rho^{(t)}$ indicates the ratio between the target depth and the source depth. Specifically, SUP-NeRF updates the translation in image space and depth as:

$$\begin{aligned} x^{(t+1)} &= x^{(t)} + v_x^{(t)} \\ y^{(t+1)} &= y^{(t)} + v_y^{(t)} \\ Z^{(t+1)} &= \rho^{(t)} Z^{(t)} \end{aligned} \tag{1}$$

where (x, y) represent the object's 2D location in image space, Z indicates the depth. The new 3D location $T_{o2c}^{(t+1)}$ is then computed from the updated image coordinates, depth and camera intrinsic matrix K. Compared to [21, 34], our definition of $\rho^{(t)}$ leads to a more straight-forward update.

During the pose updating process indicated by $\oplus$, both the rotation and translation are updated. Through such representation and updating scheme, SUP-NeRF handles objects at varying distances with the same output space. Further discusses the choice of an effective pose representation and the right coordinate frame are included in Appendices A1 and A2.

Cross-task Shortcut. Simultaneously perceiving pose, shape, and texture, requires an image encoder with shared initial layers but separate later stages. However, encoding shape, texture, and pose is a challenge as they require different features, making it tough for a shared encoder to meet all needs effectively. Shape and texture encoding demand a pose-invariant approach, as they correspond to each object's frame. On the other hand, pose encoding needs to capture sufficient pose information for later use. This difference leads to conflicting learning signals during joint training, potentially weakening the model's performance.

SUP-NeRF proposes a straightforward one-step approach to resolve this conflict. After processing through the first four CNN layers separately, SUP-NeRF introduces a cross-task shortcut that deducts pose features from the shape and texture features. This design choice eliminates the shape and texture's reliance on pose, allowing the earlier layers to focus on pose-dependent features for all tasks. This shortcut turns the conflict into a cooperative relationship, enhancing the joint training of different modules, as demonstrated in Tab. 7. The details of our image encoder is described in Appendix A3.

Training. SUP-NeRF closely follows the loss functions and occupancy mask definitions from AutoRF [29]. The total loss during training is: $L_{train} = L_{rgb} + w_{occ}L_{occ} + w_{pose}(L_{pose}^{(direct)} + \sum_t^K L_{pose}^{(t)})$, where w_{occ} and w_{pose} indicate the corresponding weights.

We compare the RGB output to the input image to calculate photometric loss L_{rgb}, and the occupancy values to the mask for occupancy loss L_{occ}. SUP-NeRF additionally compares the projected box corners from the integrative pose estimator ${B^{(t)}}_{proj}$ with the ground-truth values to compute the pose losses $L_{pose}^{(t)}$. We define the pose loss function as the mean distance between the matched corners: $L_{pose}^{(*)} = \frac{1}{8}\sum_{i=1}^{8}\sqrt{(x_i^{(*)} - \hat{x}_i)^2 + (y_i^{(*)} - \hat{y}_i)^2}$, where $(*)$ indicates it compatible with both the direct pose loss and the iterative pose loss, $(x_i^{(*)}, y_i^{(*)})$ denote a input corner position, and $(\hat{x}_i, \hat{y}_i)$ indicates a ground-truth corner po-

sition. SUP-NeRF introduces an additional MLP layer for more accurate pose estimation to directly supervise the 2D coordinates of box corners predicted from the pose code.

In training, only the network weights are treated as the optimizable variables. Moreover, the NeRF decoder uses ground-truth camera poses $[R^{gt}_{c2o}|T^{gt}_{c2o}]$ instead of predictions from the pose estimation module to ensure more reliable training.

Implementation Details. SUP-NeRF's image encoder backbone is based on ResNet50, with shared early layers and separated later layers for pose, shape, and texture encoding. For more information on this, please refer to the Appendix A3. To implement the NeRF decoder, we use CodeNeRF [12] for its simplicity and ease of use. For the differentiable rendering based on NeRF, we set the latent dimension to 256, use 64 samples along each ray, and normalize the shape space by the diagonal length of the observed object's 3D box. For the pose estimation module, we also set the latent dimension to 256 for all MLP layers. Before feeding the input image and occupancy mask into the image encoder, we pad them to make them square and resize them to 128×128 pixels. We then render 32×32 images and occupancy maps to calculate losses.

To ensure effective iterative pose estimation without sacrificing generalization, SUP-NeRF uses a heuristic to sample a random initial 3D object pose across the entire pose space. For translation, we sample in the relative space. We sample a random 2D location within the image ROI range as the initial projection of the object center, while setting the initial depth to $Z = 20$. For rotation, we sample from the full range of yaw angles combined with a random rotation angle within a $(-20°, 20°)$ range around a random axis direction. This sampling method covers almost the entire distribution of object poses in the camera frame and is used consistently in both training and inference.

During training, we use 3 iterations of the pose estimator to generate corresponding pose losses, which allow for pose regression to handle highly diverse poses. We *jointly* train the unified model for the pose estimation module, image encoder, and NeRF decoder for 40 epochs on the nuScenes dataset using a learning rate of 10^{-4} and loss weights of $w_{occ} = 0.1$, $w_{poss} = 0.01$ to compute the total loss.

Inference. During inference, SUP-NeRF optimizes the shape code, texture code, and object pose $[R_{o2c}|T_{o2c}]$ keeping the network weights frozen. The total loss L_{infer} is thus $L_{infer} = L_{rgb} + w_{occ}L_{occ}$, where w_{occ} balances the two. SUP-NeRF minimizes the loss L_{infer} to update the variables. At inference, we run 3 iterations for the *feed-forward* pose estimation module without updating the shape and color codes, and 50 iterations for the *gradient-based* joint optimization of shape, texture, and pose. We use a step size of 0.02 for shape and texture code updates and 0.01 for pose updates during joint optimization.

4 Experiments

Our experiments use three datasets: nuScenes [3], KITTI [6] and Waymo [38]. Our evaluation framework simultaneously measures rendering quality, shape reconstruction accuracy, and pose estimation precision.

Data Splits. Since these datasets are not specifically for 3D reconstruction, we sifted a subset for better evaluation. We chose daytime sequences and used Mask R-CNN [7] to get instance masks, as nuScenes lacks 2D segmentation masks. We matched these masks with 3D bounding box annotations, categorizing them into foreground, background, and unknown, similar to AutoRF [29]. More details are in Appendix A7. We use the following splits of these datasets:

- *nuScenes Train Split*: We collected 87,048 objects from the nuScenes Train split for training.
- *nuScenes Val Split*: We collected 5,000 random objects from the nuScenes Val split for evaluation.
- *KITTI Val Split*: We use 4,895 objects from KITTI Val split [6] for evaluation.
- *Waymo Val Split*: We randomly selected 5,000 objects from Waymo Val split [16] for evaluation.

Evaluation metrics. Our evaluation framework differs from traditional NeRF assessments as it simultaneously evaluates object reconstruction quality and pose accuracy. We incorporate two monocular reconstruction metrics: (1) the *PSNR* (Peak Signal-to-Noise Ratio) comparing the rendered and observed images; (2) the Depth Error between the rendered depth map and sparse LiDAR data, denoted as *DE*. We also include two pose estimation metrics: (3) rotation error denoted as *RE*; (4) the translation error denoted as *TE*. To offer a thorough evaluation, we compare various methods at different stages: post-*feed-forward (FF)*, and at the 50th iterations of NeRF *joint optimization*. Beyond monocular assessments, we also conduct *cross-view* evaluations for PSNR and depth error metrics, denoted as (5) *PSNR-C* and (6) *DE-C*, focusing on objects observed from novel views. As the nuScenes dataset provides object IDs that allows tracking objects across various views, the cross-view evaluation is exclusively conducted on the nuScenes dataset, as explained in Fig. 5.

Baselines. We compare SUP-NeRF with two baselines: AutoRF [29] and AutoRF [29]+FCOS3D [42](selected for its monocular capability, and availability of model trained on the nuScenes dataset). Both AutoRF and SUP-NeRF use the same NeRF decoder and image encoder to ensure a fair comparison. The difference lies in SUP-NeRF's additional layers for pose encoding and refinement. During testing, we apply the same random pose generation for both AutoRF and SUP-NeRF methods, whereas the FCOS3D baseline begins with its own pose prediction. We also compare with BootInv [34], which recovers pose, shape, and appearance. Leveraging our flexible design, we developed SUP-BootInv by integrating BootInv's pretrained decoder with our encoder and pose estimator.

4.1 nuScenes Monocular Reconstruction and Pose Estimation

Tab. 2 shows that SUP-NeRF outperforms two baselines, AutoRF and its combination with FCOS3D on nuScenes Val set. AutoRF alone struggles to improve pose from a random start, resulting in lower PSNR and depth error. SUP-NeRF consistently reduces depth error by over 25%, rotation error by over 30%, and translation error by over 13% compared to the FCOS3D and AutoRF com-

Table 2: nuScenes Monocular Reconstruction and Pose Estimation Results. SUP-NeRF consistently improves AutoRF-based pipelines in all metrics, particularly in the pose estimation metrics. SUP-BootInv also consistently improves BootInv in all metrics. [Key: **Best**, **Second Best**, FF= Feed Forward, C= Cross-View]

Method	PSNR (↑)	DE (m) (↓)	RE (deg.) (↓)	TE (m) (↓)	PSNR-C (↑)	DE-C (m) (↓)
	FF|50it	FF|50it	FF|50it	FF|50it	FF|50it	FF|50it
AutoRF [29]	3.6|10.6	11.21|10.09	87.52|88.07	6.04|5.95	10.0|8.8	1.31|1.41
AutoRF+FCOS	7.5|**17.2**	**1.34**|**0.81**	9.77|10.17	0.85|**0.78**	9.8|10.5	**1.29**|**1.16**
SUP-NeRF	**10.5**|**18.8**	**0.69**|**0.60**	**7.01**|**7.07**	**0.68**|**0.73**	**10.6**|10.9	**1.22**|**1.14**
BootInv [34]	9.4|14.3	5.01|3.56	28.40|28.00	2.59|2.91	**10.9**|**11.8**	1.37|1.35
SUP-BootInv	**10.9**|15.4	1.95|1.62	**7.11**|**8.40**	**0.64**|1.00	**10.9**|**11.9**	1.37|1.40

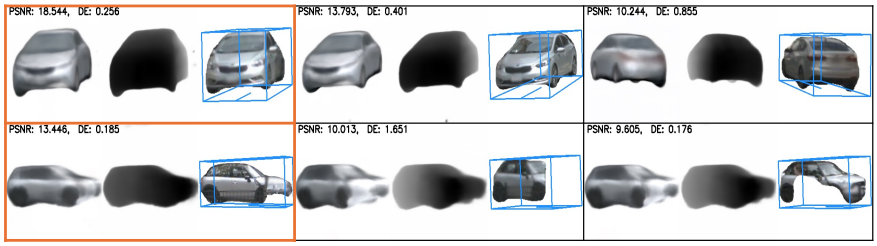


Fig. 5: nuScenes Cross-view Evaluation. Each row shows a set of images of the same object from different angles. We use the example for both monocular pose estimation and NeRF reconstruction. Other images in each row are used to evaluate the reconstructed shape and texture in PSNR and Depth Error (DE).

bination. Our feed-forward stage also shows promising potential for real-time applications, especially where rendering quality is less critical.

We next compare BootInv [34] as the baseline, although the performance of original BootInv's pose estimation was subpar on both the nuScenes and KITTI datasets, integrating our robust pose module with this NeRF framework significantly enhanced pose estimation accuracy. This underscores the adaptability of our unified pipeline in augmenting various object NeRF frameworks for improved pose estimation. Moreover, comparisons of BootInv [34] and SUP-BootInv in terms of monocular PSNR and depth error suggest that SUP-BootInv's superior pose estimation contributes to its higher PSNR and lower depth error. More detailed analysis upon BootInv-based pipelines is included in Appendix A5.

4.2 Cross-Dataset Evaluation

We next compare the generalization capability of all these NeRF models with the cross-dataset experiments. For these experiments, we directly apply nuScenes models to the KITTI and Waymo datasets and report the metrics.

KITTI. Tab. 3 shows the result of cross-dataset experiments on the KITTI dataset. It shows that SUP-NeRF achieves SoTA performance across all evaluation metrics. SUP-NeRF reduces depth error by about 16%, rotation error by

Table 3: KITTI Cross-dataset Monocular Reconstruction and Pose Estimation Results. We train all methods on nuScenes dataset, and test on KITTI dataset. Our methods SUP-NeRF and SUP-BootInv consistently show superiority in all metrics compared to the counterpart methods. [Key: **Best**, **Second Best**, FF= Feed Forward]

Method	PSNR (↑)	DE (m) (↓)	RE (deg.) (↓)	TE (m) (↓)
	FF|50it	FF|50it	FF|50it	FF|50it
AutoRF [29]	0.4|7.6	9.89|8.83	89.67|90.42	6.16|6.07
AutoRF+FCOS	2.4|**13.4**	**2.42**|**1.74**	11.95|12.5	2.2|2.09
SUP-NeRF	4.0|**14.1**	**2.19**|**1.54**	**6.79**|**6.89**	**1.06**|**1.01**
BootInv [34]	**7.0**|12.3	6.11|4.38	15.52|15.76	3.97|3.90
SUP-BootInv	**7.6**|12.7	2.79| 2.21	**9.16**|**8.84**	**1.15**|**1.67**

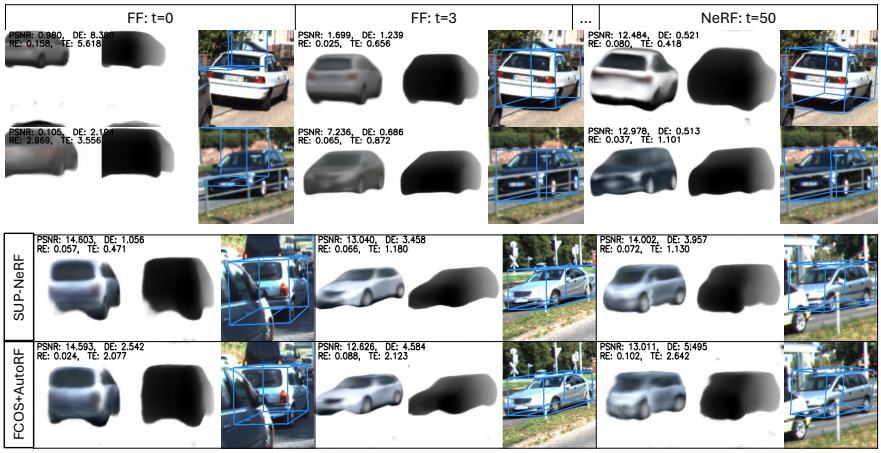


Fig. 6: KITTI (Cross-dataset) Qualitative Results. In the top panel, we demonstrate SUP-NeRF executes pose estimation reliably, fast converging from a random initial pose to the true one, and enables neural reconstruction under diverse object poses, occlusion cases under this cross-dataset setup. In the bottom panel, SUP-NeRF is visually compared to the other major competitor, demonstrates sharper rendered image, higher accuracy in shape and pose.

as large as 50% and translation error also about 50%. Thus, SUP-NeRF shows significantly better geometric performance in cross-dataset setting compared to AutoRF+FCOS, which we attributes to our deep pose module's independence from camera intrinsic parameters. Meanwhile, compared to nuScenes evaluation, all the methods' PSNR and depth error performance drop, which means the cross-dataset generalization of monocular NeRF remains an open challenge.

Fig. 6 presents the qualitative results on this experiment, demonstrating how SUP-NeRF handles diverse poses and occlusions in pose refinement and the visual comparisons between the two most competitive pipelines. Consistent improvements upon integrating our design into BootInv baseline were also observed. More visual comparisons are in Appendix A9.

Table 4: Waymo Cross-dataset Monocular Reconstruction and Pose Estimation Results. We train all methods on nuScenes dataset, and test on Waymo dataset. Our methods SUP-NeRF and SUP-BootInv consistently show superiority in all metrics compared to the counterpart methods, the only exception is the comparison to AutoRF+FCOS in Rotation Error. [Key: **Best**, **Second Best**, FF= Feed Forward]

Method	PSNR (↑)	DE (m) (↓)	RE (deg.) (↓)	TE (m) (↓)
	FF|50it	FF|50it	FF|50it	FF|50it
AutoRF [29]	0.6|9.8	6.76|6.53	86.56|87.67	9.1|9.14
AutoRF+FCOS	6.2|16.5	**2.43**|**2.32**	**6.65**|**7.2**	3.22|3.31
SUP-NeRF	4.8|**17.0**	**2.32**|**1.56**	**10.01**|**10.6**	**1.68**|**1.54**
BootInv [34]	**8.1**|11.0	8.85|8.20	30.78|31.52	5.26|6.16
SUP-BootInv	**8.6**|11.9	5.36|4.38	10.24|11.04	**1.67**|**2.53**

Waymo. Tab. 4 shows the result of cross-dataset experiments on the Waymo dataset. It shows that SUP-NeRF achieves SoTA performance across most evaluation metrics. The only exception is the superior rotation estimation achieved by FCOS3D, which is likely due to leveraging larger context for heavily occluded cases. However, our advantage in translation estimation is more significant.

4.3 Ablation Studies

The major ablation studies we conduct includes (i) the impact of pose estimation module designs and (ii) the impact of different object NeRF networks. Other ablation studies to analyze the impact of training with predicted 2D boxes, pose refinement iterations, training epochs, as well as NeRF's capability on handling initial pose errors are included in Appendix A6, owing to lack of space.

Pose Estimation Module. We evaluate multiple pose estimation designs from two main angles. First, we focus on object-centric pose estimation task alone with evaluations on nuScenes, KITTI, and Waymo datasets. The prototypical designs include: (i) Direct 6D pose regression using MLP layers; (ii) Pose estimation from predicted box corners via Keypoint-based Perspective-n-Point (PnP); (iii) NOCS-based pose estimation as implemented in BootInv [34]; (vi) External monocular 3D detection FCOS3D [42]; (v) Our iterative pose refinement method, which uses projected boxes. For fairness, methods (1), (2), and (5) share the same image encoder, pose encoding layers, differing only in their output layers. Tab. 5 shows that SUP-NeRF consistently outperforms the others, notably in cross-dataset translation estimation.

The second evaluation assesses the impact of different initial poses on object-NeRF performance within the SUP-NeRF architecture. We additionally compared (i) Ground-Truth (GT) pose; (ii) Random initial pose, to represent the oracle and worst case in pose initialization, and exclude the NOCS method which is part of a different NeRF framework. Given different initial poses, all methods undergo the same NeRF Gradient-based Pose Refinement (NGPR), evaluated immediately and after 50 iterations. Tab. 6 shows that our method significantly surpasses other initial pose estimators in geometric accuracy, demonstrating its effectiveness in enhancing 3D model quality.

Table 5: Pose Estimation Ablation Studies. We evaluate all the candidate pose estimation methods with a focus on pose accuracy alone, on nuScenes, KITTI, and Waymo validation sets. Our method shows outstanding robustness in all cases, particularly in translation accuracy in cross-dataset tests. [Key: **Best**]

Pose Module	RE (deg.)(↓)			TE (m) (↓)		
	nuScenes	KITTI	Waymo	nuScenes	KITTI	Waymo
MLP Pose	31.96	37.58	30.89	5.53	5.31	10.1
Corners+PnP	24.79	35.10	93.76	2.82	3.55	18.68
NOCS +PnP(BootInv)	28.40	15.52	23.66	2.59	3.97	3.47
FCOS3D	9.77	11.95	**6.65**	0.85	2.2	3.22
SUP-NeRF	**7.01**	**6.79**	10.01	**0.68**	**1.06**	**1.54**

Table 6: Ablation studies of Pose Initialization in Object NeRF on nuScenes. We compare different choices of initial poses under SUP-NeRF frameworks. As shown, SUP-NeRF presents superiors effectiveness in all the metrics. [Key: **Best**, FF= Feed Forward, C= Cross-View]

Initial Pose	PSNR (↑)	DE (m) (↓)	RE (deg.) (↓)	TE (m) (↓)	PSNR-C (↑)	DE-C (m) (↓)
	FF|50it	FF|50it	FF|50it	FF|50it	FF|50it	FF|50it
Random	3.6|10.6	11.21|10.09	87.52|88.07	6.04|5.95	9.1|7.4	1.42|1.82
MLP Pose	1.2|6.9	5.33|4.41	31.96|24.57	5.53|2.78	9.8|8.5	1.3|1.4
Corners+PnP	7.2|16.2	2.18|1.83	24.79|24.57	2.82|2.78	9.9|9.5	1.3|1.3
FCOS3D	7.5|17.2	1.34|0.81	9.77|10.17	0.85|0.78	9.8||10.5	1.29||1.16
SUP-NeRF	**10.5|18.8**	**0.69|0.60**	**7.01|7.07**	**0.68|0.73**	**10.6|10.9**	**1.22|1.14**
GT	10.0|18.8	0.66|0.53	0.|2.11	0.|0.13	10.9|11.1	1.19|1.14

Tab. 6 also confirms the role of NeRF Gradient-Based Pose Refinement (NGPR). By comparing the difference between feed-forward results and the later ones, one can find NGPR merely reduces rotation error or translation error. On the contrary, it can even increase pose error when the initial pose error is small enough, e.g. rotation error $< 7°$, and translation error $< 0.75m$. On the other hand, the improvements of PSNR and depth reconstruction over time indicate that NeRF optimization mostly focuses on shape and texture optimization w.r.t. the visible surface regardless the pose.

Impact of object NeRF framework. To demonstrate SUP-NeRF's benefits, we compared different network configurations using the same pose refinement approach. For fairness, we used identical subnetwork modules across all configurations. These include a ResNet-based Image Encoder (E), a NeRF Decoder (D), and our Pose Estimation Module (P). CodeNeRF [12] has just a Decoder (D), AutoRF combines an Image Encoder and Decoder (E + D), and SUP-NeRF uses all three (E + D + P). For CodeNeRF and AutoRF, we added a separately trained Pose Network (PNet, composed of E + P) for initial pose refinement. We also tested a SUP-NeRF version without the cross-task Short Cut (SC) to assess its impact. Tab. 7 shows that SC consistently improves SUP-NeRF across all metrics on nuScenes.

SUP-NeRF, with joint training, outperforms separate training setups AutoRF + PNet and CodeNeRF + PNet in monocular metrics. Interestingly, while

Table 7: Object NeRF Framework Ablations on nuScenes. We used the same subnetwork architectures - the image encoder (E), NeRF decoder (D), and pose estimation module (P) - to create various model configurations. For SUP-NeRF, we also included a version without the cross-task Short Cut (SC) for comparison. [Key: **Best**, FF= Feed Forward, C= Cross-View]

Model Training	PSNR(↑)		DE (m)(↓)		RE (deg.)(↓)		TE (m)(↓)		PSNR-C(↑)		DE-C (m)(↓)	
	FF	50it	FF	50it	FF	50it	FF	50it	FF	50it	FF	50it
CodeNeRF (D)+PNet(E+P)	7.6	16.0	0.87	0.69	10.57	9.75	0.72	0.76	**11.0**	**11.2**	**1.01**	**1.04**
AutoRF (E+D)+PNet(E+P)	9.7	17.5	0.79	0.66	8.26	7.87	0.7	0.74	5.6	10.1	1.24	1.16
SUP-NeRF (E+D+P) w/o SC	8.7	18.0	0.94	0.73	10.12	9.22	0.75	0.74	10.8	10.9	1.23	1.11
SUP-NeRF (E+D+P)	**10.5**	**18.8**	**0.69**	**0.60**	**7.01**	**7.07**	**0.68**	**0.73**	10.6	10.9	1.22	1.14

Table 8: Model Size and Running Time Comparison in feed-forward scenarios, and with 20 and 50 iterations of NeRF optimization. SUP-NeRF gets the smallest running time. [Key: **Best**, FF= Feed Forward]

Method	Params (M) (↓)	FF (s) (↓)	FF+20it (s) (↓)	FF+50it (s) (↓)
AutoRF+FCOS [29]	91.116 (36.166+54.950)	0.123 (0.114+0.009)	0.714 (0.114+0.600)	1.599 (0.114+1.485)
SUP-NeRF	**49.816**	**0.018**	**0.608**	**1.493**
BootInv [34]	182.616	0.156	3.534	8.601
SUP-BootInv	57.580	0.018	3.396	8.463

CodeNeRF lags in monocular, it excels in cross-view evaluations. This implies that using mean shape codes, rather than current image encodings, might reduce overfitting to current view.

4.4 Running Speed Analysis

Tab. 8 compares the model size and running speed on a single A5000 Graphics card. Detailed setup and analysis details are in Appendix A8. Tab. 8 shows that SUP-NeRF needs about half or one third as many parameters as others and is 6-8 times faster in the feed-forward (FF) stage. Meanwhile, a significant portion of the run time is consumed by the NeRF optimization step, whether for 20 iterations (20it) or 50 iterations (50it). However, speed limitation of NeRF can been effectively tackled by recent advances in neural rendering [15,31], which we consider as orthogonal to our contribution to the feed-forward stage.

5 Conclusion

This paper introduces SUP-NeRF: a unified network that seamlessly integrates pose estimation and object reconstruction. SUP-NeRF includes a novel pose estimation module to handle scale-depth ambiguity and a new representation invariant to camera changes. Consequently, SUP-NeRF greatly improves the robustness of object pose estimation compared to the standard NeRF baseline. SUP-NeRF outperforms previous methods in both monocular reconstruction and pose estimation tasks, especially in challenging cross-dataset applications.

References

1. Batra, D., Chang, A.X., Chernova, S., Davison, A.J., Deng, J., Koltun, V., Levine, S., Malik, J., Mordatch, I., Mottaghi, R., Savva, M., Su, H.: Rearrangement: A challenge for embodied AI. CoRR (2020) 2
2. Bian, W., Wang, Z., Li, K., Bian, J.: Nope-nerf: Optimising neural radiance field with no pose prior. In: CVPR (2023) 4
3. Caesar, H., Bankiti, V., Lang, A., Vora, S., Liong, V., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: A multimodal dataset for autonomous driving. In: CVPR (2020) 3, 4, 8
4. Chen, H., Manhardt, F., Navab, N., Busam, B.: Texpose: Neural texture learning for self-supervised 6d object pose estimation. In: CVPR (2023) 4
5. Chen, Y., Lee, G.H.: DBARF: deep bundle-adjusting generalizable neural radiance fields. In: CVPR (2023) 4
6. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the KITTI vision benchmark suite. In: CVPR (2012) 4, 8, 9
7. Girshick, R., Radosavovic, I., Gkioxari, G., Dollár, P., He, K.: Detectron. https://github.com/facebookresearch/detectron (2018) 9
8. Gkioxari, G., Johnson, J., Malik, J.: Mesh R-CNN. In: ICCV (2019) 2, 3
9. Henderson, P., Tsiminaki, V., Lampert, C.H.: Leveraging 2d data to learn textured 3d mesh generation. In: CVPR (2020) 2
10. Hodaň, T., Michel, F., Brachmann, E., Kehl, W., Glent Buch, A., Kraft, D., Drost, B., Vidal, J., Ihrke, S., Zabulis, X., Sahin, C., Manhardt, F., Tombari, F., Kim, T.K., Matas, J., Rother, C.: BOP: Benchmark for 6D object pose estimation. ECCV (2018) 4
11. Insafutdinov, E., Campbell, D., Henriques, J.F., Vedaldi, A.: Snes: Learning probably symmetric neural surfaces from incomplete data. In: Avidan, S., Brostow, G.J., Cissé, M., Farinella, G.M., Hassner, T. (eds.) ECCV. pp. 367–383 (2022) 2, 3
12. Jang, W., Agapito, L.: Codenerf: Disentangled neural radiance fields for object categories. In: ICCV (2021) 2, 3, 8, 13
13. Kanazawa, A., Tulsiani, S., Efros, A.A., Malik, J.: Learning category-specific mesh reconstruction from image collections. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV (2018) 2, 3
14. Kehl, W., Manhardt, F., Tombari, F., Ilic, S., Navab, N.: SSD-6D: making rgb-based 3d detection and 6d pose estimation great again. In: ICCV (2017) 4, 6
15. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph. (2023) 14
16. Kumar, A., Brazil, G., Corona, E., Parchami, A., Liu, X.: DEVIANT: depth equivariant network for monocular 3d object detection. In: ECCV (2022) 2, 4, 9
17. Kumar, A., Guo, Y., Huang, X., Ren, L., Liu, X.: SeaBird: Segmentation in Bird's View with Dice Loss Improves Monocular 3D Detection of Large Objects. In: CVPR (2024) 2
18. Kundu, A., Genova, K., Yin, X., Fathi, A., Pantofaru, C., Guibas, L.J., Tagliasacchi, A., Dellaert, F., Funkhouser, T.A.: Panoptic neural fields: A semantic object-aware neural scene representation. In: CVPR (2022) 2, 3, 4
19. Kundu, A., Li, Y., Rehg, J.M.: 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In: CVPR (2018) 2, 3
20. Li, F., Yu, H., Shugurov, I., Busam, B., Yang, S., Ilic, S.: Nerf-pose: A first-reconstruct-then-regress approach for weakly-supervised 6d object pose estimation. CoRR (2022) 4

21. Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: Deepim: Deep iterative matching for 6d pose estimation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV (2018) 4, 6, 7
22. Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: Deepim: Deep iterative matching for 6d pose estimation. In: ECCV (2018) 4
23. Lin, Y., Florence, P., Barron, J.T., Rodriguez, A., Isola, P., Lin, T.: inerf: Inverting neural radiance fields for pose estimation. In: IROS (2021) 3
24. Lipson, L., Teed, Z., Goyal, A., Deng, J.: Coupled iterative refinement for 6d multi-object pose estimation. In: CVPR (2022) 4
25. Lu, Y., Ma, X., Yang, L., Zhang, T., Liu, Y., Chu, Q., Yan, J., Ouyang, W.: Geometry uncertainty projection network for monocular 3d object detection. In: ICCV (2021) 2, 4
26. Merrill, N., Guo, Y., Zuo, X., Huang, X., Leutenegger, S., Peng, X., Ren, L., Huang, G.: Symmetry and uncertainty-aware object SLAM for 6dof object pose estimation. In: CVPR (2022) 4
27. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM pp. 99–106 (2021) 2, 4
28. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: CVPR (2017) 4
29. Müller, N., Simonelli, A., Porzi, L., Bulò, S.R., Nießner, M., Kontschieder, P.: Autorf: Learning 3d object radiance fields from single view observations. In: CVPR (2022) 2, 3, 4, 6, 7, 9, 10, 11, 12, 14
30. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. pp. 102:1–102:15 (2022) 2
31. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. (2022) 14
32. Park, D., Ambrus, R., Guizilini, V., Li, J., Gaidon, A.: Is pseudo-lidar needed for monocular 3d object detection? In: ICCV (2021) 4
33. Park, K., Patten, T., Vincze, M.: Pix2pose: Pix2pose: Pixel-wise coordinate regression of objects for 6d pose estimation. In: ICCV (2019) 4
34. Pavllo, D., Tan, D.J., Rakotosaona, M.J., Tombari, F.: Shape, pose, and appearance from a single image via bootstrapped radiance field inversion. In: CVPR (2023) 2, 3, 4, 7, 9, 10, 11, 12, 14
35. Rad, M., Lepetit, V.: Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. ICCV (2017) 4
36. Simonelli, A., Bulò, S.R., Porzi, L., Lopez-Antequera, M., Kontschieder, P.: Disentangling monocular 3d object detection. In: ICCV (2019) 2, 4
37. Solà, J., Deray, J., Atchuthan, D.: A micro lie theory for state estimation in robotics. CoRR (2018) 6
38. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., Anguelov, D.: Scalability in perception for autonomous driving: Waymo open dataset. In: CVPR (2020) 8
39. Tekin, B., Sinha, S.N., Fua, P.: Real-time seamless single shot 6d object pose prediction. In: CVPR (2018) 4
40. Tian, S., Cai, Y., Yu, H.X., Zakharov, S., Liu, K., Gaidon, A., Li, Y., Wu, J.: Multi-object manipulation via object-centric neural scattering functions. In: CVPR (2023) 2

41. Wang, H., Sridhar, S., Huang, J., Valentin, J., Song, S., Guibas, L.J.: Normalized object coordinate space for category-level 6d object pose and size estimation. In: CVPR (2019) 4
42. Wang, T., Zhu, X., Pang, J., Lin, D.: FCOS3D: fully convolutional one-stage monocular 3d object detection. In: ICCVW (2021) 4, 9, 12
43. Weihs, L., Deitke, M., Kembhavi, A., Mottaghi, R.: Visual room rearrangement. In: CVPR (June 2021) 2
44. Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In: RSS (2018) 4
45. Yang, B., Zhang, Y., Xu, Y., Li, Y., Zhou, H., Bao, H., Zhang, G., Cui, Z.: Learning object-compositional neural radiance field for editable scene rendering. In: ICCV (2021) 2, 3
46. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: CVPR (2021) 2, 3
47. Yu, H., Wu, J., Yi, L.: Rotationally equivariant 3d object detection. In: CVPR (2022) 4
48. Zakharov, S., Ambrus, R.A., Guizilini, V., Park, D., Kehl, W., Durand, F., Tenenbaum, J.B., Sitzmann, V., Wu, J., Gaidon, A.: Single-shot scene reconstruction. In: CoRL (2021) 2, 3
49. Zakharov, S., Kehl, W., Bhargava, A., Gaidon, A.: Autolabeling 3d objects with differentiable rendering of SDF shape priors. In: CVPR (2020) 2, 3
50. Zakharov, S., Shugurov, I., Ilic, S.: Dpod: 6d pose object detector and refiner. In: ICCV (2019) 4
51. Zhou, A., Kim, M.J., Wang, L., Florence, P., Finn, C.: Nerf in the palm of your hand: Corrective augmentation for robotics via novel-view synthesis. In: CVPR (2023) 2