Lab 10: Autonomous Manipulation

Vision-Based Block Sorting & Stacking

EE 3002 - JDS Robotics

Lahore University of Management Sciences (LUMS)

Spring Semester 2026

1 Introduction

In Lab 9, we utilized coordinate transformations to perform a 2D planar drawing task. In this lab, we advance to 3D spatial reasoning and logic-based manipulation.

You will be provided with five 30mm × 30mm blocks, each labeled with a unique AprilTag (`tag_0` through `tag_4`). Moving into the Z-axis introduces severe mathematical complexities in path planning, specifically regarding collision avoidance and dynamically calculating the target height of the growing tower.

Lab Objectives Overview:

- **Task 1:** Single Tag Pick and Place (Cartesian constraints, Yaw extraction, Soft-Grip & Z-height math).
- **Task 2:** Grasp Symmetry Optimization (Yaw constraint challenge).
- **Task 3:** Full Sorting & Stacking Pipeline .

Hardware Demonstration Video

Before beginning, watch the successful execution pipeline, including a breakdown of the kinematic singularities (the "wrist flip") that occur when trajectory constraints are not properly configured.

→ Click Here to Watch the Demo ←

2 Phase 1: Configuring the Vision System

Before launching the robot, you must instruct the `apriltag_ros` package to actively search for your specific blocks and know their physical sizes.

Navigate to the `apriltag_ros` configuration directory and modify the `tags.yaml` and `tags.param.yaml` files to define `tag_0` through `tag_4`. You must specify the tag family (36h11) and the exact size of the tag in meters (0.03m). If the size is incorrect, your Z-axis distance calculations will fail.

3 Phase 2: Unified Launch Architecture

To streamline deployment and ensure the gripper receives the correct torque parameters (`modes.yaml`), we have provided a unified Python Launch File. This compiles the hardware control, camera driver, and AprilTag nodes into a single executable command.

Download the Custom Launch File

→ Click Here to View `lab10_launch.py` ←

Instructions: Click the link above, copy the Python code, and save it inside your ROS 2 workspace's launch directory as `lab10_launch.py`.

3.1 Execution Instructions

Once the file is saved, open a single terminal and launch the entire environment:

```
ros2 launch lab10_launch.py
```

In a second terminal, execute your Python script (which you will build in Phase 3):

```
python3 stack.py
```

4 Phase 3: The Laboratory Tasks

You will build your `stack.py` script sequentially by completing the following three tasks.

4.1 Task 1: Single Tag Pick and Place

Goal: Program the manipulator to locate a *single* block, pick it up, and place it at the drop zone. This introduces Cartesian constraints, orientation, dynamic Z-height math, and gripper torque control.

1A. Reading the TF2 Buffer

To find where a specific tag is relative to the robot's base, query the TF tree:

```
trans = tf_buffer.lookup_transform('wx250s/base_link', 'tag_0', rclpy.time.Time
    ())
raw_x = trans.transform.translation.x
raw_y = trans.transform.translation.y
```

1B. Extracting Yaw from Quaternions

AprilTags return rotation as a 4D Quaternion (x, y, z, w). First, convert this to a 1D Euler angle (Yaw) so the gripper aligns with the block:

```
import math

def get_yaw_from_quaternion(q):
    siny_cosp = 2 * (q.w * q.z + q.x * q.y)
    cosy_cosp = 1 - 2 * (q.y * q.y + q.z * q.z)
    return math.atan2(siny_cosp, cosy_cosp)
```

1C. Dynamic Z-Calculations

Unlike dropping items in a bin, building a tower means the drop-off location changes dynamically based on the loop iteration (i). Ensure your hover clearance is mathematically linked to your drop height:

$$Z_{\text{place}} = Z_{\text{table\_base}} + (i \times \text{Box\_Thickness}) \tag{1}$$

1D. Commanding the Arm (Singularity Avoidance)

If you use joint angles near the table limits, the Inverse Kinematics solver may flip the wrist upside down. Force the solver to plan a straight line by utilizing Cartesian components and clamping the pitch:

```
safe_pitch = 1.45 # Clamped to prevent IK singularities

bot.arm.set_ee_pose_components(
    x=target_x,
    y=target_y,
    z=hover_z,
    pitch=safe_pitch,
    yaw=tag_yaw
)
```

1E. Soft-Grip Control

Because the foam blocks crush easily, you must reduce the motor torque *before* commanding the grasp to prevent structural instability:

```
# Hardware Initialization
bot.gripper.set_pressure(0.4) # 40% torque for a gentle hold
bot.gripper.release()

# Inside your execution sequence:
bot.gripper.grasp()
```

1F. The Stacking Drop Zone

To ensure the arm remains within its optimal kinematic workspace, your designated Drop Zone for the tower can be defined at $X = 0.20, Y = -0.25$. Use these coordinates for all `place` commands.

4.2 Task 2: Grasp Symmetry Optimization

Goal: Optimize the gripper's rotation so it takes the shortest path to a flat edge.

The raw yaw extracted in Task 1 forces the gripper to align perfectly with the sticker's orientation. However, because the blocks are square, there are four valid grasping angles separated by 90° ($\frac{\pi}{2}$ rad).

Your Challenge: Write a mathematical function using the modulo operator (`%`) that takes the raw `tag_yaw` and constrains it between $-\frac{\pi}{4}$ and $+\frac{\pi}{4}$. This will ensure the robot's wrist never twists more than 45° to grab a block.

4.3 Task 3: Full Sorting & Stacking Pipeline

Goal: Incorporate the rest of the tags. Scan the workspace, sort the 5 blocks in ascending (or descending) order, and vertically stack them into a precise tower.

The Loop

Wrap your Task 1 and Task 2 logic into a `for` loop that iterates over a list of tags (`['tag_0', 'tag_1', ...]`). Your loop must execute the following state machine continuously:

- **Search:** Spin the node until `tag_i` is found.
- **Pick:** Move to (X, Y) at a safe `hover_z` (10cm clearance). Plunge to `pick_z` (0.035m), grasp with reduced pressure, and lift back to hover.
- **Place:** Traverse to the designated Drop Zone. Plunge to the dynamically calculated `place_z`, release the gripper, and lift clear before looping to the next block.