A Qualitative Results – Sec 5.1

Our qualitative results video showcasing a few of the successful episodes of our trained agent can be found in the supplementary package. We also provide two failure examples and discuss the limitations of our trained agent.

B Training Details for the Policy Network – Sec 4.3

We train our models for 20M frames (unless otherwise specified). All the models share the same visual encoder, three convolutional layers to embed the RGBD observation. The maximum episode length is 200 frames, and the episode fails if the agent does not finish the task before the end of the episode. Using AI2-THOR [24] multi-node training, we render 500 frames per second on 4 machines, each with 4 Tesla T4 GPUs. We use Adam optimizer with a learning rate of 3e-4 and take gradient steps every 128 frames. We consider two objects being in proximity if the centers of objects are less than 20cm apart. We train our conditional segmentation model offline. We use AllenAct [47] framework.

C Training Details for Conditional Segmentation Model – Sec 4.2

We train our conditional segmentation models for 100 epochs. We use Adam optimizer with a learning rate of $1e-4$. We initialize the ResNet backbones with ImageNet pre-trained weights for faster training. Input images and query images are of size 224x224. We use random crop and flipping the image for data augmentation during training. We do not use color jittering as we found it to negatively impact the performance of our model on the validation set.

C.1 Dataset Details.

We also generate a dataset of images from the training scenes to train our segmentation model. Our dataset consists of more than 23K images, including 116K object masks from 120 scenes (where 80 are used for training, 20 for validation, and 20 for test sets). We ensure that the validation and test scenes used to train m-VOLE are not used to train the segmentation model. As explained in Section 3, the model requires query images as input. We collect these query images by taking a crop containing the object of interest from the images of this dataset. We only use the query images collected from the training scenes of AI2-THOR, so the instances depicted in the query images do not overlap with the instances of target objects during inference.

D Reward Ablation – Sec 4.3

Section 4.3 introduced three additional reward elements that are not used in the standard ArmPointNav setup. We ablate the performance gain that each component brings to our model. This analysis can be useful not only for evaluating the performance of our method but also for use in future works. Table 6 includes the results in absence of each of these components. Arm control reward, which encourages the agent to move its arm close to the target objects, is shown to be the most important element of the reward and the agent does poorly without this reward. Getting training off the ground with sparse RL reward for object manipulation is extremely difficult, and by motivating the agent to bring its arm closer to the target object, the training becomes faster and more efficient.

Exploration	Arm Control	Object Visibility	PU	SR	SRwD
✓	-	✓	0.9	0	0
-	✓	✓	47.8	15.4	5.86
✓	✓	-	79.4	53.6	30.1
✓	✓	✓	**81.2**	**59.6**	**31.0**

Table 6. Reward ablations. We investigate the performance of the model in absence of each reward component. Exploration, arm control, and object visibility refer to the rewards for visiting a new state, δ distance to object and observing the object, respectively.

E Details of the Memory Model – Sec 4.1

We have two memory modules, the explicit memory storing object locations and the implicit recurrent network's memory. The first module, calculates a weighted average of the previous estimates of the object's locations. Formally, $\bar{\hat{d}}_O = \frac{1}{T(T+1)/2} \sum_{t=0}^{T} t \times \tau(\hat{d}_O^t, t, T)$, where $\tau(P, i, j)$ is the transformation of point P from the agent's coordinate frame at time i to time j (note that this transformation can be noisy due to noise in agent's motion). The output of this memory, combined with previous visual observations is given to our second memory module, which is a GRU with a hidden layer of size 512. This recurrent network implicitly encodes the previous actions, observations, and object and agent states, and is used to estimate the next action.

F Details of the Noise Models – Sec 5.2

This section provides additional details and visualizations on the noise models used in Section 5.2.

F.1 Agent Motion Noise Model

Murali et al. [32], introduced a noise model for agent's movements and rotation, capturing the noise in motion for a real robot, the Locobot. We use their noise model to ablate the impact of the noise in the agent's movements on our model's performance. To better understand how the noise multiplier (the x-axis in Figure 5 in the main submission) impacts the predicted trajectory, we illustrate a sample trajectory. Figure 6 shows agent's groundtruth and predicted trajectory for different noise multipliers $x = 0.1 - 1$. Note that the paths start from the same initial location, and the predicted paths diverge from the actual trajectory traversed by the agent as the episode progresses and the errors accumulate.

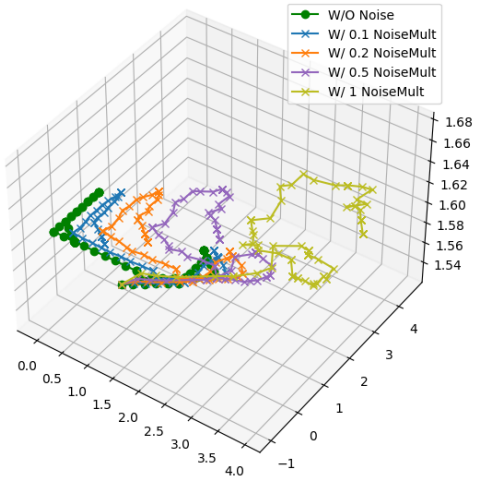

Fig. 6. Noise in Trajectory. The green path shows the trajectory traversed by the agent, and the other paths show the estimated trajectory of the agent for a variety of noise multipliers. Note that even a small noise multiplier $x = 0.2$ can result in a big divergence in the trajectories.

F.2 Depth Noise Model

Redwood distortion noise model, introduced in [7], is designed to model the noise of actual depth sensors (Kinect cameras) with a fixed resolution. This noise model can act as a proxy to estimate the impact of using a real-world noisy depth camera.

F.3 How Does Segmentation Affect the Final Performance? – Sec 5.3

We compare the performance of our approach using different segmentation models in Section 5.3. In this section, we ablate how the performance of the segmen-

tation model affects the final performance of our approach on the task of Object Displacement.

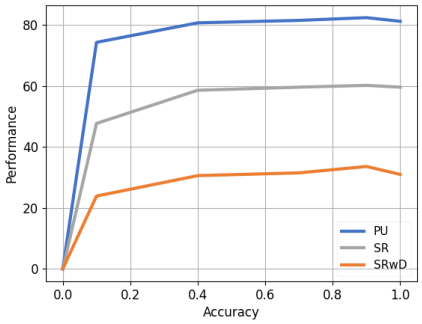

Fig. 7. Impact of Partial Masks.

First, we evaluate how the performance on the final task changes if the target objects are detected correctly, but the mask does not perfectly segment the observed object. In other words, if we use IoU as a metric for the accuracy of the segmentation model, how the accuracy of the segmentation network affects the final performance. For this experiment, at each timestep that the object is visible, we randomly choose $x\%$ of the segmentation mask of the target object to preserve and remove the rest. Figure 7 plots the change in the final performance as we increase x. Note that $x = 0$ presents the ablation where no mask is provided, and $x = 1$ is equivalent to providing the groundtruth segmentation mask. This experiment shows that our network can achieve strong results even if only 10% of the target object is segmented.

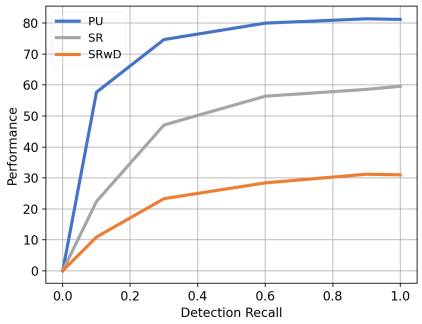

Fig. 8. Impact of Missing Masks.

We also show that if the segmentation mask of the object is only retrieved 30% of the times, across all the timesteps that the target object is visible, our network can achieve approximately similar performance as the one using the ground-truth segmentation mask (Figure 8). For every x on the plot, at each timestep, we remove the segmentation mask of the target object with the probability $1 - x$ and calculate the final performance of the model. Similar to the previous plot, $x = 0$ shows the performance when no mask is provided, and $x = 1$ is equivalent to the groundtruth segmentation mask.

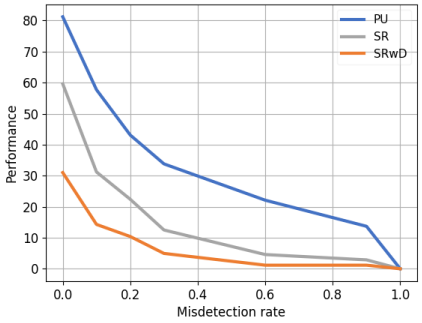

Fig. 9. Impact of Category Confusion.

One of the other main issues that the segmentation models have is detecting wrong objects. For instance, as shown in the supplementary video, the segmentation model might segment a pan in the scene while the query object asks for a pot. Figure 9 shows the final performance of the model for different rates of mis-detections. At each timestep, with the probability of $x\%$, instead of the segmentation mask of the target object, we randomly select the segmentation mask of another object in the scene as the input mask to the model. For instance, metrics at $x = 0.2$ show the model's performance using a segmentation network that detects a wrong object with the probability of 20%.

The conclusion is that even segmentation models with high precision and low recall are beneficial for the Object Displacement task.

G Our method (m-VOLE) outperforming ArmPointNav (APN). – Sec 5.1

APN has access to the exact direction towards the target, so it tends to choose the direct path towards the goal, which is not necessarily a plausible solution. For example, as shown in Fig. 10, the red path represents a shorter path, but the arm collides with other objects if it follows that path. Moreover, we did a further quantitative analysis to investigate m-VOLE superiority to APN (Tab 7). EpLen

Fig. 10. m-VOLE vs APN paths towards the target. The red path illustrates the greedy solution an agent with access to GT direction sensors might take to reach the bowl (Failing due to collision with the oven).

PU and EpLen Success, respectively, represent episode length for the pickup stage and the full task. Our analysis shows that these metrics are lower for APN than m-VOLE. However, the SRwD of APN (success without collision) is lower. This observation shows that APN is more efficient in the number of steps (as it uses the exact direction) but does not handle collisions well.

Model	EpLen PU	EpLen Success	SRwD
ArmPointNav	46.3	78.8	28.5
m-VOLE w/ GT mask	49.3	87.1	31.0

Table 7. m-VOLE vs APN.

H Code and Data

Our code is based on ManipulaTHOR [11] and AllenAct [47]. We use the APND dataset provided in [11]. The dataset can be downloaded from their website. The code can be found on the website.

I Limitations

Here, we discuss two main limitations of the work. First, we consider separate modules for segmentation and depth perception. Hence, their errors have a cascading effect. Another design choice we made was to abstract away grasping. While this allowed us to focus on other challenging aspects of the problem, moving beyond simulation will likely require methods to account for real-world graspers. These are some of our study's primary limitations, which serve as avenues for future work.