I. METHOD

A. Perception Module

Building graph $\boldsymbol{z}_t^\omega = (\mathcal{O}_t, \mathcal{E}_t)$ from point cloud $\bar{y_t} \in \mathbb{R}^{N\times 3}$ contains two phases. First, we sample $\mathcal{O}_t^{\text{fps}}$ from the point cloud using farthest-point sampling. We found that sampled particles from the point cloud are likely to be at the edge of underneath objects. To bias sampled particles towards object centers, we define $\mathcal{O}_t$ as mass centers of $\mathcal{O}_t^{\text{fps}}$ neighbor points. For example, for $o_t^{i,\text{fps}} \in \mathcal{O}_t^{\text{fps}}$, we find the corresponding $o_t^i \in \mathcal{O}_t$ by applying

$$\begin{aligned} \bar{y_t}' &= \{y | y \in \bar{y_t}, ||y - o_t^{i,\text{fps}}||_2 \leq r_{\text{center}}\} \\ o_t^i &= \frac{1}{|\bar{y_t}'|} \sum_{y \in \bar{y_t}'} y. \end{aligned} \tag{1}$$

r_{center} is the hyperparameter to determine the neighbor points.

To find edges $\mathcal{E}_t$, we first approximate particle displacements $\Delta\mathcal{O}_t$ using the action u_t. We compute the sweeping region given the action u_t. If o_t^i is within the sweeping region, Δo_t^i is the vector from o_t^i to the pusher end. We define $\hat{\mathcal{O}}_t = \Delta\mathcal{O}_t + \mathcal{O}_t$. For $\hat{o_t}^i$, $(\hat{o_t}^i, \hat{o_t}^j) \in \mathcal{E}_t$ if it satisfies the following criteria:

$$\begin{aligned} ||\hat{o_t}^i - \hat{o_t}^j||_2 < r_{\text{edge}} \\ \hat{o_t}^j \in \text{kNN}(\hat{o_t}^i). \end{aligned} \tag{2}$$

r_{edge} is the hyperparameter to determine the distance needed for two nodes to interact. $\text{kNN}(\hat{o_t}^i)$ is the set of k-nearest-neighbor of the node $\hat{o_t}^i$ and k is a hyperparameter.

B. Regressor Module

We use a regularizer $R(\omega)$ to encourage efficiency as mentioned in Equation 7. $R(\omega)$ is defined as the following:

$$R(\omega) = w_R c(\boldsymbol{z}_o, y_g)\omega.$$

w_R is a pre-defined hyperparameter. $c(\boldsymbol{z}_0, y_g)$ is the task objective for the current representation. This means that $R(\omega)$ increases linearly as ω increases. We include $c(\boldsymbol{z}_0, y_g)$ in $R(\omega)$ because we observe that $c^*(\omega)$ tends to be larger when the initial task objective is larger.

C. Control Module

The task objective takes in the representation $\boldsymbol{z}_t^\omega = (\mathcal{O}_t, \mathcal{E}_t)$ and binary heatmap y_g. Given camera intrinsics, we transform $\mathcal{O}_t$ to $\mathcal{P}_t \in \mathbb{R}^{\omega\times 2}$ in image space. We could also extract points within goal region $\mathcal{Q}_t \in \mathbb{R}^{M\times 2}$ using the binary heatmap y_g. Using the farthest-point sampling, we sample $\mathcal{Q}_t' \in \mathbb{R}^{M'\times 2}$ from $\mathcal{Q}_t$. Given these definitions, we compute the task objective using the following equation:

$$\begin{aligned} &c(\boldsymbol{z}_t, y_g) \\ &= \sum_{p_i \in \mathcal{P}_t} \min_{q_j \in \mathcal{Q}_t} ||p_i - q_j||_2 + \sum_{q_j \in \mathcal{Q}_t'} \min_{p_i \in \mathcal{P}_t} ||p_i - q_j||_2. \end{aligned} \tag{3}$$

Instead of using $\mathcal{Q}_t$ in the second term, we use $\mathcal{Q}_t'$ so that the first and second terms can be balanced.

For the MPC hyperparameter in Algorithm 1, we use 20 sampled action sequences with $T = 1$. The number of gradient descent iterations is 200.

II. EXPERIMENT

A. Distribution Distance

The distribution distance is computed similarly to the task objective. We use the RGBD observation to segment out the foreground object and obtain foreground pixels $\mathcal{F}_t \in \mathbb{R}^{F\times 2}$. Similar to Equation 4, the distribution distance d is defined using the following equation:

$$d = \sum_{f_i \in \mathcal{F}_t} \min_{q_j \in \mathcal{Q}_t} ||f_i - q_j||_2 + \sum_{q_j \in \mathcal{Q}_t} \min_{f_i \in \mathcal{F}_t} ||f_i - q_j||_2 \tag{4}$$

B. High-Level Planner for ***Sort*** Task

We use the A* search algorithm to find the high-level path. We represent every blob of object piles as a circle with a fixed radius in image space. Therefore, the state for one blob can be represented as its 2D blob center in image space. For the search algorithm, if there are k blobs in the scene, the node is the concatenation of k blob centers. Neighbors of one node are those nodes that can be reached by changing one blob center in a single step with no collision.

To accelerate motion planning, we divide the image space into a sparse grid. The blob center will only be on the grid. In addition, to encourage a path with fewer steps, we add a constant cost for each path so that a path with fewer steps is preferred.