Dual-Arm Manipulation and Policy Learning at IPRL: System Design, Control, and Performance Baselines

Aarya Sumuk
Advisor: Prof. Jeannette Bohg
Mentors: Jingyun Yang, Carlota Parés, Tyler Lum

December 2025

Abstract

This report summarizes the design, implementation, and evaluation of a dual-arm manipulation platform based on two Franka FR3 robots at IPRL, developed in collaboration with Agile Robotics. The work covers simulation environments, dual-arm workspace design, teleoperation and impedance control, collision avoidance, and learning-based controllers for contact-rich assembly tasks, including tight-tolerance USB insertion. The primary focus is on robust system design and clear performance baselines for key tasks.

1 Introduction

This project focused on building and evaluating a dual-arm manipulation platform for assembly-style tasks using two Franka FR3 arms. The main objectives were:

- design and compare dual-arm mounting configurations in simulation;
- implement safe, practical teleoperation pipelines, including impedance control;
- transition to real hardware and debug integration issues;
- implement collision avoidance for dual-arm operation; and
- establish performance baselines for imitation learning and reinforcement learning on insertion tasks.

The resulting system includes:

1. dual-arm simulation and real-robot setups suitable for table-top assembly;
2. teleoperation pipelines for both OSC and Cartesian impedance control;
3. data collection pipelines for multimodal robot datasets; and
4. performance baselines on pick-and-place and insertion tasks, including tight-tolerance USB insertion.

2 Simulation and Workspace Design

2.1 MuJoCo and Robosuite Environments

The work began in simulation using MuJoCo, with two Franka arms modeled from Menagerie assets to understand joint layouts, collision geometries, and basic kinematics.

To move faster, the project transitioned to Robosuite, which provides:

- modular task and environment abstractions;

- built-in operational space control (OSC) controllers; and
- support for multiple arms and objects in a single scene.

Custom dual-arm environments were created with:

- multiple mounting configurations for the two Frankas;
- standard objects and custom CAD-derived assets (e.g., gears, cup-and-saucer);
- convex-decomposed collision meshes to better approximate real contact behavior.

2.2 Workspace and Mounting Evaluation

Several dual-arm mounting configurations were evaluated in simulation, including:

- upright arms facing forward;
- arms tilted downward at 45°;
- arms tilted downward and inclined forward;
- arms pointing horizontally outwards in opposite directions; and
- an Agile-style configuration with arms mounted at $\sim$45° upwards with slight forward inclination ("elbows-up").

A scripted pick-and-place coverage test was used:

- each arm attempted multiple grasp types over a grid of cube start positions on the table;
- a fixed target region in the center of the table defined a successful placement;
- success/failure heatmaps were generated per arm and grasp type.

Key finding: the Agile-style "elbows-up" configuration achieved the best overall coverage, particularly for side grasps, with reduced table and self-collisions compared to upright or downward-facing mounts. This configuration was later adopted for the physical setup.

3 Teleoperation and Control Stack

3.1 OSC-Based Teleoperation

A dual-arm teleoperation pipeline was implemented using Robosuite's OSC controller as the low-level interface. Oculus controllers provided 6D pose deltas that were mapped to Cartesian end-effector commands.

The teleoperation scheme used:

- a **deadman switch**: arm motion only while a grip button is pressed;
- **binary gripper commands**: trigger press toggles gripper closure;
- rate-limited Cartesian deltas and position/velocity clamps for safety.

This pattern was used consistently across simulation and real hardware, enabling quick iteration and straightforward operator training.

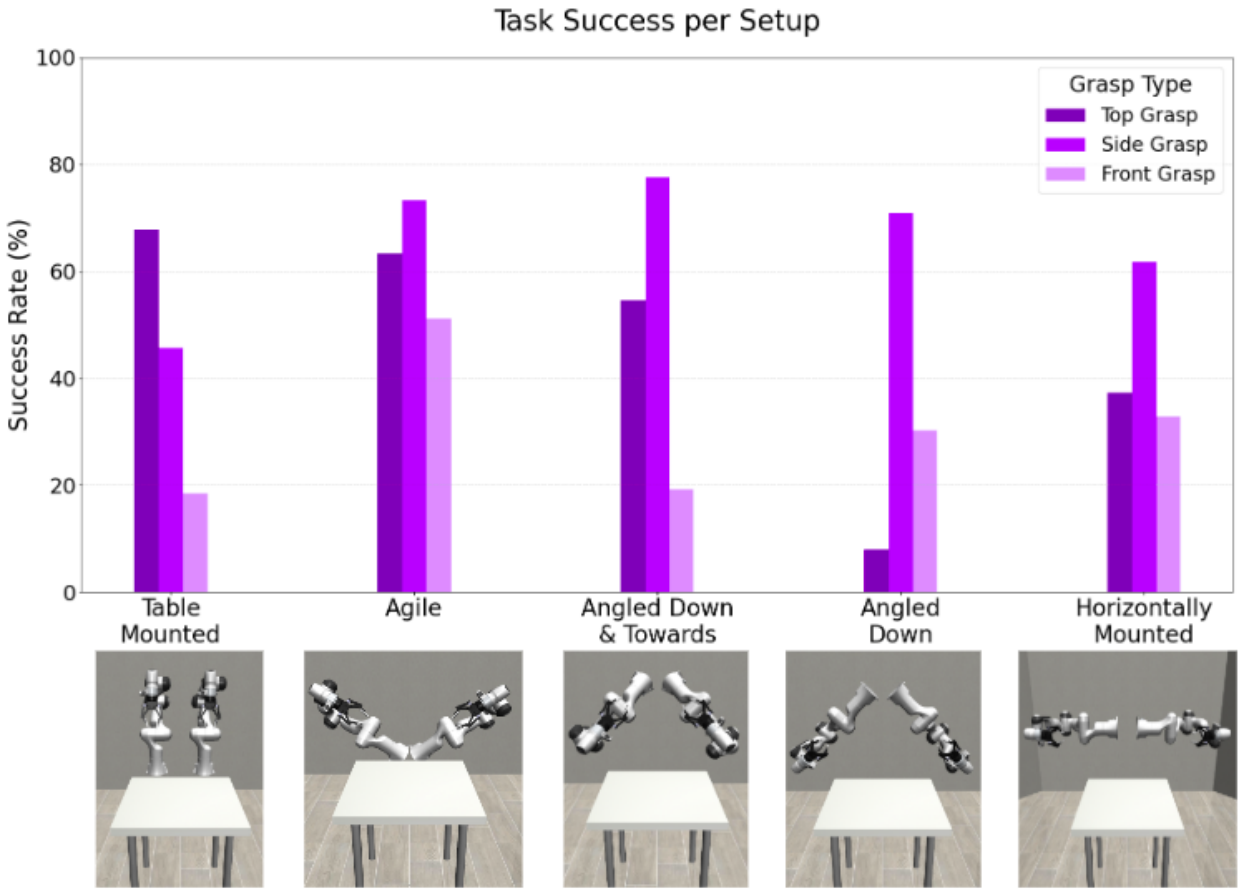


Figure 1: Top: Front views of the candidate Franka FR3 mounting configurations evaluated in simulation. Bottom: Success rates of cube placing trials across configurations, stratified by grasp type.

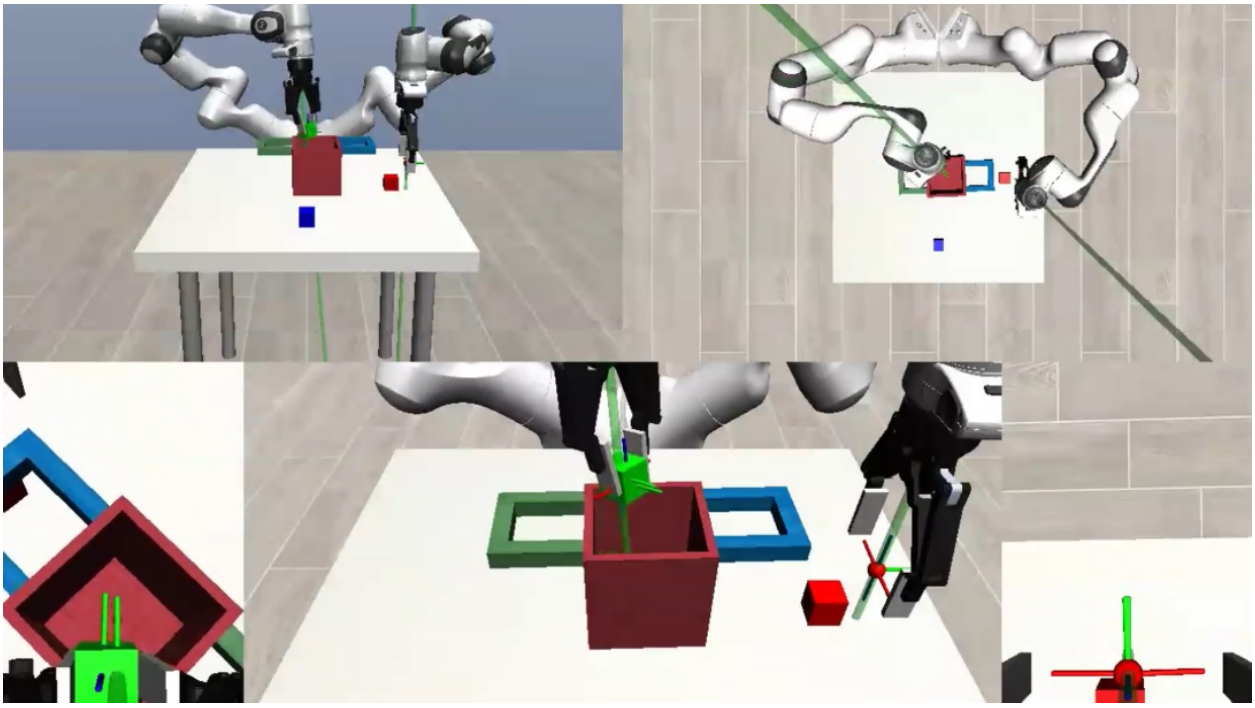


Figure 2: Simulated teleoperation setup showcasing multiple synchronized views.

3.2 Haptic Devices and Workspace Mapping

Haply haptic devices were explored as alternative teleoperation interfaces. Integration required:

- reading and processing device state and force feedback via the Haply API;
- mapping the device's small workspace to the robot workspace using drift and re-centering;
- adapting control to binary gripper commands.

In simulation, force feedback based on simulated force-torque sensors was found to be noisy and unreliable for precise haptic interaction, so Oculus-based teleoperation remained the primary interface.

3.3 Impedance Control for Contact-Rich Tasks

While stiff OSC worked well for free-space motion and simple pick-and-place, it was brittle for insertion tasks, often generating high contact forces and triggering safety thresholds.

To address this, a ROS 2-compatible Cartesian impedance controller (from the LucasG2001 codebase) was integrated into the Franka stack. The teleoperation pipeline was modified so that:

- teleoperation commands define desired Cartesian poses or motion references;
- the impedance controller enforces desired stiffness and damping, allowing compliance in contact.

Careful gain tuning and motion limit adjustments were required to avoid both instability and excessive compliance. The resulting setup provided smoother insertion behavior and fewer abrupt

safety stops.

4 Real-Robot Integration

4.1 Mechanical Setup and Controllers

Once the physical Frankas arrived, an initial dual-arm workspace was assembled using Vention components, with upright mounting as a temporary configuration. Later, the Agile-style mount was installed to match the configuration validated in simulation.

Controller integration involved:

- testing existing lab controllers (Polymetis, OpenSAI), which were incompatible with the latest Franka firmware;
- adopting a lightweight Cartesian controller (`franky`) for arm control;
- integrating `pyrobotiqgripper` for the Robotiq grippers.

The earlier OSC teleoperation design was ported onto these controllers, yielding real-world teleoperation behavior closely matching the simulation.

4.2 Gravity Compensation and Desk Configuration

After remounting the arms on the inclined Agile-style mount, persistent "force threshold reached" errors occurred even at rest. The underlying cause was a mismatch between:

- the physical base orientation; and
- the gravity vector configured in the Franka Desk software.

Using the Franka Desk Swagger API, the gravity orientation was updated to match the new mount. This resolved the erroneous forces and highlighted the importance of verifying gravity compensation whenever the base orientation changes.

4.3 Standardized Test Objects

To evaluate teleoperation and learning on realistic tasks, 3D-printed objects were prepared based on the NIST board and InDUSTReal items. These objects enabled:

- structured pick-and-place and alignment tasks;
- repeatable evaluation across teleoperation, diffusion policies, and RL.

5 Collision Avoidance

5.1 Self-Collision Avoidance

The lab controller included a self-collision avoidance mechanism based on potential fields:

- links were modeled as repulsive fields;
- as the arm approached self-collision, commanded motion was redirected away from dangerous configurations based on link velocities and distances.

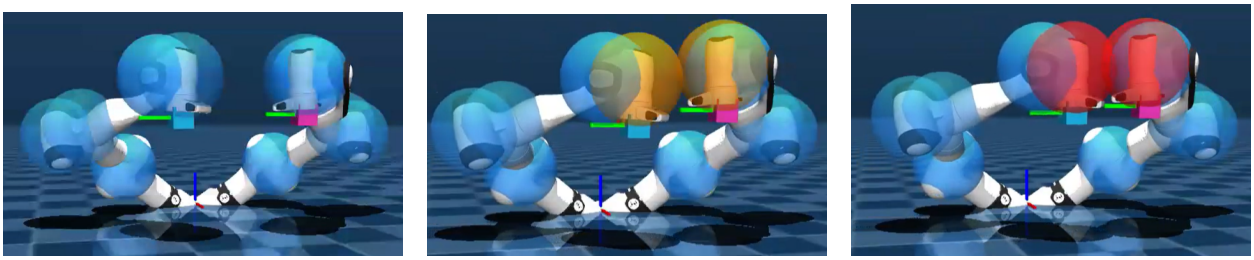
This mechanism provided a practical baseline for safe single-arm operation.

5.2 Dual-Arm Potential-Field Controller

No suitable off-the-shelf dual-arm collision avoidance solution was found, so a custom MuJoCo-based controller was implemented:

- selected joints of each arm were approximated as spheres;
- pairwise distances between spheres across arms were monitored;
- when a distance fell below a warning threshold, a repulsive velocity component proportional to penetration was added;
- when a critical distance was reached, commanded velocities in the collision direction were suppressed.

This method worked well in practice under moderate motion speeds. Very rapid motions could still lead to contacts before repulsion fully engaged, illustrating the trade-off between responsiveness and conservatism.



(a) Proxy spheres attached to both manipulators.

(b) Arms entering the warning zone.

(c) Critical overlap; inward motion suppressed.

Figure 3: Dual-arm collision avoidance in simulation.

6 Imitation Learning with Diffusion Policies

6.1 Dataset Collection

To evaluate imitation learning, a single-arm cube pick-and-place task was implemented on the real robot. A multimodal dataset was collected consisting of:

- full robot state and joint information;
- gripper state;
- time-synchronized image streams from ZED and RealSense cameras.

Approximately fifty episodes were recorded via teleoperation, then reformatted into the Zarr structure required by the REAL diffusion policy framework, with custom configuration files defining the observation and action spaces.

6.2 Representation and Observation Design

Initial policies trained with:

- quaternion-based orientation representations; and
- continuous gripper signals

performed poorly in real rollouts, exhibiting drift and out-of-distribution behavior.

Practical modifications that improved performance included:

- representing orientation using the first two columns of the rotation matrix instead of quaternions, reducing ambiguity;
- discretizing the gripper state to a binary open/closed variable;
- simplifying observations to a single, stable camera view;
- temporal downsampling to reduce sequence length.

6.3 Sanity Checks and Final Policy

Before real deployment, several sanity checks were performed:

- feeding demonstration observations into the trained policy and comparing predicted actions to ground truth;
- gradually replacing stored observations with live data while monitoring action deviations;
- verifying normalization and denormalization by reconstructing known state-action pairs.

After these adjustments, a reliable diffusion policy was obtained for single-arm cube pick-and-place on the real robot, showing that diffusion-based imitation learning can work well when dataset design and representations are carefully chosen.

7 Reinforcement Learning for Insertion Tasks

7.1 SERL in Simulation

The SERL framework for real-world RL was first re-implemented in simulation for a Franka cube pick-and-place task. This required:

- connecting SERL to the simulated environment;
- resolving CUDA and dependency issues on local GPU hardware;
- ensuring consistent interfaces between simulation, policy, and replay buffer.

This stage established a working SERL setup tailored to the lab environment.

7.2 Real-World USB Insertion

The main RL target was a tight-tolerance USB insertion task on the real Franka. The SERL stack was adapted to:

- interface with the lab's robot drivers and camera setup;
- define an environment exposing appropriate observations (robot state, vision) and action spaces;
- clip and constrain actions following SERL safety conventions.

With separate learner and actor processes running on the real system, the policy achieved approximately 88% success on USB insertion after around one hour of training.

7.3 Peg Insertion Baseline

To contextualize the USB results, a peg insertion task modeled after FMB was also implemented. This task has looser mechanical tolerances. Using SERL with similar hyperparameters, the system reached nearly 100% success within roughly twenty minutes of real training, consistent with previously reported performance.

7.4 Comparison with Behavior Cloning

In parallel, a behavior cloning (BC) policy for USB insertion was developed by another student using real demonstrations and DAgger-style corrective rollouts. Despite these efforts, the BC policy achieved only about 10% success.

This contrast highlighted that:

- tight-tolerance insertion is highly sensitive to small misalignments and contact dynamics;
- pure imitation learning is brittle when demonstrations do not cover the full range of off-nominal states;
- RL with online interaction can adapt to contacts and misalignments, albeit at the cost of real-world exploration.

8 SERL Performance Summary

Table 1 summarizes the observed SERL performance across three insertion tasks, in terms of approximate training time and final success rate.

Task	Training time	Final success rate
FMB peg insertion (without force)	~ 20 min	$\approx 100\%$
USB insertion (without force)	~ 60 min	$\approx 88\%$
USB insertion (with force)	~ 40 min	$\approx 100\%$

Table 1: SERL performance across different insertion tasks. Training times and success rates are approximate and refer to the end of training runs on the real system.

For the *USB insertion (with force)* configuration, the wrist force–torque sensor on the Franka was calibrated, integrated into the SERL observation pipeline, and used during training with the same actor–learner setup. Compared to the no-force configuration, adding force feedback reduced the required training time from roughly 60 minutes to about 40 minutes (a reduction of ~ 20 minutes, or around 33%) and increased the final success rate from approximately 88% to about 100% (an improvement of roughly 12 percentage points).

9 Discussion

Across the components described above, several higher-level observations emerge:

- **Mounting and workspace matter.** The Agile-style elbows-up configuration significantly improved workspace coverage and reduced collisions, validating the value of simulation-based mounting evaluation before hardware deployment.
- **Control stack design is critical.** A simple, consistent OSC-based teleoperation interface,

upgraded with Cartesian impedance for contact-rich tasks, provided a robust foundation for both human operation and data collection.

- **Configuration details can dominate.** Gravity compensation alignment, controller compatibility with firmware, and camera pose consistency had large effects on system stability and learning performance.
- **Learning is often data and representation-limited.** Orientation and gripper representations, dataset size and structure, and normalization sanity checks were often the difference between failure and success for diffusion policies.
- **RL, BC, and force sensing play different roles.** For easy tasks with generous tolerances, BC and diffusion policies can be sufficient given good demonstrations. For tight-tolerance insertion, SERL-based RL clearly outperformed BC, and incorporating force sensing further reduced training time and improved final success rates on USB insertion.

10 Summary of Contributions

The main technical contributions of this work are:

- **Dual-arm simulation and workspace design:** construction of dual-arm Franka FR3 environments in MuJoCo/Robosuite, integration of the Agile Robotics mount, and quantitative workspace evaluation showing the advantages of an elbows-up configuration.
- **Teleoperation and impedance control:** implementation of a robust OSC-based teleoperation pipeline with Oculus input, extension to Haply devices, and integration of a Cartesian impedance controller for safer insertion-style tasks.
- **Real-robot integration:** physical setup of a dual-arm Franka FR3 workspace using Vention hardware and the Agile mount, resolution of controller compatibility issues, and correction of gravity compensation via the Franka Desk Swagger API.
- **Collision avoidance:** implementation of a dual-arm potential-field collision avoidance controller in simulation, and preliminary exploration of OSCBF-based safety constraints for possible future use.
- **Imitation learning with diffusion policies:** collection of multimodal real-world demonstration datasets, adaptation of the REAL diffusion policy framework to these data, and identification of practical representation and dataset design choices enabling a reliable pick-and-place policy.
- **Force sensing for insertion:** calibration of the wrist force–torque sensor, integration of force readings into the SERL observation pipeline, and demonstration that adding force feedback for USB insertion reduced training time from roughly 60 to about 40 minutes while increasing the final success rate from $\sim 88\%$ to about 100%.
- **RL baselines for insertion:** re-implementation of SERL for Franka FR3 tasks in simulation and on real hardware, achieving strong performance on FMB peg insertion and USB insertion (with and without force sensing), and empirical comparison against a behavior cloning baseline that highlighted the advantages of RL for tight-tolerance insertion.