Pick and Place project report

1. Introduction

This project focuses on designing, implementing, and evaluating a complete autonomous pick-and-place pipeline for rigid objects using a UR5 robotic arm simulated in Gazebo. The goal is to produce a fully functional system that detects objects in the environment, plans collision-free and kinematically feasible motions, grasps the object using an actuated gripper, and relocates it to a designated target area. All components were implemented within ROS, and the resulting system performs the full manipulation loop without human intervention.

The system architecture consists of a perception subsystem responsible for detecting and localizing objects in the environment, a high-level task planner that sequences the manipulation operations, a motion planner that computes feasible trajectories based on inverse kinematics (IK) and a time-scaling algorithm, and a low-level control interface that executes joint trajectories and gripper commands on the UR5 robot. The solution includes deliberate design choices such as redundancy resolution for the manipulator's 6-DOF kinematics, singularity avoidance, object-dependent grasping strategies, and a custom trajectory parameterisation technique that ensures smooth motion suitable for simulated execution.

This report presents the full perception pipeline, the motion and task planning approaches, the integration strategy between components, design challenges encountered and solved, and a discussion of the system's performance. All explanations are grounded in the provided source code of the motion planner (motion_planning_node.cpp), the task planner (task_planning_node.cpp), and the UR5 robot controller.

2. Visual Perception

The perception subsystem is implemented as a ROS node that converts synchronized RGB and depth images into 6-DOF object pose estimates expressed in the robot base_link frame. Object detection is performed using a YOLOv8 neural network applied to the RGB stream, producing 2D bounding boxes for the known block classes. The RGB and registered depth images are synchronized using an approximate time policy, and camera intrinsics are obtained once from the CameraInfo topic.

For each detected bounding box, valid depth pixels are back-projected into 3D points in the camera frame using the calibrated intrinsics. The object position is computed as the median of these 3D points, providing a robust estimate of the object's visual centroid even under partial occlusion or noisy depth measurements. Object orientation is estimated using Principal Component Analysis (PCA): when enough

3D points are available a full 3D orientation is recovered, while in sparse cases a 2D PCA fallback provides a yaw estimate with roll and pitch fixed.

The resulting pose is transformed from the camera frame to base_link using TF and published as a PoseStamped message on a per-class topic. A simple but effective height-based filter is applied in base_link to reject detections inconsistent with the table height, together with an additional cutoff to eliminate false detections on the robot arm. A debug image topic visualizes accepted detections only.

The published poses should represent visual centroids suitable for grasping and motion planning, unfortunately we couldn't find a way to use the published poses correctly since they seem wrong, so at this time the vision node is passive.

3. Robot Motion Planning

Motion planning is performed by the motion_planning_node, which receives high-level grasp or place goals from the task planner and computes time-parameterised joint trajectories compatible with the UR5 controller. The motion planner is built on several core components: an IK solver using the KDL library, a post-processing step to clean and validate solutions, a pose-sequencing mechanism that divides motion into pre-grasp, grasp, lift, transfer, pre-place, place, and retreat stages, and a cubic time-scaling trajectory generator.

3.1 Inverse Kinematics

Inverse kinematics is computed with KDL's ChainIkSolverPos_NR_JL solver, which enforces joint limits and performs a Newton-Raphson iterative method to converge toward a joint-space solution that reproduces the target end-effector pose. Since the UR5 is a 6-DOF manipulator and the task requires only 5 DOFs to be met (position and yaw orientation), IK redundancy appears naturally. The approach taken in this project is to constrain the tool's roll and pitch angles to fixed values and allow yaw to vary according to perception input. This reduction simplifies the IK problem while still enabling successful grasps on objects located anywhere within the workspace.

To avoid singularities, the IK solver is augmented with a best-solution selection strategy. Multiple candidate orientations are tested—not only the yaw coming from perception but also its antipodal orientation (yaw + π). The system prefers the orientation whose IK solution has joint values that are less extreme and whose elbow configuration minimizes the risk of crossing singular regions. After obtaining potential IK solutions, the motion planner selects the candidate with the largest minimal distance to joint limits.

3.2 Trajectory Generation and Time Parameterisation

Once target joint positions are produced, the trajectory is converted into a time-parametrised form using a cubic polynomial interpolation. For each joint, the

position, velocity, and acceleration are computed at discrete timesteps to ensure smooth motion. The cubic time-scaling function ensures that trajectories start and end at zero velocity, avoiding abrupt acceleration peaks.

The trajectory discretization frequency is intentionally slow (10 Hz by default) because the UR5 controller in simulation performs its own smoothing and low-pass filtering. This prevents simulation instabilities or joint vibrations, which are common when sending trajectories that are too finely discretised or have abrupt curvature.

The planner produces complete sequences of trajectories for each stage of manipulation, chaining them in order, with optional blending between segments when possible.

3.3 Grasp Pose Construction

A considerable portion of the planning logic is dedicated to generating intermediate poses for successful grasp execution. For example, the pre-grasp pose is generated by offsetting the object's pose along the z-axis, ensuring a vertical approach consistent with many industrial grasping strategies, especially when perception uncertainty remains.

Similarly, place operations compute a pre-place pose that hovers above the desired placement location before descending. These hierarchical waypoints provide geometric robustness and reduce the likelihood of collision with the environment. Because the simulation does not include advanced collision checking, the planner uses a conservative approach: grasping always occurs vertically, gripper opening is commanded only when close to the object, and retreat motion is always upward.

4. High-Level Task Planning

Task planning is handled by the task_planning_node, a finite-state controller that sequences the full pick-and-place operations. Its role is to interpret perception data, decide when a new object is available, compute appropriate grasping parameters, and send goals to the motion planner. It also monitors acknowledgments from both the motion planner and the gripper service to ensure actions have completed successfully before progressing.

The task planner operates in a cyclical loop:

1. **Detection and Selection** – When the perception system publishes object detections, the task planner selects one based on ordering and validity checks, including object height consistency and workspace limits.
2. **Grasp Planning** – The grasp pose is computed by extracting the object's centroid and height from the bounding box. The grasp yaw is passed to the motion planner, which resolves the optimal yaw candidate using IK.

3. **Grasp Execution** – The planner sends a pre-grasp goal followed by a grasp descent command. It then directs the gripper to close and waits for confirmation.
4. **Transfer and Placement** – A fixed or dynamically computed placement location is chosen, and the planner instructs the motion planner to generate the corresponding pre-place and place trajectories.
5. **Retreat and Reset** – The robot retreats to a safe position before searching for the next object.

The finite-state structure ensures robustness: the system never transitions to a new stage unless the current stage acknowledges completion. It also makes the manipulation repeatable, allowing the robot to pick multiple objects one after another.

The task planner also anticipates the possibility that some objects may not be reachable or may fail IK. In such cases, it skips the current object and attempts the next one, ensuring the system does not deadlock.

5. System Integration

Integration across modules is accomplished through ROS topics, services, and TF transformations. The interactions between nodes follow a clean, modular structure that isolates responsibilities:

- The **perception node** publishes object detections.
- The **task planner** subscribes to detections and publishes high-level manipulation goals.
- The **motion planner** receives goals, computes joint trajectories, and forwards them to the low-level UR5 command node.
- The **controller** executes joint trajectories and handles gripper commands.
- TF maintains the spatial relationship between frames such as /base_link, /world, and camera frames.

Coordination between components required addressing real-world issues such as communication delays, the need for acknowledgment messages between the planner and controller, and the prevention of race conditions where a gripper command could be sent before the robot reached the correct pose. These synchronization details were essential to making the system robust.

The result is a seamless workflow in which perception provides data, planning interprets and transforms it, and control executes the final behaviour. Each module operates independently but contributes to a unified robotic system.

6. Results and Discussion

The final system successfully performs autonomous pick-and-place operations on randomised rigid blocks in simulation. Testing with random block placements

confirmed that the combination of pre-grasp waypoints, conservative descending motions, and robust yaw selection produces reliable grasps even when perception is imprecise. The UR5 executes the planned trajectories smoothly, with no instability or oscillation, thanks to the use of cubic interpolation and appropriate discretisation of waypoints.

One of the most significant challenges encountered during development was kinematic feasibility near the workspace boundaries. Blocks placed near the edge of the table or near singularity-prone regions required careful orientation selection and occasionally the alternative yaw candidate to produce a valid IK configuration. The scoring function used to evaluate IK solutions proved essential in resolving these cases and contributed to highly stable grasping performance.

Another challenge involved coordinating the gripper with the motion execution. Because grasp success depends on the timing of the gripper closure relative to the descent, the task planner had to enforce strict acknowledgment-based synchronization. Incorporating explicit confirmations from both the motion planner and the gripper ensured that the grasp happened at the right moment in the sequence.

Overall, the system performs consistently and demonstrates all the required behaviours: detecting rigid blocks, selecting appropriate poses, planning collision-free motions, grasping with the end-effector, transporting the block, and placing it in a target region. The performance is reliable and repeatable across varied scenarios.

7. Conclusion

This project implements a complete autonomous pick-and-place pipeline using a UR5 robotic arm in Gazebo. The system integrates perception, task planning, motion planning, and robot control into a cohesive structure capable of handling multiple rigid objects placed randomly in the workspace. By simplifying orientation requirements, using staged grasp sequences, selecting the best IK solution, and generating smooth time-parametrised trajectories, the system achieves robust manipulation behaviour suitable for a simulated industrial environment.

The work demonstrates a realistic and modular approach to robotic manipulation: perception determines where objects are, the task planner decides what must be done, the motion planner determines how the robot should move, and the controller executes the physical behaviour. Each component is robust on its own but especially powerful when combined. The final system meets all project requirements and performs the pick-and-place task reliably and autonomously.