# 7 Appendix

## 7.1 Skills

In this section, we describe the skills used, their preconditions, parameter generation methods, and termination conditions. The rod width is $w$, length is $l$, and height is $h$.

A common precondition for all skills is that any desired pose is within reachable ranges of the robot of the robots workspace where control works well, which are specified as a $0.5m \times 0.6$ rectangle in $x$ and $y$.

Low-level motion plans for *Pick* and *LiftAndDrop* move in 2D in an obstacle free $z_{\text{free}}$ region to get to a target pose. *OpenDrawer* also moves in $z_{\text{free}}$ until it needs to open the drawer.

Some common hyperparameters:

1. $\epsilon_{gripper}$: An amount beyond $w$ to close the gripper, which should be very small or the skill will not terminate.
2. $\epsilon_p$: Maximum position error for $[x, y, z]$
3. $\epsilon_\psi$: Maximum yaw error.
4. $w_{\text{open}}$ : The width at which the gripper is considered open, which is wider than $w$.

### 7.1.1 Pick

This skill moves the robot to a grasp around an object. The parameters are $[x_d, y_d, z_d, \psi_d]$ which represent the desired final pose of the end-effector. We assume rods are grasped at their narrowest axis so the gripper width is $w - \epsilon_g$

The precondition is that no rod is within $l/2$ of the robot gripper, and the open gripper position will not cause collisions with other rods. The termination condition is that the end-effector pose is close to the desired pose: $||[x_d, y_d, z_d] - [x, y, z]|| < \epsilon_p$, $|\psi_d - \psi| < \epsilon_\psi$ and the gripper width is less than some $w - \epsilon_g$

Pick is relatively easy to model with a kinematic model, though failures can occur from collision with the drawer not modeled in the analytical model in `RodInDrawer`, reaching joint limits, or perception error. In our setup with our parameter generators, these errors were rare, but would not be ruled out by the skill precondition.

Grasps around the center of the rod, and a fixed distance from the ends are used to compute $[x_d, y_d, z_d, \psi_d]$.

### 7.1.2 *LiftAndDrop*

*LiftAndDrop* lifts the rod from the ground, goes to $[x_d, y_d, z_d, \psi_d]$, and then opens the gripper, allowing the rod to drop.

The precondition is that a rod is within $l/2$ of the gripper. The termination condition, like *Pick*, requires $||[x_d, y_d, z_d] - [x, y, z]|| < \epsilon_p$, $|\psi_d - \psi| < \epsilon_\psi$ and that the gripper width is greater than $w_{\text{open}}$.

Task-specific parameters $[x_d, y_d, z_d]$ are sampled within a rectangle defined by the goal (box or drawer) at a fixed height above the object. The rectangle is some fraction of the goal region, which was conservatively set at 2 cm.

*LiftAndDrop* is a placement task, which is much harder to predict, especially at a height because the rod interacts with the ground, goal box, and goal drawer, potentially bouncing unpredictably. Some analytical model failures result from hitting the walls of the box while grasping a rod from the side, and dropping the rod on the top of the chest holding the drawer. The simulator model is fairly accurate though it does not perfectly model all interactions with the drawer.

### 7.1.3 *OpenDrawer*

For this skill, a robot opens a drawer by placing the gripper parallel to the front edge of the drawer and then moves the gripper to a second goal pose along the drawer axis. This skill assumes the

drawer is opened along the negative $y$ axis in world frame. The parameters are $[x_d, y_d, z_d, \psi_d, y_{\text{open}}]$ The first goal pose is $[x_d, y_{d,i}, z_d, \psi_d]$. The robot pushes the drawer by moving along the negative $y$ axis to a final $y_{d,f}$ The precondition is that there is no rod already in the hand, and that the drawer is not already open. The parameters are $[x_d, y_{d,i}, z_d, \psi_d, y_{d,f}]$. Parameter generation is object-centric, sampled based on the drawer pose to a random opened amount.

The precondition for *OpenDrawer* is that the gripper does not have any objects within $l/2$ of its starting pose.

The termination condition is reached once the robot has reached the final pose at $[x_d, y_d - y_{\text{open}}, z_d, \psi_d]$.

Parameters are sampled by selecting $[x_d, y_d, z_d]$ so the robot is a fixed distance (2 cm) behind the front of the drawer in y, 1 cm offset from the center in $x$, and 4 cm above the drawer bottom in $z$. and $\psi_d$ is selected to the gripper is parallel to the long side of the front of the drawer. $y_{\text{open}}$ was randomly sampled from a small range: [14,17] cm.

### 7.2 Skill-Level Models

#### 7.2.1 Simulator Models

We use a simulator to model complex interactions with rods, target objects, and the robot. As a result, the simulator should be accurate in at least some contexts to be useful. It is non-trivial to tune the parameters to get realistic behavior with small objects that rotate in-hand and behave realistically with thin objects such as drawer walls or box walls, so simulator parameter tuning is important. The parameters used for each IsaacGym simulator that do not match default parameters are shown in Table 3. We also show the joint properties and gains used for the Franka below the simulator properties. Lastly we show simulator environment properties. The friction coefficient between the rods and other objects in the scene including the gripper is sampled from a small range: $[0.50, 0.51]$. The torsional friction coefficient is $1 \times 10^{-13}$. The mass is sampled from the range $[50, 51]$ g. and the dimensions are the same as they are in the real world: 2.3 x 18.5 x 2.3 cm. The drawer is modeled as a prismatic joint with approximated dynamics and limits from visual inspection.

Because the state in our experiments does not include joint angles, the robot state for the simulator model is underconstrained. To solve this problem, we use inverse-kinematics, optimizing for an upright elbow using the tools provided in [27].

| Property | Value |
|---|---|
| engine | physx |
| dt | $2.5 \times 10^{-3}$ for `RodInBox` and $5 \times 10^{-3}$ for `RodInDrawer` |
| num_position_iterations | 8 |
| rest_offset | 0 |
| contact_offset | $1 \times 10^{-3}$ |
| friction_offset_threshold | $1 \times 10^{-3}$ |
| friction_correlation_distance | $5 \times 10^{-4}$ |
| stiffness | 1000 |
| damping | 500 |
| joint stiffness | [0, 0, 0, 0, 0, 0, 0, $1 \times 10^3$, $1 \times 10^3$] |
| joint damping | [0, 0, 0, 0, 0, 0, 0, 200, 200] |
| effort | [0, 0, 0, 0, 0, 0, 0, 10, 10] |

Table 3: Simulation parameters for environment including physics simulator properties, attractor properties, and joint properties.

#### 7.2.2 Analytical Models

When there is no object in the gripper, *Analytical (Pick & Place)* predicts that the robot goes to the commanded goal pose described by the action parameters, ${}^{\text{goal}}T_{ee}$, and all other objects do not move. If there is an object close to the gripper, the transform from robot gripper to object, ${}^{\text{ee}}T_{rod}$, is computed from the initial state: ${}^{\text{ee}}T_{rod} = {}^{\text{world}}T_{ee}^{-1} \cdot {}^{\text{world}}T_{rod}$, using the closest rod to the gripper as the target rod. The end effector is assumed to move to the desired pose then drop the rod to the ground so the new pose of the rod is computed using the initial ${}^{\text{ee}}T_{rod}$ by applying the transform

to the end-effector at the goal pose. Let $h_{\text{rod}}$ be the height of the rod, and $h_{\text{drawer}}$ the height of the drawer at the rod's final pose. The final $z$ value after the rod is dropped is $\frac{h_{\text{rod}}}{2}$ if on the floor, and $h_{\text{drawer}} + \frac{h_{\text{rod}}}{2}$ if the rod is dropped over the drawer.

*Analytical (Drawer)* uses the geometry of both the drawer and the grippers to compute whether at least one gripper is in contact with the drawer edge. If they are in contact, then the drawer moves the amount specified in the skill parameters. Otherwise the drawer does not move. This model does not account for the joint limits of the drawer.

### 7.3 MDE Training

In this section, we describe how we train the MDEs. As mentioned in Section 4.1, training data is collected by computing plans using all models, and then executing them. We collected 26 trajectories solving `RodInBox` and 17 trajectories solving `RodInDrawer`. 15% of samples were held out for testing. Since real-robot data is expensive to collect, data augmentation is critical to good MDE performance. In addition to doubling the dataset by swapping the two rod locations, we also added random noise from a normal distribution with standard deviation 1 cm for state features and 3 cm for parameter features. Of the augmented dataset, 5% were removed from the training set to use for validation which informed early stopping to prevent overfitting. The MLP hyperparameters are shown in Table 4. The optimizer we use is the Adam optimizer [28].

For all models, $c_1 = 3$ (underestimate loss coefficient) and $c_2 = 1$ (overestimate loss coefficient).

| Hyperparameter | Value |
|---|---|
| Number layers | 2 |
| Activation | ReLU |
| Learning rate | $5 \times 10^{-3}$ |
| Hidden units per layer | 64 |
| Adam L2 weight decay | $5 \times 10^{-3}$ |

Table 4: MLP hyperparameters

### 7.4 Planning Algorithm

Here, we describe more details for the planner, including hyperparameters, heuristics and pseudocode.

The planner hyperparameters for `RodInBox` are $w = [10, 1]$, $\epsilon = 5$ (for WA*). For `RodInDrawer` the planner hyperparameters are $w = [10, 1.1, 1]$, $\epsilon = 10$

The heuristic we use is task dependent. For `RodInBox`, the heuristic is the distance between the target rod and the center of the box. For `RodInDrawer`, the heuristic is the distance between the target rod and center of the box, plus the distance between the current drawer location and the minimum required to open it to place a rod in.

### 7.4.1 Algorithm pseudocode

**Algorithm 1** Planning using multiple queues given:
start state $s_0$,
goal set $\mathcal{S}^g$,
skills that generate $a$ given $s$,
list of models from slowest to fastest $[M_0, M_1, \ldots, M_K]$, and
model cost weights $[w_0, w_1 \ldots w_K]$

**procedure** SEARCH($s_0, \mathcal{S}^g, \mathbb{A}, [M_0, M_1, \ldots, M_K]$, **w**)
  Initialize start node with $s_0$ and $A_{inc}(s_0) \leftarrow \emptyset$
  insert start node into open queues with $f \leftarrow g + wh(s_0)$
  **while** any open queue is nonempty **do**
    $i \leftarrow \arg\min_{i<=K} w_i \text{OPEN}_i.MinKey()$
    $s_{\text{best}} \leftarrow Pop(\text{OPEN}_i.MinKey())$
    **if** $A_{inc}(s_{\text{best}})$ is nonempty **then**
      $A \leftarrow A_{inc}(s_{\text{best}})$
    **else**
      Generate $A$ using parameter generators from each skill which satisfy $\text{pre}(a)$
    **end if**
    **if** $s_{\text{best}} \in \mathcal{S}^g$ **then**
      Return path from node
    **end if**
    **if** $i = 0$ **then**
      $successors \leftarrow$ FULLEXPANSION($s_{\text{best}}$)
      Add $s_{\text{best}}$ to all closed sets
    **else**
      $successors \leftarrow$ PARTIALEXPANSION($s_{\text{best}}$)
      Add $s_{\text{best}}$ to CLOSED$_i$
    **end if**
    **for** $s' \in successors$ **do**
      $A_{inc}(s') \leftarrow \emptyset$
      **for** $i \in \{0, K\}$ **do** ▷ Update all graphs
        **if** $s' \notin$ CLOSED$_i$ and $g(s') > g(s) + c(s, a, s')$ **then**
          $g(s') \leftarrow g(s) + c(s, a, s')$
          Add $s'$ to all open queues
        **end if**
      **end for**
    **end for**
  **end while**
**end procedure**

**Algorithm 2** Full expansion of node

**procedure** FULLEXPANSION($s$)
  $successors \leftarrow \emptyset$
  **for** $a \in A$ **do**
    **if** $s \in \text{pre}(a)$ **then**
      Compute $s'$ using highest (fastest) $i$ such that $(s, a) \in \text{pre}(M_i)$
      Add $s'$ to $successors$
    **end if**
  **end for**
  **return** $successors$
**end procedure**

**Algorithm 3** Partial expansion procedure for only computing successors in the precondition for $M_i$

**procedure** PARTIALEXPANSION($s, i$)
  $successors \leftarrow \emptyset$
  **for** $a \in A$ **do**
    **if then**$(s, a) \in \text{pre}(M_i)$
      Compute $s'$ using $M_i(s, a)$
      Add $s'$ to $successors$
    **else**
      Add $a$ to $A_{inc}(s)$
    **end if**
  **end for**
  return $successors$
**end procedure**

Last we show the $d_{\text{max}}$ parameters for each skill in Table 5. These currently are chosen by hand based on task accuracy needs. In future work, we plan to explore automating selecting $d_{\text{max}}$, as well as use a different $d$ to reflect the consequences of predicting an incorrect contact mode. Predicting the wrong contact mode can lead to very different dynamics despite Euclidean distances being very similar.

| Task | Skill | $d_{\text{max}}$ |
|---|---|---|
| `RodInBox` | *Pick* | 3 cm |
| | *LiftAndDrop* | 8 cm |
| `RodInDrawer` | *OpenDrawer* | 6 cm |
| | *Pick* | 3 cm |
| | *LiftAndDrop* | 5 cm |

Table 5: $d_{\text{max}}$ values used for each skill and task.