Uncertainty-aware Active Learning of NeRF-based Object Models for Robot Manipulators using Visual and Re-orientation Actions

Saptarshi Dasgupta*, Akshat Gupta*, Shreshth Tuli+ and Rohan Paul+
Indian Institute of Technology Delhi, India {*} Equal contribution {+} Equal advising

***Abstract*— Manipulating unseen objects is challenging without a 3D representation, as objects generally have occluded surfaces. This requires physical interaction with objects to build their internal representations. This paper presents an approach that enables a robot to rapidly learn the complete 3D model of a given object for manipulation in unfamiliar orientations. We use an ensemble of partially constructed NeRF models to quantify model uncertainty to determine the next action (a visual or re-orientation action) by optimizing informativeness and feasibility. Further, our approach determines *when* and *how* to grasp and re-orient an object given its partial NeRF model and re-estimates the object pose to rectify misalignments introduced during the interaction. Experiments with a simulated Franka Emika Robot Manipulator operating in a tabletop environment with benchmark objects demonstrate an improvement of (i) 14% in visual reconstruction quality (PSNR), (ii) 20% in the geometric/depth reconstruction of the object surface (F-score) and (iii) 71% in the task success rate of manipulating objects *a-priori* unseen orientations/stable configurations in the scene; over current methods. Additional details appear at: https://actnerf.github.io/.**

I. Introduction

We consider the problem of acquiring a 3D visual and geometric representation of an object for sequential robot manipulation tasks. In recent years, Neural Radiance Fields (NeRF) has emerged as a useful implicit representation that allows synthesis of novel views aiding in downstream planning, manipulation, and pose estimation tasks. Such a representation is acquired by collecting a set of views from known poses in the environment. The process for collecting such views is either in (i) batch mode [1]–[3] by exhaustively collecting observations covering a region or (ii) actively by determining a set of informative views [4]–[7]. Although effective in rapidly constructing an object model, such approaches can only reconstruct the visible regions of the object, failing to model obscured parts such as the base, internal contents, and other occluded regions. The inability to accurately model the object owing to occlusions in the scene translates to poor manipulation ability for subsequent manipulation tasks.

This work considers the possibility of *directly interacting* via grasping, re-orientation, and stably releasing the object to expose previously *unexposed* regions for subsequent model building. Fig. 1 presents an overview of our model acquisition technique. Introducing physical interaction during model acquisition poses two key challenges. First, finding stable grasping points using a *partially* built model is challenging due to depth uncertainty in unobserved or poorly observed regions. Second, re-orientation introduces uncertainty in the object's pose, affecting the incremental fusion of the radiance field arising from new observations. Further, as opposed to scene-based representations, we seek the ability to acquire object-centric radiance fields to support semantic tasks that may require sequential manipulation actions (e.g., clearing objects from a region). Overall, this paper makes the following contributions:

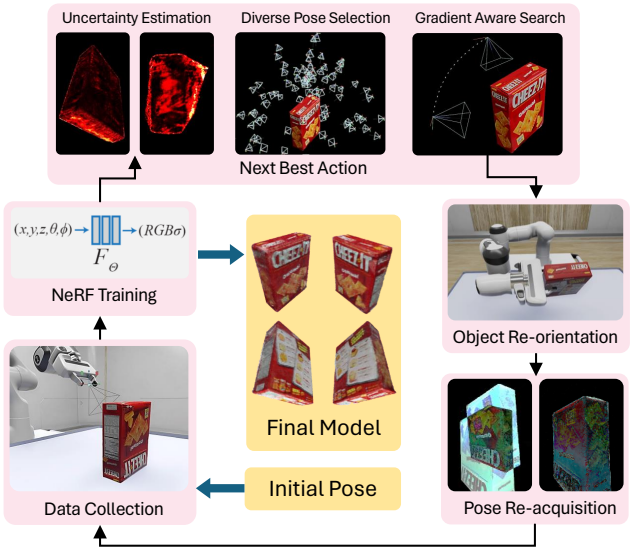


Fig. 1: **Overview.** We present an active learning approach for building a NeRF model of an object that allows the robot to re-orient the object while collecting visual observations. Our approach is guided by a measure of model uncertainty in the partially constructed model, which is used to determine the most informative feasible action, aiding model construction.

- Leveraging vision foundation models to isolate the object of interest to disentangle its uncertainty from that of other background objects in the scene.
- A search procedure that estimates the next most informative action (visual or re-orientation). The procedure relies on a coarse-to-fine optimization of the continuous viewing space incorporating (i) model uncertainty in the partially-built model (adapting [5]), (ii) motion costs, and (iii) kinematic constraints.
- An approach for grasping while accounting for the uncertainty in the partially constructed model and re-estimating the pose of the object after interaction for fusing the incrementally acquired model.

In essence, this work takes a step in the direction of acquiring a rich NeRF model of an object to support future robot manipulation tasks and hope that this work will invite discussion and feedback at the proposed RoboNeRF workshop.

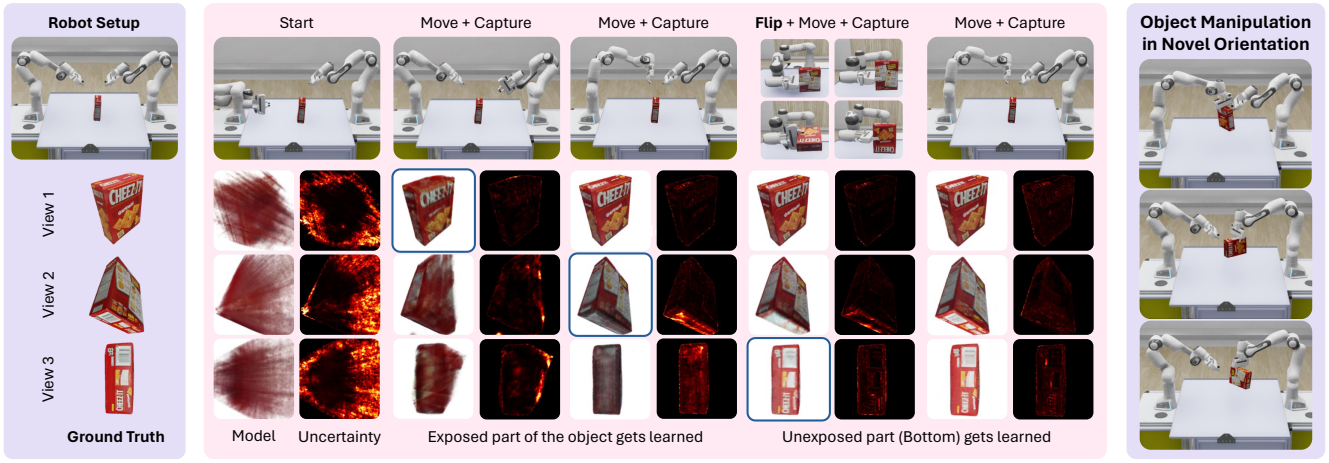


Fig. 2: **Overview.** We show the RGB images and uncertainty maps rendered from trained models during our active learning process. The GT images are shown for reference. Before flipping, the bottom surface of the object has high uncertainty, which only diminishes once we perform the flip and acquire information about the bottom surface. The acquired object model enables future manipulation action of the object in any orientation.

II. Problem Formulation

Our problem concerns a robot manipulator in a tabletop environment and an object placed near the table's center. The robot's task is to obtain a 3D representation of the object for manipulation in unseen orientations. Actions available to the robot A include `Move(`p_i`)`: moving its arm to pose p_i, `Flip()`: flipping the object, and `Capture()`: capturing an image with its attached camera. Further, let $\Gamma(a)$ denote the cost of an action $a \in A$. The robot must execute a sequence of actions $A^* = (a_1, a_2, \ldots, a_n)$, where each a_i represents a combination of actions from A. After executing each action a_i, the robot applies a $Capture()$ action to obtain an image. The collected images are then used to train a NeRF model F_θ. Given a partially trained model $F_{\Theta_{k-1}}$, based on images $i_1, i_2, \ldots, i_{k-1}$, the goal is to identify the next action a_k to capture an image from a viewpoint with the highest uncertainty, while also minimizing the action cost $\Gamma(a_k)$:

$$a_k = \arg\max_a \left[U(F_{\Theta_{k-1}}, p) - \lambda \Gamma(a) \right], \tag{1}$$

where, p represents the pose achieved by the robotic arm upon executing action a, and $U(F_{\Theta_{k-1}}, p)$ quantifies the uncertainty in the model from pose p. This formulation is equivalent to minimizing the loss function $L(a)$, defined as:

$$L(a) = \lambda \Gamma(a) - U(F_{\Theta_{k-1}}, p) \tag{2}$$

Next, we turn our attention to estimating the actions minimizing this object using an estimate of model uncertainty in an active learning setup.

III. Technical Approach

A. Estimating Model Uncertainty

Following [5], we train M NeRF models using the same set of images but randomly initialized weights. By rendering images from these M models for any camera pose, we render an uncertainty heat map (see Fig. 3) where uncertainty associated with a pixel corresponding to ray $\mathbf{r}$ and estimated

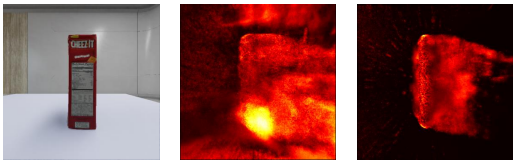


Fig. 3: **Effect of background on uncertainty estimation.** Uncertainty heat maps of NeRF models trained on segmented v/s original images. Segmentation of the object removes artefacts arising from modeling background visual appearance.

colors $\hat{\boldsymbol{C}}_i(\mathbf{r})$ as:

$$\sigma^2(\mathbf{r}) = \frac{1}{M} \sum_{k=1}^{M} \|\boldsymbol{\mu}(\mathbf{r}) - \hat{\boldsymbol{C}}_k(\mathbf{r})\|^2, \text{where} \tag{3}$$

$$\boldsymbol{\mu}(\mathbf{r}) = \frac{1}{M} \sum_{k=1}^{M} \hat{\boldsymbol{C}}_k(\mathbf{r}), \tag{4}$$

and $\boldsymbol{\mu}(\mathbf{r})$ and $\hat{\boldsymbol{C}}_i(\mathbf{r})$ are vectors representing the RGB color channels. Using the expression for the uncertainty of a ray, $\sigma^2(\mathbf{r})$, the uncertainty for a pose p can be quantified as the sum of the uncertainties for all rays emanating from p.

In a 3D scene representation, this approach yields uncertainty estimates covering both the object and its surroundings. A next-best-view (NBV) strategy based on such uncertainty tends to minimize background uncertainty, diverging from our main goal, especially in cluttered scenes. To focus solely on the object's uncertainty, we adopt Grounded-SAM [8], which leverages textual prompts to generate object masks by integrating Grounding DINO [9] and SAM [10]. This method enables training NeRF models on segmented images, offering a more precise evaluation of uncertainty centered on the object of interest (see Fig. 3).

B. Uncertainty-guided Next Action Selection

We now tackle the challenge of identifying the next best action within the context of our active learning framework.

TABLE I: Comparison of our method with ActiveNeRF and other baselines. All the baselines include `Flip()` action as detailed in the appendix (VI-B) and are trained with segmented images. F-score* represents the F-score values multiplied by 10. As evident from the tables, our approach outperforms other methods by a significant margin.

Method	Basket		Cheezit Box		Mug		Rubik's Cube		Spam Can		**Total**	
	PSNR↑	F-score*↑	PSNR↑	F-score*↑	PSNR↑	F-score*↑	PSNR↑	F-score*↑	PSNR↑	F-score*↑	PSNR↑	F-score*↑
Model quality after 20 iterations **without** grasping												
Ours	**17.2** ± **0.1**	**4.2** ± **0.5**	**21.8** ± **0.2**	**4.3** ± **0.2**	26.3 ± 0.1	6.5 ± 0.2	**30.3** ± **0.3**	3.8 ± 0.1	23.9 ± 0.6	4.1 ± 0.3	**23.9** ± **0.1**	**4.6** ± **0.1**
Random	17.0 ± 0.2	4.0 ± 0.6	21.4 ± 0.4	4.0 ± 0.4	**26.5** ± **0.2**	**6.6** ± **0.1**	29.5 ± 0.2	**4.1** ± **0.3**	**24.7** ± **0.2**	**4.4** ± **0.0**	23.8 ± 0.1	4.6 ± 0.2
Furthest	17.1 ± 0.1	3.0 ± 0.2	21.2 ± 0.1	4.2 ± 0.2	26.5 ± 0.3	5.8 ± 0.2	28.6 ± 0.5	3.6 ± 0.2	23.9 ± 0.3	4.0 ± 0.2	23.5 ± 0.1	4.1 ± 0.1
Active [7]	15.5 ± 0.5	2.6 ± 0.2	19.0 ± 0.1	2.9 ± 0.3	24.8 ± 0.9	4.4 ± 0.1	27.0 ± 0.6	3.4 ± 0.3	19.7 ± 1.3	3.1 ± 0.1	21.2 ± 0.4	3.3 ± 0.1
Model quality after 20 iterations **with** grasping												
Ours	**16.9** ± **0.1**	3.4 ± 0.3	**21.3** ± **0.3**	**4.0** ± **0.2**	**26.9** ± **0.2**	**5.9** ± **0.1**	**31.6** ± **0.5**	**4.0** ± **0.3**	**23.4** ± **0.5**	**3.8** ± **0.1**	**24.0** ± **0.2**	**4.2** ± **0.1**
Random	16.0 ± 0.6	**3.5** ± **0.8**	20.9 ± 1.0	3.9 ± 0.4	22.2 ± 0.8	4.6 ± 0.2	29.1 ± 0.9	3.5 ± 0.5	22.8 ± 1.8	3.4 ± 0.4	22.2 ± 0.5	3.8 ± 0.2
Furthest	16.5 ± 0.2	3.1 ± 0.2	19.8 ± 0.7	3.7 ± 0.4	24.1 ± 0.3	4.8 ± 0.7	27.8 ± 0.9	3.8 ± 0.4	21.1 ± 1.5	3.3 ± 0.3	21.9 ± 0.4	3.7 ± 0.2
Active [7]	16.1 ± 0.2	2.6 ± 0.2	19.2 ± 0.2	2.5 ± 0.9	23.6 ± 3.2	5.5 ± 0.3	27.6 ± 0.8	3.9 ± 0.6	22.1 ± 0.2	3.2 ± 0.1	21.7 ± 0.7	3.5 ± 0.2
Model quality attained given a cost budget of 2 **without** grasping												
Ours	**17.1** ± **0.1**	**3.9** ± **0.3**	**21.4** ± **0.2**	**4.3** ± **0.1**	**26.0** ± **0.3**	**5.8** ± **0.5**	**29.8** ± **0.7**	**4.5** ± **0.2**	**23.7** ± **0.4**	**4.1** ± **0.2**	**23.6** ± **0.2**	**4.5** ± **0.1**
Random	17.0 ± 0.2	3.8 ± 0.4	19.3 ± 1.1	3.7 ± 0.4	25.2 ± 0.9	5.7 ± 0.7	28.6 ± 0.6	3.8 ± 0.1	23.3 ± 1.4	3.6 ± 0.3	22.7 ± 0.4	4.1 ± 0.2
Furthest	16.4 ± 0.4	2.6 ± 0.1	19.2 ± 0.2	3.5 ± 0.1	24.7 ± 1.1	4.4 ± 0.1	26.6 ± 1.2	4.1 ± 0.1	22.6 ± 0.4	3.9 ± 0.1	21.9 ± 0.3	3.7 ± 0.0
Active [7]	15.4 ± 0.4	3.1 ± 0.1	18.4 ± 0.2	3.1 ± 0.1	24.1 ± 0.4	4.1 ± 0.2	26.3 ± 0.6	3.4 ± 0.3	19.5 ± 1.7	3.6 ± 0.1	20.7 ± 0.4	3.5 ± 0.1
Model quality attained given a cost budget of 2 **with** grasping												
Ours	**16.9** ± **0.1**	**3.3** ± **0.4**	**21.2** ± **0.2**	**3.9** ± **0.2**	**26.8** ± **0.4**	**5.8** ± **0.3**	**30.9** ± **0.7**	**4.0** ± **0.3**	**23.7** ± **0.6**	**3.8** ± **0.2**	**23.9** ± **0.2**	**4.2** ± **0.1**
Random	16.0 ± 0.3	3.0 ± 0.3	20.9 ± 1.0	3.8 ± 0.2	22.3 ± 0.8	4.4 ± 0.2	28.5 ± 1.3	3.6 ± 0.3	22.8 ± 1.9	3.4 ± 0.4	22.1 ± 0.5	3.6 ± 0.1
Furthest	16.5 ± 0.2	1.5 ± 0.8	19.9 ± 0.2	3.6 ± 0.2	23.5 ± 0.4	4.3 ± 0.3	27.7 ± 0.9	3.5 ± 0.4	22.1 ± 1.7	3.3 ± 0.3	21.9 ± 0.4	3.2 ± 0.2
Active [7]	15.8 ± 0.2	3.3 ± 0.2	19.2 ± 0.3	3.2 ± 0.1	21.9 ± 2.0	5.1 ± 0.2	26.4 ± 0.8	2.7 ± 1.0	21.6 ± 0.5	3.3 ± 0.2	21.0 ± 0.8	3.5 ± 0.2

First, we consider the scenario where only `Move()` action is allowed. In this case, the minimization variable a in (2) is substituted with $p \in SE(3)$, representing the 6-DoF pose of the camera affixed to the robot's end-effector. The designated action for a pose p corresponds to maneuvering the end-effector to position the camera at p. To circumvent the limitations of naive gradient descent approaches, which falter due to the presence of numerous local minima within the objective function, we propose a bi-level optimization strategy. First, we select a *sparse* and *diverse* subset of k candidate poses from n randomly sampled poses, all oriented towards the work space's center. Second, we execute a gradient descent search from each candidate pose. The final solution is selected from the second level that yields the lowest objective function value. Next, in scenarios where `Flip()` actions are also considered, the problem is decomposed into two sub problems: 1) Identifying the optimal action without allowing flips, and 2) Adjusting the coordinate axes to simulate a flip and determining the optimal subsequent action. The cost associated with `Flip()` is accounted for exclusively in the second sub problem. The final optimal action is chosen based on the lower $L(a)$ value from these sub problems. Finally, let $\Gamma(a)$ denote the action cost (2) defined as the the cost for `Move()` as a weighted sum of translation distance and angular distance between initial and final poses. For the `Flip()` action we assume constant cost.

C. Object Re-orientation during Model Acquisition

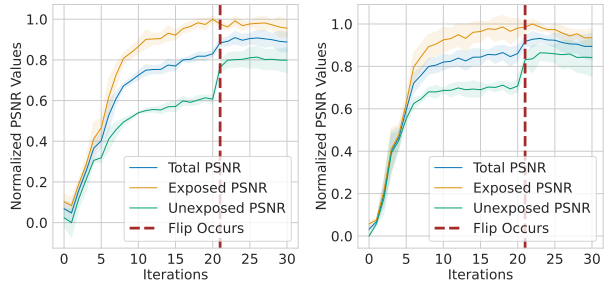
This section details the approach for grasping and re-orientation. From the RGB and depth images rendered by our partial model, we compute the lateral grasp poses using AnyGrasp [11], which generates grasp pose-confidence pairs. Note that during learning the object model may be partially-known often containing spurious geometric features which lead to poor grasp pose estimates. Hence, we prune grasp poses based on distance from center of the point cloud and average opacity of the grasp patch. We then score each grasp pose, to select the most suitable grasp. Our grasp score is defined as $G_s = (1-\theta)/U_d$, where θ is the angle between the grasp pose and surface normal and U_d is the variance of rendered depths summed over the rays of the grasp patch. In essence, the grasp estimates are weighed according to the depth uncertainty, increasing the score of those grasps where the NeRF model is certain about the object's surface.

Upon grasping the object is rotated using a pre-defined skill and placed on the supporting surface. Note that such actions may lead to uncertainty in the final resting pose of the object. This leads to errors if the NeRF models pre and post interaction are directly blended. To ameliorate the error, we estimate the pose of the object by leveraging the acquired NeRF model using the iNeRF methodology [12]. We optimize the camera pose to minimize the Sum of Squared Differences (**SSD**) between the captured and NeRF's predicted image. We observe that directly applying the original iNeRF optimization framework for pose estimation does not have a wide basin for convergence. Alternatively, we first use multiple images for pose estimation and further combine the best solution of three non-gradient based methods: Nelder-Mead [13], COBYLA [14], [15], and Powell [16]. Our experiments show better solution quality with this approach.

IV. Evaluation and Results

A. Evaluation Setup

We perform experiments with two Franka Emika Robots in *Isaac Sim* simulator. Object models are selected from the



(a) Cheezit Box (b) Spam Can

Fig. 4: **Effect of Flip on Model Quality**.We demonstrate the impact of flipping objects on model quality (PSNR) for **Exposed** and **Unexposed** subsets of the validation set, representing camera poses above and below the object center, respectively. Unexposed PSNR shows a significant increase, leading to an overall improvement in total PSNR. PSNR values are min-max normalized in each plot. The complete figure including all the objects is in the appendix (Fig. 8).

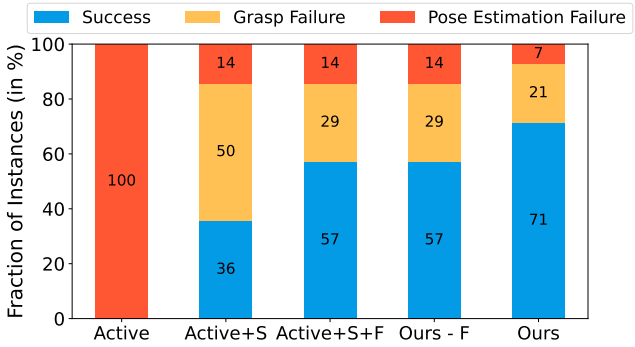


Fig. 5: **Grasping Performance Analysis.** Analysis of grasp task success rate and failure scenarios. **Active** denotes vanilla ActiveNeRF trained on captured images without segmentation and with no `Flip()` action. **S** denotes *object segmentation*, **F** denotes possibility of `Flip()` action.

YCB dataset [17], focusing on objects suitable for lateral grasping and flipping. We compare the following baselines: (i) Random View, where views are randomly selected, (ii) Next Furthest View, which maximizes distance from existing training views, and (iii) ActiveNeRF [7], implemented using the Kaolin-Wisp framework [18]. Evaluation metrics include PSNR (Peak Signal-to-Noise Ratio) for visual fidelity, using 64 images per object for validation sets, and F-score [19] for geometric accuracy, derived from NeRF models through Marching Cubes [20].

B. Evaluation of Model Quality

We evaluate the quality of the model acquired using the proposed approach in relation to the baseline models under two settings (i) when `Flip()` actions are prohibited and the robot can only collected visual observations, and (ii) when `Flip()` actions permitted along with visual observations. Further, we analyse the resultant model quality by first imposing a bound on the total number of training iterations. This fixed iteration count was set at 20 to afford ample iterations for all methods to converge to reasonable NeRF models for manipulation. Results were consistent for other values as well. Next, we analyse the model quality for a total cost budget for actions taken by the robot. The budget is empirically set to a value which is lower than the total cost to allow quality saturation. Here, the cost budget is set

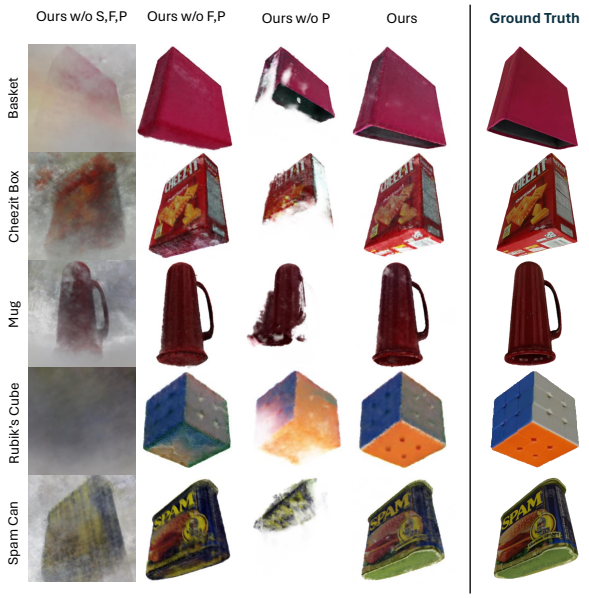


Fig. 6: **Model Quality Comparison under Pipeline Modifications.** Comparison of learned model quality with ground truth meshes. The different models are obtained after removing certain components from our pipeline. **S** denotes *object segmentation*, **F** denotes *flip*, and **P** denotes *pose re-acquisition* after the flip has been executed.

to 3 for experiments allowing the `Flip()` action and to 2 for those that do not, with the difference directly correlating to the cost associated with a flip action. Table I shows the results. The proposed method consistently performs better than baselines. In particular, it improves PSNR (by 14%) and F-Score (by 20%) over the ActiveNeRF baseline.

C. Benefit of Object Re-orientation in Model Learning

Figure 4 shows that the model quality improves significantly after flipping and exposing previously unseen surfaces. We also show images rendered from models (trained for 20 iterations each) without `Flip()` and compare it against our best model (Fig. 6). Further, we perform task-level evaluation by constructing a dataset with objects placed in random positions and orientations in the robot's workspace. The robot is tasked to construct a NeRF model and then perform picking/placing from a randomly sampled poses. Figure 5 reports the Grasp Success Rate (GSR) and failure modes for all methods. Results indicate that the proposed model outperforms alternate approaches.

V. Conclusions

This paper introduces an active learning approach to acquire a NeRF model using both visual observation and re-orientation actions. This facilitates modeling of previously occluded surfaces allowing future pick/place interactions when the object may be positioned aribitrarily in any stable mode. Our approach estimates model uncertainty using an ensemble of models and using such uncertainty to estimate the next action (visual or re-orientation) to improve the model. Future work will explore improving the timing efficiency of ensembling and incorporate multi-step interactions and experiments on real platforms.

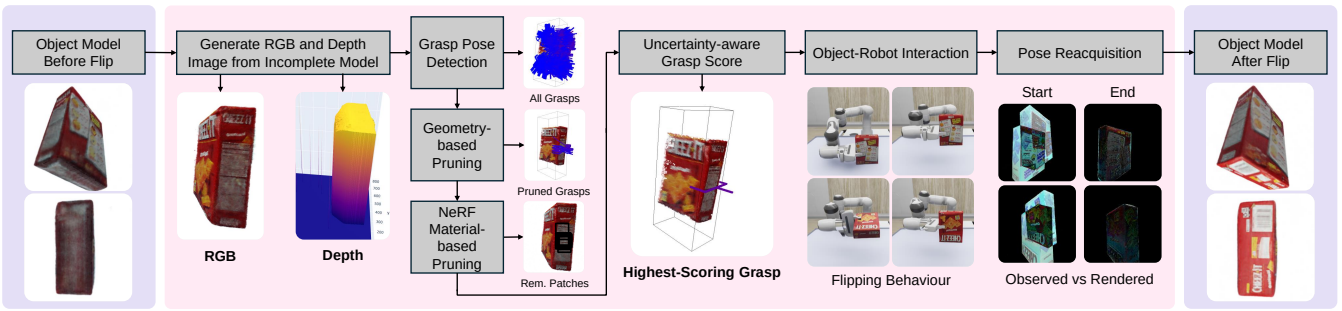


Fig. 7: **Object Re-orientation Approach.** First, the RGB and Depth images are rendered from the object's current NeRF model. Using these, AnyGrasp detects potential grasps, which are then pruned based on the geometry of the generated point cloud and NeRF's material density on grasp patches. The best grasp is selected from the remaining using our uncertainty-aware grasp score. The robot executes the chosen grasp to re-orient the object, and the modified iNeRF is employed to re-acquire the object's pose in its new orientation. We show the quality of the object models before and after the flip. The post-flip model is obtained by capturing images in a re-oriented position and adding them to the training dataset.

VI. Appendix

A. Grasping Methodology Details

As mentioned in Section III-C, after obtaining grasp poses from AnyGrasp, we first prune them on the basis of distance from distance from center of the point cloud and the average opacity of the grasp patch, that is average predicted opacity averaged over the rays of the grasp patch. The predicted opacity $\hat{O}(\mathbf{r})$ corresponding to ray $\mathbf{r}$ is calculated as follows:

$$\hat{O}(\mathbf{r}) = \sum_{i=1}^{N} \alpha_i, \tag{5}$$

$$\alpha_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)\left(1 - \exp(-\sigma_i \delta_i)\right), \tag{6}$$

where σ_i and $\mathbf{c}_i$ respectively denote the density and color predicted by the NeRF model at a sampled point $\mathbf{r}_i = \mathbf{o} + t_i\mathbf{d}$ along ray $\mathbf{r}$ having origin $\mathbf{o}$ and direction $\mathbf{d}$, and $\delta_i = t_{i+1} - t_i$ represents the distance between adjacent sampled points.

Next, we assign a grasp score to each of the filtered grasp pose to select the best grasp among them. Our grasp score is defined as follows

$$G_s = \frac{1-\theta}{U_d} \tag{7}$$

where θ is the angle between the grasp pose and surface normal. θ should be minimized as grasping from a non-lateral grasp increases the probability of the object toppling during or after the flip. U_d is the variance of rendered depths summed over the rays of the grasp patch. We minimize U_d to be certain about the location of the object surface in 3D space near the grasp pose. Its computation is similar to that of (3) with predicted depth $\hat{D}(\mathbf{r})$ being used instead of color. $\hat{D}(\mathbf{r})$ corresponding to ray $\mathbf{r}$ is calculated as follows:

$$\hat{D}(\mathbf{r}) = \sum_{i=1}^{N} \alpha_i t_i, \tag{8}$$

where α_i, t_i are defined as in (5) and (6).

This approach ensures that the grasp poses chosen are not only theoretically viable according to AnyGrasp's criteria but also practically applicable within the constraints and current state of our partial object models. Fig. 7 shows our whole re-orientation pipeline.

B. Implementation Details

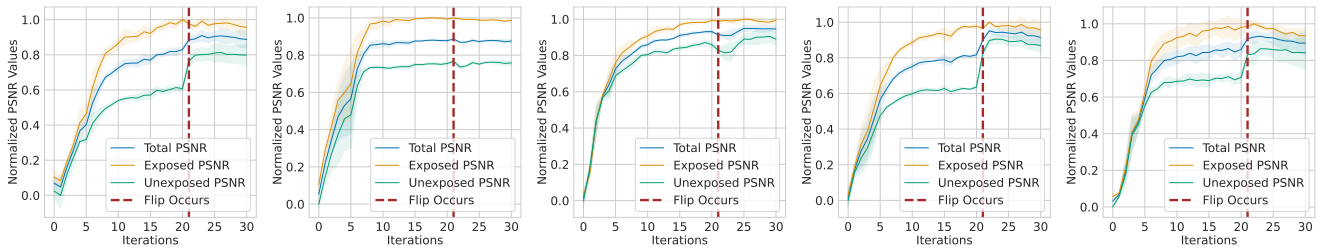
For our experiments, we define $\Gamma(a)$ for a `Move()` action, which transitions the end-effector from (r_1, q_1) to (r_2, q_2) (with r indicating position and q representing rotation in quaternion form), as follows:

$$\Gamma(a) = \alpha_1(1 - d(q_1, q_2)) + \alpha_2 d(r_1, r_2), \tag{9}$$

whereas, for a `Flip()` action, we set $\Gamma(a) = \alpha_3$. The cumulative cost Γ for a sequence of actions is the sum of the costs for individual actions. The cost function parameters, including λ and α_i, play a pivotal role (see 2 and 9). For our purposes, λ is fixed at 1, and the α values are determined based on the relative average durations of their corresponding actions executed by the robot, reflecting a practical consideration of action cost in terms of time. However, the observations in Section IV and VII translate to other values of hyper-parameters as well.

Our methodology is implemented on the Kaolin-Wisp framework [18], utilizing the InstantNGP model [22]. We train an ensemble of five models on a single NVIDIA A40 GPU. For a given pose, we consider the prediction of the ensemble as the mean of the predictions of individual models. On average, training a single NeRF model takes approximately 36 seconds, while determining the next best action requires about 3 minutes. These processes are amenable to parallelization, potentially reducing computation times significantly. The models are trained with images of 800×800 resolution, and PSNR evaluations are conducted using images at their full resolutions.

To align the baselines (see IV-A) with our framework where `Flip()` actions are permissible, We integrated a `Flip()` action. Specifically, for Random and Furthest, a `Flip()` is executed at the first iteration that meets a predefined grasp score threshold (7). This approach is infeasible for ActiveNeRF due to its generation of partial models lacking precise surfaces, which complicates grasp pose determination. Consequently, in the case of ActiveNeRF, we resort to a predetermined flip via an external manipulator at a specific iteration, assuming accurate post-flip object pose knowledge to ensure that the generated models are of the highest quality.



(a) Cheezit Box (b) Basket (c) Mug (d) Rubik's Cube (e) Spam Can

Fig. 8: **Effect of Flip on Model Quality**.We demonstrate the impact of flipping objects on model quality (PSNR). The dashed line indicates the iteration at which the object is flipped. The PSNR is shown for the **Exposed** and **Unexposed** subsets of the validation set, representing camera poses above and below the object center, respectively. In most cases, the Exposed PSNR remains almost constant, while the Unexposed PSNR shows a significant increase, leading to an overall improvement in total PSNR. PSNR values are min-max normalized in each plot.

TABLE II: Ablation results for uncertainty estimation technique. Epi, Total stands for epistemic and overall uncertainty as proposed by [21] (with modifications stated in section VII). F-score* represents the F-score values multiplied by 10

Uncertainty	Basket		Cheezit Box		Mug		Rubik's Cube		Spam Can		**Total**	
	PSNR↑	F-score*↑	PSNR↑	F-score*↑	PSNR↑	F-score*↑	PSNR↑	F-score*↑	PSNR↑	F-score*↑	PSNR↑	F-score*↑
Model quality after 20 iterations **without** grasping												
Ours	**17.2** ± **0.1**	**4.2** ± **0.5**	**21.8** ± **0.2**	**4.3** ± **0.2**	**26.3** ± **0.1**	**6.5** ± **0.2**	**30.3** ± **0.3**	3.8 ± 0.1	**23.9** ± **0.6**	**4.1** ± **0.3**	**23.9** ± **0.1**	**4.6** ± **0.1**
Entropy [4]	14.2 ± 0.3	3.0 ± 0.2	19.8 ± 0.2	3.4 ± 0.1	25.0 ± 0.6	5.6 ± 0.2	27.7 ± 0.6	4.0 ± 0.4	20.3 ± 0.3	4.0 ± 0.2	21.4 ± 0.2	4.0 ± 0.1
Epi [21]	15.5 ± 0.6	3.2 ± 0.3	20.4 ± 0.3	3.7 ± 0.1	25.8 ± 0.3	6.0 ± 0.4	29.0 ± 0.1	**4.5** ± **0.7**	21.0 ± 0.4	3.8 ± 0.1	22.3 ± 0.2	4.2 ± 0.2
Total [21]	16.6 ± 0.4	3.5 ± 0.2	19.7 ± 0.6	3.8 ± 0.3	21.2 ± 2.2	4.1 ± 0.7	25.6 ± 0.5	3.0 ± 0.2	22.1 ± 1.0	3.9 ± 0.1	21.0 ± 0.5	3.7 ± 0.2
Model quality attained given a cost budget of 2 **without** grasping												
Ours	**17.1** ± **0.1**	**3.9** ± **0.3**	**21.4** ± **0.2**	**4.3** ± **0.1**	**26.0** ± **0.3**	**5.8** ± **0.5**	**29.8** ± **0.7**	**4.5** ± **0.2**	**23.7** ± **0.4**	**4.1** ± **0.2**	**23.6** ± **0.2**	**4.5** ± **0.1**
Entropy [4]	14.8 ± 1.0	3.0 ± 0.2	19.3 ± 0.5	3.6 ± 0.1	24.9 ± 0.5	5.5 ± 0.1	27.7 ± 0.6	4.0 ± 0.4	20.3 ± 0.3	4.0 ± 0.2	21.4 ± 0.3	4.0 ± 0.1
Epi [21]	15.2 ± 0.2	3.1 ± 0.2	19.3 ± 0.2	3.4 ± 0.1	25.2 ± 0.3	5.3 ± 0.3	28.4 ± 0.7	4.5 ± 0.4	20.7 ± 0.2	4.0 ± 0.0	21.8 ± 0.2	4.1 ± 0.1
Total [21]	16.6 ± 0.4	3.5 ± 0.1	19.7 ± 0.6	3.8 ± 0.3	21.2 ± 2.2	4.1 ± 0.7	25.6 ± 0.5	3.0 ± 0.2	22.1 ± 1.0	3.9 ± 0.1	21.0 ± 0.5	3.7 ± 0.2

TABLE III: Fraction of Correct Pose Estimation Instances within Bounds
Tight Bound := (Rotation Error $< 2°$ AND Translation Error < 0.5 cm)
Loose Bound:= (Rotation Error $< 5°$ AND Translation Error < 1.0 cm)
Loss- *Single*: SSD of a single image, *Multi*: summed SSD of 4 images

Optimization Method	Tight Bound (in %)		Loose Bound (in %)	
	Single	Multi	Single	Multi
Nelder-Mead [13]	17.9	33.3	30.8	59.0
COBYLA [14] [15]	5.1	23.1	7.7	51.3
Powell [16]	51.3	61.5	56.4	64.1
Ours (Combined)	**51.3**	**69.2**	**61.5**	**82.1**

VII. Ablations

First, to show the necessity of each component of our pipeline, we remove them step-by-step and run for 20 iterations each. The qualitative results are shown in Fig. 6. Next, we delve into the effectiveness of different optimization techniques for pose re-acquisition, including Nelder-Mead [13], COBYLA [15], and Powell's method [16], alongside our approach of selecting the minimum loss among these. The comparative analysis extends to single versus multi-image optimization strategies, as elucidated in Section III-C, with outcomes presented in Table III. Notably, Fig. 6 demonstrates the crucial role of pose re-acquisition, highlighting that its absence results in significantly degraded NeRF models, rendering them impractical for robotic applications.

Finally, we conduct ablation studies on various uncertainty estimation methodologies. We assess the entropy-based uncertainty metric introduced by Lee et al. [4] and the epistemic and total uncertainties delineated by Sünderhauf et al. [21]. They propose quantifying epistemic uncertainty via the aggregation of termination probabilities for points sampled along each ray, noting that uncertainty peaks when rays fail to intersect with the scene. However, this method is not applicable to our scenario, as our focus lies on quantifying object-specific uncertainty rather than that of the entire scene. Their epistemic uncertainty takes on the maximum possible value of 1 for the pixels lying outside the segmented image of the object. Since these pixels should not contribute to object uncertainty, we assign them a value of 0 and run experiments on these modified epistemic and total uncertainties. The results are shown in Table II.

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 *ECCV*, 2020.

[2] A. Yu, V. Ye, M. Tancik, and A. Kanazawa, "pixelNeRF: Neural radiance fields from one or few images," in *CVPR*, 2021.

[3] M. M. Johari, Y. Lepoittevin, and F. Fleuret, "Geonerf: Generalizing nerf with geometry priors," *IEEE international conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[4] S. Lee, L. Chen, J. Wang, A. Liniger, S. Kumar, and F. Yu, "Uncertainty guided policy for active robotic 3d reconstruction using neural radiance fields," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 12 070–12 077, 2022.

[5] K. Lin and B. Yi, "Active view planning for radiance fields," in *Robotics Science and Systems*, 2022.

[6] L. Jin, X. Chen, J. Rückin, and M. Popović, "Neu-nbv: Next best view planning using uncertainty estimation in image-based neural rendering," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023, pp. 11 305–11 312.

[7] X. Pan, Z. Lai, S. Song, and G. Huang, "Activenerf: Learning where to see with uncertainty estimation," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*, Springer, 2022, pp. 230–246.

[8] T. Ren, S. Liu, A. Zeng, *et al.*, *Grounded sam: Assembling open-world models for diverse visual tasks*, 2024. arXiv: 2401.14159 [cs.CV].

[9] S. Liu, Z. Zeng, T. Ren, *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[10] A. Kirillov, E. Mintun, N. Ravi, *et al.*, "Segment anything," *arXiv:2304.02643*, 2023.

[11] H.-S. Fang, C. Wang, H. Fang, *et al.*, "Anygrasp: Robust and efficient grasp perception in spatial and temporal domains," *IEEE Transactions on Robotics*, 2023.

[12] 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.

[13] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[14] M. J. Powell, *A direct search optimization method that models the objective and constraint functions by linear interpolation*. Springer, 1994.

[15] M. J. Powell, "Direct search algorithms for optimization calculations," *Acta numerica*, vol. 7, pp. 287–336, 1998.

[16] M. J. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The computer journal*, vol. 7, no. 2, pp. 155–162, 1964.

[17] B. Calli, A. Singh, J. Bruce, *et al.*, "Yale-cmu-berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017.

[18] T. Takikawa, O. Perel, C. F. Tsang, *et al.*, *Kaolin wisp: A pytorch library and engine for neural fields research*, 2022.

[19] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–13, 2017.

[20] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *Seminal graphics: pioneering efforts that shaped the field*, 1998, pp. 347–353.

[21] N. Sünderhauf, J. Abou-Chakra, and D. Miller, "Density-aware nerf ensembles: Quantifying predictive uncertainty in neural radiance fields," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2023, pp. 9370–9376.

[22] 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.