FrankaShapeFitRL

Dipika Khullar

September 24, 2024

Contents

1 Introduction

In this work, we design an environment to insert three shapes using the 7-DOF Franka Panda robot in Isaac Gym. The robot is tasked with manipulating three distinct geometric objects (rectangular prism, triangular prism, and hexagonal prism) and sequentially inserting them into their corresponding apertures on a wall. Our goal is to use Proximal Policy Optimization (PPO) to generalize the shape-fitting task across different object geometries and environment conditions.

We initially focused on the simpler task of placing a cube into a large cubby by leveraging joint torque control, which provided better handling of contact forces compared to operational space control (OSC). This early phase allowed us to explore the robot's ability to perform basic object manipulation using torque-based control, especially for precision tasks like insertion.

Next, we engineer the reward function to enforce a sequential task structure: first inserting the cube, followed by the hexagonal prism, and finally the triangular prism. This was achieved by using a flag system to signal task completion, allowing the policy to focus on one object at a time. We also introduced observation space masking to accelerate the learning process by selectively focusing on relevant parts of the state space.

Finally, to encourage policy generalization, we randomized the object focus of the reward function in each episode, enabling the robot to handle different shapes and positions dynamically. The work can be found on GitHub

2 Environment and Simulation

The simulation was built using the Isaac Gym environment, where the robot is modeled with its stock configuration from the library. The objects and the wall were designed using CAD software (OnShape) and imported into IssacGymEnvs for the simulation. This section covers the robot setup, object design, wall iterations, and simulation configuration.

2.1 Franka Robot Setup

The Franka robot is a 7-DOF articulated arm robot with a two-finger gripper on the end effector. It has 7 rotational degrees of freedom (DOF) in the arm and 2 translational DOF in the gripper. Its state space encompasses its joint angles and angular velocities for the joints: q_i and $\dot{q}_i$ for each joint $i \in [0, 6]$, and the gripper finger positions, d, and velocities, $\dot{d}$, for both the left and right finger prismatic joints: $[d_{lf}, \dot{d}_{lf}, d_{rf}, \dot{d}_{rf}]$.

$$s \in [q_0, q_1, q_2, q_3, q_4, q_5, q_6, \dot{q_0}, \dot{q_1}, \dot{q_2}, \dot{q_3}, \dot{q_4}, \dot{q_5}, \dot{q_6}, d_{lf}, \dot{d}_{lf}, d_{rf}, \dot{d}_{rf}]$$

The action space of the Franka robot consists of joint torque values for all 7 revolute joints (τ_i), and the force commands for the two gripper fingers ($f_{left\_finger}, f_{right\_finger}$).

$$\mathbf{a} \in [\tau_0, \tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, f_{lf}, f_{rf}]$$

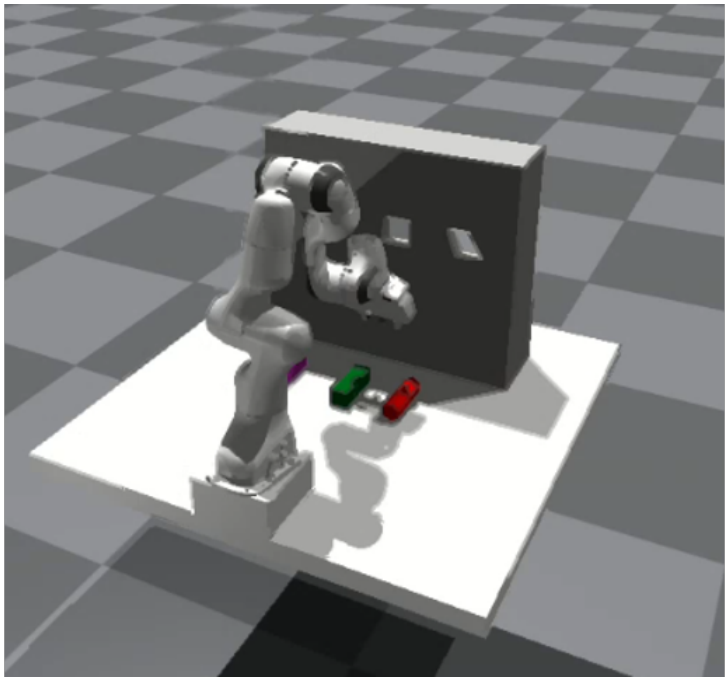

Figure 1: Final wall design with three distinct shapes.

2.2 Insertion Wall and Object Design

The final insertion wall contains three horizontally aligned holes, as illustrated in Figure 2. Each hole is uniquely designed to accommodate one of the test objects: a rectangular prism, a triangular prism, and a hexagonal prism. The holes have a diameter of 15 cm, while the test objects have a uniform diameter of 5 cm and a length of 15 cm. For each object, the goal position is defined as the bottom center of the corresponding hole, with an offset applied to account for the object's height.

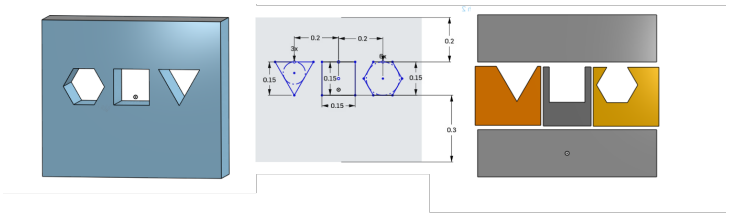

Figure 2: Final concept, diagram and implementation of the wall.

The design of the insertion wall underwent multiple iterations to balance task complexity and optimize robot-object interaction. The initial design consisted of a large, vertically aligned hole, created by stacking pre-defined box geometries from the IsaacGym library. This configuration was utilized during initial experiments to understand differences between insertion and placement tasks.

In the second iteration (not shown), a modular wall design was introduced to facilitate more efficient convex decomposition for physics-based calculations. However, the vertical alignment of the holes presented a new issue: the robot experienced difficulties when attempting to insert objects into the higher holes, primarily due to limitations in joint configurations and reachability.

The wall was ultimately redesigned with a horizontal alignment of the holes, as illustrated in Figure 2. This horizontal arrangement ensures all three insertion points (cube, hexagon, and triangle) fall within the robot's operational radius. The horizontal alignment allowed the robot to access each hole without excessive joint articulation, enabling smoother execution of the insertion tasks. This setup was finalized

for experimentation after confirming the robot could reliably reach all target positions.

3 Methodology

The goal of this project is to train a policy $a_t = \pi_\theta(s_t)$ that takes in an observation of the robot and objects, and provides the appropriate actions that allow the robot to insert the object into the hole. We test three progressively difficult tasks for training a policy, as well as test two different reward styles.

3.1 Basic Cube Insertion

In this scenario, as seen in Figure **??**, the Franka robot must insert a simple 5 cm cube into a large hole (30 cm x 30 cm). This is a simple task to prove that a policy is learning on an insertion task.

The observation space includes:

- The pose of the cube's center of mass (represented by the a cartesian position and rotational quaternion): $\text{cube}_{\text{pos}}(x, y, z), \text{cube}_{\text{rot}}(w, xi, yi, zi)$
- The distance from the cube to the goal position in the hole: $\text{cube}_{\text{pos}} - \text{goal}_{\text{pos}} = \text{dist}_{cube\_to\_hole}(x, y, z)$
- The pose of the end effector: This is from the fixed link that connects the fingers to the arm. The pose is represented by a three dimensional cartesian position and a four dimensional rotational quaternion: $\text{eef}_{\text{pos}}(x, y, z), \text{eef}_{\text{rot}}(w, xi, yi, zi)$
- The joint angles (q_i) and gripper joint values (d_{lf}, d_{rf}) of the franka arm and gripper: $[q_0, q_1, q_2, q_3, q_4, q_5, q_6, d_{lf}, d_{rf}]$

$$\mathbf{o} \in [\text{cube}_{\text{pos}}, \text{cube}_{\text{rot}}, \text{dist}_{cube\_to\_hole}, \text{eef}_{\text{pos}}, \text{eef}_{\text{rot}}, q_0, q_1, q_2, q_3, q_4, q_5, q_6, d_{lf}, d_{rf}]$$

The action space returned by the policy is the joint torques for each Franka revolute joint, and a single boolean for closing the gripper (after the boolean is received from the policy, it is copied to both the left and right finger as the same value).

$$\mathbf{a} \in [\tau_0, \tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \text{bool\_gripper}]$$

Reward Function:

$$\text{dist\_reward} = 1 - \tanh\left(10 \cdot \frac{\|cp - gp\| + \|cp - lfp\| + \|cp - rfp\|}{3}\right)$$

$$\text{is\_lifted} = (cp - tp) > 4\,\text{cm}$$

$$\text{align\_reward} = (1 - \tanh(10.0 \cdot \|cp - hp\|)) \cdot \text{is\_lifted}$$

$$\text{completion\_flag} = \text{cube\_xy\_aligned} \,\&\, \text{cube\_z\_aligned} \,\&\, \text{gripper\_released}$$

$$\text{cube\_xy\_aligned} = \|\,(cp - hp)\,[:2]\|$$

$$\text{cube\_z\_aligned} = \|(cp - hp)[2]\|$$

$$\text{gripper\_released} = \|cp - gp\| > 2\,\text{cm}$$

$$\text{Reward} = \begin{cases} 16, & \text{if completion\_flag is true} \\ 0.1 \cdot \max(\text{dist\_reward}, \text{align\_reward}) + & \\ \quad 1.5 \cdot \text{is\_lifted} + 2.0 \cdot \text{align\_reward}, & \text{otherwise} \end{cases}$$

where cp is the cube position, gp is the gripper position, lfp is the left finger position, rfp is the right finger position, tp is the table surface position, and hp is the hole position.

3.2 Sequential Reward

This section addresses the more challenging problem of a three-hole wall with smaller holes, formulated as a sequential insertion task. The objective is to learn a reward function that guides the agent through sequential insertions, where each object (cube, hexagon, and triangle) must be placed into its corresponding hole in a specific order.

To enforce this, a binary flag mechanism is introduced, which restricts the agent from interacting with the hexagon until the cube has been successfully placed. This flag prevents any hexagon-related movements, ensuring that the agent focuses solely on completing the cube insertion first. The same procedure is then transferred to the triangle once the hexagon task is completed.

To train the policy $a_t = \pi(s_t)$ with this sequential reward method, we modify the observations passed into the policy. The full observation space includes both the cube, hexagon, and triangle related states, along with other states like end-effector positions and Franka joint angles. The dimensionality of the observation space is increased, but not all of the information is always relevant to the current object. Irrelevant observations for each task were masked, effectively clipping the observation space and maintaining a fixed input size. For example, the observation space for the hexagon is masked (set to zeros) until the cube task is completed. Once the cube is placed correctly, the hexagon's observations are unmasked, allowing the agent to start interacting with it. The action space remains identical as experiment 3.1.

The observation space o_t at time step t is defined as:

$$o_t = [\mathbf{o}_{\text{cube}}, \mathbf{o}_{\text{hexagon}}, \mathbf{o}_{\text{eef}}, \mathbf{o}_{\text{joints}}]$$

Where:

- $\mathbf{o}_{\text{cube}}$: $\text{Tcube}_{\text{pos}}(x, y, z), \text{cube}_{\text{rot}}(w, xi, yi, zi)$
- $\mathbf{o}_{\text{hexagon}}$: $\text{hexagon}_{\text{pos}}(x, y, z), \text{hexagon}_{\text{rot}}(w, xi, yi, zi)$
- $\mathbf{o}_{\text{triangle}}$: $\text{triangle}_{\text{pos}}(x, y, z), \text{triangle}_{\text{rot}}(w, xi, yi, zi)$
- $\mathbf{o}_{\text{eef}}$: $\text{eef}_{\text{pos}}(x, y, z), \text{eef}_{\text{rot}}(w, xi, yi, zi)$
- $\mathbf{o}_{\text{joints}}$: The joint angles (q_i) and gripper joint values (d_{lf}, d_{rf}) of the franka arm and gripper: $[q_0, q_1, q_2, q_3, q_4, q_5, q_6, d_{lf}, d_{rf}]$

We define t binary flags $\mathbf{F}_{\text{cube}}$, such that:

$$\mathbf{F}_{\text{cube}} = \begin{cases} 1, & \text{if the cube has been placed} \\ 0, & \text{otherwise} \end{cases}$$

$$\mathbf{F}_{\text{hexagon}} = \begin{cases} 1, & \text{if the hexagon has been placed} \\ 0, & \text{otherwise} \end{cases}$$

The masked observation space o'_t becomes:

$$o'_t = [\mathbf{o}_{\text{cube}}, \mathbf{F}_{\text{cube}} \cdot \mathbf{o}_{\text{hexagon}}, \mathbf{F}_{\text{cube}} \cdot \mathbf{F}_{\text{hexagon}} \cdot \mathbf{o}_{\text{triangle}}, \mathbf{o}_{\text{eef}}, \mathbf{o}_{\text{joints}}]$$

The total reward r_t is defined as:

$$r_t = r_{\text{cube}}(t) + \mathbf{F}_{\text{cube}} \cdot r_{\text{hexagon}}(t) + \mathbf{F}_{\text{cube}} \cdot \mathbf{F}_{\text{hexagon}} \cdot r_{\text{triangle}}(t)$$

Where:

- $r_{\text{cube}}(t)$: Reward for placing the cube
- $r_{\text{hexagon}}(t)$: Reward for placing the hexagon
- $r_{\text{triangle}}(t)$: Reward for placing the triangle

3.3 Generalized Reward

The sequential reward task is limited in its scalability. Each new object requires expanding the policy's observation space. One solution to this is to train a policy that is agnostic to the object it is trying to insert. A single policy can then be used sequentially for any number of object insertions in a task.

To allow the policy to learn a more general insertion task, we randomize the objects and starting positions in each episode. The observation space can be reverted back to the size used for the single object insertion task. The action space also remains unchanged from the single object insertion task.

The observation space includes:

- The state of the current object (position and rotation of the center of mass): $\text{obj}_{\text{pos}}(x, y, z), \text{obj}_{\text{rot}}(w, xi, yi, zi)$
- The distance from the cube to the goal position in the hole: $\text{obj}_{\text{pos}} - \text{goal}_{\text{pos}} = \text{dist}_{obj\_to\_hole}(x, y, z)$
- The pose of the end effector: This is from the fixed link that connects the fingers to the arm. The pose is represented by a three dimensional cartesian position and a four dimensional rotational quaternion: $\text{eef}_{\text{pos}}(x, y, z), \text{eef}_{\text{rot}}(w, xi, yi, zi)$
- The joint angles (q_i) and gripper joint values (d_{lf}, d_{rf}) of the franka arm and gripper: $[q_0, q_1, q_2, q_3, q_4, q_5, q_6, d_{lf}, d_{rf}]$

$$\mathbf{o} \in [\text{obj}_{\text{pos}}, \text{obj}_{\text{rot}}, \text{obj}_{cube\_to\_hole}, \text{eef}_{\text{pos}}, \text{eef}_{\text{rot}}, q_0, q_1, q_2, q_3, q_4, q_5, q_6, d_{lf}, d_{rf}]$$

The same reward structure utilized for the single object insertion task is applied in this experiment. During testing, the policy is run continuously. Once an object reaches its target state, the script transitions to a new object, updating the policy's observation space accordingly to focus on the next object for insertion.

The randomization of objects and initial states was not sufficient for the policy to generalize effectively. This approach did not yield the desired results. Additional reward engineering will be required to introduce more intermediate sub goals and guide the policy more effectively toward successful task completion.

4 Results

Table 1: Summary of Experiment Results

Experiment	Environments	Steps	Cumulative Reward	Control Method	Generalization	WANB Run
Basic Cube Insertion	16384	6576	3988	OSC	N/A	wanb run
Basic Cube Insertion	16384	6576	3988.37	Joint Torque	N/A	wanb run
Small Hole Insert - Sequential Reward	16384	14216	3886.94	Joint Torque	No	wanb run
Small Hole Insert - Generalized Reward	16384	4832	394.88	Joint Torque	No	wanb run

4.1 Basic Cube Insertion

This experiment demonstrated that we are able to perform a sudo insertion task, and gave us confidence to move forward with the harder task of inserting multiple shapes into a wall with smaller holes. The policy was able to easily learn the single object insertion for the cube into the large hole. We use 16384 environments and train for 6576 steps, achieving a cumulative reward of 3988.

This first experiment was instrumental in understanding the specific requirements of insertion tasks, especially when compared to object placement tasks, such as those in the Franka Cube Stack environment. One of the key considerations was selecting between operational space control (OSC) and joint torque control. Unlike object placement, insertion tasks demand precise force control, particularly at the

moment of contact with the target hole. Joint torque control proved to be more effective than OSC for managing the fine-tuned force application needed during the insertion process. The insights from these trials also highlighted the increased sensitivity to alignment errors during insertion, which differs from the less force-sensitive requirements of standard object placement.

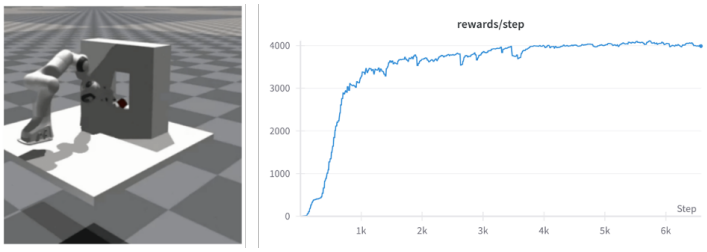

Figure 3: Successfully place a cube in a large cube shaped hole.

4.2 Sequential Reward

In this experiment we tried to create a joint reward function for insertion by masking the observation space and aggregating rewards for each object. The following works for a single object, and fails to let go of the second object.

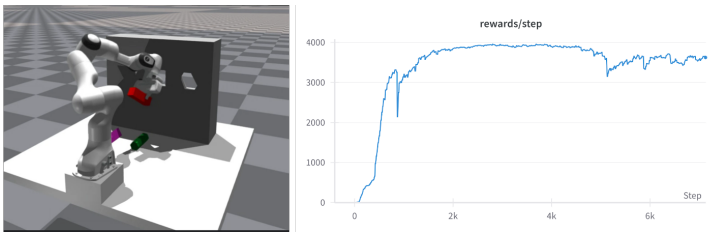

Figure 4: Sequential reward setup for 3 object insertion

4.3 Generalized Reward

This method aimed to improve generalization by training the robot to pick up and insert different objects in each episode. While it succeeds in picking up the first object, it fails on the second and third due to task complexity, possible reward structuring issues, and the challenge of managing dependencies between different steps of the task (object placements).

1. Overfitting to a Single Object: When randomizing objects between episodes, the robot's policy likely overfitted to the first object it encountered. Since the policy was reset or updated at the end of each episode, the model may not have had enough continuity or training time with other objects, causing it to over-specialize in one object at a time.

2. Insufficient Exposure to Object Variability: Even though the object was randomized, the robot was trained on one object per episode. This could have led to insufficient exposure to the variation in object types within a single training session, preventing the policy from effectively generalizing across different shapes.

3. Limited Object Variability per Update: Since the robot was only interacting with one object per episode, it missed the chance to train on multiple objects during the same policy update. As a result, the policy couldn't properly balance learning across all object types, leading to sub-optimal performance when switching between them.

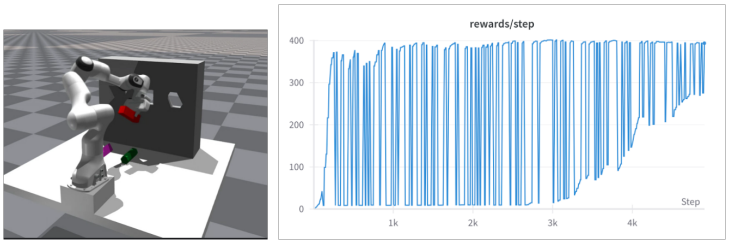

Figure 5: Generalized reward setup for 3 object insertion

5 Future Work

5.1 Multiple Object Generalization

In experiment 3.2, we create a set of experiments to deduce what part of the sequential reward is causing the failure.

- Sequential Object Manipulation: Test the ability of the policy to sequentially pick up and release each object (cube, hexagon, triangle) in isolation. This will simplify the task, reducing the complexity of the observation space, and help identify if the issue stems from handling the increased observation space size in the original setup. This will allow me to rule out the increased observation space size as a potential bottleneck to learning. If this simple task succeeds, we know the general setup of the sequential policy is correct.
- Adjust Gripper Goal Position: Modify the goal position of the gripper to be closer to the edge of the object (e.g., the cube) instead of its center, potentially improving insertion accuracy and grip control.
- Prolonged Training: Increase training duration to determine if the current model is getting stuck in a local minimum. Prolonged training may help the policy discover better strategies for object manipulation.
- Orientation Subgoals: Introduce intermediate subgoals for object orientation, addressing the issue where the robot grips the object from an incorrect angle, such as the back of the prism. This will ensure the robot rotates the object to the correct position (e.g., a 180-degree rotation) before attempting insertion.

In Experiment 3.3, several avenues can be explored to improve the policy's ability to generalize object insertion tasks across varied scenarios.

- Parallel Environments with Object Randomization: Train the policy across multiple parallel environments, each initialized with different objects (cube, hexagon, triangle) in randomized positions. This approach aims to reduce overfitting by exposing the policy to a broader distribution of object shapes and positions, thereby improving generalization to unseen configurations. Implementing such an enhancement may require significant changes to the current code architecture, but it could yield better robustness by ensuring diverse training examples across environments
- Wall and Hole Position Randomization: Randomize the positions of the wall and holes at the beginning of each episode, varying the spatial configuration of the environment.Spatial randomization will prevent the policy from memorizing specific object-to-hole placements, encouraging a more adaptable insertion strategy.
- Reward Scaling for Sequential Tasks: Implement an off-policy learning algorithm to incorporate experience replay. Off-policy methods such as Deep Q-Networks (DQN) or Soft Actor-Critic (SAC) could be explored for this purpose, leveraging experience replay buffers to store and re-use diverse

interaction data. This allows the policy to learn from previous interactions, not just the most recent episode. This change would enable the policy to update using experiences from previous episodes, thereby utilizing a broader range of interactions with different objects and goals. The ability to learn from past experiences, even those involving different objects or hole configurations, could accelerate convergence and improve the policy's capability to generalize across tasks.

5.2 Sim-to-Real Transfer

In simulation, precise object and goal locations are readily available, but this is not the case in the physical world. We will need to integrate a vision-based model using a camera for object detection and pose estimation (YOLO, Resnet, Faster R-CNN, etc). We can use a vision model for object and hole detection, paired with a depth sensor (like Intel RealSense or a stereo camera) to achieve accurate 3D localization of the test objects and corresponding hole positions.

Moreover, in simulation, the policy is provided with explicit knowledge of which object belongs in which hole. In a real-world setup, this mapping will require an additional logic layer. The system must not only detect the holes but also classify their shapes and dimensions using techniques such as contour detection or template matching. Once the shapes are identified, we will need to implement a reward function to match the detected objects with their compatible holes, using size and shape constraints.

Training in the real world is resource-intensive and time-consuming. To mitigate this, the majority of training will be performed in simulation, using a camera-based setup to simulate the real-world sensory input. This requires creating a virtual camera in the simulation environment to capture images of the scene, and then using those images for object and hole detection and localization, thus mimicking real-world conditions for observation space variables.

We will need to introduce domain randomization in order for simulation to transfer effectively to the real world. This will involve randomizing aspects of the environment, such as lighting conditions, object colors, textures, and even the shapes of the holes. By introducing variability in these factors during simulation, the learned policy will become more robust to discrepancies between the simulation and real-world environments. Noise should be added to sensor data to account for inaccuracies in real-world perception, ensuring that the robot can handle slight discrepancies in object localization and interaction. We could also explore a hybrid approach combining operational space control (OSC) and joint-torque control could be considered, as these allow for better compliance and adaptability to external forces encountered in physical interactions.