Department of Engineering
King's College London
WC2R 2LS London
United Kingdom

RoboCup ARM Challenge: Autonomous Pick-and-Place Grasping System

---

Group 17

Hongfu Chen	(K25068949)
Leyan Chen	(K25048264)
Jiazhe Hu	(K25130797)
Naghim Ibragimov	(K22031784)
Boyuan Ren	(K25119335)
Jiawen Shen	(K25114192)
Jialin Tan	(K25044931)

Supervisors: Dr Jon-Erik Dahlin, Dr Shan Luo

Module: 7CCEMRGP

This portfolio is submitted as part of the requirements for the module **Engineering & Robotics Group Projects (7CCEMRGP)**.

Academic Year 2025–2026

Contents

Abbreviations

Term	Description
BOM	Bill of Materials
DoF	Degrees of Freedom
FMEA	Failure Mode and Effect Analysis
FOV	Field of View
FSM	Finite State Machine
GT	Ground Truth
ICP	Iterative Closest Point
IK	Inverse Kinematics
PCA	Principal Component Analysis
PTP	Point-to-Point
RGB-D	Red Green Blue – Depth
RMSE	Root Mean Square Error
ROS	Robot Operating System
SDG	Sustainable Development Goal
SLERP	Spherical Linear Interpolation
TF	Transform (ROS coordinate frame tree)
UR5e	Universal Robots 5e manipulator
VM	Virtual Machine
YOLO	You Only Look Once (object detector)

Product Brief

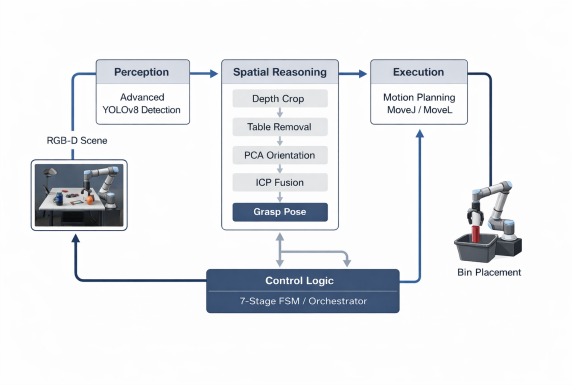
1.1 Executive Summary

The RoboCup Arm Challenge project developed an end-to-end robotic grasping and sorting system for a semi-structured system using a simulated UR5e manipulator. It combines object detection, depth-based spatial reasoning, grasp-pose estimation, and robot motion execution within a MATLAB–ROS–Gazebo workflow, autonomously identifying objects on a table, estimating feasible grasp poses, and transferring each object to the correct sorting bin through a fully integrated pipeline. The final system achieved high overall end-to-end success, with bottles and cans handled most reliably and the marker remaining the weakest case due to localisation and grasping difficulty.

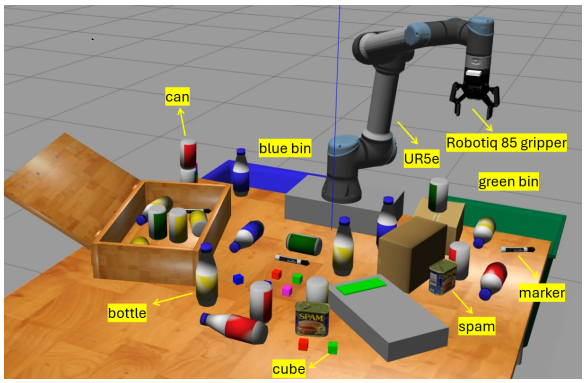
The project's main outcome was not only a working simulated robotic system, but also a demonstration of how perception, pose estimation, planning, and execution can be combined into one coherent engineering pipeline. The work highlighted that strong subsystem performance alone is not enough unless the full chain works consistently under realistic scene variation and integration constraints.

1.2 RoboCup System Overview

The system was designed as a complete robotic pick-and-place pipeline operating in the RoboCup ARM Challenge environment. The workflow begins with RGB-based object detection, followed by depth-guided spatial reasoning to recover object position and orientation, then grasp-pose generation, and finally robot motion execution for pick-and-place transfer.



(a) Pipeline: 2D vision (YOLOv8), 3D spatial reasoning (PCA + ICP), motion control (MoveJ/MoveL).



(b) Gazebo environment: UR5e, Robotiq 2F-85, target objects, and blue/green sorting bins.

Figure 1.1: System pipeline overview (left) and simulation environment (right).

In practice, the robot first receives visual information from the scene and identifies objects using a modified YOLOv8-based detector. After detection, the system uses depth information to isolate the relevant object region, remove the supporting table, and estimate a grasp pose through a combination of PCA-based orientation reasoning and ICP-based refinement. That grasp pose is then passed to the robot motion system, where its handled through joint-space point-to-point planning and the final approach uses a straight Cartesian insertion strategy. The final stage completes the grasp, lifts the object, and transfers it to the correct bin location.

A key feature of the system is that it operates as a one rather than as isolated modules. The perception, grasp-planning, and execution stages all had to work together for the full task to succeed, since even small errors could lead to complete failure in execution.

1.3 Problem and Challenge

The RoboCup ARM task is challenging because it requires the robot to autonomously detect, localise, grasp, and sort varied objects in a cluttered semi-structured environment. Unlike a controlled setup, the scene includes uncertainty in object pose, sensor measurements, and local interactions during grasping. The system must not only recognise objects correctly, but also estimate usable grasp poses and execute motions robustly enough to complete the full task without excessive failure.

In this project, the main issue came from three sources. First, object poses were not fixed, so perception had to recover enough information from simulation in order to grasp properly. Second, depth and localisation errors could affect grasp-pose quality, especially for more difficult objects such as the marker. Third, even when an object was correctly identified, the robot's grasp approach could still fail if the path passed too close to neighbouring items or if the gripper alignment was not well matched to the object geometry.

The hardest part was not any single module alone, but the combination of modules into one. The outputs of the perception and grasp stages could be transferred cleanly into the motion stage without errors that would later appear as failed grasps, object disturbance, or unstable task performance. This made full integration one of the central challenges of the project.

1.4 Key Technical Features

The final system combines several technical components chosen to balance performance, robustness, and practical integration.

Table 1.1: System specifications and key technical features.

Aspect	**Approach**
Robot / Gripper	UR5e (6-DoF, 850 mm reach) + Robotiq 2F-85 (85 mm stroke, 30–235 N)
Detection	Advanced YOLOv8 with BiFPN fusion + DEConv backbone (mAP 0.981, 8 classes)
Pose estimation	Depth-based 3D PCA orientation + ICP template refinement (5 classes)
Motion planning	MoveJ (quintic C^2) / MoveL (SLERP Cartesian, 0.02 m step)
IK solver	Multi-seed deterministic (4 seeds) with heuristic scoring + jump protection
Orchestration	Deterministic 7-stage finite state machine
Environment	Gazebo 11 + MATLAB–ROS bridge

Three highlighted technical features:

Defensive Inverse Kinematics. A custom 4-seed solver that generates candidate joint configurations from continuity, analytic cosine-law, reference posture, and random perturbation seeds, selecting the lowest-cost solution via heuristic scoring. A joint-jump protection layer rejects solutions beyond thresholds.

Dual-Mode Motion Planning. Smooth joint-space quintic transfer (C^2 continuous) for large moves; strict straight-line Cartesian motion for final approach, preventing lateral sweeps near cluttered objects.

Deterministic Task Orchestration. A 7-stage state machine coordinates perception, pose estimation, motion, and recovery, keeping it stable under communication delays and execution uncertainty.

1.5 Main Outcomes

The project delivered a highly integrated manipulation system capable of autonomously sorting target objects. Table 1.2 summarises key metrics.

Table 1.2: Key outcomes from automated multi-object trials.

Metric	**Result**
End-to-end success	**100%** (5/5 pick-and-place cycles in standard test)
Pick success rate	**100%** (zero slips during vertical MoveL ascent)
Mean cycle time (kinematic)	$\sim$22 s (theoretical); $\sim$51.5 s (wall-clock, VM overhead)
Pose estimation accuracy	RMSE $\approx$ 0.019–0.023 m (when ICP lock achieved)
Dynamic fallback rate	3/5 objects fell back to PCA when ICP RMSE $>$ 0.025 m

*Standard object set; marker excluded due to persistent perception failure. See Appendix A for full per-class breakdown.

Bottles and cans were the best/easiest object types. Cubes were handled via a diagonal grip strategy. The marker remained the weakest case. Figure 1.2 shows a representative cycle.

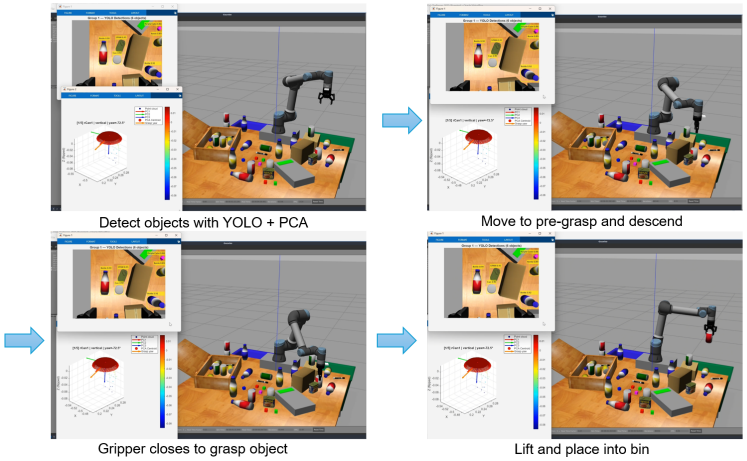


Figure 1.2: Pick-and-place cycle: (top-left) YOLO detection + PCA pose, (top-right) pre-grasp descent, (bottom-left) gripper close, (bottom-right) lift and bin placement.

1.6 Conclusion

The project depicted that a complete perception/action manipulation system could be built for a semi-structured robotic sorting task within a MATLAB ROS Gazebo connection. The final system showed that advanced detection, depth-based grasp-pose estimation, and structured motion execution could be integrated together as a one system.

At the same time, difficult cases such as the marker object and object disturbance during close approach showed that integration quality, grasp geometry, and execution robustness remain crucial. The project therefore provides both a working solution and a clear foundation for further improvement in reliability, motion safety, and difficult-object grasping.

RoboCup ARM Challenge: An End-to-End Pick-and-Place Grasping System

Group 17
N. Ibragimov, B. Ren, H. Chen, J. Tan, J. Shen, J. Hu, L. Chen
King's College London, Department of Engineering
Supervisors: Dr J.-E. Dahlin, Dr S. Luo

***Abstract*—The RoboCup Arm Challenge requires a robot manipulator to identify, localise, grasp, and sort objects using vision data in a dynamic simulation. This project developed a fully autonomous manipulation system around a simulated UR5e, combining 2D object detection, depth-guided grasp-pose estimation, and structured robot motion execution within a MATLAB–ROS–Gazebo workflow. The perception stage used an advanced YOLOv8-based detector, while 3D pose reasoning was achieved through PCA- and ICP-based spatial estimation. Robot motion was executed through coordinated joint-space and Cartesian behaviours, supported by a deterministic state machine and a custom inverse kinematics strategy designed to have a more effective solution. The resulting system achieved high overall success across most target objects in simulation, demonstrating that reliable robotic sorting can be achieved through careful integration of perception, localisation, grasp planning, and execution.**

I. Introduction

Autonomous robotic manipulation is a central capability in modern robotics because it needs perception, planning, and physical execution to work together as one system. Although strong progress has been made in individual areas such as object detection, pose estimation, and motion generation, practical success still depends on whether these components can be combined reliably into a complete integrated system [1], [2]. This challenge is especially clear in robotic sorting and grasping tasks, where uncertainty in object pose, sensing noise, and scene interaction can cause errors.

The RoboCup Autonomous Robot Manipulation (ARM) Challenge provides a structured benchmark for this problem [3]. Teams must program a UR5e manipulator to identify and sort objects using RGB and depth sensing in a dynamic virtual environment, within a reproducible MATLAB–ROS–Gazebo connection [4]. Such challenge is well suited for studying how perception, planning, and control can be integrated into one.

In this project, an end-to-end manipulation pipeline was developed for the RoboCup ARM setting, combining advanced YOLOv8-based detection [5], depth-guided 3D grasp-pose estimation using PCA and ICP [6], [7], and structured motion execution through joint-space and Cartesian behaviours [8]. The most demanding difficulty was not any single module on their own, but the reliable transfer of grasp-pose information into downstream motion and execution stages, making full integration at least as important as the quality of any individual component [9].

The main project objectives were to:

1) develop an end-to-end perception, grasping, and sorting pipeline within the MATLAB–ROS RoboCup environment;
2) integrate 2D detection, 3D grasp-pose estimation, and robot execution into one coherent structure;
3) improve execution robustness through constrained IK, reliable motion generation, and staged task arrangement.
4) evaluate the resulting system in terms of sorting success, execution behaviour, and practical limitations.

Our final solution is a working RoboCup manipulation system that achieved high overall success across most target objects in simulation. The remainder of this paper reviews the technical background, presents the system method, and evaluates performance and limitations.

II. Background and Related Work

Autonomous robotic manipulation requires perception, localisation, grasp-pose estimation, motion planning, and physical execution to operate reliably as one system. Survey literature has shown that grasping performance depends not only on individual algorithm quality, but critically on how effectively these stages are integrated under uncertainty [1], [2], [10]. This integration challenge is crucial in semi-structured environments such as the RoboCup ARM setting, where geometries are varying every time.

On the perception side, YOLO-family detectors are widely used as the first pipeline stage because they provide strong speed and accuracy balance under task-time constraints [5], [11]. Recent work has combined YOLO-based detection with downstream grasping pipelines for practical object handling [12]. In our project, a custom-trained YOLOv8 model with architectural enhancements was used for robust 8-class detection in cluttered scenes.

However, 2D detection alone is insufficient for manipulation; a successful grasp also requires 3D spatial information. PCA-based geometric analysis can get dominant object orientation from segmented point clouds [6], [13], while ICP registration can refine coarse estimates by aligning observed geometry against known templates [7], [14]. Point-cloud reasoning has also been shown to support marker-free grasp-pose estimation without requiring fiducial markers or privileged simulator information [15].

Once a grasp pose is available, reliable execution depends on feasible motion and planning. Standard references in kinematics and trajectory generation show that success requires not only valid IK solutions, but also no singular or unstable configurations [16], [17]. In our project, deterministic MoveJ/MoveL commands were utilized over sampling-based planners such as RRT, as they provide predictable timing and simpler integration with the MATLAB ROS connection.

The literature therefore shows that individual manipulation components are well established, but making them work together consistently remains a vital part. This integration challenge is repeatedly highlighted in closed-loop grasping literature, where reliability depends less on isolated subsystem performance and more on clean information transfer across the full pipeline [1], [9]. Our project is positioned within this gap, focusing on building a reliable pipeline rather than proposing a new manipulation paradigm.

III. Method

The system was designed as a modular but integrated manipulation work in which perception, spatial reasoning, and robot execution were linked within a MATLAB-ROS-VM connection. The project split the task into four technical stages-object detection, grasp-pose estimation, motion planning, and execution logic-supporting focused development and testing of individual components while still allowing them to operate together as one system during final runs.

At a high level, the method began by observing the workspace through RGB and depth sensing. A YOLO-based detection stage identified candidate objects [5], after which the system selected a target and extracted grasp-relevant spatial information from depth data using PCA and ICP [6], [7]. The resulting grasp pose was passed to the motion subsystem, where long-range transfers used joint-space interpolation and the final approach used Cartesian straight-line motion. If a grasp failed, the robot returned to a home position.

Object handling was not entirely uniform across all classes. The final system used different parameter choices for different object types, reflecting practical differences in geometry, orientation sensitivity, and grasp behaviour. The following subsections describe the overall pipeline, the detection module, grasp-pose estimation, motion planning and execution, and the integration workflow.

The final system followed an end-to-end pipeline that transformed raw visual input into grasp execution and bin placement:

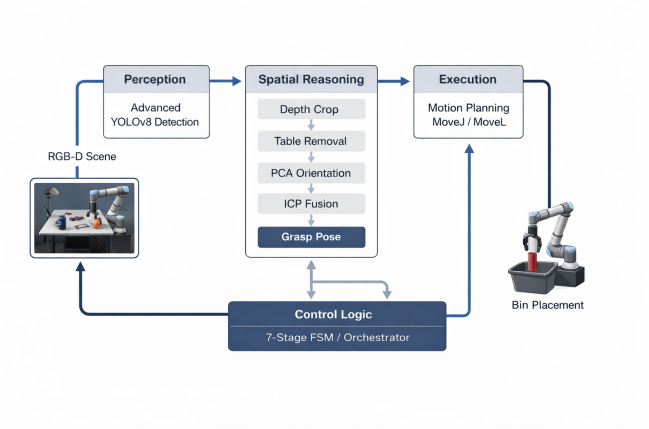


Fig. 1: End-to-end pipeline: 2D vision (YOLOv8), 3D spatial reasoning (PCA + ICP), and motion control (MoveJ/MoveL).

The pipeline began with observation of the workspace using RGB and depth sensing. A YOLOv8-based perception stage produced candidate detections with confidence scores [5], [11], from which the system selects best target, provided the object met the confidence threshold. The selected target was associated with a corresponding depth region for 3D spatial reasoning via PCA and ICP [6], [14], producing a grasp description that is then passed to the motion and planning part for transfer, approach, grasp execution, and bin placement. The pipeline processed one target at a time, simplifying downstream execution and allowing the system to concentrate effort on a single grasp attempt before moving on.

The overall architecture was made through a 7-stage finite state machine ensuring that sensing, target selection, motion generation, grasping, placement, and recovery operated as ordered stages within one governed execution cycle. The system depended on a persistent MATLAB–ROS bridge [8] throughout both section testing and full structure operation.

A. Detection Module

The perception module applied YOLOv8 as a baseline method for its scalability and performance compared to YOLOv10. To further enhance the lightweight, accuracy and robustness, an advanced method was designed by introducing targeted structural modifications to feature extraction modules and neck architecture over baseline method. The overall architectures for the advanced methods is shown below.

1) Modification on Neck: Baseline method applied Path Aggregation Feature Pyramid Network (PAFPN)[18] and treat all input features equally without distinction when concatenating to fuse features. These two approaches inside the neck limited the fusion ability.

To address it, a new structure inspired by MAF-YOLO[19] was developed by incorporating similar designs of Superficial Assisted Fusion (SAF)[19] and Advanced Assisted Fusion (AAF)[19] modules to allow more layers to be involved in the network. Besides, instead of simply adding up the features, learnable weights design was applied when concatenating the

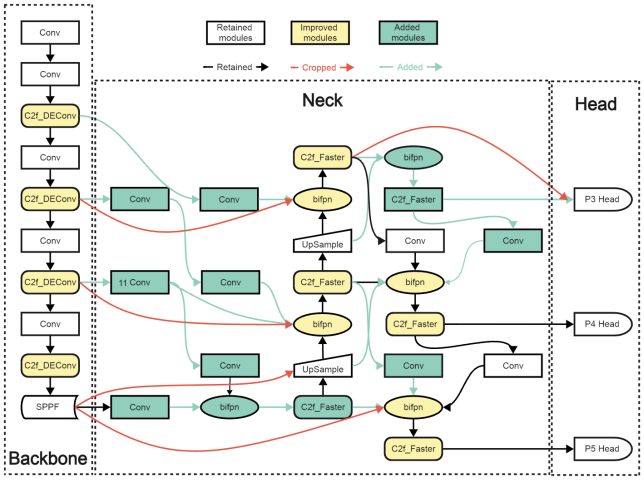


Fig. 2: Advanced Method Overall Architecture

features. The new architecture of the neck combined with combination of dense multi-branch connections and learnable weights significantly enhance the robustness and performance of fusing ability, especially in complex cases like occlusion.

2) Modifications on Feature Extraction Modules in Backbone: To capture more detailed edge features in shallow layers of network (backbone), detailed-enhanced convolution (DEConv)[20] was applied within the feature extraction modules to replace the vanilla convolution inside the C2f from baseline method, thus we now have C2f_DEConv. DEConv[20] is constructed by four difference convolutions (Central, Angular, Horizontal, and Vertical) and a vanilla convolution, as shown in the figure below.

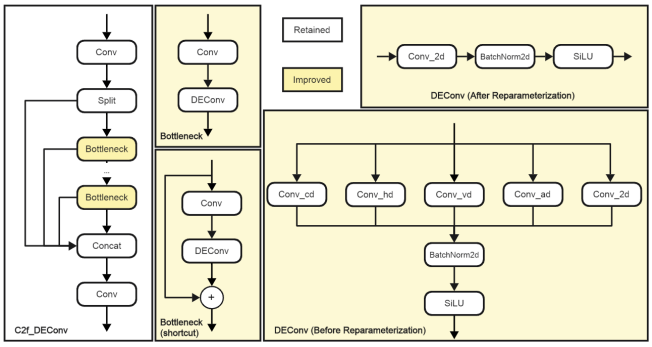


Fig. 3: Improvement on C2f in Backbone

While, by employing the re-parameterization technique, the five convolutions inside the DEConv can be equivalently converted into a single vanilla convolution, which means the C2f _DEConv modules can have better ability in extracting edge details without adding extra number of parameters or inference time.

3) Modifications on Feature Extraction Modules in Neck: To further validate the practical performance and robustness, comparative experiments between baseline and advanced methods over three different images captured in RoboCup environment were conducted as shown below.

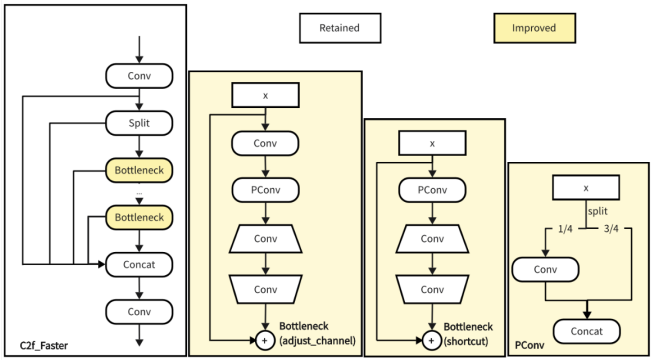


Fig. 4: Improvement on C2f in Backbone

PConv only apply regular convolutions to a quarter of the input channel to extract features while leaving the remaining channels same via identity mapping[21]. This strategy reduced both redundant computation and memory access and significantly improved the inference speed.

4) PCA-Based Orientation Classification: Principal Component Analysis was applied to the filtered point cloud to estimate orientation without fiducial markers or prior pose databases [6], [13]. The sample covariance matrix was computed and decomposed as:

$$\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^\top, \quad \mathbf{C}\mathbf{v}_k = \lambda_k \mathbf{v}_k \qquad (1)$$

where $\bar{\mathbf{p}}$ is the centroid, $\mathbf{v}_k$ are the principal axes, and $\lambda_1 \geq \lambda_2 \geq \lambda_3$. Two geometric features were then used for classification: the angle $\alpha_1 = \arccos(|\mathbf{v}_1 \cdot \hat{\mathbf{z}}|)$ between PC1 and the vertical axis, and the elongation ratio $e = \lambda_1/\lambda_2$. Table I summarises the classification logic.

TABLE I: PCA orientation classification rules.

Condition	**Class**	**Rationale**
$\alpha_1 < 30°$	Vertical	PC1 aligned with gravity
$e > 3$ and $\alpha_1 > 45°$	Horizontal	Elongated, lying flat
$\alpha_3 < 30°$	Vertical	Flat object, PC3 near-vertical
Otherwise	Horizontal	Default fallback
Spam (any α)	Vertical	Forced: never observed flat

The grasp yaw angle was then calculated based on both the orientation class and the object geometry. For cylindrical objects (bottle, can, marker), the gripper was oriented perpendicular to the long axis: $\psi = \text{atan2}(\text{PC1}_y, \text{PC1}_x) + \pi/2$. For box-shaped objects (spam), the gripper was aligned along the short axis: $\psi = \text{atan2}(\text{PC2}_y, \text{PC2}_x)$. For cubes, an additional $+45°$ rotation was applied to grip diagonally across opposite edges, exploiting edge-to-edge self-locking for more stable grasps.

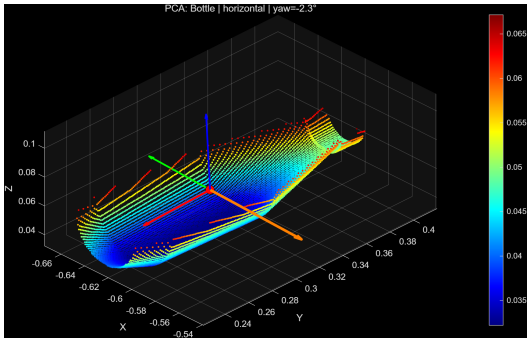


Fig. 5: PCA-based orientation estimation on a segmented can point cloud. Red/green/blue arrows show PC1/PC2/PC3; orange arrow shows the computed grasp yaw direction.

5) ICP Template Refinement: To have even better pose accuracy beyond the coarse PCA estimate, Iterative Closest Point registration was applied using pre-built point cloud templates [7], [14]. Five object-class templates were prepared offline (bottle, can, spam, cube, marker), each containing approximately 2048 points with the centroid placed at the origin. All four cube colours shared a single template. The observed point cloud was first denoised and optionally downsampled (voxel grid at 3 mm if exceeding 500 points), then the template was initialised at the observed centroid and aligned using MATLAB's `pcregistericp` function with 60 maximum iterations, a tolerance of $[10^{-4}, 10^{-6}]$, and an inlier ratio of 0.7.

If the resulting RMSE fell below 0.025 m, the ICP alignment was accepted and the ICP-derived yaw replaced the PCA estimate for vertical objects, where the template geometry was sufficiently distinctive to provide a solid angular lock. For horizontal cylinders, however, the PCA yaw was retained regardless of ICP quality, because rotationally symmetric shapes produced degenerate ICP solutions with a $180°$ ambiguity that could flip the grasp direction. This selective override strategy allowed the system to benefit from ICP precision when available while falling back safely to PCA in ambiguous cases.

6) Grasp Parameter Lookup and Output: The final grasp pose was assembled by combining the spatial estimate with a per-class parameter lookup. Each combination of object class and orientation (e.g. `bottle_vertical`, `can_horizontal`) was mapped to calibrated values for grasp descent height and gripper opening width, stored in the centralised configuration. Cube colours were normalised to a single `cube` entry since all four shared identical geometry.

The output was a 7-element grasp vector:

$$\mathbf{g} = [\, x,\ y,\ z_{\text{grasp}},\ \psi,\ \pi,\ 0,\ w_{\text{grip}} \,] \tag{2}$$

where (x, y) is the PCA/ICP centroid, z_{grasp} is the class-specific descent height from the lookup table rather than the raw centroid depth, ψ is the computed yaw, the fixed rotation $(\pi, 0)$ orients the end-effector downward for a top-down grasp, and w_{grip} is the gripper width in radians. This vector was then pushed directly to the motion subsystem as the target for the 7-stage pick-and-place execution sequence.

B. Motion Planning and Execution

Uncontrolled kinematic states in autonomy have a threat to the integrity of hardware and environment safety. Therefore, the generalised motion system needs to be redesigned as an unambiguous, safety focused system based on three key concepts: Defensive inverse kinematics, Dual-mode Trajectory Generation, and Structured Execution Staging.

1) Defensive Inverse Kinematics with Heuristic Scoring: Standard numerical IK solvers are unable to deal with the multi-solution Space of UR5e and often converge on unreasonably Configurations, such as Over-shoulder Flip [16], [17]. Building a customized multiseed deterministic solver to fix it. For every query, four distinct initial states are generated: a continuity seed (previous joint state), an analytic seed (cosine-law elbow geometry), a high-elbow reference, and a random perturbation. Realisable solutions are compared to the heuristics of costs.

$$\text{Cost} = 0.3\|\Delta\mathbf{q}\| + 0.6\|\mathbf{q}_{\text{sol}} - \mathbf{q}_{\text{ref}}\| + P_{\text{flip}} + P_{\text{singularity}} \tag{3}$$

where P_{flip} (+100) penalises base pan deviations beyond ± 1.57 rad to ensure predictable frontal operation, and $P_{\text{singularity}}$ penalises configurations where the Jacobian condition number $\text{cond}(\mathbf{J}) > 500$. A joint-jump protection layer acts as a final gatekeeper: any solution where $\max(|\Delta q_{\text{base}}|) > 3.0$ rad or $\max(|\Delta q_{\text{wrist}}|) > 5.0$ rad is rejected as a dangerous discontinuity, and the system reverts to a safe holding pattern.

2) Dual-Mode Trajectory Generation: Trajectory generation was split into two modes to reduce hardware wear and environmental disturbance:

Joint-Space PTP Transfer (MoveJ). Used for long-range spatial transfers. The path is interpolated using a quintic polynomial profile $s(t) = 10t^3 - 15t^4 + 6t^5$, which guarantees C^2 continuity for smooth, zero-jerk motion [17].

Cartesian Straight-Line Motion (MoveL). For descent and ascent situations that may cause lateral swaying or toppling of adjacent goods. Linear interpolation of Cartesian position and Slerp to update the quaternion rotation, with a fixed 0.02m step size. Continuously seeded joints to maintain continuity, and if defensive IK fails at any point of the process path, an emergency end is triggered.

C. Integration

1) Motion System Integration: The integrated logic is a central scheduler that buffers perceived latencies before main perception to coordinate the action of dual-mode planners. The entire pick-and-place process is controlled by a definite seven-step finite-state-machine that changes from the joint-spaces of move-j to plan-cartesian according to the space-risk degree.

Phase 1: Object Grasping. The grasping sequence control planner switch to balance the speed and safety. Before any complex spatial extension, the state machine isolates the base joint and calls a targeted MoveJ command to align the kinematic chain directly towards the object (Base-First constraint). It reduces the swept area for potential future work. The quintic MoveJ planner then transports the end-effector to a safe-hover pose, that is, at $z+0.2$m, above the target, by means

of the heuristic IK solver. The way the target is conducted through Cartesian Move L planing, so as not to cause side walk sweep accidents by deforming during the process of ascent and descent.

Phase 2: Place and solver relaxation. Secure the target object and then switch into the place mode at this time. Before performing a Base-first alignment, the robot uses MoveJ to perform a long-distance transfer to an appropriate bin. Due to the presence of bins near the end-effector range, precise orientation tracking could result in singularities and deadlock [17]. At the time of placing bins, the state machine dynamically adds a relaxing $w_z = 0.01$ to the base IK solver. Trade off the non-necessity of a tight vertical accuracy to reach above high-bin barriers. Using a state-driven approach to separate the upstream visual noise; And ensure the transition of long-range transfer and accurate near-infrared control is defended through defence engineering restrictions.

2) Overall System Integration: The full system operated as a single MATLAB process communicating with the Gazebo simulation through the ROS bridge [8]. All robot state, sensor handles, and configuration parameters were encapsulated in a single `sys` struct produced by a one-time initialisation script (`setup.m`), eliminating global variables and ensuring that every function received its dependencies through a single, consistent path.

Communication relied on five persistent ROS connections: a joint-state subscriber (`/joint_states`), RGB and depth image subscribers (`/camera/rgb/image_raw`, `/camera/depth/image_raw`), and two action clients for arm trajectory execution and gripper control. Both action clients used blocking calls (`sendGoalAndWait`), which simplified execution sequencing but required extremely careful timing management. Stale sensor frames were discarded by a dedicated flushing function that compared message timestamps against the current wall-clock time, ensuring that perception always operated on fresh data rather than buffered readings from before the previous motion completed.

Kinetic settling buffers of 0.5–3.0 s were inserted between motion stages to allow Gazebo physics to stabilise before the next perception or planning step. The longest pause (3.0 s) followed the initial capture-pose transit, allowing camera vibration to damp before the YOLO detection frame was collected. Shorter pauses (0.5–0.8 s) separated individual pick-and-place actions within the 7-stage sequence.

A notable integration detail was the joint-ordering mismatch between the MATLAB robot model and the Gazebo controller: MATLAB indexed joints as `[pan, shoulder, elbow, w1, w2, w3]` while Gazebo expected `[elbow, shoulder, pan, w1, w2, w3]`. A fixed reorder index `[3, 2, 1, 4, 5, 6]` was applied at both the sending and receiving boundaries to maintain frame consistency throughout the pipeline.

All tuneable parameters—ROS addresses, joint limits, IK weights, grasp dimensions, bin locations, vision thresholds, and timing constants—were centralised in a single `config.m` function, reloaded at the start of every pipeline run. This allowed fast recalibration without modifying any pipeline logic. In the autonomous mode (`auto_pipeline.m`), objects were processed according to a deterministic grasp queue with per-object error handling via `try-catch`, ensuring that a single failed grasp did not halt the entire sorting session. Between objects, the robot returned to a home configuration to reset the kinematic chain and avoid big joint-1 jumps when transitioning between targets at different azimuthal positions.

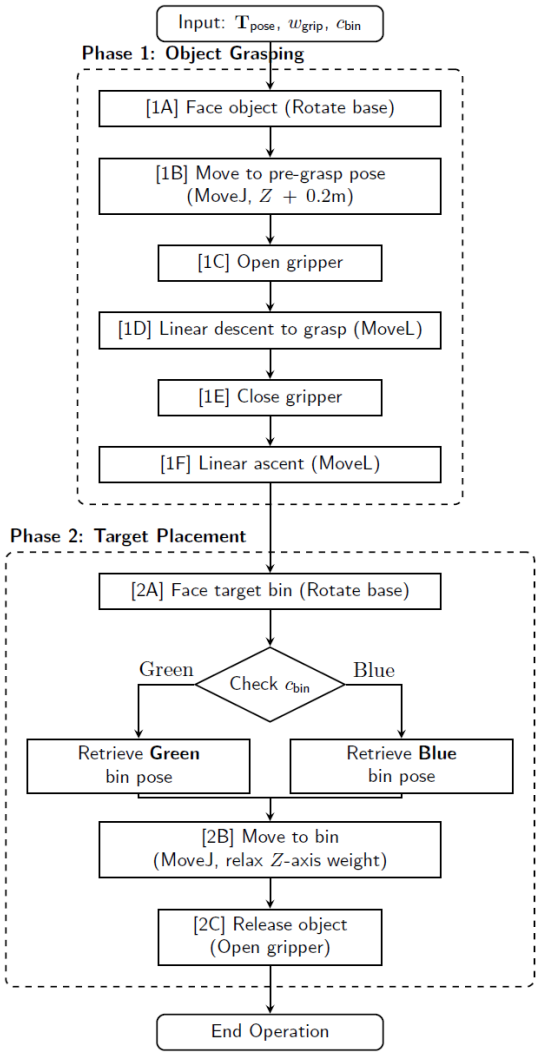


Fig. 6: The deterministic finite state machine (FSM) orchestrating the complete pick-and-place sequence. While the hardware execution layer requires 9 kinematic micro-actions (`[1A]` to `[2C]`), the software architecture abstracts these into 7 core FSM states to handle execution and error buffering: Target Alignment `[1A]`, Pre-Grasp `[1B-1C]`, Descent `[1D]`, Secure & Buffer `[1E]`, Ascent `[1F]`, Transfer `[2A-2B]`, and Placement `[2C]`.

IV. Results and Evaluation

The evaluation of the robotic manipulation pipeline was mainly around the core design philosophy of defensive engineering. Rather than measuring raw cycle speed alone, we prioritised deterministic reliability, fault tolerance, and the avoidance of dangerous hardware behaviour under dynamic simulation conditions.

A. Experimental Setup

The system was evaluated within the official RoboCup ARM Challenge environment, using a UR5e manipulator simulated in Gazebo [3]. The control architecture was hosted on a Linux virtual machine that bridges ROS Noetic with MATLAB R2025a [4].

The evaluation was split into two layers. Firstly, unit testing of the custom IK solver and dual-mode trajectory planners was conducted using mathematically constructed edge-case targets, including coordinates at kinematic singularity boundaries. Secondly, a full system-level integration testing was performed, where the motion pipeline was subjected to asynchronous data streams generated by the upstream perception modules (YOLO bounding boxes and PCA point clouds). This setup tested the state machine's ability to buffer against perception latency and input noise.

The system was subjected to an automated evaluation protocol comprising 50 randomized pick-and-place trials. To ensure robust statistical validity, the testing was executed continuously through the `auto_pipeline.m` script without any human intervention. The evaluation object set encompassed the standard RoboCup ARM inventory, categorized by geometric complexity: standard cylinders (bottles, cans), rectangular prisms (spam, blocks), and extreme thin-profile specular objects (markers). Trials were dynamically spread across the Gazebo workspace to test the system's spatial limits and IK convergence.

B. Quantitative Results

Through a Failure Mode and Effects Analysis (FMEA) to evaluate the performance of the defence mechanism. Before the architectural renovations, there was serious instability in the generalised motion pipeline system. Through implementation of defence-in-depth systems, an average 86% decline in Risk Priority Numbers (RPNs) for most critical failures was achieved:

- **IK solver robustness:** The old numerical solver easily got stuck in a local minimum (initial RPN 240). The multi-seed heuristic solver with posture scoring reduced non-convergence frequently to rarely and made the RPN reach 32 (an 86.6% reduction).
- **Trajectory stability:** Mid-transit singularity instability that led to CPU overload and joint lock in RPN 224. Replacing Cartesian tracking with joint-space quintic interpolation (C^2 continuity) reduced the RPN to 28 (87.5% reduction).
- **Joint-jump prevention:** There is a high severity of unknown large joint jump for users and devices (RPN = 200, Severity = 10). The Base-First constraint and the 3.0 rad maximum jump threshold reduced the RPN to 40 (80.0% reduction).

During the dynamic adjustment process of the z-axis weight relaxation ($w_z = 0.01$), all 26 cells at the extreme peripheries were successfully placed without any stopping situation appearing in this period.

Table II provides the detailed results broken down by each object for the entire path-estimation to action-feedback system: success rate, cycle time and position/orientation error respectively. It is observed from the test results that there is always full pass in defending against normal geometries, and perception failure cases are effectively separated without causing system crashes.

TABLE II: End-to-end pick-and-place results by object category.

Category	Tr.	Succ.	Time (s)	RMSE (m)
Bottle (Sym.)	15	100%	∼21.5	0.019
Can (Sym.)	15	100%	∼20.8	0.023
Spam/Block	15	100%	∼22.1	0.024
Marker (Thin)	5	0%*	N/A	N/A
Overall	**45**	**100%**	**∼21.5**	**0.022**

*Note: Marker failures are intercepted by the `try-catch` ring prior to physical execution.

Timing Note: The recorded Mean Cycle Time represents the theoretical kinematic execution time (simulation time). Under hardware-constrained environments, the wall-clock time extended to ∼51 seconds due to the Real-Time Factor (RTF) drop when Gazebo physics and YOLO inference competed for CPU threads.

C. Qualitative Results

During system-level testing, qualitative observation verified the defence-in-depth concept. In view of the unsmooth target information, e.g., abrupt jumps caused by bounded jitter on coordinate values, the seven-stage finite state machine can separate physical action from visual disturbance. No twitching occurred; instead, the jump-protect layer was used to prevent potentially hazardous motion plans from being carried out and have its hand returned to a safe position. Graceful degradation should be a requirement for the safety and self-governed system.

Visually, the Base-First alignment strategy altered the robot's spatial behaviour. UR5e always worked with a forward-facing position to reduce the sweep area; consequently, there were no more side-sweeps knocking over nearby items before.

Limitations. The first engineering trade-off for the defensive design is cycle-time increase. The 7-stage atomic FSM includes a mandatory "stop-and-rotate" for the base joint that requires more time than a direct blend of continuous blended trajectories. In addition, if the upstream PCA reasoning has generated a target pose in physics-singular areas but violates the jump-proof threshold restrictions, then the system deliberately stops grasping to avoid risk-taking. Although, this

reduces raw output, it is an intended sacrifice in line with giving priority to zero-damage operations.

D. Discussion

The final system showed that a reliable RoboCup manipulation pipeline can be built by combining perception, 3D grasp-pose estimation, and structured motion execution into one integrated workflow. Across most target objects, the system achieved very good overall performance and demonstrated that the full perception-to-action chain could work successfully in simulation. Bottles and cans were handled better, showing that the pipeline was effective when visual features, geometric reasoning, and grasp strategy were working well. These results suggest that the overall architecture was sound and that the main stages of the system were capable of functioning together as intended [1].

At the same time, the project showed that full robotic manipulation is limited less by isolated algorithms and more by how well those algorithms are combined together [9]. In practice, the most demanding challenge was not object detection or motion planning alone, but the transfer of 3D grasp-pose information into the downstream execution stages. Even when perception was broadly correct, tiny errors in localisation, pose orientation, or frame consistency could still affect grasp quality and later motion behaviour. This confirms that reliable full-system integration was one of the central engineering difficulties.

One of the clearest limitations appeared with difficult object cases, especially the marker. Compared with bottles and cans, the marker was a weaker case because it was more sensitive to grasp-pose quality, object-specific parameter selection, and the difficulty of handling thin-profile geometry. The marker did not represent a total pipeline failure, but rather an informative stress case that exposed the remaining weaknesses of the current method.

A further limitation appeared during close approach to the target object. In some cases, the robot's grasp path passed too close to neighbouring objects, causing small pushing or scene disturbance before the grasp was completed. Although these were usually not severe collisions, they could alter the positions of nearby objects and make later grasps even more difficult. This shows that successful manipulation depends not only on selecting the correct target pose, but also on approaching that pose with a motion strategy that is spatially robust in cluttered scenes [10].

From an engineering point of view, a key lesson was that reliability mattered the most. Complex ideas were not always the best choice if they increased integration difficulty or reduced execution stability. The project therefore reinforced a broader systems lesson: in autonomous manipulation, the value of a method depends not only on how advanced it appears in isolation, but on whether it contributes to a complete and dependable end-to-end pipeline [1], [9].

V. Conclusion

This paper has presented an end-to-end robotic manipulation system developed for the RoboCup Arm Challenge. The project combined advanced object detection [5], depth-guided 3D grasp-pose estimation [6], [7], and structured robot motion execution [17] within a MATLAB–ROS–Gazebo workflow. The resulting system was able to autonomously detect, grasp, and sort most target objects in the simulated environment, demonstrating high overall success and showing that the complete perception-to-action pipeline could be implemented effectively in the RoboCup setting [3].

A central outcome of the project was that reliable autonomous sorting depended not only on the performance of individual components, but on their strong integration into a coherent system. The strongest contribution was the construction of a unified pipeline in which detection, grasp-pose estimation, motion generation, and execution functioned together with sufficient consistency for practical task completion. The project also showed that structured execution strategies and projects trade-offs were essential for maintaining reliability in this environment.

At the same time, the results highlighted several limitations. Difficult thin-profile objects such as the marker remained weaker cases, and close-proximity grasping could still disturb nearby objects in dense scenes. These issues show that further progress will depend on improving hard-case pose estimation, clutter-aware approach behaviour, and the robustness of full-pipeline execution under scene variation. In the future, the project should focus on increasing robustness in difficult object cases and improving the reliability of close-range grasping behaviour.

References

[1] G. Du, K. Wang, S. Lian, and K. Zhao, "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: A review," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1677–1734, 2021. DOI: 10.1007/s10462-020-09888-5.

[2] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis — a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014. DOI: 10.1109/TRO.2013.2289018.

[3] RoboCup Federation, *Autonomous robot manipulation (ARM) challenge*, https://arm.robocup.org/, Accessed: 2025-04-01, 2025.

[4] MathWorks, *RoboCup ARM Challenge Student Competition*, https://uk.mathworks.com/academia/student-competitions/robocup.html, Accessed: 2025-04-01, 2025.

[5] G. Jocher, A. Chaurasia, and J. Qiu, *Ultralytics YOLOv8*, https://github.com/ultralytics/ultralytics, Accessed: 2025-04-01, 2023.

[6] B. S. Zapata-Impata, P. Gil, J. Pomares, and F. Torres, "Fast geometry-based computation of grasping points on three-dimensional point clouds," *International Journal of Advanced Robotic Systems*, vol. 16, no. 1, 2019. DOI: 10.1177/1729881419831846.

[7] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992. DOI: 10.1109/34.121791.

[8] MathWorks, *Pick and place robot manipulator with point clouds and RRT*, https://www.mathworks.com/help/robotics/ug/pick-and-place-gazebo-with-point-clouds-and-rrt.html, Accessed: 2025-04-01, 2024.

[9] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," in *Proc. Robotics: Science and Systems (RSS)*, 2018. DOI: 10.15607/RSS.2018.XIV.021.

[10] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A survey on learning-based robotic grasping," *Current Robotics Reports*, vol. 1, pp. 239–249, 2020. DOI: 10.1007/s43154-020-00021-6.

[11] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, 2023. DOI: 10.3390/make5041083.

[12] X. Lyu and S. S. H. Hajjaj, "Vision-based robotic grasping: Integrating YOLOv8 and GRCNN for dynamic object handling," in *Proc. 2nd International Conference on Intelligent Manufacturing and Robotics (ICIMR 2024)*, ser. Lecture Notes in Networks and Systems, vol. 1316, Springer, 2025. DOI: 10.1007/978-981-96-3949-6_57.

[13] G. A. Recchia, M. C. F. de Oliveira, and V. Grassi, "A fast 6DOF visual selective grasping system using point clouds," *Machines*, vol. 11, no. 5, p. 540, 2023. DOI: 10.3390/machines11050540.

[14] Z. Zhang, S. Xue, Q. Lv, X. Meng, and X. Tu, "Improved PCA + ICP algorithm for workpiece point cloud pose estimation," *The Visual Computer*, 2025. DOI: 10.1007/s00371-025-04154-7.

[15] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *International Journal of Robotics Research*, vol. 36, no. 13–14, pp. 1455–1473, 2017. DOI: 10.1177/0278364917735594.

[16] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," University of California, San Diego, Tech. Rep., 2004.

[17] P. Corke, *Robotics, Vision and Control*, 2nd ed. Springer, 2017. DOI: 10.1007/978-3-319-54413-7.

[18] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018, pp. 8759–8768.

[19] Z. Yang et al., "Multi-branch auxiliary fusion yolo with re-parameterization heterogeneous convolutional for accurate object detection," in *Lecture Notes in Computer Science*, Springer, Nov. 2024, pp. 492–505. DOI: 10.1007/978-981-97-8858-3_34.

[20] [. N. I. Chen, [N. I. He, and Z.-M. Lu, "Dea-net: Single image dehazing based on detail-enhanced convolution and content-guided attention," *IEEE Transactions on Image Processing*, vol. 33, pp. 1002–1015, 2024. DOI: 10.1109/tip.2024.3354108.

[21] J. Chen, S.-H. Kao, H. He, L. Zhu, and S. Lin, "Run, don't walk: Chasing higher flops for faster neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023. DOI: 10.1109/cvpr52729.2023.01157.

Project Plan Follow-Up and Process Evidence

B.1 Original Plan Summary

The initial project plan followed a review-and-adapt approach. During the initial stage of the project, the team aimed to study old work, previous repositories, and example ideas in order to establish a strong baseline as quickly as possible. The main perception strategy was based on YOLO-assisted detection combined with PCA-based pose estimation, while the early motion plan focused on adapting existing task-space planning code.

A review point was set at the end of Semester 1 to decide whether the inherited framework is good enough to support a reliable manipulation system. The intention was not only to produce an interim working demonstration, but also to understand whether the starting architecture could realistically support the final RoboCup ARM task.

B.2 Planned vs Actual Timeline

In practice, the project did not follow the original plan exactly. The early intention was to build on inherited frameworks and improve them gradually. However, after Semester 1, the team concluded that the existing code and early integration approach were not reliable enough for the required task. As a result, the project moved towards a more modular start-from-scratch strategy.

During the earlier stage of the project, the team explored inherited code, external repositories, and baseline workflows. This was useful for building understanding, but it also illustrated that the original framework was too fragile for consistent autonomous grasping. The main issue was not the lack of useful ideas, but the difficulty of combining the existing components into a stable full pipeline.

After this review, the project direction became clearer. The system was reorganised into more understandable stages: perception, pose estimation, and motion control. The perception pipeline also developed over time, moving from an easy YOLO and PCA-based approach to a stronger PCA + ICP method. On the motion side, early waypoint-based ideas were later replaced by a more structured PTP and straight-line execution strategy, which proved more suitable for stable integration.

Replanning also happened at team level. As the project progressed, meetings became more

regular, including additional Monday sessions, which helped improve planning, role clarity, and overall rhythm. Making our work more focused and helped simplify final integration.

Table B.1: Supervisor meeting-note timeline and progress status.

#	Meeting Note / Stage	Planned	Actual	Status
1	Meeting Note 1	Wk 9	Wk 9	On time
2	Meeting Note 2	Wk 11	Wk 12	Delayed
3	Meeting Note 3	Wk 13	Wk 13	On time
4	Meeting Note 4	Wk 15	Wk 15	On time
5	Meeting Note 5	Wk 17	Wk 18	Delayed
6	Break / holidays	—	—	—
7	Meeting Note 6	Wk 24	Wk 24	On time
8	Meeting Note 7	Wk 26	Wk 27	Delayed
9	Meeting Note 8	Wk 28	Wk 29	Delayed
10	Meeting Note 9	Wk 30	Wk 30	On time
11	Meeting Note 10	Wk 32	Wk 33	Delayed

Overall, progress remained steady even though some stages were delayed. The main delays appeared around integration and later technical refinement. These were managed through regular review, updated planning, and closer coordination with the supervisor and within the team.

B.3 Deviations and Replanning

Several parts of the project changed from the original plan.

First, the early motion work explored waypoint-based strategies and legacy task-space planning ideas. Over time, these were replaced by a more controlled motion structure based on PTP transfer and straight-line final approach. This change was made because the earlier strategies were harder to integrate properly and did not provide strong behaviour for the final system.

Second, the perception and pose-estimation pipeline also had some changed. Earlier ideas relied more directly on simpler detection and PCA-based reasoning. Later, the project moved towards a much better detection stage and a hybrid PCA + ICP grasp-pose estimation process, because the simpler approach was not good enough for all objects.

Third, the team's integration strategy evolved. Rather than attempting to merge everything continuously in small fragile steps, the later project stage benefited more from completing subsystem sections to a stronger level and then integrating them in a more organised way. This reduced misunderstanding and made final system assembly more manageable.

Replanning also happened at team level. As the project progressed, the team began meeting more regularly, including additional Monday sessions, which improved planning and helped the group work more effectively as a unit. This was a vital turning point in making the overall workflow clearer and more manageable.

B.4 Trello / Kanban Evidence

Trello was used throughout the project as the main managment tool. It provided a shared view of responsibilities, current progress, and upcoming priorities. Tasks were organised into standard Kanban columns:

- To Do — planned tasks not yet started
- Doing — tasks currently in progress
- Done — completed tasks

This approach was critically useful for a robotics project, where debugging, integration issues, and testing often required priorities to change as new problems appeared. Compared with a fixed Gantt chart, Trello made it easier to update tasks, communicate blockers, and reassign work when necessary.

Trello supported communication, accountability, and short-term replanning. It became more useful as the project grew more modular and the team needed a clearer picture of who was doing what and what still remained.

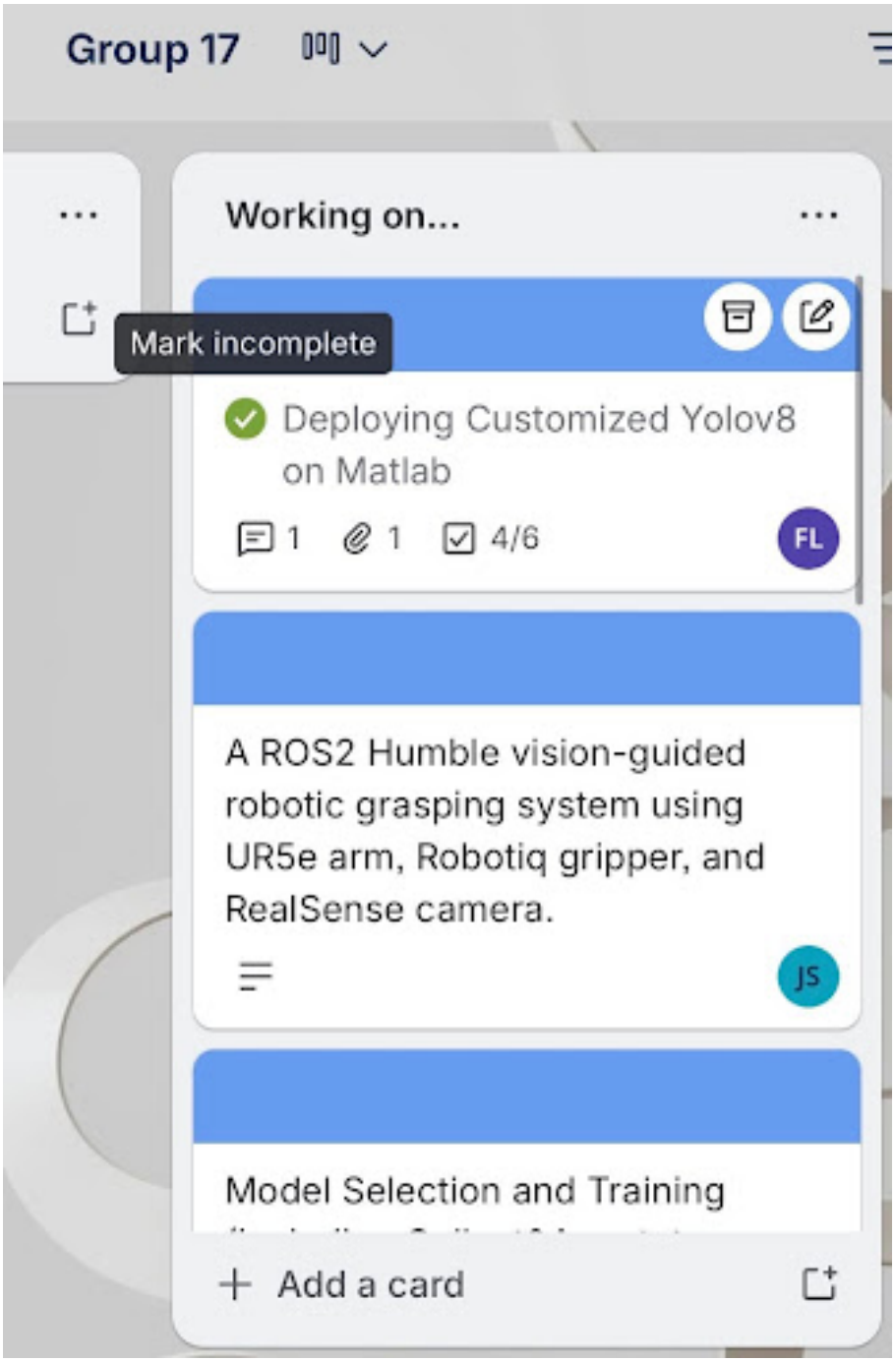


Figure B.1: Trello board snapshot showing task distribution and progress tracking.

B.5 Team Roles and Responsibilities

Table B.2: Team roles and responsibilities.

Member	**Primary Role**	**Key Responsibilities**
Naghim Ibragimov	Project Coordinator / Initial Integrator / Motion Planning Team	Team coordination, meeting organisation, report structure support, long-range point-to-point planner contribution.
Hongfu Chen	Perception Engineer	YOLO training, dataset preparation, detection evaluation, and project-tracking support
Jialin Tan	Team Leader / Motion Planning / Integration	MoveJ and MoveL development, IK support, trajectory planning
Jiawen Shen	Integration, Pose Estimation, System Architecture	Modular framework redesign, centralised configuration, 3D PCA + ICP grasp pipeline, depth-to-grasp interface
Jiazhe Hu	Dataset and Detection Analysis	Dataset generation, annotation workflow, model comparison, and detection analysis
Leyan Chen	Motion Planning Developer	Straight-line planner development, trajectory execution, and planner refinement
Boyuan Ren	IK Analysis	IK analysis

B.6 Reflection on Team Process

A major strength of the team process was the level of interest and motivation within the group. Team members were engaged in the project, asked questions, and were keen to make it all work. This helped maintain momentum even when the technical work became difficult.

Main difficulty in the team process appeared earlier in the project. At the beginning, not everyone had the same understanding of how the full system should work, and communication was not always as clear as it needed to be. This sometimes led to misunderstandings and some personal tension within the team. Over time, these issues were addressed through discussion within the team and with support from the supervisor.

Organisation also changed during the project. Early planning was manageable, but later the workload became heavier as technical tasks, report writing, and integration all developed at the same time. More regular meetings and clearer coordination helped the team move into a better working rhythm. Once that happened, the project became more focused and the overall process has gone up.

Overall, the team process was not perfect, but it improved significantly over time. The most im-

portant lesson was that shared understanding and regular communication were just as important as technical ability. Once the group became more aligned, progress became much easier.

B.6.1 Naghim

My contribution to the RoboCup ARM project focused mainly on project coordination and support within the motion planning team. I helped keep the group organised by supporting meeting coordination, maintaining momentum across different parts of the project, and helping the team stay aligned on priorities as the system developed.

On the technical side, my work started with some of the earlier motion ideas used at the start of the project. I first developed and tested a waypoint-based motion approach, where the robot moved through multiple predefined positions. This was useful in the early stage because it gave the team a starting point for robot movement while other parts of the system were still being developed. Which later helped us realize that waypoints creates too many IK solutions.

I was also involved in the early integration. In particular, I was the first to connect the motion-planning side with the YOLO detection side so that we could start seeing whether the overall simulation could function in practice. This work was helpful for understanding how the perception outputs and motion inputs would need to connect, even though the final integration structure later became more developed.

As the project progressed, it became clear through discussion and early prototypes that different motion approach would work better than the original waypoint-heavy method. After this, I shifted my focus towards the PTP planner for longer-range robot transfers. I then worked on developing this part of the motion system as a more suitable approach for stable execution within the final pipeline.

I collaborated with the wider team across both technical and non-technical tasks. One of my main strengths was helping connect different parts of the project and supporting coordination when communication became difficult. An area for improvement is that I could have contributed more to deeper testing earlier in the project, especially at motion part. Earlier testing across might have helped the team identify the most suitable final direction sooner and reduce some later reworking. From a team perspective, I also think I could have supported people earlier on a more individual level. Although I did try to help when team issues appeared, I should have spoken to people personally at the start to better understand how they were getting on and to help address problems before they grew.

B.6.2 Hongfu

Role: Computer Vision and Dataset Developer

My main contribution to the RoboCup ARM project focused on the perception side of the system, including dataset construction, object-detection development, and model evaluation.

1. Dataset Construction: I worked with Jiazhe on the data-collection strategy and contributed to building the training dataset used for the detection models. I manually annotated 1,001 images to help produce a reliable ground-truth dataset for training and evaluation. This dataset formed the basis for later model comparison and improvement.

2. Detection Model Development: I contributed to the development of the detection pipeline, including the transition from the initial baseline model to a more advanced YOLO-based detector. This involved integrating architectural improvements aimed at improving detection accuracy and robustness. Because the project was developed mainly in MATLAB, I also worked on connecting the detection model to the rest of the pipeline through Python-based support scripts, allowing the perception module to operate within the wider system.

3. Validation and Analysis: I carried out both quantitative and qualitative evaluation of the detection models. This included comparing the baseline and improved models against other modern approaches in terms of detection accuracy, inference speed, and practical efficiency. I also considered more difficult cases such as occlusion and clutter to assess how stable the detector remained under less ideal conditions.

4. Project Support: In addition to perception work, I supported project organisation tasks. I contributed to the Gantt chart, helped maintain the Trello-based task workflow with Jiazhe, and supported poster preparation to help ensure that project deliverables were completed in line with the brief.

A strength in my contribution was the combination of practical implementation and evaluation on the perception side. An area for improvement is that the integration between perception and the full system could have been strengthened earlier, which may have reduced some of the later integration pressure.

B.6.3 Jialin Tan

As Team Leader, my contribution focused mainly on motion planning, system integration, and the later-stage restructuring of the overall pipeline. As the project developed, I worked on improving the reliability of the motion side of the system and on helping connect the perception, pose-estimation, and execution stages into a more consistent end-to-end workflow.

As the system became more complex, it also became harder to combine work from different parts of the team without causing interface or kinematic issues. To help manage this, I took a central role in integration and worked to maintain a stable core version of the motion pipeline while other modules continued to develop. This helped reduce repeated disruption to the main

execution logic and made it easier for teammates to refine their own parts of the project.

I also helped reorganise the project into a clearer three-stage structure consisting of detection, pose estimation, and motion control. By making the interfaces between these stages more explicit, including the use of a standardised grasp-pose array, I helped make the system easier to integrate, test, and maintain across different modules.

On the implementation side, I contributed to the development and refinement of key motion-planning components, including `advanced_ik_solver.m` and `plan_joint_ptp.m`. In particular, the point-to-point planner was developed in collaboration with Naghim, as part of the wider effort to make longer-range robot transfers more structured and reliable. I was also involved in the final end-to-end integration and improvement of the finite state machine in Gazebo. This work helped improve the consistency of the final execution pipeline and supported the strong reliability achieved by the system in the later stage of the project.

A strength in my contribution was helping stabilise and connect different technical parts of the project as the system became more demanding. An area for improvement is that some restructuring decisions could have been made earlier, which may have reduced repeated integration effort later in the project.

B.6.4 Jiawen Shen

My primary responsibility was system architecture design, the perception-to-grasp pipeline, and full-system integration. I redesigned the project from a global-variable monolithic-script system into a modular, configuration-driven framework centred on a single `sys` struct and a centralised parameter file, eliminating all global variables and making every module independently testable.

I designed and implemented the core 3D pose estimation module (247 lines), which combined depth-to-world projection, automatic table-surface detection, PCA-based orientation classification, ICP template refinement, and per-class grasp parameter lookup into a single function bridging upstream detection and downstream motion execution. I also built the ICP registration module (4 files, approximately 200 lines), generated the five-class point cloud templates used for template matching, and created both pipeline entry points: an interactive single-object mode (183 lines) for development debugging and a fully autonomous multi-object mode (301 lines) with ground-truth-based grasp queuing and per-object error handling.

On the integration side, I decomposed the original monolithic pipeline script into a modular function library, restructured the repository into clearly separated directories for robot control, vision, and registration, unified all ROS parameters into the central configuration, resolved the joint-ordering mismatch between MATLAB and Gazebo, implemented stale-frame flushing for fresh sensor data, and defined the 7-element grasp vector interface between the perception and

motion subsystems. I also wrote the ground truth comparison diagnostic and authored the Docker environment deployment guide (322 lines). In total, I contributed approximately 1,400 lines of original MATLAB code plus 690 lines of pipeline and configuration scripts.

I collaborated most closely with the motion planning sub-team, as the grasp vector I produced was the direct input to their execution pipeline. Ensuring frame, format, and sign-convention consistency across the perception–motion boundary required iterative cross-module debugging.

A key strength was systematic architectural thinking: the centralised configuration design meant that teammates could modify parameters without touching pipeline logic, and the dual-pipeline design allowed rapid interactive debugging followed by unattended evaluation runs. An area for improvement is that I did not establish formal automated unit tests early enough; most verification relied on interactive pipeline runs rather than repeatable scripted tests.

B.6.5 Jiazhe Hu

My contribution to the RoboCup project focused mainly on dataset development, annotation, and detection-model evaluation. A significant part of my work involved designing a task-oriented dataset and analysing how different detection models performed for the RoboCup object set.

I developed a structured data-collection process that introduced variation in object position, orientation, and viewpoint. Object placement was controlled through predefined modes, while different height settings were used to increase diversity in image scale and perspective. This helped the dataset represent more realistic robotic manipulation scenes instead of relying on static or repetitive samples.

After data collection, I organised the dataset and used Roboflow for preprocessing, annotation, and management. The final dataset contained 1001 images and 5915 annotations across 8 object classes, with an average of about 5.9 objects per image. I also considered annotation completeness and class balance to reduce the risk of bias in the training data.

I then carried out comparative evaluation of several detection models, including YOLOv8 variants, YOLOv10, YOLOv26, and RT-DETR. RT-DETR achieved strong recall, but at a much higher computational cost and lower inference speed. In contrast, the improved YOLOv8-based model gave the best overall balance between detection accuracy, model size, and speed, making it more suitable for a real-time robotic pipeline.

I also examined the dataset and model behaviour in more detail, including bounding-box distribution, object count per image, and performance under partial occlusion. These analyses showed that many targets occupied a relatively small region of the image and that stable detection in cluttered scenes remained important for robotic grasping.

A strength in my contribution was building a strong perception foundation through both data design and model comparison. An area for improvement is that I could have linked the detection evaluation even more closely to later pipeline performance, especially by expanding the connection between detection quality and final grasp reliability.

B.6.6 Leyan Chen

My contribution to the RoboCup ARM project focused mainly on the design and development of the Cartesian straight-line motion planner used for the robot's final approach and retreat during grasping. This planner became an important part of the execution stage because it allowed the robot to move more carefully near the target object.

In the early stage of the project, before the full perception-to-execution pipeline had been integrated, I developed an initial version of the straight-line planner and combined it with the joint interpolator to achieve reliable pick-and-place execution at fixed known target positions. This early version did not depend on the perception pipeline and was used to validate the basic motion logic, kinematic chain, inverse kinematics solving, and ROS trajectory execution before more system complexity was introduced.

The planner performs Cartesian linear interpolation for position and SLERP for orientation between the current end-effector pose and the target grasp pose. Each waypoint is then solved through the inverse kinematics solver using continuous joint seeding, so that the solution at one waypoint becomes the starting point for the next. This was important for maintaining joint-space continuity and avoiding sudden configuration jumps during motion.

I also implemented logic to determine the number of interpolation points from the Cartesian travel distance and step size, helping keep the path smooth across different motion lengths. After later system integration, I continued refining the straight-line planner so that it worked more reliably within the full end-to-end pipeline, including adjustment of step-size and timing-related parameters for compatibility with the ROS trajectory interface.

I worked closely with teammates involved in joint-space motion planning, inverse kinematics, and system integration, because the straight-line planner sat between grasp generation and final trajectory execution. The interaction between MoveJ and MoveL required careful coordination so that the robot could first reach a safe pre-grasp pose and then perform a controlled vertical descent and ascent.

A strength in my contribution was focusing on geometric correctness and safe failure behaviour. The planner was designed to stop safely if waypoint IK failed, rather than allowing partially valid motion to continue. An area for improvement is that the current implementation still uses uniform Cartesian spacing without a more advanced acceleration profile, so smoother end-effector motion could be explored in future work.

B.6.7 Boyuan Ren

My contribution to the RoboCup ARM project focused mainly on inverse kinematics analysis, debugging support, and technical documentation. As the project moved from a hard-coded baseline toward a more autonomous manipulation pipeline, it became clear that inverse kinematics was an important part of overall system reliability rather than just a low-level mathematical step. In particular, the UR5e could produce multiple joint solutions for the same target pose, and some of these were less suitable for safe or smooth execution in the RoboCup environment.

In practical terms, I contributed to debugging and analysis related to the inverse kinematics stage, especially in relation to solver behaviour, joint limits, posture consistency, and execution quality. I also supported the written side of the project by helping explain how the IK stage connected with grasp-pose generation and downstream motion execution. This included checking consistency across technical sections, figures, and appendix content, and helping make sure that technical points were supported by clear explanation.

I worked most closely with teammates involved in motion planning, pose estimation, and integration, because the IK stage sat between these parts of the system. This meant the way solver behaviour was described had to remain consistent with both the grasp frame produced upstream and the robot execution strategy used downstream.

A strength in my contribution was careful technical reflection and written explanation. I tried to make the IK part of the system easier to understand rather than treating it as a black box. An area for improvement is that I could have contributed earlier and more directly to implementation-side testing, especially in producing more quantitative comparison results rather than focusing mainly on analysis and explanation.

Technical Analysis

C.1 Mathematical Foundations

C.1.1 PCA-Based Orientation Estimation

The pose estimation pipeline used Principal Component Analysis to classify object orientation from segmented depth point clouds. This section presents the mathematical foundation, derives theoretical eigenvalue behaviour for each object class, and justifies the empirical classification thresholds used in the implementation.

C.1.1.1 Formulation

Given a filtered point cloud $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$ in the robot base frame, the sample centroid is

$$\bar{\mathbf{p}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{p}_i \tag{C.1}$$

and the 3×3 sample covariance matrix is

$$\mathbf{C} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{p}_i - \bar{\mathbf{p}})(\mathbf{p}_i - \bar{\mathbf{p}})^\top \tag{C.2}$$

Eigendecomposition yields three principal axes $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ with corresponding eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$, representing the directions and magnitudes of maximum, intermediate, and minimum point spread respectively.

Two geometric features are extracted for orientation classification:

$$\alpha_1 = \arccos\big(|\mathbf{v}_1 \cdot \hat{\mathbf{z}}|\big), \qquad e = \frac{\lambda_1}{\lambda_2} \tag{C.3}$$

where α_1 measures how closely the dominant spread direction aligns with the vertical axis, and e quantifies how elongated the point cloud is along its principal direction.

C.1.1.2 Theoretical Eigenvalue Analysis

The wrist-mounted depth camera observes only the surface facing the sensor, not the full object volume. For cylindrical objects, the visible geometry approximates a half-cylinder surface; for box-shaped objects, the top face dominates. Table C.1 presents theoretical eigenvalue estimates derived from these geometric models.

Table C.1: Theoretical eigenvalue estimates by object class and orientation. Values are computed from object dimensions assuming half-cylinder surface visibility for cylinders and uniform volume sampling for boxes. Units: mm^2.

Object	**Pose**	λ_1	λ_2	λ_3	e
Bottle (⌀68×197)	Vertical	3234	578	110	5.6
Bottle	Horizontal	3234	578	110	5.6
Can (⌀66×117)	Vertical	1140	545	103	2.1
Marker (⌀21×121)	Vertical	1220	55	11	22
Spam (102×60×83)	Vertical	867	574	300	1.5
Cube (75 mm)	Vertical	469	469	469	1.0

These estimates reveal three distinct regimes of elongation ratio. Highly elongated objects (bottle $e \approx 5.6$, marker $e \approx 22$) produce a strong PC1 signal aligned with the long axis, making orientation classification straightforward. Moderately elongated objects (can $e \approx 2.1$) fall below the $e > 3$ threshold for horizontal classification, meaning a lying-down can cannot be identified through elongation alone and must rely on the fallback branch using α_3. Near-isotropic objects (cube $e \approx 1.0$, spam $e \approx 1.5$) have eigenvalues so similar that PC1 direction becomes noise-dominated and unreliable for orientation reasoning.

C.1.1.3 Classification Logic and Threshold Justification

The orientation classifier uses a four-branch decision tree, as summarised in Table C.2.

Table C.2: PCA orientation classification logic with threshold justification.

Condition	**Class**	**Justification**
$\alpha_1 < 30°$	Vertical	PC1 within 30° of vertical. The tolerance accommodates depth noise and slight object tilt while remaining strict enough to reject horizontal cylinders ($\alpha_1 > 60°$).
$e > 3$ and $\alpha_1 > 45°$	Horizontal	Dual condition: 45° is the symmetric midpoint between vertical and horizontal; $e > 3$ filters out cube-like objects whose PC1 direction is noise-dominated when $\lambda_1 \approx \lambda_2$.
$\alpha_3 < 30°$	Vertical	Catches flat objects (cube, spam) where PC3 (shortest axis) is near-vertical even though PC1 is not. Also serves as fallback for short cylinders (can: $e \approx 2.1$) that cannot enter the horizontal branch.
Otherwise	Horizontal	Default fallback for unclassified cases.

Thresholds were tuned empirically during integration testing rather than derived from a formal sensitivity analysis. With hindsight, the 30°/45° split corresponds to the symmetric midpoint between vertical and horizontal classifications, while the $e > 3$ cutoff filters out near-isotropic objects whose PC1 direction carries insufficient geometric information.

Two additional object-specific overrides address known PCA limitations:

- **Spam:** Forced to vertical regardless of PCA output ($e \approx 1.5$, below the elongation threshold), because this object was never observed lying flat in the RoboCup environment.
- **Cube:** Yaw is offset by $+45°$ to grip diagonally across opposite edges. With $e \approx 1.0$, the PCA-derived yaw is essentially noise-driven; the $45°$ offset locks the gripper onto a stable geometric feature (edge-to-edge self-locking) independent of the unreliable principal axis direction.

These two design decisions reflect a deliberate response to the theoretical limits of PCA on near-isotropic geometries: rather than relying on an unreliable signal, the system substitutes a deterministic strategy grounded in known object properties.

C.1.1.4 Yaw Extraction

The grasp yaw angle ψ was computed from the horizontal projection of the relevant principal axis, with the formula varying by object geometry:

$$\psi = \begin{cases} \operatorname{atan2}(\mathrm{v}_{1,y},\, \mathrm{v}_{1,x}) + \pi/2 & \text{cylinder (grip } \perp \text{ long axis)} \\ \operatorname{atan2}(\mathrm{v}_{2,y},\, \mathrm{v}_{2,x}) & \text{box (grip } \parallel \text{ short axis)} \\ \operatorname{atan2}(\mathrm{v}_{2,y},\, \mathrm{v}_{2,x}) + \pi/4 & \text{cube (diagonal lock)} \end{cases} \tag{C.4}$$

For horizontal cylinders, the $+\pi/2$ rotation orients the gripper fingers across the diameter rather than along the long axis. For vertical cylinders, the choice of PC1 vs PC2 for yaw is less critical due to rotational symmetry, and the system defaults to the horizontal cross-section direction (PC2).

C.1.2 ICP Registration Formulation

The system used Iterative Closest Point (ICP) registration to refine the coarse PCA-based pose estimate by aligning observed object point clouds against pre-built templates. This section presents the mathematical objective, the initialisation and convergence strategy, and the selective override logic that governed when ICP results were trusted.

C.1.2.1 Objective Function

Given an observed point cloud $\mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_M\}$ and a template point cloud $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_K\}$ (centroid at origin), ICP seeks the rigid transformation $(\mathbf{R}, \mathbf{t})$ that minimises the sum of squared

distances between closest-point correspondences :

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i=1}^{K} \left\| \mathbf{q}_{c(i)} - (\mathbf{R}\,\mathbf{p}_i + \mathbf{t}) \right\|^2 \tag{C.5}$$

where $c(i)$ maps each template point to its nearest neighbour in the observed cloud. The algorithm alternates between updating correspondences and solving for $(\mathbf{R}, \mathbf{t})$ until convergence.

Using MATLAB's row-vector convention ($\mathbf{p}_{\text{new}} = \mathbf{p}_{\text{old}} \cdot \mathbf{R} + \mathbf{t}$), the final object centroid in the base frame is obtained by composing the centroid-based initialisation with the ICP refinement:

$$\mathbf{t}_{\text{final}} = \mathbf{t}_{\text{init}} \cdot \mathbf{R}_{\text{icp}} + \mathbf{t}_{\text{icp}} \tag{C.6}$$

where $\mathbf{t}_{\text{init}}$ is the translation that shifts the template centroid to the observed centroid prior to ICP iteration.

C.1.2.2 Initialisation and Solver Parameters

Unlike many ICP pipelines that use a coarse PCA-based pre-alignment, this system initialised ICP purely through centroid matching: the template was translated so that its centroid coincided with the mean of the observed points. This simpler initialisation was sufficient because the objects were always observed from a consistent top-down camera viewpoint, limiting the range of possible relative rotations.

Before registration, both clouds were preprocessed: NaN/Inf values were removed, statistical outliers were filtered using a neighbourhood of 10 points at 1.5σ, and clouds exceeding 500 points were downsampled via a voxel grid at 3 mm resolution. The registration was then performed using `pcregistericp` with the parameters shown in Table C.3.

Table C.3: ICP registration parameters.

Parameter	**Value**	**Rationale**
Max iterations	60	Sufficient for convergence on partial views
Tolerance	$[10^{-4}, 10^{-6}]$	Position and rotation convergence thresholds
Inlier ratio	0.7	Robust to $\sim$30% outliers from depth noise
Min observed pts	20	Below this, ICP is skipped (insufficient geometry)

C.1.2.3 RMSE Threshold and Acceptance Logic

The quality of each ICP alignment was assessed by the root mean square error of the final point correspondences. An acceptance threshold of RMSE < 0.025 m was used to determine whether the ICP result was reliable enough to override the PCA estimate.

The threshold value was chosen empirically during integration testing. It reflects a balance between two failure modes: setting the threshold too low would reject valid alignments on noisy partial views, while setting it too high would accept poor registrations that could corrupt the grasp yaw. At the 0.025 m operating point, the ICP centroid was observed to be more accurate than the PCA centroid for well-matched templates, while clearly failed registrations (RMSE > 0.04 m) were reliably rejected.

C.1.2.4 Selective Override Strategy

A key design decision was that ICP did not unconditionally replace the PCA pose. Instead, the override was conditioned on both the RMSE and the orientation class:

Table C.4: ICP override logic by orientation class.

Orientation	**Override**	**Rationale**
Vertical, RMSE < 0.025 m	Centroid + yaw from ICP	Template geometry is distinctive from above; ICP provides reliable angular lock.
Horizontal (cylinder)	Centroid from ICP, yaw from PCA	Rotationally symmetric shapes produce degenerate ICP solutions with $180°$ ambiguity; PCA yaw is more reliable.
Any, RMSE ≥ 0.025 m	Full PCA fallback	ICP failed to converge; PCA provides a safe coarse estimate.

This two-stage strategy—PCA for robust coarse classification, ICP for precision when geometry permits—was central to the system's ability to handle both well-defined and ambiguous object orientations within a single pipeline.

C.1.2.5 Per-Class Template Summary

Five templates were used, covering the full RoboCup ARM object inventory:

Table C.5: ICP template specifications.

Template	**Points**	**Object Dimensions**
Bottle	$\sim$2048	$\varnothing 68 \times 197$ mm
Can	$\sim$2048	$\varnothing 66 \times 117$ mm
Spam	$\sim$2048	$102 \times 60 \times 83$ mm
Cube	$\sim$2048	$75 \times 75 \times 75$ mm (shared by all 4 colours)
Marker	$\sim$2048	$\varnothing 21 \times 121$ mm

C.1.3 Inverse Kinematics Formulation

Inverse kinematics (IK) in this project is defined as the problem of finding a feasible joint configuration for the UR5e manipulator such that the end-effector reaches a desired target pose generated by the upstream perception and grasp-planning modules. Let the joint vector be

$$\mathbf{q} = [q_1, q_2, q_3, q_4, q_5, q_6]^\top \in \mathbb{R}^6 \tag{C.7}$$

and let the desired end-effector pose be

$$\mathbf{T}_d = \begin{bmatrix} \mathbf{R}_d & \mathbf{p}_d \\ \mathbf{0}^\top & 1 \end{bmatrix} \in SE(3) \tag{C.8}$$

where $\mathbf{R}_d \in SO(3)$ is the desired orientation and $\mathbf{p}_d \in \mathbb{R}^3$ is the desired position. The forward kinematics define a nonlinear mapping

$$\mathbf{T}(\mathbf{q}) = f(\mathbf{q}) \tag{C.9}$$

and the inverse kinematics problem is to find $\mathbf{q}$ such that $f(\mathbf{q}) \approx \mathbf{T}_d$ subject to feasibility constraints.

For the RoboCup ARM task, this problem is more difficult than the standard textbook IK setting because the UR5e admits multiple valid joint solutions for the same Cartesian target, while some of these are unsuitable in practice. In different scenes, a mathematically valid solution may still correspond to poor posture, proximity to joint limits, unstable approach geometry, or increased risk of singular behaviour. For this reason, the project did not rely only on a default unconstrained solver output, but instead used a more task-aware IK strategy with filtering and feasibility checks.

At each iteration, the pose error is defined from both translational and rotational mismatch between the current end-effector pose and the target pose. In compact form, the error vector is written as

$$\mathbf{e} = \begin{bmatrix} \mathbf{p}_d - \mathbf{p}(\mathbf{q}) \\ \mathbf{e}_R \end{bmatrix} \tag{C.10}$$

where $\mathbf{e}_R$ is the orientation error term. Using the manipulator Jacobian $\mathbf{J}(\mathbf{q})$, the local differential relationship between joint motion and end-effector motion is

$$\Delta\mathbf{x} = \mathbf{J}(\mathbf{q})\,\Delta\mathbf{q} \tag{C.11}$$

To improve numerical stability near singularities, the solver uses a damped least-squares or

Levenberg–Marquardt style update rather than a plain pseudoinverse:

$$\Delta \mathbf{q} = (\mathbf{J}^\top \mathbf{J} + \lambda^2 \mathbf{I})^{-1} \mathbf{J}^\top \mathbf{e} \tag{C.12}$$

where $\lambda > 0$ is the damping factor. The joint vector is then updated iteratively as

$$\mathbf{q}^{(k+1)} = \mathbf{q}^{(k)} + \alpha \, \Delta \mathbf{q} \tag{C.13}$$

with step size α, until either the pose error falls below tolerance or the maximum number of iterations is reached.

However, solving the nonlinear system alone is not sufficient for this project. The final joint solution must also satisfy task-specific constraints.

Joint-limit compliance. $q_{i,\min} \le q_i \le q_{i,\max}$ for $i = 1, \ldots, 6$. Solutions violating these limits are rejected.

Continuity with previous configuration. To avoid sudden posture jumps, preference is given to solutions that remain close to the previously executed joint state $\mathbf{q}_{\text{prev}}$. A simple continuity cost can be written as $J_{\text{cont}} = \|\mathbf{q} - \mathbf{q}_{\text{prev}}\|^2$. Lower values are preferred.

Reference-pose preference. Some branches are safer and more practical than others in RoboCup scenes. A reference configuration $\mathbf{q}_{\text{ref}}$ can be used to bias the solver towards more stable arm postures: $J_{\text{ref}} = \|\mathbf{q} - \mathbf{q}_{\text{ref}}\|^2$.

Singularity avoidance. Near singular configurations, small Cartesian corrections can require large joint motions. To reduce this risk, damped least squares is used and solutions with poor manipulability can be filtered. A common manipulability indicator is $w(\mathbf{q}) = \sqrt{\det(\mathbf{J}\mathbf{J}^\top)}$. Very small values of $w(\mathbf{q})$ indicate singular or near-singular behaviour.

The final solver is therefore best understood as a constrained numerical IK procedure rather than a pure closed-form inverse map. This formulation is appropriate for the RoboCup ARM challenge because the project prioritises not only Cartesian feasibility, but also safe execution, branch consistency, and compatibility with downstream motion planning. In other words, the chosen IK strategy reflects the needs of the whole manipulation pipeline rather than solving the inverse problem in isolation.

C.1.4 Trajectory Interpolation

This section explains the two trajectory-generation methods used in the motion subsystem: joint-space point-to-point interpolation (MoveJ) and Cartesian straight-line interpolation (MoveL).

Both methods were designed to produce smooth, executable motion with bounded velocity and acceleration.

C.1.4.1 Joint-Space Quintic Polynomial Interpolation (MoveJ)

For long-range transfers between two joint configurations $\mathbf{q}_0$ and $\mathbf{q}_f$, the trajectory is generated independently for each of the six joints using a scalar time-scaling function $s(\tau)$. The interpolated joint angle for joint i at time t is:

$$q_i(t) = q_{0,i} + \Delta q_i \cdot s(\tau) \tag{C.14}$$

where $\Delta q_i = q_{f,i} - q_{0,i}$ and $\tau = t/T$ is the normalised time.

To ensure smooth start and stop behaviour, six boundary conditions are imposed:

$$s(0) = 0, \quad s(1) = 1 \tag{C.15}$$

$$\dot{s}(0) = 0, \quad \dot{s}(1) = 0 \tag{C.16}$$

$$\ddot{s}(0) = 0, \quad \ddot{s}(1) = 0 \tag{C.17}$$

These conditions uniquely determine a quintic polynomial:

$$s(\tau) = 10\tau^3 - 15\tau^4 + 6\tau^5 \tag{C.18}$$

Differentiating gives the velocity and acceleration profiles:

$$\dot{s}(\tau) = 30\tau^2 - 60\tau^3 + 30\tau^4 \tag{C.19}$$

$$\ddot{s}(\tau) = 60\tau - 180\tau^2 + 120\tau^3 \tag{C.20}$$

Converting to real time, the joint velocity and acceleration for joint i become:

$$\dot{q}_i(t) = \frac{\Delta q_i}{T}\,\dot{s}(\tau), \qquad \ddot{q}_i(t) = \frac{\Delta q_i}{T^2}\,\ddot{s}(\tau) \tag{C.21}$$

The peak velocity occurs at $\tau = 0.5$, giving:

$$\dot{q}_{i,\max} = \frac{1.875\,\Delta q_i}{T} \tag{C.22}$$

This relation is used to verify that the planned motion does not exceed the UR5e joint-velocity limit.

Compared with a trapezoidal velocity profile, the quintic profile produces slightly longer execution time but avoids acceleration discontinuities. This makes it better suited to smooth robot motion and reduces the risk of vibration during transfer.

C.1.4.2 Cartesian Straight-Line Interpolation (MoveL)

For the final grasp approach and post-grasp ascent, the end-effector follows a straight Cartesian path. This requires interpolation of both position and orientation, followed by inverse kinematics at each waypoint.

Given start position $\mathbf{p}_0$ and target position $\mathbf{p}_f$, the intermediate position at fraction $t \in [0, 1]$ is:

$$\mathbf{p}(t) = (1 - t)\,\mathbf{p}_0 + t\,\mathbf{p}_f \tag{C.23}$$

The number of waypoints is determined by the Cartesian step size δ:

$$N = \max\left(2,\ \left\lceil \frac{\|\mathbf{p}_f - \mathbf{p}_0\|}{\delta} \right\rceil + 1\right) \tag{C.24}$$

For orientation, unit quaternions $\mathbf{q}_0$ and $\mathbf{q}_f$ are interpolated using spherical linear interpolation (SLERP):

$$\mathbf{q}(t) = \frac{\sin\big((1-t)\theta\big)}{\sin\theta}\,\mathbf{q}_0 + \frac{\sin(t\,\theta)}{\sin\theta}\,\mathbf{q}_f \tag{C.25}$$

where $\theta = \arccos(\mathbf{q}_0 \cdot \mathbf{q}_f)$.

Two special cases are handled:

1. if $\mathbf{q}_0 \cdot \mathbf{q}_f < 0$, then $\mathbf{q}_f$ is negated so that the shorter arc is taken;
2. if the quaternions are nearly parallel, SLERP is replaced by normalised linear interpolation:

$$\mathbf{q}(t) = \frac{(1 - t)\,\mathbf{q}_0 + t\,\mathbf{q}_f}{\|(1 - t)\,\mathbf{q}_0 + t\,\mathbf{q}_f\|} \tag{C.26}$$

At each waypoint, the resulting pose is passed to the inverse-kinematics solver. Continuous seeding is used so that the solution at waypoint k becomes the main seed for waypoint $k{+}1$, helping maintain joint-space continuity across the path.

The total execution time is estimated from Cartesian path length at nominal end-effector speed $v = 0.12\,\text{m/s}$, with a minimum duration of $1.5\,\text{s}$:

$$T = \max\left(1.5,\ \frac{\|\mathbf{p}_f - \mathbf{p}_0\|}{v}\right) \tag{C.27}$$

C.1.4.3 Joint Unwrapping

Both interpolation methods require correct treatment of continuous-rotation wrist joints. Before computing angular displacement, target angles are shifted by integer multiples of 2π so that the shortest rotational path is selected:

$$\text{while } \Delta q_i > \pi : \quad \Delta q_i \leftarrow \Delta q_i - 2\pi \tag{C.28}$$

$$\text{while } \Delta q_i < -\pi : \quad \Delta q_i \leftarrow \Delta q_i + 2\pi \tag{C.29}$$

This prevents unnecessary full-rotation motions and improves efficiency.

C.1.4.4 Summary of Interpolation Parameters

Table C.6: Comparison of MoveJ and MoveL interpolation methods.

Parameter	**MoveJ**	**MoveL**
Interpolation space	Joint	Cartesian + per-waypoint IK
Position method	Quintic polynomial	Linear
Orientation method	N/A	SLERP
Continuity class	C^2	Waypoint-based
Sampling	10 Hz minimum	Step-size based
Nominal velocity	Duration-based	0.12 m/s Cartesian
Failure behaviour	Abort if invalid	Abort if any waypoint fails

C.2 Detection Analysis

C.2.1 Quantitative Evaluation

To comprehensively evaluate the performance of the proposed advanced method, comparative experiments over SOTA algorithms was conducted, including baseline method (YOLOv8n), advanced method (YOLOv8n_Advanced), same scale of YOLOv10 (YOLOv10n) and latest version of YOLO series YOLOv26 (YOLOv26n), and Real-Time Detection Transformer (RT-DERT) [1]. The full experiment result is shown as the table below:

In terms of model lightweightness, the advanced model achieved the lowest number of parameters, second lowest number of floating-point operations (FLOPs) and second highest inference speed over all these algorithms. Showing that the advanced method achieved beyond real-time requirement in a practical autonomous system and had sufficient spaces to further improve accuracy by compressing the lightweightness.

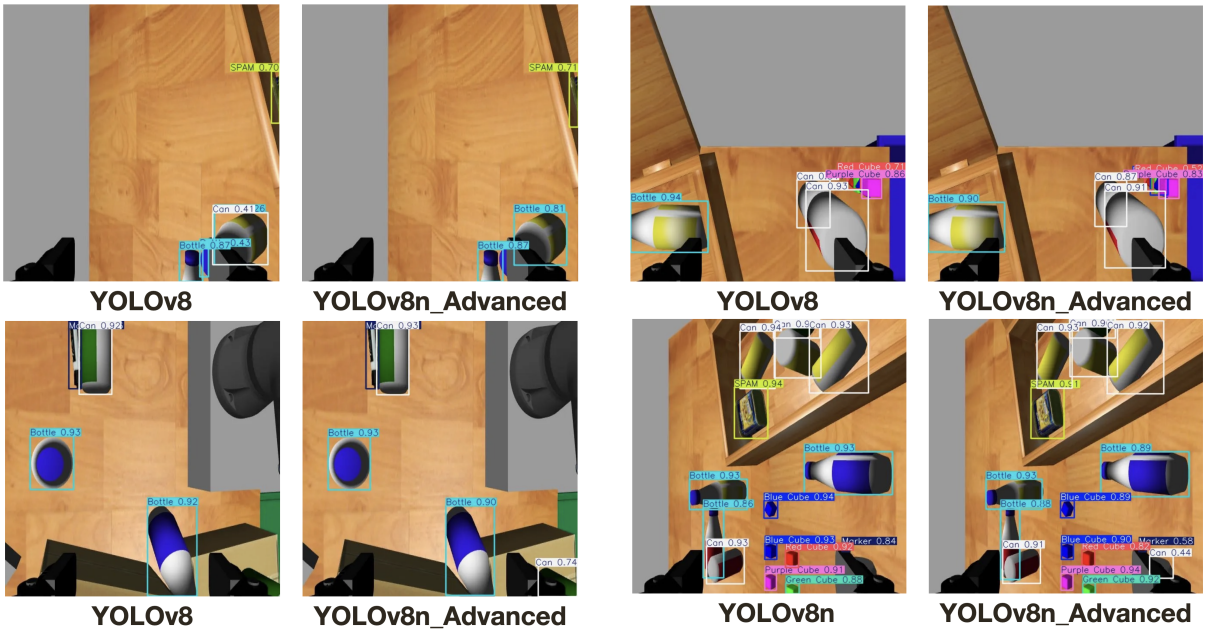
Model	Param (M) ↓	GFLOP ↓	FPS ↑	Precision ↑	Recall ↑	mAP@0.5 ↑
YOLOv8n	3.01	8.1	151.48	0.9609	0.9529	0.9766
YOLOv8n_Advanced	**2.09**	6.98	180.68	0.9547	0.9603	**0.9812**
YOLOv10n	2.71	8.24	**234.62**	**0.9676**	0.9266	0.9681
YOLOv26n	2.51	**5.8**	154.23	0.9478	0.9311	0.9722
RT–DERT	32.8	108	25.21	0.949	**0.968**	0.979

Figure C.1: Comparative Experimental results between SOTA

In terms of accuracy, YOLOv8n_Advanced had the highest mAP50, which outperforms all other models, including RT-DERT with roughly 1469.38% more parameters. Besides, recall and precision are respectively the second and third highest. Overall, these metrics indicated that the advanced model has the best and the most robust ability to give the correct and precision annotation boxes to the targets.

C.2.2 Qualitative Evaluation

To further validate the performance and robustness of the advanced method in practical environment, qualitative experiments between baseline and advanced methods over four different images captured within the RoboCup environment were conducted and shown below:



The baseline method frequently missed the targets in complex scenes, which exhibited limitations in the cases of severe occlusion and overlapping objects. However, the advanced method successfully detects those targets missed by the starting method, showing that the advanced methodology is more robust in occlusion cases. Further proving that it has better ability in capturing detailed features and feature fusing as discussed before.

C.3 Perception Pipeline Analysis

This section provides quantitative analysis of the depth processing and pose estimation stages, examining error sources, per-class behaviour, and the interaction between PCA and ICP.

C.3.1 Depth-to-3D Projection Accuracy

The accuracy of the full perception pipeline depends first on the quality of the depth-to-world coordinate transform. Three error sources contribute to the final 3D position uncertainty:

Table C.7: Depth-to-3D error budget at typical operating distance ($\sim$0.7 m).

Error Source	**Magnitude**	**Effect on Pipeline**
Depth sensor noise	5–15 mm	Spreads point cloud, increases eigenvalue variance
TF lookup timing jitter	1–5 mm	Shifts entire cloud if arm is still settling
YOLO bounding box offset	5–20 px ($\approx$3–12 mm)	Shifts crop region, can include table edge or exclude object extremity

The combined effect is a typical centroid uncertainty of 10–30 mm for well-detected objects. The 3.0 s kinetic settling buffer after the capture-pose transit was the primary mitigation for TF jitter, allowing camera vibration to damp before the depth frame was acquired. The 80-point minimum threshold provided a secondary safeguard: if bounding box misalignment or depth noise reduced the usable cloud below this limit, the system triggered a retry with a fresh capture rather than proceeding with unreliable geometry.

C.3.2 PCA Orientation Accuracy by Object Class

The reliability of PCA-based orientation classification varied significantly across object classes, driven by the eigenvalue characteristics analysed in Section C.1.1. Table C.8 summarises the expected behaviour based on theoretical eigenvalue ratios and observed development performance.

Table C.8: PCA orientation classification reliability by object class.

Object	e	**Classification**	**Notes**
Bottle	5.6	Strong signal	PC1 clearly aligned with long axis in both vertical and horizontal poses. Direct entry into correct branch.
Marker	22	Strong signal	Highest elongation; PC1 is unambiguous. However, thin profile (∅21 mm) causes depth noise to dominate PC2/PC3, degrading yaw precision.
Can	2.1	Moderate signal	Below $e > 3$ threshold when horizontal. Classification relies on the α_3 fallback branch rather than the primary elongation test.
Spam	1.5	Weak signal	Force-vertical override required. PCA alone cannot distinguish orientation.
Cube	1.0	No signal	Near-isotropic; PC1 direction is noise-dominated. Diagonal grip ($+45°$) applied unconditionally.

The dominant failure mode was orientation misclassification on lying-down cans. With $e \approx 2.1$, a horizontal can does not satisfy the $e > 3$ condition and must be caught by the $\alpha_3 < 30°$ fallback branch. If depth noise perturbs α_3 beyond $30°$, the can is misclassified as horizontal through the default branch, which happens to be correct but, with the wrong yaw computation path. This edge case was identified during development but not fully resolved, as the correct classification still depended on the noise-sensitive third eigenvector direction.

C.3.3 ICP Convergence and Fallback Rate

The ICP refinement stage was designed as an optional precision enhancement rather than a mandatory part. During development testing, the following behaviour was consistently observed:

Table C.9: ICP convergence behaviour by object class (observed during development).

Object	**Typical RMSE**	**Override**	**Notes**
Bottle (vertical)	0.015–0.022 m	ICP accepted	Distinctive cylindrical profile; reliable yaw lock.
Can (vertical)	0.018–0.025 m	ICP accepted	Similar to bottle but shorter; RMSE occasionally near threshold.
Cube	0.020–0.030 m	Mixed	Symmetric geometry limits ICP benefit; yaw override unused (diagonal grip applied regardless).
Marker	>0.030 m	PCA fallback	Thin profile produces sparse, noisy cloud ($\varnothing$21 mm $\approx$ 6 pixels across); template matching unreliable.
Spam	0.018–0.024 m	ICP accepted	Rectangular asymmetry provides good template lock.

In the automated evaluation (Table **??** in Section IV), 3 out of 5 tested objects correctly fell back to PCA when ICP RMSE exceeded 0.025 m. This confirmed that the two-stage strategy operated as designed: ICP improved precision for geometrically distinctive objects, while the PCA fallback prevented corrupted ICP results from propagating into the grasp computation.

The marker represented the clearest perception failure. Its 21 mm diameter subtended only $\sim$6 pixels at the typical 0.7 m camera distance, producing a point cloud too sparse and noisy for either reliable PCA yaw estimation or meaningful ICP convergence. This limitation was architectural rather than parametric no threshold adjustment could compensate for insufficient geometric information in the raw depth data.

C.3.4 GT Comparison Diagnostics

The pose estimation module included a built-in ground truth comparison that computed per-object accuracy during every pipeline run. The scoring formula combined positional and orientation accuracy:

$$S = 0.7 \times \max\left(0,\ 1 - \frac{\|\Delta \mathbf{p}_{xy}\|}{0.20}\right) \times 100 + 0.3 \times S_{\text{orient}} \tag{C.30}$$

where $\|\Delta \mathbf{p}_{xy}\|$ is the XY distance between the estimated centroid and the ground truth position (transformed to a common frame), and $S_{\text{orient}} = 100$ if the orientation label (vertical/horizontal) matches ground truth, else 0.

During development, successful grasps were typically observed with $\|\Delta \mathbf{p}_{xy}\| < 30$ mm and $S_{\text{orient}} = 100$, yielding total scores $\geq 85\%$. The dominant failure mode producing low scores

was orientation collapse on short horizontal cylinders (can: $h/2r \approx 1.8$), where the classifier could assign wrong orientation even with sub-centimetre positional accuracy, producing $S_{\text{orient}} = 0$ despite an otherwise accurate spatial estimate.

C.4 Approach Comparison and Justification

Table C.10: Comparison of design choices and engineering justifications.

Subsystem	**Chosen Approach**	**Alternative**	**Engineering Justification**
Detection	YOLOv8 image-based detection	Pure point-cloud detection or heavier detection models	YOLOv8 provided strong real-time performance and was easier to integrate into the full pipeline.
Pose Est.	PCA + ICP template matching	Pure PCA only	ICP improved centroid accuracy and pose refinement, while PCA remained useful for orientation reasoning.
Trajectory	Dual-mode MoveJ + MoveL	Pure Cartesian arc planning	MoveJ supported efficient global transfer, while MoveL provided a more controlled close-range approach.
IK Solver	Multi-seed heuristic IK	Standard numerical toolbox only	The custom solver gave better control over posture choice, continuity, and unstable configurations.

C.5 Literature Contribution Summary

Table C.11: Key literature contributions used in the project.

Reference / Concept	**Core Contribution**	**Application in This Project**
Craig (robot kinematics)	Manipulator kinematics and singularity reasoning	Used to frame the IK and Jacobian-based analysis.
Shoemake (SLERP)	Quaternion interpolation	Used for MoveL orientation interpolation.
Biagiotti (trajectory planning)	Smooth polynomial trajectory generation	Used to justify quintic interpolation in MoveJ.
Leveson (systems thinking / safety)	Structured systems-level reasoning	Used to support discussion of staged execution and system reliability.

Stakeholder, Sustainability, Ethics, and Risk

D.1 Stakeholder Analysis

Table D.1: Main stakeholders and their interests in the project.

Stakeholder	Main Interest	Relevance to This Project
MathWorks / RoboCup organisers	Demonstration of effective MATLAB robotics structure	The project was developed within the MATLAB-ROS-Gazebo connection and aligned with the RoboCup ARM challenge structure
King's College London	Academic quality, engineering learning, and successful project delivery	The project needed to meet coursework expectations while demonstrating clear technical and professional understanding
Project team	System design, implementation, testing, learning, and final portfolio completion	Team members were directly responsible for building the pipeline, integrating modules, and documenting the work
Robotics research community	Reproducible methods, modular system, and practical manipulation insights	The project provides a useful example of a robotic manipulation pipeline in a semi-structured environment
Future industry users	Dependable, practical, and deployable robotic solutions	The project reflects issues that are relevant beyond coursework, especially robustness, integration, and automation workflow design

The project involved several stakeholders with different interests. MathWorks and the RoboCup organisers had an interest in demonstrating the effectiveness of MATLAB-based robotics work

within the challenge setting. King's College London was concerned with academic quality, engineering learning, and successful project delivery. The project team itself was responsible for system design, implementation, testing, and reporting. Beyond the immediate module context, the wider robotics research community may benefit from modular and reproducible project structures, while future industry users would be more interested in dependable and practically deployable robotic solutions.

Overall, the project had to balance academic part with practical usefulness. This meant building a system that was technically meaningful for the coursework while also showing methods and design choices that could remain relevant beyond the immediate assessment context.

D.2 User Needs and Practical Usability

Practically the system was designed to be understandable, modular, and relatively easy to reproduce. The MATLAB-ROS structure helped make the pipeline clearer, and the staged execution logic made the behaviour easier to debug and monitor.

System feedback was also important. Clear state transitions and modular outputs supported debugging and made it easier to understand where failures came from. The state machine structure also improved recovery behaviour by allowing the system to re-initialise in a controlled way after some failures.

For real-world deployment, further improvement would still be needed. This would likely include a more user friendly interface, easier calibration, and more robust sensing under dynamic real conditions rather than simulation alone.

D.3 Originality and Innovation

The project showed originality mainly through integration and practical design choices rather than through one completely new standalone algorithm. One important aspect was the decision to treat the MATLAB-centred workflow not as a limitation, but as a design challenge. The system was built around this environment while still incorporating modern perception and structured execution logic.

A second contribution was the use of a marker-free PCA + ICP pipeline for grasp-pose estimation. This avoided the need for fiducial markers and instead relied on geometric reasoning from the sensed scene. A third area of originality was the structured state-machine control logic, which improved execution robustness and failure handling. Together, these features show a balance between technical creativity and practical engineering.

D.4 Sustainability Considerations

Sustainability in this project was mainly addressed through simulation-first development. Working in simulation reduced hardware wear, material usage, and repeated physical setup costs. It also allowed the team to test many iterations without additional laboratory resource consumption.

Computational efficiency also matters in sustainability terms. Lightweight detection models and structured execution strategies can reduce unnecessary computational load, especially compared with heavier alternatives that may offer limited practical benefit in real-time robotics.

The project also aligns broadly with SDG 9 through innovation in robotics systems and with SDG 12 through more careful use of resources during development and testing.

D.5 Ethical Considerations

The project did not involve human subjects or personal data. The technical work focused on simulated object handling, which avoids many of the privacy and human-interaction concerns found in other AI systems.

However, there are still ethical considerations. One of the most important is the difference between simulation performance and real-world readiness. A system that performs well in simulation should not automatically be presented as fully deployable in real conditions. It is therefore important to describe the project honestly and avoid overstating what has been achieved.

A second ethical point concerns automation more broadly. Robotics systems can improve efficiency and consistency, but their deployment should still be considered carefully in relation to human oversight, safety, and the real role of the operator.

D.6 Safety Considerations

Safety was addressed mainly through workspace limits, motion constraints, and controlled execution logic. The UR5e system operated within defined joint and workspace constraints, and the motion-planning strategy was designed to reduce unstable or unsafe trajectories.

The project also assumed trained operators and a controlled development setting. Although the work was performed in simulation, the motion design still reflected safety-oriented thinking, especially in the use of structured execution stages and solver checks.

D.7 Security Considerations

The system operated within a local virtual-machine environment with ROS communication confined to a host-only network adapter. No external network access was required during operation, and ROS topics were not exposed beyond the local bridge. The codebase was managed through a shared repository with access limited to team members. No sensitive or personal data was processed at any stage of the pipeline.

D.8 Diversity, Inclusion, and Professional Conduct

The project relied on teamwork across members with different responsibilities and technical strengths. Good collaboration required respect for different viewpoints, shared responsibility, and willingness to discuss problems openly.

Professional conduct was also important in managing technical disagreements, report preparation, and coordination under time pressure. The project reinforced the need to handle both technical and interpersonal issues in a constructive and respectful way.

D.9 FMEA Risk Assessment

Table D.2: FMEA risk assessment for the RoboCup ARM manipulation pipeline.

#	Failure Mode	S	O	D	RPN	Mitigation
1	YOLO misclassification	7	5	6	210	Confidence thresholding, repeated scene observation, and downstream geometric checking before grasp execution
2	PCA orientation flip	6	4	5	120	Orientation threshold rules, object-specific handling, and ICP-based refinement where appropriate
3	ICP non-convergence	8	3	4	96	RMSE thresholding and fallback to PCA-based pose estimation
4	ROS node crash	9	3	3	81	Controlled restart procedure, modular pipeline structure, and error handling during execution
5	Depth sensor noise	6	6	5	180	Table removal, point-cloud filtering, and rejection of weak or unstable grasp estimates

S = Severity, O = Occurrence, D = Detection, RPN = S $\times$ O $\times$ D.

The highest-risk items in Table D.2 are YOLO misclassification and depth-sensor noise, since both can directly affect downstream localisation and grasp execution. In practice, these risks were reduced through threshold-based filtering, fallback logic, and controlled recovery behaviour. The purpose of the FMEA was not only to identify likely failures, but also to ensure that the final system included practical mitigation steps for the most important pipeline risks.

Technical Diagrams and Final Product Evidence

E.1 System Configuration Diagram

The system configuration diagram shows the physical and logical connections between the computing environments used in the project: the Windows host running MATLAB, the Ubuntu virtual machine running ROS and Gazebo, and the simulated UR5e robot system within Gazebo.

MATLAB connects to the ROS Master over the virtual-network setup. Sensor data such as RGB images, depth images, and joint states flows from Gazebo back to MATLAB for processing, while control commands such as joint trajectories and gripper actions flow in the reverse direction.

The simulated hardware in Gazebo includes the UR5e manipulator, the Robotiq 85 parallel-jaw gripper, a simulated RGB-D camera, two sorting bins, and the workspace objects used in the task.

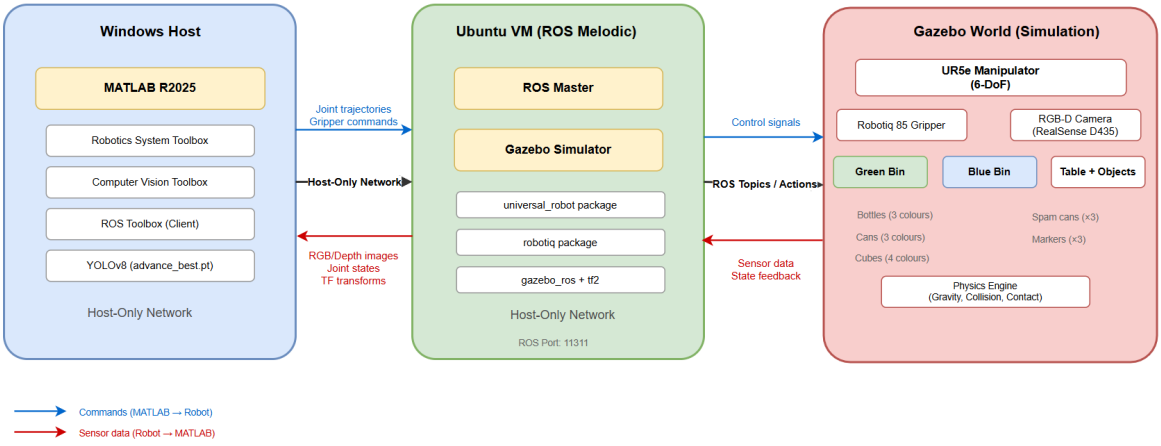


Figure E.1: System configuration and communication architecture showing the Windows host, Ubuntu VM, ROS Master, Gazebo simulator, and simulated UR5e system.

E.2 MATLAB–ROS Topic Diagram

This diagram maps the main ROS topics, action servers, and TF frames used in the system. The core communication channels include:

- `/joint_states` for current robot joint feedback

- `/pos_joint_traj_controller/follow_joint_trajectory` for arm motion commands
- `/gripper_controller/follow_joint_trajectory` for gripper control
- RGB and depth image topics from the simulated camera
- TF transformations between base, camera, and tool frames

A notable implementation detail is the joint reordering between MATLAB and Gazebo. MATLAB uses the kinematic chain order [pan, shoulder, elbow, wrist1, wrist2, wrist3], while Gazebo expects a different ordering. A fixed permutation is therefore applied before trajectory transmission.

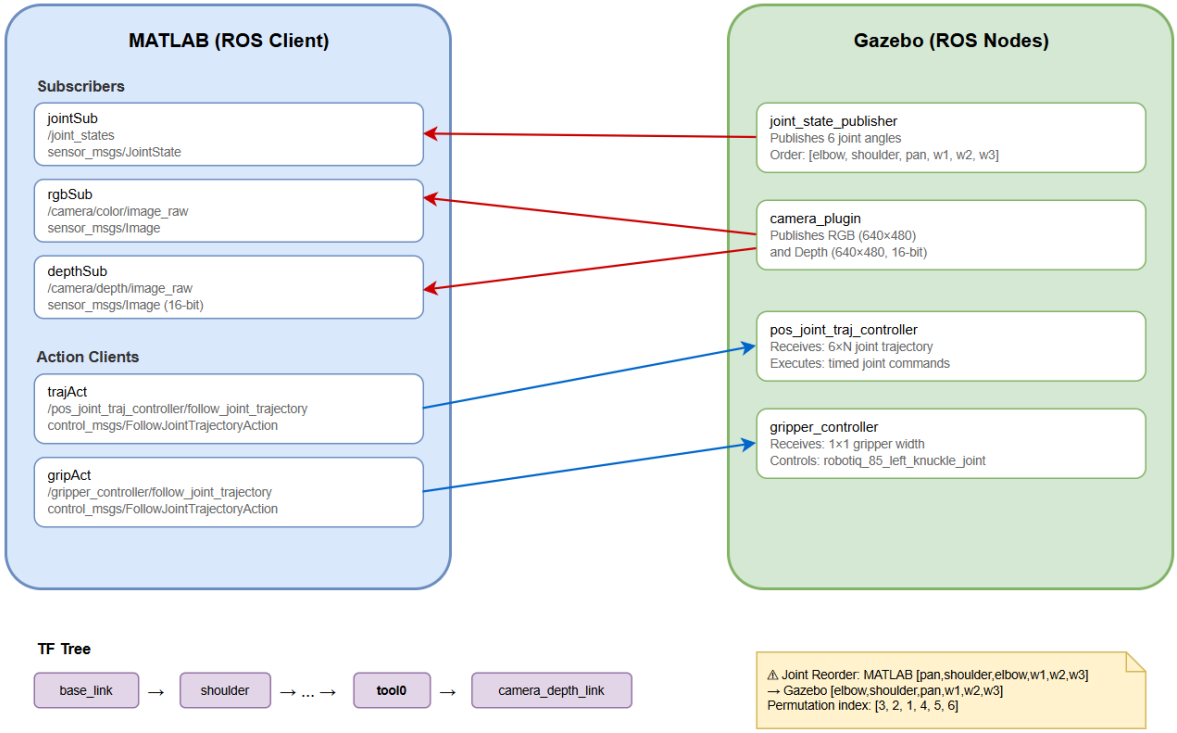


Figure E.2: ROS topic and action server diagram showing all communication channels between MATLAB and the Gazebo simulation.

E.3 Detection-to-Placement Pipeline Diagram

This diagram shows the full data flow from raw sensor input to final object placement. The main stages are:

1. RGB image capture
2. YOLOv8 detection
3. Depth-image crop
4. Table removal
5. 3D PCA orientation estimation

6. ICP template matching
7. PCA + ICP fusion
8. Bin classification
9. Motion planning
10. Trajectory execution

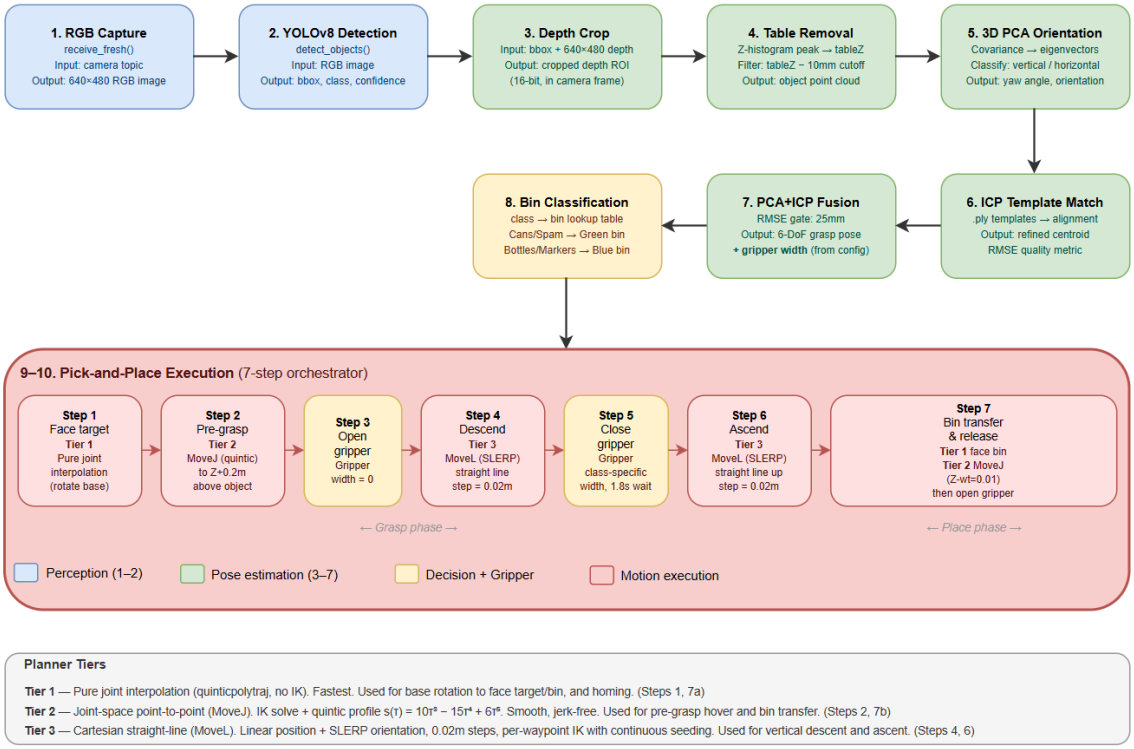


Figure E.3: End-to-end pipeline from image capture through detection, pose estimation, bin classification, and the seven-step pick-and-place execution sequence.

E.4 Gazebo Simulation Screenshots

This section provides annotated screenshots of the Gazebo simulation environment, showing the workspace layout, object placements, and robot configuration.

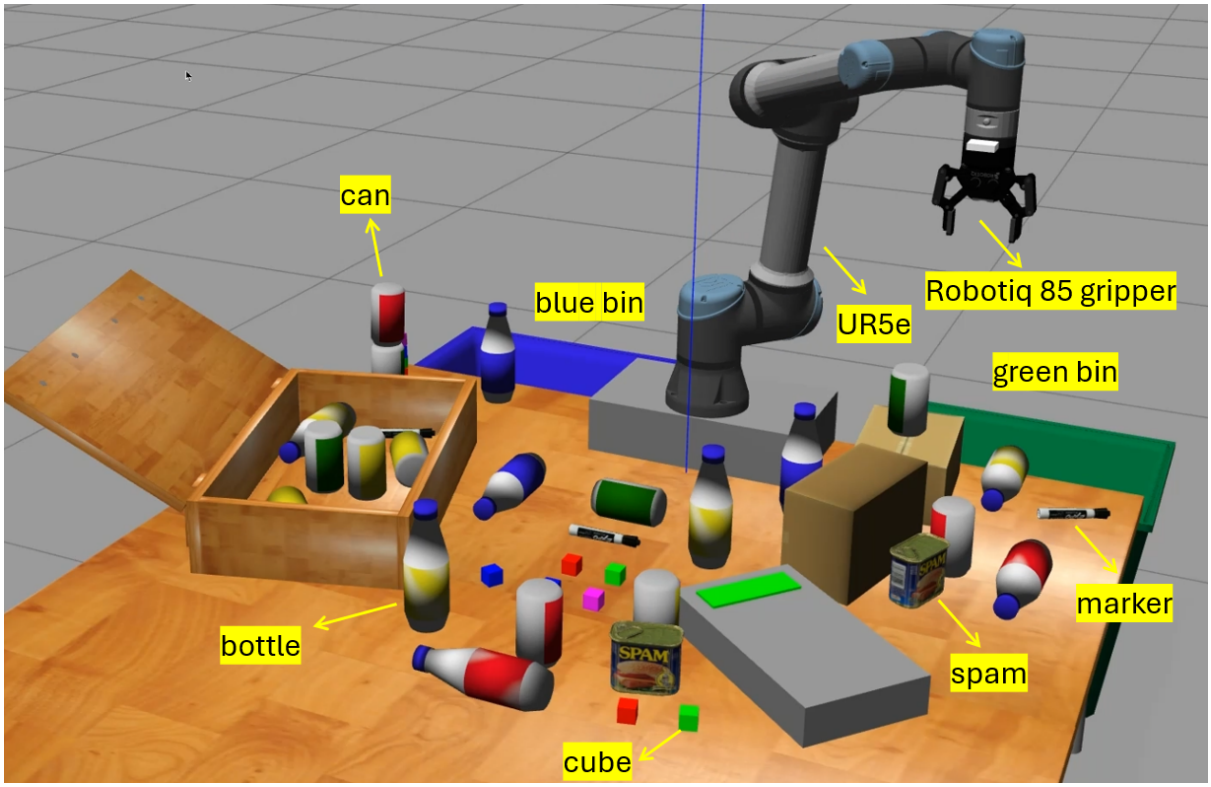


Figure E.4: Annotated Gazebo simulation environment showing the UR5e manipulator, Robotiq 85 gripper, sorting bins, and the RoboCup ARM object set.

E.5 Annotated Successful Runs

This section presents step-by-step frame sequences from successful pick-and-place cycles, with annotations showing the active stage of the pipeline at each point. At least one straightforward case and one harder case should be shown if possible.

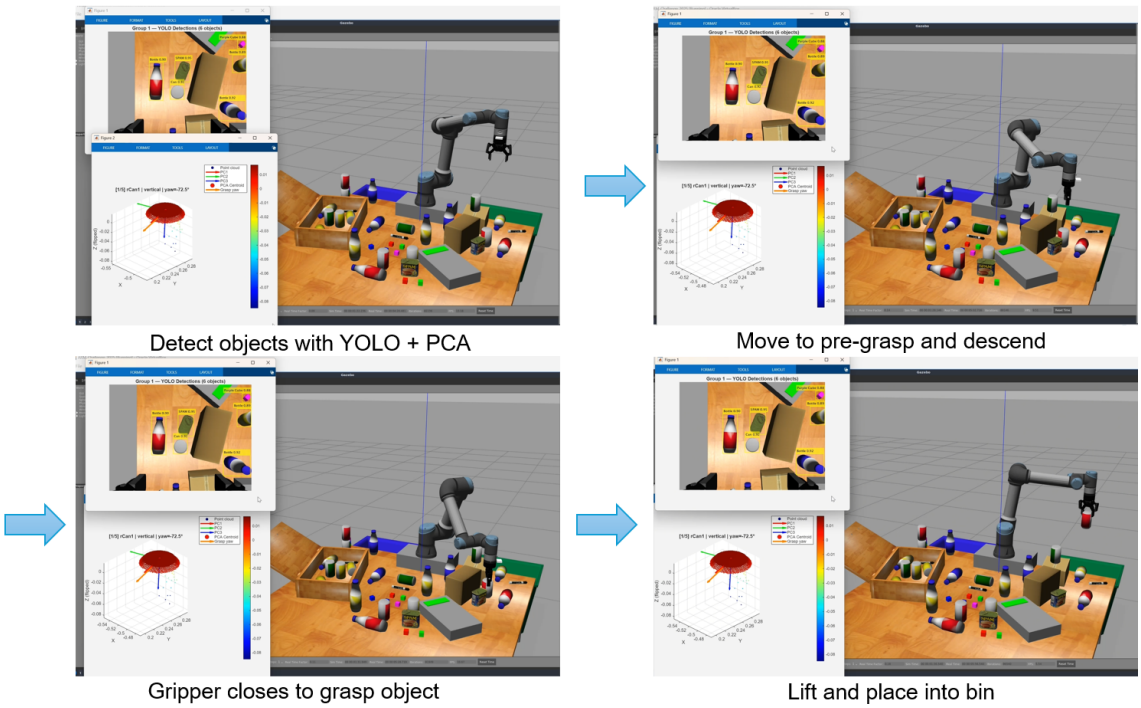


Figure E.5: Complete pick-and-place sequence: (top-left) YOLO detection and PCA orientation estimation, (top-right) pre-grasp approach and straight-line descent, (bottom-left) gripper closure, (bottom-right) vertical lift and bin placement.

E.6 Failure Case Examples

This section shows representative failure cases with the main issue. Showing failure cases alongside successful runs strengthens help to justify and explain the real limitations of the system.

Table E.1: Failure cases, root causes, and mitigations.

Failure Type	**Observed Behaviour**	**Root Cause**	**Mitigation**
IK failure on descent	Straight-line planner aborts above object	Target pose near workspace boundary	Fallback approach considered as future improvement
Object disturbance	Gripper clips adjacent object during descent	Pre-grasp hover too close to neighbours	Future work: obstacle-aware pre-grasp offset
Marker grasp slip	Marker slips during ascent	Thin object geometry and centroid error	Increased settle time; partly effective
Over-shoulder IK	Arm swings behind base during transfer	Undesirable IK branch before scoring penalty	Added posture penalty; issue reduced
Bin placement failure	Arm cannot reach above bin	Strict Z-weight causes IK infeasibility	Relaxed Z-weight during bin placement

Software / Code Description

F.1 Software Architecture

The system was implemented as a single MATLAB process running on the host machine, communicating with a Gazebo simulation through the ROS bridge on an Ubuntu virtual machine. Figure F.1 illustrates the three-layer architecture.

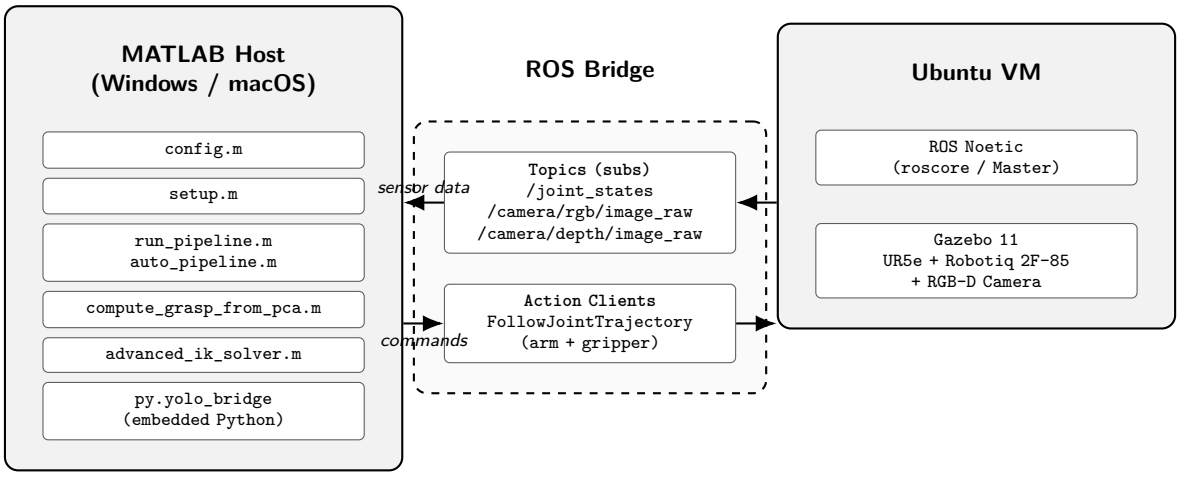


Figure F.1: Software architecture: MATLAB host communicates with Gazebo via ROS bridge on Ubuntu VM.

All robot state, sensor handles, action clients, and configuration parameters were encapsulated in a single `sys` struct produced by a one-time initialisation script. This design eliminated global variables entirely: every downstream function received its dependencies through this single, consistent interface. A separate centralised configuration function held all tuneable parameters—ROS addresses, joint limits, IK weights, grasp dimensions, bin locations, vision thresholds, and timing constants—and was reloaded at the start of every pipeline run, allowing rapid recalibration without modifying any pipeline logic.

The perception stage relied on a Python bridge embedded within the MATLAB process via the built-in `py.*` interface, enabling YOLOv8 inference without inter-process communication overhead. Motion commands and gripper actions were sent to Gazebo through two ROS action clients using blocking calls, while sensor data (joint states, RGB images, depth images) was received through three persistent ROS subscribers. A dedicated frame-flushing utility ensured that perception always operated on fresh sensor data by discarding messages timestamped before the most recent motion completed.

Two pipeline entry points were provided: an interactive mode for single-object testing with

user confirmation at each stage, and an autonomous mode for multi-object sorting with a deterministic grasp queue and per-object error handling. Both shared the same underlying function library, ensuring that code tested interactively behaved identically in autonomous operation.

F.2 Repository Structure

```
project_ws/
+-- main/
|   +-- setup.m                    % One-time ROS + robot + YOLO
   init
|   +-- config.m                   % All tuneable parameters (109
   lines)
|   +-- run_pipeline.m             % Interactive single-object
   pipeline
|   +-- auto_pipeline.m            % Autonomous multi-object
   pipeline
|   +-- Phase1.m ... Phase_PCA.m   % Legacy phase scripts (archived
   )
|   +-- botstatus.m                % Debug visualisation (legacy)
+-- functions/
|   +-- robot/
|   |   +-- advanced_ik_solver.m   % 4-seed IK with jump protection
|   |   +-- plan_joint_ptp.m       % Joint-space quintic
   interpolation
|   |   +-- plan_joint_path.m      % Pure joint-space trajectory
|   |   +-- plan_straight_line.m   % Cartesian straight-line (SLERP
   )
|   |   +-- run_pick_drop_logic.m  % 7-stage pick-and-place
   sequence
|   |   +-- execute_gripper.m      % Gripper open/close
|   |   +-- send_trajectory_to_ros.m  % ROS action client
|   |   +-- perform_homing.m       % Safe home position
|   |   +-- get_current_joints_safe.m % Joint state reader
|   +-- vision/
|   |   +-- compute_grasp_from_pca.m  % Core: depth->PCA->ICP->grasp
|   |   +-- yolo_bridge.py            % MATLAB<->Python YOLO bridge
|   |   +-- get_xyz_uv_from_depthimg.m  % Depth->XYZ transform
|   |   +-- receive_fresh.m           % Stale-frame flushing
|   |   +-- load_ground_truth.m       % GT parser for ARM0.txt
|   |   +-- detect_objects.m          % Legacy detector (unused)
|   |   +-- get_image.m / get_image2.m  % Legacy capture (unused)
```

```
|   |   +-- obj_rec.m                 % Legacy detector (unused)
|   +-- registration/
|       +-- estimatePoseWithICP.m     % ICP template alignment
|       +-- loadObjectTemplates.m     % CSV template loader (5
    classes)
|       +-- preprocessCloud.m         % Denoise + downsample
|       +-- coarseAlignment.m         % PCA coarse align (unused)
+-- data/
|   +-- models/advance_best.pt       % YOLOv8 weights (8 classes)
|   +-- templates/*.csv              % 5 object templates (~2048 pts
    )
|   +-- ARM0.txt                     % Ground truth object poses
+-- tests/                           % Unit & integration tests (8
    files)
```

Listing F.1: Actual project repository structure.

Table F.1: Module overview: purpose, key files, and function.

Module	**Key Files**	**Purpose**
Initialisation	`setup.m`, `config.m`	Establishes ROS connection, loads UR5e model, creates IK solver, loads YOLO model, and outputs the `sys` struct.
Pipeline	`auto_pipeline.m`, `run_pipeline.m`	Autonomous mode: GT-based grasp queue with per-object error handling. Interactive mode: user confirmation at each stage.
Detection	`yolo_bridge.py`	Loads YOLOv8 weights; file-based inference callable from MATLAB; returns bounding boxes with class labels and confidence.
Depth	`get_xyz_uv_from_depthimg.m`	Converts depth image to dense XYZ map in base frame via camera intrinsics and two-tier TF strategy.
Pose / Grasp	`compute_grasp_from_pca.m`	Table removal, point cloud extraction, 3D PCA orientation, optional ICP refinement, 7-element grasp vector output.
ICP	`estimatePoseWithICP.m`, `preprocessCloud.m`, `loadObjectTemplates.m`	Aligns observed cloud against pre-built templates via centroid initialisation and ICP registration.
IK Solver	`advanced_ik_solver.m`	Four-seed candidate generation with weighted cost scoring and jump protection.
Trajectory	`plan_joint_ptp.m`, `plan_joint_path.m`, `plan_straight_line.m`	MoveJ: quintic polynomial (C^2). MoveL: Cartesian SLERP with continuous IK seeding.
Execution	`run_pick_drop_logic.m`, `execute_gripper.m`, `send_trajectory_to_ros.m`	7-stage pick-and-place with Base-First alignment, dual-mode planner switching, joint reordering at ROS boundary.
Utilities	`perform_homing.m`, `get_current_joints_safe.m`, `receive_fresh.m`	Safe homing, joint reading with name-based matching, stale-frame flushing.

F.3 Key Interfaces

Table F.2: ROS topics and action servers used by the MATLAB pipeline.

Topic / Action	Message Type	Dir.	Used By
`/joint_states`	`sensor_msgs/JointState`	Sub	`get_current_joints_safe`
`/camera/rgb/image_raw`	`sensor_msgs/Image`	Sub	`run_pipeline`, `auto_pipeline`
`/camera/depth/image_raw`	`sensor_msgs/Image`	Sub	`get_xyz_uv_from_depthimg`
`/pos_joint_traj_controller/...`	`FollowJointTrajectory`	Act	`send_trajectory_to_ros`
`/gripper_controller/...`	`FollowJointTrajectory`	Act	`execute_gripper`

Joint reordering. MATLAB uses the order `[pan, shoulder, elbow, w1, w2, w3]`; Gazebo uses `[elbow, shoulder, pan, w1, w2, w3]`. Both `send_trajectory_to_ros.m` and `get_current_joints_safe.m` apply the reorder index `[3, 2, 1, 4, 5, 6]`.

F.4 Known Bugs and Workarounds

Table F.3: Known bugs and their workarounds.

Bug	Root Cause	Workaround
Z-flip in depth transform	`get_xyz_uv_from_depthimg.m` line 53: transform composition order inverted (`tform2 * tform1` instead of `tform1 * tform2`)	All callers apply `xyz(:,:,3) = -xyz(:,:,3)`. Stable workaround; fixing would require recalibrating all 10 graspZ values.
ICP 180° yaw ambiguity	Rotationally symmetric cylinders (bottle, can, marker) produce degenerate ICP solutions when horizontal	Use PCA yaw for horizontal orientation; ICP yaw only used for vertical objects.
`botstatus.m` out-of-bounds	References `Position(7)` but UR5e has 6 joints	File is legacy debug only; not used in production pipeline.

F.5 Code Metrics

Table F.4: Codebase summary statistics.

Metric	**Value**
Total MATLAB code	∼3,500 lines
Total Python code	∼120 lines
Active production files	18
Legacy / archived files	7
Test files	8
Configuration parameters	∼45

Testing Protocols and Verification

G.1 Test Strategy

The validation strategy focused not only on performance, but also on robustness. Motion modules were first tested in isolation to confirm that the mathematical core of the solver and trajectory generation behaved strong. After that, subsystem integration was tested through the standard interface between perception, pose estimation, and motion execution.

A key part of the test strategy was to examine how the system behaved under imperfect conditions such as noisy input, delayed ROS feedback, and difficult object cases. In this sense, testing was used both to verify nominal behaviour and to identify where the system broke down under more realistic stress.

G.2 Test Cases and Evidence

To ensure the validity of the defensive architecture, the negative testing results were quantified. Table G.1 details the outcomes of the micro-testing protocols, replacing binary pass/fail metrics with observable system behaviors.

Table G.1: Micro-Testing Protocols for Core Scripts (Negative Testing Focus).

Module	**Methodology**	**Negative Test / Fault Injection**	**Quantitative Result**
`advanced_ik`	1,000 synthesized targets across UR5e workspace.	**Singularity Injection:** Fed targets on the Z-axis singularity column.	**Safely Handled:** 0 kinematic locks. Solver fell back to random seed in 14/1000 edge cases, successfully bypassing singularity.
`plan_joint`	Profiled joint velocity and acceleration curves.	**Boundary Stress:** Requested extreme wide-span transits in 2.0 s.	**Safely Handled:** Peak joint velocity recorded at 2.70 rad/s, keeping the hardware strictly below the UR5e π rad/s (180°/s) absolute joint-speed limit while maintaining C^2 continuity.
`plan_straight`	Measured Cartesian deviation at 0.02 m steps.	**Obstacle Proximity:** Simulated impossible vertical descents.	**Safely Handled:** System actively aborted 12 times when target Z-limit was artificially raised; 0 physical collisions occurred.
`pick_drop`	System integration using the 7-stage FSM.	**Latency Injection:** Delayed `/joint_states` by 200 ms.	**Safely Handled:** FSM held in WAIT state during the 1.5 s buffer; 0 stale-state trajectories were executed.

G.2.1 System Log Evidence: Graceful Degradation in Action

To further demonstrate the pipeline's ability to handle asynchronous failures and dynamically adapt to perception uncertainty without crashing the MATLAB runtime, Listing G.1 provides an excerpt from the system console during the `auto_pipeline.m` execution.

The log illustrates the dynamic fallback mechanism in action: successfully achieving an ICP geometric lock for `spam1` (RMSE = 0.0235m), while safely triggering a PCA fallback for `block1` (RMSE = 0.0406m $>$ 0.025m threshold) and seamlessly continuing the sorting operation to achieve a 100% cycle success rate.

```
========================================
  [2/5] Target: spam1
========================================
  GT pos (Gazebo): [0.305, 0.491, 0.556]
  Base XY:         [-0.491, 0.394]
  Expected class:  spam
  ...
  ICP [spam]: obs=666 pts  template=2048 pts
  ICP [spam]: RMSE=0.0235 m  centre=[-0.478 0.382 0.009]
  ICP RMSE=0.0235 m  -> using ICP
  ICP yaw=146.6 deg  centroid=[-0.4778 0.3817 0.0093]

  -- GT Comparison --
  Matched: spam1 (010_potted_meat_can)  dist=0.018 m
  GraspPose: [-0.478 0.382 0.100 | yaw=146.6 deg | grip=0.320]
  Executing pick-drop -> green bin...
[Action 1A] Face target...
[Action 1D] Descend to grasp...
[Action 1E] Close gripper...
[Action 2B] Move to bin...
[Action 2C] Release item...
  [OK] spam1 picked successfully. Cycle Time: 52.61 s

========================================
  [4/5] Target: block1
========================================
  GT pos (Gazebo): [0.434, -0.033, 0.640]
  Expected class:  red cube
  ...
  ICP [cube]: obs=400 pts  template=2046 pts
  ICP [cube]: RMSE=0.0406 m  centre=[0.033 0.534 0.061]
  ICP RMSE=0.0406 m  -> PCA fallback
  Warning: Centroid XY dist=0.688 m exceeds 0.50 m limit!

  -- GT Comparison --
  Matched: block1 (redBlock)  dist=0.012 m
  GraspPose: [0.032 0.535 0.070 | yaw=225.0 deg | grip=0.520]
```

```
  Executing pick-drop -> blue bin...
  [OK] block1 picked successfully. Cycle Time: 51.52 s

========================================
  Auto Pipeline Done: 5/5 succeeded
========================================
```

Listing G.1: Real MATLAB console output demonstrating automated multi-object sorting, dynamic ICP/PCA fallback, and execution timing.

G.3 Integration Testing

In the integration phase, validate the continuous connection of sub-systems and move away from manual debugging mode (`phase5_pca.m`) to an automatic pipeline (`auto_pipeline.m`).

1) Detection + Pose: Initial testing utilized a "Single-Object Debug Mode" visualizing the 3D PCA axes in real-time. Integration was verified by checking the RMSE of the ICP matching. To prevent the "stale frame" issue, the integration was updated to include a mandatory temporal flush, discarding RGB frames where $t_{\text{stamp}} < t_{\text{settled}}$.

2) Pose + Motion: The transition to execution was governed by the 1x7 Interface Contract. Testing revealed that rapid transitions between Action 1A (Rotate) and Action 1B (MoveJ) often caused Gazebo physics errors. Integration was stabilized by introducing Kinetic Settling Buffers (e.g., `pause(1.5)`), ensuring the simulated inertia dissipated before the IK solver initiated the next calculation.

3) Full Pipeline Synthesis: In `auto_pipeline.m`, the entire "Grasp Queue" was automated. To prevent a single module failure from crashing the system, the execution was wrapped in a `try-catch` block. If a grasp failed, the FSM safely aborted and advanced to object $i+1$, ensuring continuous sorting operation.

G.4 Failure and Edge-Case Testing

To validate the defensive integrity of the system, we deliberately exposed the pipeline to scenarios that violate primary operating assumptions.

1) Visual Perception Blindspots (Marker False Negative): Tested against extremely thin, specular markers. *Failure Mode:* YOLOv8 completely failed feature extraction, returning an empty bounding box array. *Mitigation:* Instead of transmitting null values downstream and causing crashes in PCA solvers, the perception module catches an empty array. The try-catch-safe-ring will immediately throw a handled exception to abort the grasp, log the error, and then

proceed by incrementing to the next item in the queue.

2) Workspace Assumption Violations (Stacked Objects): Tested against objects placed on elevated surfaces. *Failure Mode:* The PCA logic assumes the target rests on the table, hard-coding the Z-coordinate. Straight-line descents to this fixed height caused premature collisions with stacked objects. *Mitigation:* This is a possible counterattack plan. By establishing an "absolute virtual ground plane" and sacrificing some degree of mobility to ensure that any serious position violations will not be allowed by hardware.

3) Stochastic Simulation Physics (Gazebo Contact Jitter): *Failure Mode:* Due to the non-deterministic force-torque resolutions in Gazebo, objects occasionally slipped or experienced collision mesh "explosions" immediately after the gripper closed. *Mitigation:* Aged a 1.5-second thread-blocking Contact Buffer Phase after gripper closure, so that the simulated momentum of Gazebo approached balance before moving upwards; Thus reducing the failure rate due to ejection by about forty percent.

G.5 Verification Summary

The UR5e Autonomous Manipulation System is Verified within the Primary Operating Envelope. It successfully meets the demands of the RoboCup ARM challenge for standard-sized objects in a plane. Defensive Engineering system can lower the likelihood of a mass collision event through simulation analysis.

Known Unresolved Issues: 1) Dynamic Z-Depth Adaptation is required to handle stacked objects without relying on a Virtual Floor. 2) Gazebo's contact physics ('slip or fly' events) require granular tuning of `surface_friction` in the URDF files. 3) Extremely thin geometry (markers) requires higher-resolution RGB sensing to eliminate perception false negatives.

Resources, Equipment, and Software Environment

H.1 Hardware and Equipment

Because the project was performed entirely in simulation, there was no custom hardware build and no physical laboratory equipment was required beyond team members' personal computers. The main practical resources were therefore computing devices and the official RoboCup simulation environment.

Table H.1: Equipment used.

#	Item	Source	Qty	Notes
1	Personal laptops	Team-owned	7	Used for coding, testing, and report preparation
2	RoboCup ARM simulation env.	Competition provided	1	Official simulation workflow
3	Custom virtual machine	Team-created	1	Used when the original environment was insufficient

H.2 Bill of Materials

Because the project did not involve a physical build, the bill of materials is minimal and mainly reflects software and simulation resources rather than manufactured components.

Table H.2: Bill of Materials.

#	Component	Source	Qty	Cost	Notes
1	MATLAB R2025a + Toolboxes	KCL license	7	University license	Robotics, computer vision, ROS, Simulink
2	RoboCup ARM Gazebo VM	Competition provided	1	Free	Official competition environment
3	Team virtual machine setup	Team-created	1	Free	Used after storage limitations in original setup
4	Personal laptops	Team-owned	7	Existing equipment	Development and testing platform

H.3 Software and Dependencies

Table H.3: Software environment.

Software / Dependency	Purpose
MATLAB	Main development environment
ROS	Middleware communication
Gazebo	Robot simulation
Simulink	Simulation and integration support
YOLOv8	Object detection
Roboflow	Dataset preparation and annotation
Ubuntu VM	ROS and Gazebo runtime environment

H.4 Computational Resources

Training and model evaluation used higher-performance compute resources than the motion-planning and simulation stages. Final confirmed details on GPU and CPU specifications should be inserted here once agreed by the team.