

Assignment 2

Mattia Cozza (mattia.cozza@studenti.unipd.it)
Lorenzo Tomai (lorenzo.tomai@studenti.unipd.it)
Nicola Berti (nicola.berti.6@studenti.unipd.it)

January 18, 2026

1 System Architecture

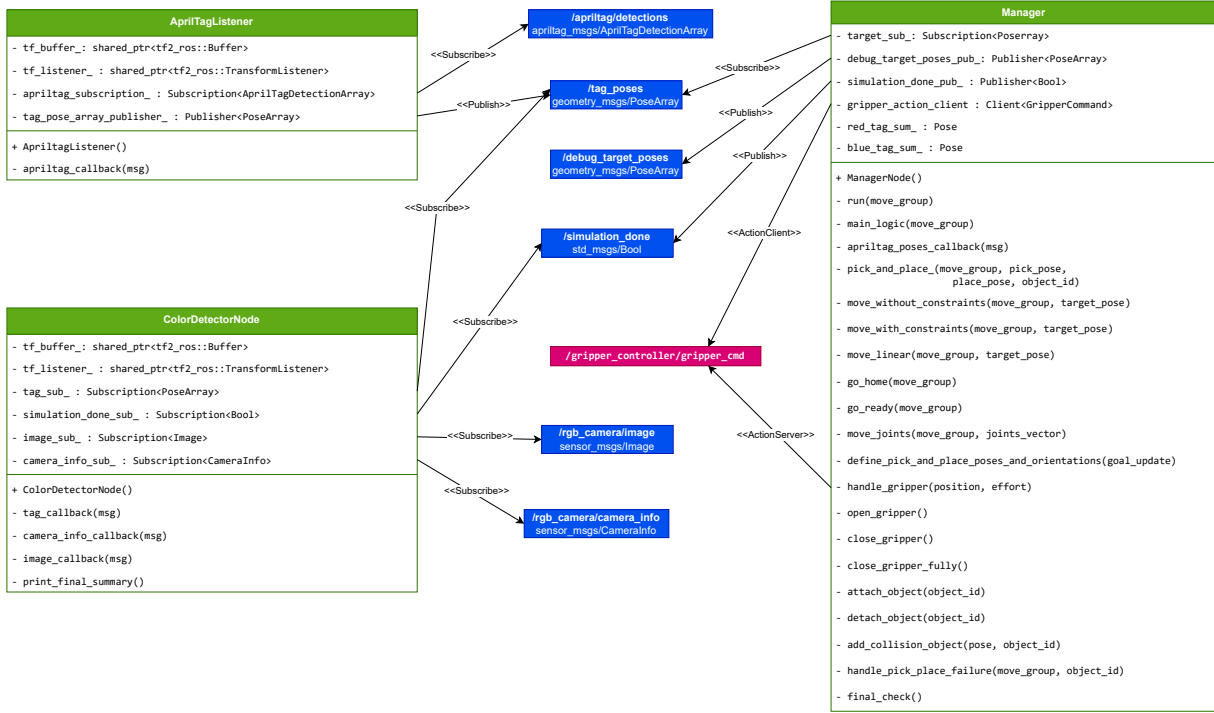


Figure 1: UML diagram

The proposed architecture implements a robust Pick&Place pipeline to swap two cubes using a UR5 manipulator. The system follows a *Sense-Plan-Act* paradigm orchestrated by a central **manager_node**.

The perception layer relies on an RGB camera to detect AprilTags and colors. Raw detections are transformed into the robot's reference frame ("*base_link*") and filtered temporally to ensure stability. The decision layer corresponds to the manager node, which handles the high-level logic, including scanning, picking, placing, and verification. Finally, the actuation layer is handled on MoveIt nodes, already provided thanks to ROS2 modularity.

2 High-Level Algorithm

This sequence is executed autonomously by the **manager_node** and represents the complete logic required to accomplish the cube swap task.

Detect AprilTags → Filter and average poses → Move to ready pose → Pick red cube → Place red cube → Move to ready pose → Intermediate verification → Pick blue cube → Place blue cube → Final verification → Return to home pose

3 Nodes

3.1 Manager Node (`manager_node`)

This node acts as the central orchestrator. It contains the main pipeline for the requested task and all the functions to properly use the MoveIt motion planning system. During the coding and experimentation, several issues were encountered related to gripper positions and movements. For these reasons we used different techniques to solve them.

- **Robust Perception:** First of all, to mitigate sensor noise in the tag detections, a simple average filter was implemented. The system accumulates $N = 20$ valid pose samples before computing the mean pose. This approach effectively resolved initial inaccuracies in pick and place definitions caused by random errors. Additionally, a systematic error observed in the x-coordinate estimate was identified and manually compensated for in the code.
- **Relevant Poses and Orientation:** To determine the optimal Pick and Approach poses, extensive testing was conducted within the RViz environment and through iterative code validation. A fixed downward-facing gripper orientation was selected, such that the end-effector's axes align with the target object. Furthermore, key gripper dimensions were defined to fine-tune the approach and grasp offset.

It is worth noting that a single orientation was employed for picking and placing both cubes. This decision was based on the AprilTags' orientation, which maintained a consistent alignment. However, the implemented logic remains adaptable: if the objects require specific orientations, the system can either adopt the tag's orientation directly (if aligned correctly) or adjust by applying rotations around the Z-axis by multiples of $k \cdot \frac{\pi}{2}$ in the specific function.

- **Hybrid Motion Planning:** The most problematic part was the motion handling. It was observed that enforcing constraints during the robot's initial movement frequently caused the planner to fail in finding a valid solution, even with permissive constraints. Consequently, unconstrained planning was used to transition the robot to a 'ready pose', from which it was able to consistently produce collision-free paths with enforced gripper orientation. Furthermore, for the other transitions, the manager node prioritizes a linear movement function, but due to observed inconsistencies with the linear solver, a fallback mechanism using only orientation constraints is triggered if the linear plan fails.
- **Error Handling & Recovery Strategy:** Given the challenges mentioned above, along with occasional simulation instabilities in Gazebo, the swap operation did not always perform as expected. To address these potential failures, which can also occur in real-world scenarios, the system implements a sort of state machine that identifies specific failure states (e.g., `LIFT_FAILURE`).

Based on the error type, an appropriate recovery routine is triggered. Strategies range from retrying with different motion planners to resetting to a safe 'ready' pose or executing secure maneuvers. Crucially, if a failure occurs while the robot is grasping an object, a specific routine is executed to prevent damage: the system logically detaches the object, opens the gripper, and performs a vertical escape maneuver to avoid potential collision states.

Finally, a critical exception is handled when failure occurs in the terminal phase, with the cube already near the target table. In this scenario, attempting "blind" recovery motions or returning to the pick pose is difficult and dangerous. Thus, the system prioritizes safety by stopping the autonomous loop and requesting manual operator intervention.

3.2 AprilTag Node (`apriltag_listener_node`)

This node serves as a spatial localization bridge for the `manager_node`. It subscribes to the `/apriltag/detections` topic and leverages the TF2 library to convert the position of each tag from its local coordinate frame into the robot's reference frame. To set up the system for this project, the AprilTag configuration file in the `cfg` folder was modified to match the specific tags used. Moreover, the detection specifically processes only tags 1 and 10, which are respectively the red and blue cubes.

3.3 Color Detector Node (`color_detector_node`)

In the context of color detection, the solution provided makes the robot unnecessary for achieving the goal, leveraging the *AprilTag* poses, which serve as spatial anchors to define *Regions of Interest* (ROIs) centered on the cubes, allowing for targeted analysis without the need for physical exploration with the robot.

Furthermore, image processing for the red cube required the implementation of two different masks. This is because, in the HSV color space, red wraps around the color wheel, occupying both the lower (approx. 0–10) and upper (approx. 170–180) ranges.

4 Challenges and Issues

- For the pre-pick and post-pick height pose, a small approach offset of 3 cm was selected. While a higher trajectory was initially preferred, to ensure collision-free motion relative to the other cube or possibly other objects on the table, this strategy resulted in frequent `goal_invalid` planning errors. These failures occurred even though the requested poses appeared to be valid for the robot's reachable workspace. This could suggest some kinematic constraints or joint limits at higher altitudes.

Given the simplicity of the environment, complex high-level strategies were considered unnecessary for this task. However, particular attention was paid to the definition of the placing poses. Specifically, the target position for the red cube was carefully selected to ensure it would not collide with the following motion path required for the blue cube.

- To ensure process robustness and account for occasional instability during the swap operation, an intermediate and a final check on the AprilTag poses were performed. The intermediate check detects whether the blue cube fell or shifted during the handling of the red cube, enabling the update of the target pose if necessary. The final check detects whether the swap was actually successful, as the cubes would sometimes fall after placement, resulting in a false positive success being reported by the robot.
- Despite the challenges described above, the implemented system is able to successfully complete the cube swap task in the majority of execution runs. Thanks to the adopted perception filtering, motion planning strategies, and recovery routines, the robot reliably performs the full pick-and-place pipeline under nominal conditions. Occasional failures are mainly related to simulation instability or planner limitations and do not compromise the overall correctness of the solution.

5 Work Distribution

The work was evenly divided among the group members. All team members contributed to the development of the `manager_node` and to the overall system integration. Mattia Cozza worked on the `color_detector_node`, its integration with the perception pipeline and filtering logic, as well as video preparation. Lorenzo Tomai focused on the `apriltag_listener_node`, pose transformation, and error handling, and provided support in documentation. Nicola Berti was responsible for system tuning, integration testing, and validation in simulation, and also contributed to the documentation.

Git Repository

https://github.com/mattreturn1/assignment_2.git

Video

Video of the main pipeline and the extra part:

https://drive.google.com/drive/folders/1KDNUZFiBInTwQr171Pyh6cg1gB06emUH?usp=drive_link

Acknowledgments

This report includes content assisted by the AI system **ChatGPT**. AI was used for support in drafting the LaTeX structure and template. All technical descriptions and project content were authored and verified by the team.

References

- [1] Open Robotics, *ROS 2 Documentation*, 2025. <https://docs.ros.org/en/jazzy/>.
- [2] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. MIT Press, 2011.
- [3] PickNik Robotics, *MoveIt Documentation*, 2025. <https://moveit.picknik.ai/main/index.html>.