RoboCup Arm Challenge Proposal

MEng Group Project (7CCE4EGP / 7CCEMRGP)

King's College London, Department of Engineering

Team Members

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

Supervisors: Dr Jon-Erik Dahlin, Dr Shan Luo

Date: November 2025

Contents

1 Introduction & Background

1.1 Context and Overview

Robotic manipulation is the capability of a robot to perceive, plan, grasp and physically interact with the surrounding/ environment. It is a foundational skill across industrial, research and service robotics. The **Universal Robots UR5e**, a 6DOF collaborative manipulator, which is the one that will be discussed in more detial thorugh out this project.

We participate in the **RoboCup Autonomous Robot Manipulation (ARM) Challenge**, supported by **MathWorks** and **Universal Robots**. The challenge evaluates how well teams can program a manipulator to identify, grasp, and sort objects in a simulated environment. All development is carried out entirely in MATLAB (Robotics System Toolbox, Computer Vision Toolbox, Simulink ROS blocks), ensuring a single-language workflow for perception, planning, and control.

To keep the introduction focused on motivation and scope, the detailed software/hardware setup and the MATLAB stuff are described later.

1.2 Background and Motivation

Automated sorting and manipulation are vital for sustainability and efficiency. Recycling facilities depend on accurate classification of materials such as metals and plastics and many more, and these are extremely difficult tasks, because they are usually time consuming, error-prone, and physically demanding for humans. Even experienced workers struggle with shape variability and lighting changes. A robotic system that can recognise and sort waste on its own contributes directly to efficiency and safety.

Automating recycling enhances both environmental and economic performance: it reduces landfill usage, improves resource recovery rates, and lowers many costs. From a research perspective, these tasks also a solution to many open problems in robotics integrating perception, motion planning, and control under uncertainty.

Academic competitions such as RoboCup are essential drivers of innovation. They give the opportunity for amazing engineering and teamwork to come in place. Each year, the ARM Challenge evolves to include new object types and dynamic scenarios, forcing teams to improve perception robustness and control efficiency. For students, it offers a valuable opportunity to test their analytical skills and ability to work in a team, which is essential in any engineering environment.

1.3 Challenge Description

The RoboCup ARM Challenge requires programming the UR5e to identify, grasp, and sort objects randomly distributed on a table. Each object belongs to one of three colour-coded

difficulty levels:

- **Green objects:** fixed shape – basic colour classification.
- **Yellow objects:** variable shape – shape recognition required.
- **Red objects:** variable orientation – orientation correction required before placement.

Higher difficulty yields higher points (*red ¿ yellow ¿ green*). The robot must rely solely on its on-board RGB-D camera for perception; direct access to internal simulator states or APIs for object manipulation is prohibited. All perception, planning, and control logic are implemented in MATLAB and executed via the ROS interface provided in the official competition environment.

Our system goal is to develop an **accurate and fast sorting algorithm**, prioritising classification and placement accuracy and once achieved, we will iteratively minimise total task time through controller and motion-planning optimisation.

1.4 Literature and Benchmark Review

The ARM Challenge builds upon MathWorks' example *"Pick and Place Robot Manipulator with Point Clouds and RRT"*, which showcases perception and trajectory generation entirely in MATLAB. That example demonstrates reliable motion planning for static scenes but offers limited adaptability to dynamic objects or time controlm which are essential.

Past competition reports indicate that top teams rely heavily on MATLAB's Computer Vision and Robotics toolboxes to segment point clouds and plan grasps. Classical image segmentation and geometric heuristics yield high accuracy but lower flexibility when object orientation varies. Deep-learning-based perception can be computationally heavy and is often avoided under real-time competition constraints.

Beyond RoboCup, research benchmarks such as *OCRTOC* (Open Cloud Robot Table Organisation Challenge) and *CausalWorld* explore object manipulation with reinforcement learning, but they depend on non-MATLAB frameworks and cloud resources. In contrast, the ARM Challenge constrains teams to MATLAB, encouraging students rely on human knowledge rather than reinforced models.

Table 1: Summary of related works and benchmarks.

System / Benchmark	Methodology	Outcome	Limitation
MathWorks Pick & Place (2024)	Point-cloud + RRT planner	MATLAB-only pipeline	Static setup
RoboCup 2024 Winner	Colour/shape heuristics	Reliable fixed-item grasping	Sensitive to orientation
OCRTOC (2023)	RL policy + cloud execution	Flexible real-robot sorting	Non-MATLAB framework
Proposed Project (2025)	MATLAB vision + motion planning	Accuracy then speed focus	Early stage

1.5 Transition to Problem Statement

Robotic manipulation in dynamic environments remains a challenging task even considering all the technological and code advancement that happened throughout years. Through the RoboCup ARM Challenge, our team aims to design, implement and test a complete MATLAB manipulation capable of accurate, efficient object sorting in simulation and, later, on the physical UR5e. The next section formalises the problem definition, scope and measurable objectives guiding our implementation.

2 Problem Statement

Autonomous robotic grasping requires good understanding of perception, planning, and precise execution. Despite strong progress, systems still struggle with reliability and efficiency in dynamic settings. We target a MATLAB/Simulink-based control system to perform grasping and sorting within the *RoboCup Autonomous Robot Manipulation (ARM) Challenge* [**?**]. The challenge provides a realistic benchmark for validating real-time manipulation in simulation and, later, on physical hardware.

2.1 Critical Challenges and Our Focus

From perception through execution the research projects sows several challneges which we address using MATLAB/Simulink methods.

2.1.1 Perception Module Robustness

Custom, optimised DL detectors (e.g., YOLO variants) are essential for tilted items but introduce library conflicts and unsupported layers when integrated in MATLAB. These incompatibilities delay deployment of competition level perception; resolving them is a priority for reliable detection, which we will wrok on relaly hard during the project.

2.1.2 Multi-Modal Sensor Fusion

Reliable understanding in RoboCup requires fusing RGB and depth. Standard detectors (e.g., YOLOv8) are RGB-only. Our provisional method maps an RGB bounding-box centre to its depth pixel, it works but is very senstitive to mistakes, small box shifts under motion/occlusion yield large 3D pose errors. A more stable, frame-consistent fusion system is needed.

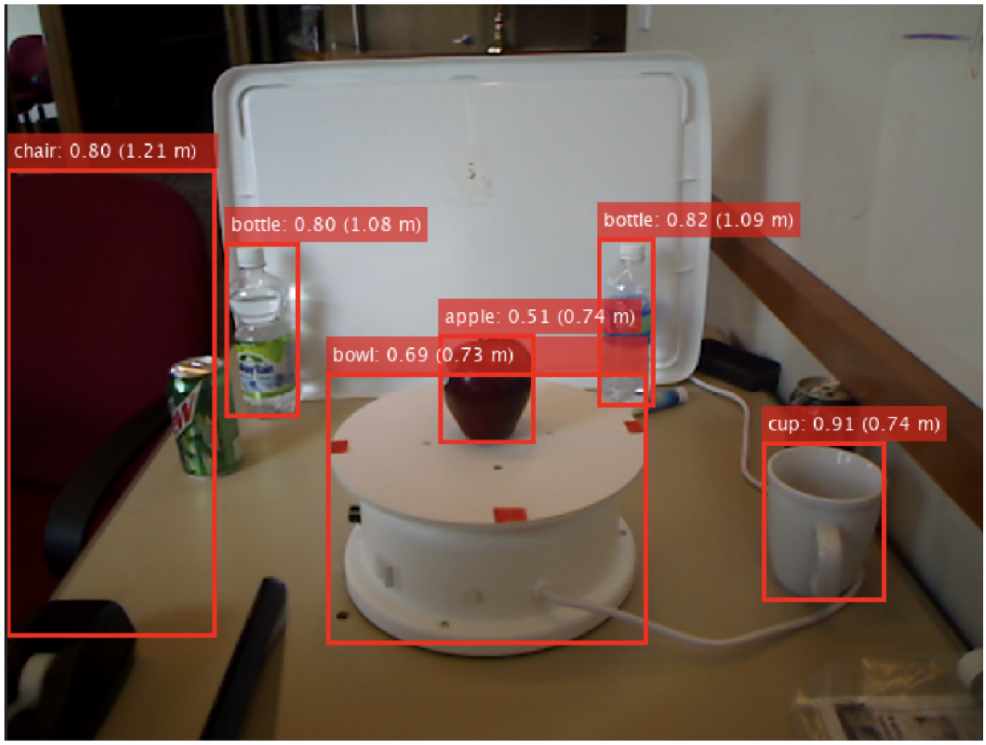


Figure 1: Perception test in Gazebo/MATLAB: RGB detections with associated distance estimates illustrate current fusion approach and its sensitivity to box drift.

2.1.3 Constrained Field of View Hinders Global Monitoring

Camera FOV limits workspace coverage; objects at the edge of the camera can leave the frame, breaking global situational awareness and delaying grasp selection. Mitigations include dynamic camera repositioning or multi view fusion.

2.1.3 Constrained Field of View Hinders Global Monitoring:

Figure 2 Camera view in Gazebo

Figure 2: Gazebo camera viewpoint illustrating limited FOV and occlusions around the UR5e workspace.

2.1.4 Inverse Kinematics Complexity

The UR5e's 6-DOF kinematics yield multiple solutions and singularities, inverseKinematics may return non-executable or weird poses. A more reliable, task-aware IK strategy (constraints, seeding, and solution filtering) is required, which we will hradly work on.

2.1.5 Motion Planner Efficiency

ManipulatorRRT is complete but can produce long detours in narrow passages typical of RoboCup scenes. We must improve path quality and the speed of completion to hit competition cycle-time targets.

2.1.6 Multi-Task Order Management

Left-to-right picking ignores scoring and travel-time trade-offs. We will design an order-selection policy that prioritises reachable, high-value items to maximise total score under time limits.

2.1.7 Gripper Control Sensitivity

Millimetre-scale localisation and also noise distorts width estimates, causing fingertip-only contact and unstable grasps. We will implement compliant force/position strategies and tolerance bands to maintain stable contact despite perception error.

2.2 Project Aims

- **High Reliability:** $>90\%$ single-attempt grasp success on partially occluded objects.
- **Operational Efficiency:** $\approx$3 s perception-to-grasp cycle time.
- **System Robustness:** Performance stability within $\pm 5 - 7\%$ across lighting and scene variations.

Metrics align with RoboCup criteria and Section 3.

2.3 Stakeholder Value

- **Industrial (MathWorks / Organisers):** Model-Based Design demonstration with MATLAB/Simulink and code generation for real life control.
- **Academia (KCL):** Teaching framework linking theory and simulation.
- **RoboCup Community:** Documented perception planning control system that lowers entry barriers.

- **Wider Industry:** Transferable workflow for sustainable automated sorting.

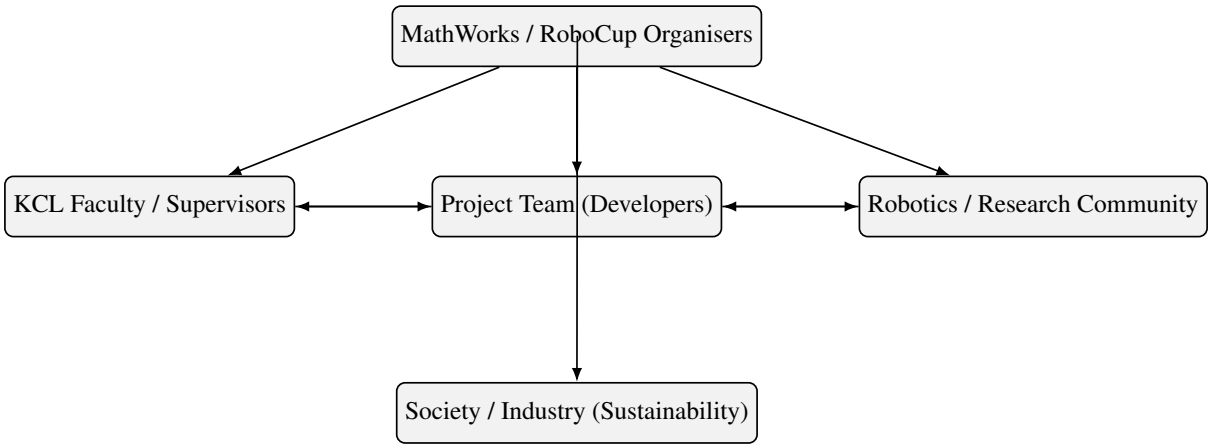


Figure 3: Stakeholder relationships for the RoboCup ARM project.

Table 2: Stakeholder needs versus project objectives.

Stakeholder	**Need / Expectation**	**Project Objective Addressing It**
MathWorks / RoboCup Organisers	Demonstrate MATLAB-based new code.	Validated Simulink grasp/sort ideas in simulation and UR5e lab tests.
KCL Faculty	Educational integration and technical quality.	Reproducible MATLAB workflow and datasets for teaching and research.
Team Members	End-to-end robotics experience.	Implement, test, and evaluate a full MATLAB/Simulink system.
Robotics Community	Accessible idea implementation.	Release documented code and metrics for future teams.
Society / Industry	Sustainable, practical automation.	Methods applicable to intelligent sorting and recycling.

2.4 Project Objectives

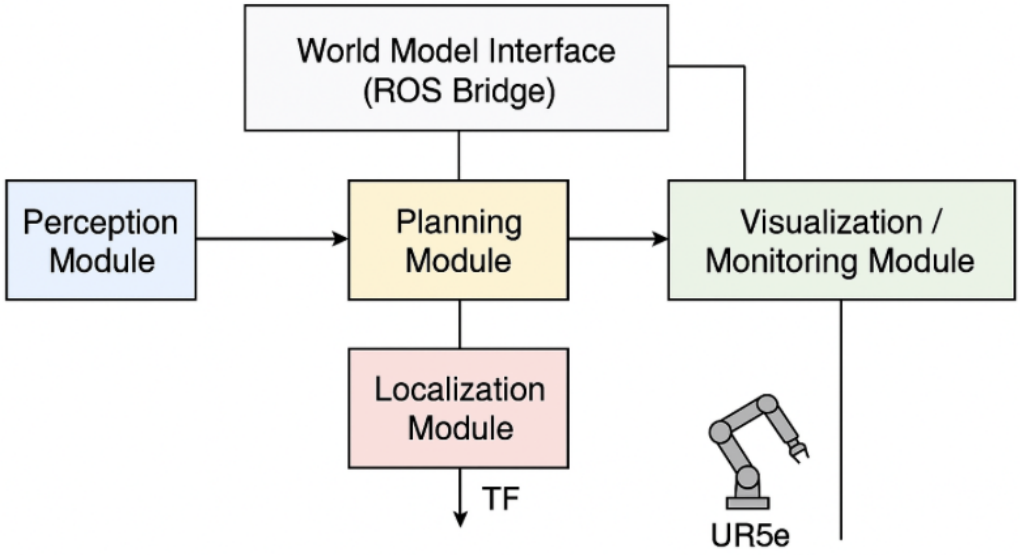


Figure 3 – System Architecture / Block Diagram

Figure 4: System Architecture / Block Diagram showing the interaction between the Perception, Planning, Localization, Visualization, and ROS Bridge modules integrated with the UR5e manipulator.

Primary Objectives

1. **End-to-end system in MATLAB:** Detect, classify, and find objects in the RoboCup ARM Gazebo VM using *Computer Vision Toolbox*.

2. **Planning and control for UR5e:** Use *Robotics System Toolbox* and Simulink ROS interfaces (Noetic) to execute pick and place trajectories.

3. **Performance target (simulation):** Achieve $\geq$ **90%** task success across $\geq$ five object categories under nominal lighting and clutter.

4. **Unified model:** Integrate perception, planning, and control into a single Simulink model communicating *entirely* via MATLAB and also ROS as stated previously.

5. **Documentation & tests:** Produce architecture docs, simulation logs, and design justification with reproducible run scripts, which will be completed throughout the whole project.

Stretch Objectives

1. **Hardware transfer:** Run the integrated controller on the KCL UR5e with almost no changes (subject to lab time and safety).
2. **Online re-planning:** Enable re-planning when objects move or grasps fail.
3. **Speed optimisation:** Reduce cycle time while maintaining accuracy (e.g., controller gains, trajectory timing; any RL kept within MATLAB constraints).

2.5 SMART Objectives (Table 3)

Table 3: SMART objectives for the RoboCup ARM project.

Objective	Specific	Measurable	Achievable	Time-bound
Develop MATLAB/Simulink perception–control pipeline	Yes	Success $\geq 90\%$	Toolboxes available	Mar–Apr 2026
Simulink model with Gazebo ROS	Yes	Runs end-to-end without errors	Supported by RoboCup VM	Mar 2026
Execute pick–place on UR5e (simulation)	Yes	5 object categories completed	Feasible using Robotics System Toolbox	Feb–Mar 2026
Hardware demo (stretch objective)	Yes	Replicates simulated task on UR5e	Requires KCL lab access	Apr 2026

2.6 Expected Outcomes

The project will deliver: (i) a Simulink model (perception planning control) for the RoboCup ARM simulator; (ii) a good evaluation of success rate, cycle time, and success under lighting/clutter variation; (iii) a technical report with architecture, parameters, and test results; and (iv) a live UR5e demo if stretch objectives are achieved. Supporting stuff include datasets, logs, and a short user manual for replication.

2.7 Constraints

Time. Oct 2025–Apr 2026 following the milestones in Section 5 (Gantt).
Software. MATLAB R2025+ with Robotics System Toolbox, Computer Vision Toolbox, and Simulink ROS; **MATLAB-only** development. ROS Noetic within the official Linux VM.
Hardware. RoboCup ARM Gazebo VM; UR5e in KCL lab for final tests; workstation ($\geq$ i9, 32 GB RAM) (What we expect).
Practical limits. The x86 Linux VM can be slower on macOS hosts; ROS Noetic limits us to ROS1 bridges; MATLAB-only. These are accounted for in Section 5 scheduling and Section 6 risk mitigations.

2.8 Feasibility Analysis

Technical

All required components are natively supported: ROS publishers/subscribers for control (Robotics System Toolbox), image segmentation and pose estimation (Computer Vision Toolbox), and Simulink integration for deterministic execution. The modular model allows independent testing of subsystems and rapid iteration. No custom middleware is required; MATLAB bridges directly to the Gazebo VM.

Resources

Resources available at KCL include: (1) the official RoboCup ARM Gazebo VM, (2) licensed MATLAB with required toolboxes, also provided by KCL university (3) access to a UR5e arm for final validation, and (4) workstations capable of real-time simulation. Implementation and verification align with the timeline and dependencies in Section 5 (WBS & Gantt). Ethical and safety considerations are handled per Section 6 in general everything is completed safely with no danger.

3 Pre-Study

3.1 Executive Summary

This pre-study demonstrates how good is a multi-phase autonomous manipulation system in the RoboCup ARM Gazebo environment. Over three weeks we implemented three progressive phases to assess technical complexity and scope. The simulation confirms that perception and control are functional and usable.

3.2 Project Objectives and Expected Outcomes

The pre-study aimed to produce a working prototype demonstrating progress:

- **Phase 1:** Pick up and place, using known object poses.
- **Phase 2:** sorting with RGB detection and class-driven grasp parameters.
- **Phase 3:** Depth-based 3D localisation and orientation estimation (towards fully autonomous grasping).

Deliverables include a Gazebo simulation controlled from MATLAB, short demo videos for each phase, and a technical report with architecture, algorithms, and metrics.

3.3 Simulation Work Conducted

Environment Setup An Ubuntu VM (ROS Melodic) runs the simulation and ROS Master; MATLAB on the host connects as a ROS client. It will contain a UR5e, a Robotiq 85 gripper, an RGB-D camera, bottles/cans/markers on a table, and colour-coded sorting bins. Gazebo provides gravity, collisions, and contact dynamics. Just like a reallife robot.

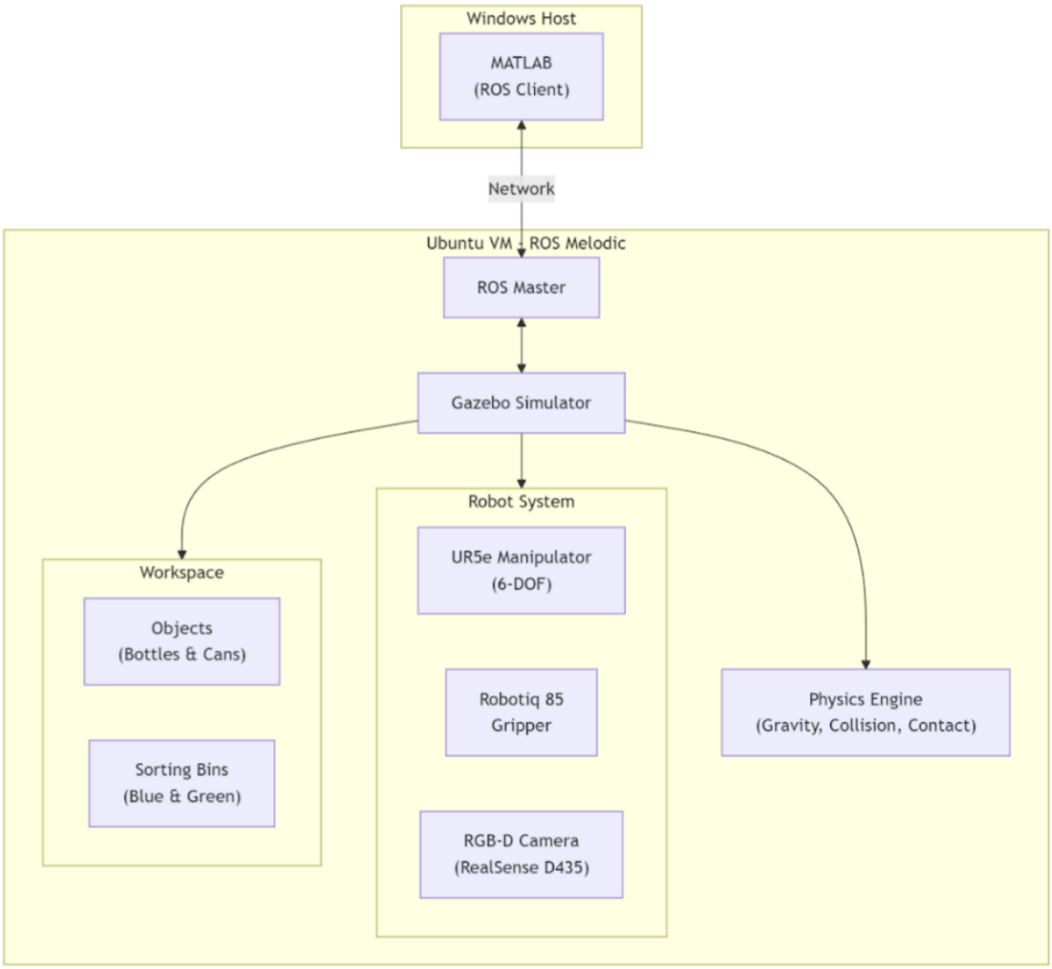


Figure 5: Simulation architecture: MATLAB–ROS–Gazebo pipeline with workspace objects, sensors, physics, and the UR5e system.

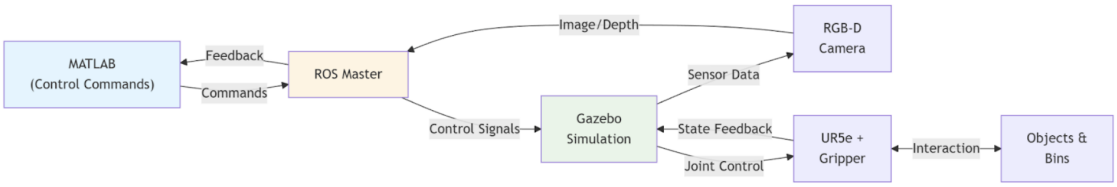


Figure 6: Data-flow diagram: commands/feedback via ROS, image/depth to perception, state feedback and joint control to Gazebo and UR5e.

Phase 1 — Hardcoded Position Control Six predefined 6-DoF poses were executed via inverse kinematics and timed trajectories.

Phase 2 — Vision-Assisted Sorting RGB images are processed by a pre-trained CNN (`RobocupDetector.mat`) producing boxes, labels, confidences. Objects are ordered left-to-right; grasp parameters adapt by class (eg: bottle/can).

Phase 3 — Autonomous Depth Processing Depth images (480×640, 16-bit) are unprojected using camera details, transformed to the base frame with TF, and filtered to object clouds using RGB boxes. PCA on hull points gives principal axis and in-plane orientation.

Table 4: Pre-study summary (Phase → Inputs, Methods, Outcomes/Insights).

Phase	Inputs	Methods	Outcomes / Insights
1	Hardcoded 6-DoF poses; gripper width	IK + trajectory execution; pre-grasp offsets	Reliable pick–place baseline; smooth control;
2	RGB images;	Detection, confidence filtering, class based grasp adaptation	Correct sequencing; closed-loop from vision to movement.
3	Depth + RGB; TF tree	Unprojection, TF transforms, cloud cropping, PCA orientation	Accurate 3D location of the object; orientation obtained; grasp-pose mapping .

3.4 Tools and Software

ROS Melodic (VM) with `universal_robot`, `robotiq`, `gazebo_ros`, `tf2`; MATLAB (Robotics System, ROS, Computer Vision Toolboxes); Gazebo physics and sensor simulation; RViz/`rqt_graph` for debugging. The detector (`RobocupDetector.mat`) classifies bottles/cans.

3.5 Technical Feasibility

- MATLAB–ROS communication operates at high level.
- IK, detection, depth processing, and TF transforms run end-to-end.
- Simulation is stable; sensor outputs are suitable for algorithm development.

3.6 Discovered Challenges

Grasp Planning - Mapping orientation to executable grasp poses is more complex than estimated; allocate 2–3 weeks, with a lot of testing.

Fallback - Phase 2 demo (manual orientation) if schedule tightens. Depth Sensor Noise - Simulated depth outliers affect hull/PCA. Mitigation: statistical outlier removal, median filtering, or RANSAC plane fitting ($\sim$1 week).

3.7 Summary

The pre-study demonstrates feasibility of the MATLAB–ROS–Gazebo system and shows each subsystem. Only grasp planning remains for full autonomy.

4 Project Plan

This section defines the management framework used to deliver the RoboCup ARM project. It explains how the work is decomposed, scheduled, and tracked from week to week.

4.1 Work Breakdown Structure

he execution of this project is built through a **Work Breakdown Structure (WBS)**. In order to well organised all technical development and integration, alongside preparatory work for assignments. We subdivide the complex robotic challenge into a series of clearly defined, controllable phased tasks. All assignments have been actively planned and clearly finalized within the Figure 7.

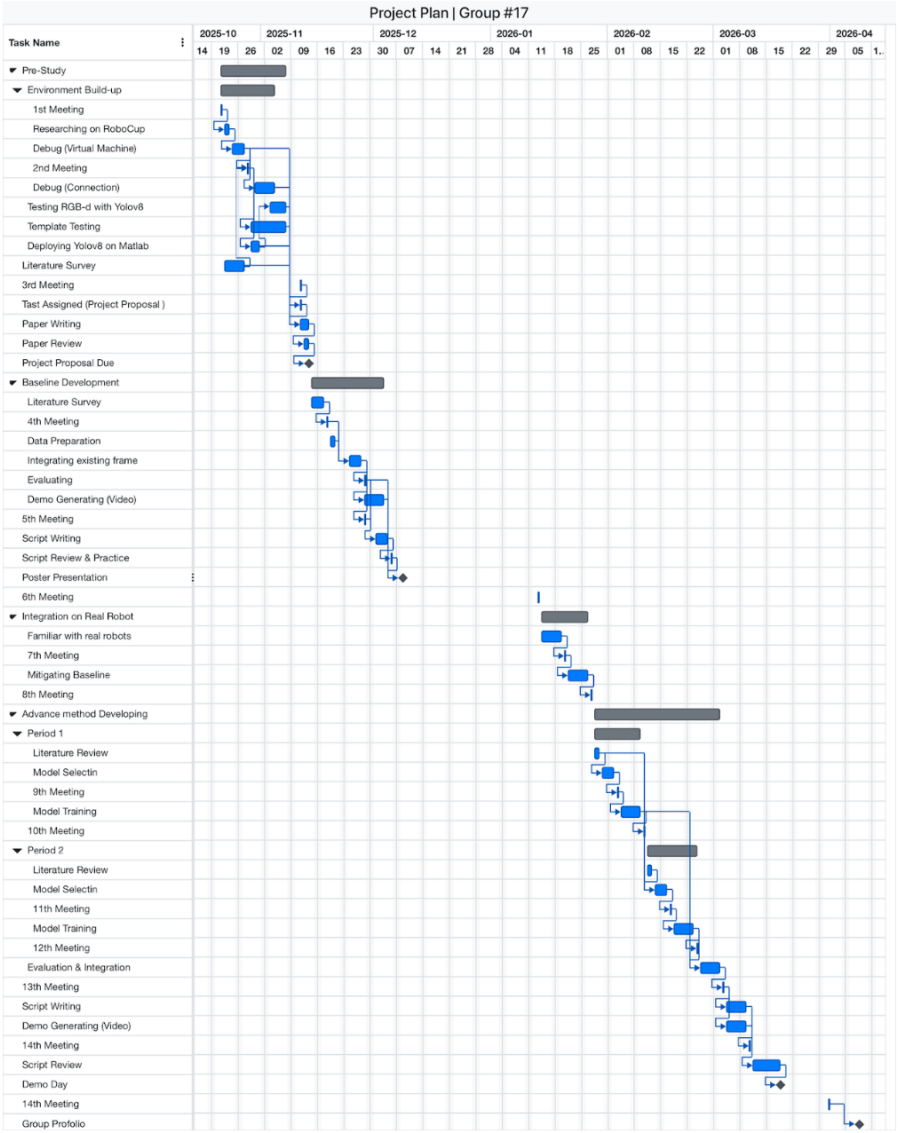


Figure 7: Gantt chart (Oct 2025–Apr 2026) showing weekly work packages, dependencies, and milestones for Group 17.

Each milestone also informs the risk mitigation planning described in Section 6, ensuring that schedule risk is monitored alongside technical risk.

4.2 Continuous Follow-up Management

Our board organises tasks into distinct lists according to project session and working phases, covering the entire workflow like scripting and baseline development to deployment optimisation.

Colour-coded card labels intuitively convey task urgency and timeframe, to help the team in prioritising focus:

- **Orange** — urgent/high-priority deliverables (e.g., proposal components).
- **Blue** — medium-term/standard tasks (e.g., baseline development).
- **Grey** — long-term/low-priority items (e.g., model training).

As shown in Figures 8, Trello clearly defines responsibility for each task through member assignments on cards, ensuring each work package has clear ownership and ensuring transparency. Through this method, the team can rationally allocate tasks based on members' expertise and utilise Trello's clear assignment mechanism to effectively minimise work overlap and redundancy, thus significantly enhancing the team's overall efficiency.

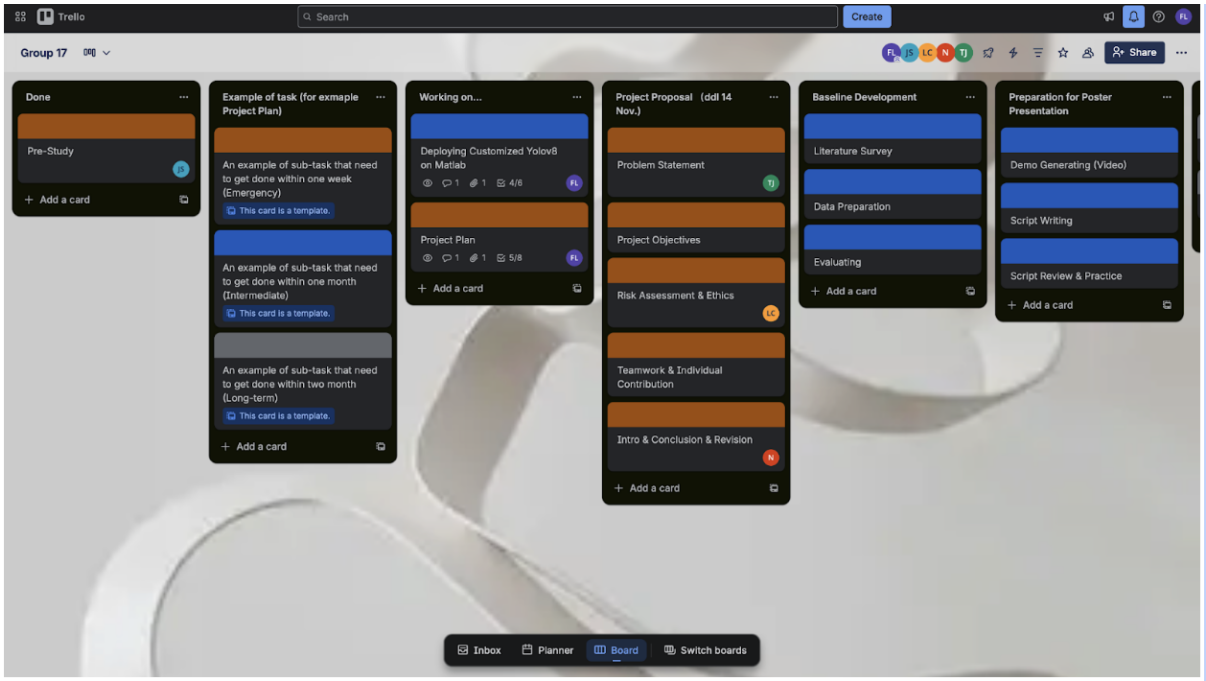


Figure 8: Trello board overview: list and colour-coded priorities for active work targets.

Simultaneously, as shown in 9 each task card incorporates sub-task checklists to break down and track minute steps. For instance, the checklist within the 'Deploying Customised Yolov8

on Matlab' card is illustrated below. By promptly updating the completion status of lists, team members can efficiently synchronise the project's actual progress. This enables us to flexibly adjust work priorities and resource allocation based on visualised urgency and progress data when facing technical challenges or time constraints, ensuring project objectives are met on schedule.

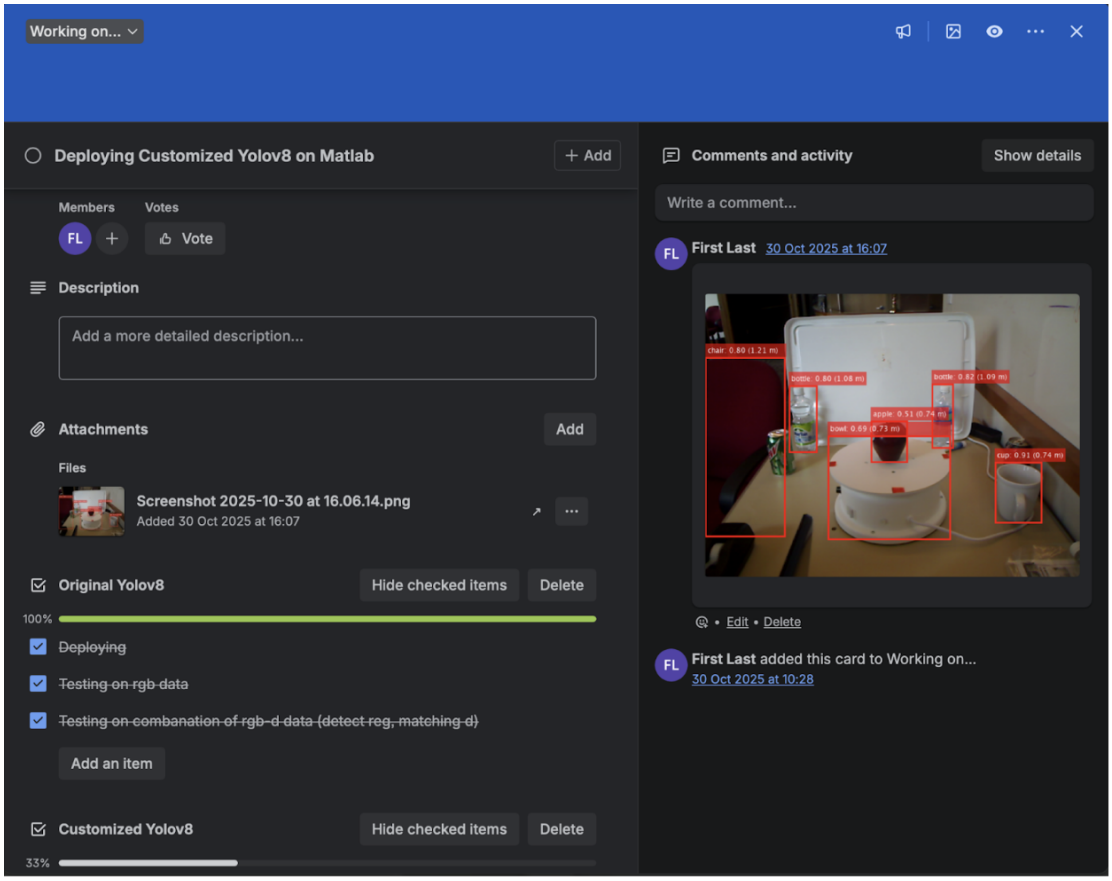


Figure 9: Example Trello card with member assignment, progress, and checklist for a specific task.

5 Risk Assessment and Ethics

5.1 Risk Assessment

5.1.1 Purpose

This section identifies the key risks associated with developing the RoboCup Autonomous Robot Manipulation system. The project integrates a UR5e robotic arm with MATLAB, ROS, and Gazebo for autonomous detection, localisation, and manipulation. Because of the technical complexity and collaborative structure, risk analysis focuses on performance, safety, and on time delivery. The aim is to detect issues early and apply correct actions throughout the project.

5.1.2 Risk Identification and Classification

Risks identified during the pre-study and integration stages fall into three categories:

Technical Risks. These include failures in perception, control, and communication, such as lighting variability affecting vision accuracy, depth sensor noise, ROS–MATLAB problems, and unstable IK, FK basically any mathematical caluclation. The simulation to reality mismatch may further impact motion accuracy and grasp stability.

Management Risks. Examples include uneven workload distribution, unclear responsibilities, version mismatches between MATLAB/ROS/Gazebo, and limited access to the UR5e robot or lab workspace, although this is really dependent on the KCL engineering department

Operational Risks. These arise during testing, such as unsafe robot motion, malfunction, hardware failure, data loss, or simulation crashes.

5.1.3 Risk Monitoring and Mitigation Plan

Risks are monitored weekly in a shared Risk Register (Table 5). High-priority risks are reviewed during supervision meetings. Mitigation strategies include GitHub version control, incremental system integration, validation scripts for simulation tests, and strict safety procedures during hardware use. If risks escalate, tasks and timelines are adjusted accordingly. Figure 10 shows the risk heat map.

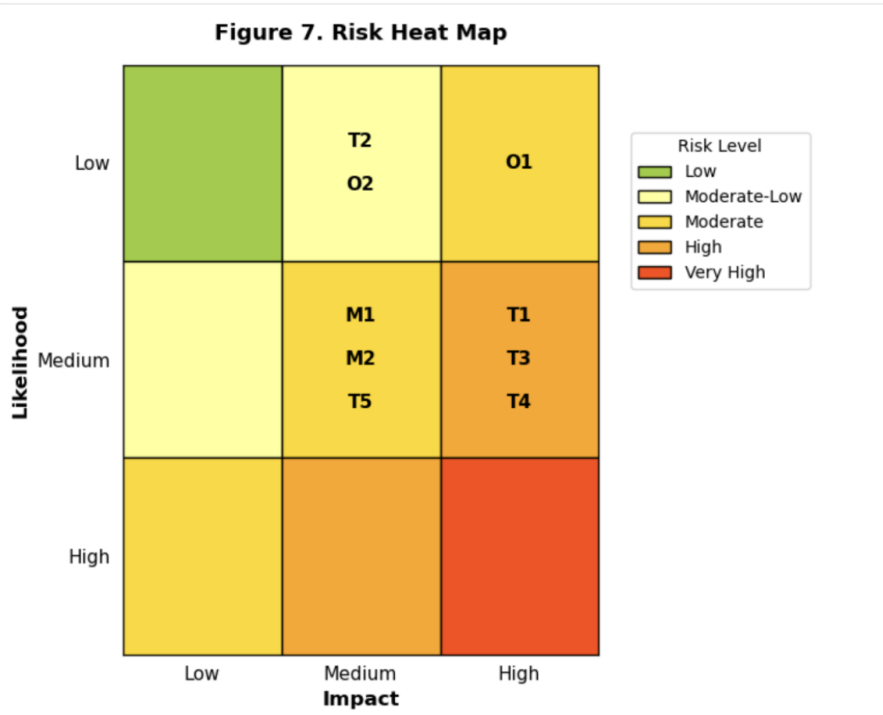


Figure 10: Risk heat map showing likelihood and impact levels for technical, managerial, and operational risks.

Table 5: Risk Register for RoboCup Autonomous Robot Manipulation Project

ID	Description	Likelihood	Impact	Mitigation Strategy	Owner
T1	Sensor noise or poor lighting reduces detection accuracy.	Medium	High	Apply filtering and calibration; test under varied lighting.	Vision Engineer
T2	ROS–MATLAB mismatch causes loss of commands.	Low	Medium	Add timeout and auto-reconnect; log errors.	Control Engineer
T3	Simulation to reality mismatch leads to unstable grasping.	Medium	High	Gradual sim-to-real transfer; maintain offset logs.	System Integrator
T4	Unstable IK or unsafe trajectories cause motion errors.	Medium	High	Joint-limit enforcement; trajectory validation.	Control Engineer
T5	Delays due to underestimated workload.	Medium	Medium	Use weekly milestones;	Project Manager
M1	Poor communication or unclear tasks slow integration.	Medium	Medium	Trello tracking; weekly meetings.	Project Manager
M2	Limited access to UR5e delays real testing.	Medium	Medium	Prioritise simulation; early booking; fallback plan.	Team Lead
O1	Unsafe motion or E-stop failure creates collision risk.	Low	High	Enforce safety zones; test E-stop each session.	Safety Officer
O2	Hardware failure or power surge causes data loss.	Low	Medium	Pre-operation checks; use surge protection; regular backups.	Lab Technician

5.2 Ethical and Sustainability Considerations

5.2.1 Safety, Data, and Fairness

All hardware experiments follow strict UR5e safety rules. Joint limits, motion constraints, and an active E-stop are used throughout testing. Only trained and authorised team members may operate the robot. No personal or human data is used; all datasets are open-source and documented. Fairness is maintained by testing with varied object types and lighting conditions.

5.2.2 Sustainability

Simulation-first development reduces energy use and hardware wear. Hardware testing is grouped to minimise resource waste. The project aligns with UN SDG 9 and SDG 12 by promoting efficient automation and responsible use of materials.

5.2.3 Ethical Approval and Compliance

The project does not involve human participants and does not require formal ethical approval. The team follows recognised Engineering Ethical Principles. If future work involves human–robot interaction or new data sources, formal approval will be sought.

6 Roles, Responsibilities & Individual Contribution Plan

This section describes each team member's role, their main responsibilities, and what they will deliver for the RoboCup ARM Challenge project. Every member contributes both technically and collaboratively to make sure the project is well coordinated, tested, and ready for competition.

Boyuan Ren – Support Developer and Documentation Assistant

Boyuan mainly responsible for bringing together our team's research findings and turning them into a clear report. **Main Deliverables:**

- Debugging and testing support for main algorithms.
- Checking data sources and references to support the evidence in the report.
- Help in preparing the final written report and figures.

Hongfu Chen – Perception Engineer and Task Coordinator

Hongfu integrates the SOTA neural network architecture to ensure the robots can perceive the target objects precisely and accurately, and ensures the team's technical workflow remains smooth and efficient by assigning tasks and monitoring task dependencies. **Main Deliverables:**

- Large Model Selection, Optimisation, Training and Validation
- Data cleaning and data preparation workflow.
- Integrating neural network models into existing framework.
- Assigning tasks cards and monitors task dependencies.
- Recording the notes of weekly meetings.

Jialin Tan – Motion Planning Engineer

Jialin focus on designing and improving the motion planning system in MATLAB/Simulink. Ensuring the robot arm can move efficiently and avoids collisions during grasping task. **Main Deliverables:**

- Motion planning model in MATLAB/Simulink.
- Checking test report on motion accuracy and reliability.
- Integrating the motion planner into other modules.

Jiawen Shen – System Integration and Testing Engineer

Jiawen is responsible for making sure all project parts (perception, planning, and control) work together smoothly in the Gazebo simulation. His work supports the "Integration and Testing" part of the project. This inlcudes a good connection between robot and MATALB ie ROS work **Main Deliverables:**

- ROS2 setup for communication between all modules.
- Testing procedures and automatic test scripts for simulation.
- Reports showing system performance and timing analysis.
- Clear documentation of how each part connects and exchanges data.

Jiazhe Hu – Grasp Pose Algorithm Engineer

Jiazhe designs the main algorithm that decides how and where the robot should grasp each object. Her work connects the perception results with the motion planning system to make sure the grasp is accurate and stable. **Main Deliverables:**

- MATLAB algorithm for grasp pose.
- Testing results from the Gazebo simulation.
- Adjusted parameters for better speed and accuracy.
- Technical documentation to explain the method clearly.

Leyan Chen – Manipulation Algorithm Developer

Leyan develops the robot's basic movement and control methods once the environment setup is ready. She works on inverse kinematics, motion paths, and grasping actions to make sure the robot performs stable and repeatable movements in both simulation and real operation. **Main Deliverables:**

- Inverse kinematics solver and movement scripts.
- Functions that convert perception data into robot grasping actions.
- Reports showing algorithm stability and improvement over time.

Naghim Ibragimov – Project Coordinator and Decision-Making Engineer

Naghim is responsible for keeping the whole project on track and making sure deadlines are met with the help of Hongfu. He manages communication with the university and RoboCup organisers and also contributes technically by improving the decision-making and evaluation stages of the robot's control system. **Main Deliverables:**

- Overall project coordination, report structure, and final editing.
- MATLAB algorithm for robot decision-making and path choice.
- Testing and reviewing robot performance in the final stage.
- Handling competition registration and contact with organisers in South Korea.

Collaboration and Workflow

The team uses Trello to track weekly progress and GitHub for version control and code sharing. Weekly meetings with supervisors help the team to find the right path to solution and also some planning for improvements. Every member's work contributes to the full MATLAB model tested in Gazebo, ensuring that the robot's perception, motion, and control systems work together as one complete design.

7 Conclusion

This proposal presents a clear and practical path that we will follow. From a simple concept of robot arm to participating in a proper tournament for the UR5e robot in the *RoboCup ARM Challenge*. The system design combines perception, planning, and control using MATLAB/Simulink, connected with a ROS–Gazebo simulation environment. This setup gives us exactly what we need, which is testing, repeatable results, and an easy transfer to the physical platform in King's College London's robotics lab. It also directly supports the SMART objectives and technical plans outlined earlier.

The initial study confirms that the project is feasible through three development stages: (1) Hard coded pick and place routine to check robot motion and kinematics, (2) vision-assisted sorting to link sensing with control, and (3) depth-based perception to estimate object pose and orientation for full autonomy. The main remaining challenge converting the estimated pose and orientation into a 6DOF position is well defined and achievable. The plan is to start with a simple lookup method for quick integration, then improve it into a what we call it a smart grasper, ie it will know the colours and orientation.

All essential resources and software are available, including MATLAB toolboxes (Robotics System, Computer Vision, and ROS), the official Gazebo virtual machine, and the UR5e manipulator for real life validation. The modular Simulink structure allows all sorts of testing and

integration, which reduces technical risk and simplifies development across the October–April schedule.

Risk management is ongoing and systematic. The team maintains a shared risk register with weekly reviews and clear task ownership. Mitigation steps are already active such as improving depth accuracy through calibration, gradually testing between simulation and reality, and verifying safe motion paths to prevent serious mistakes. Project management tools like GitHub and Trello help us to control our time, communication, and workload tracking, keeping the process well organised and preventing last-minute issues.

Ethics and sustainability are considered throughout. Simulation-first testing helps save energy and reduce wear on hardware, while all lab experiments follow strict safety and supervision procedures. The project is fully supports transparency by documenting all datasets, algorithms, and licences properly.

Going forward, the team will focus on completing the grasp pose generation stage and improving system stability under different conditions (eg object orientation). With a strong foundation and clear direction, the team is confident in delivering a reliable, sustainable, and high-performing robotic manipulation system for the RoboCup ARM Challenge.

References

[1] MathWorks. (2024) *Pick and Place Robot Manipulator with Point Clouds and RRT.* Available at: https://www.mathworks.com/help/robotics/ug/pick-and-place-gazebo-with-point-clouds-and-rrt.html (Accessed 11 November 2025).

[2] RoboCup Federation. (2025) *Autonomous Robot Manipulation (ARM) Challenge – Participation Guide.* Available at: https://www.robocup.org/news/173 (Accessed 11 November 2025).

[3] MathWorks. (2025) *RoboCup ARM Challenge Student Competition.* Available at: https://uk.mathworks.com/academia/student-competitions/robocup.html (Accessed 11 November 2025).

[4] King's College London News. (2024) *Competing in the RoboCup ARM Challenge: Challenges and Triumphs.* Available at: https://www.kcl.ac.uk/competing-in-the-robocup-arm-challenge-challenges-and-triumphs (Accessed 11 November 2025).

[5] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path Aggregation Network for Instance Segmentation," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8759–8768, June 2018.

[6] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2117–2125, 2017.

[7] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10781–10790, June 2020.

[8] J. Chen et al., "Run, Don't Walk: Chasing Higher FLOPs for Faster Neural Networks," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1157–1166, June 2023. doi:10.1109/CVPR52729.2023.01157.

[9] C. Ma et al., "YOLO-UAV: Object Detection Method for Unmanned Aerial Vehicles," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Appendix

```
function pickDrop(poseAndGripwidth, bin)
    global gripAct gripGoal jointSub initialIKGuess

    poseAndGripwidth

    % reset initialIKGuess to current config
    jointMsg = receive(jointSub);
    currentJointStates = jointMsg.Position;
    for j = 1:numel(initialIKGuess)
        initialIKGuess(j).JointPosition = currentJointStates(j);
    end

    % go to pre-grasp
    targetPose = poseAndGripwidth(1:6);
    targetPose(3) = poseAndGripwidth(3)+0.2;
    move(targetPose);

    % go to grasp
    targetPose = poseAndGripwidth(1:6);
    move(targetPose);

    % grasp object
    gripGoal=packGripGoal(poseAndGripwidth(7),gripGoal);
    sendGoal(gripAct,gripGoal);
    pause(10)

    % go to pre-grasp again
    targetPose = poseAndGripwidth;
    targetPose(3) = poseAndGripwidth(3)+0.2;
    move(targetPose)

    % go to bin
    if bin == "blue"
        binPose = [0.5 -0.3 0.3 pi/2 pi 0];
    elseif bin == "green"
        binPose = [-0.5 -0.3 0.3 -pi/2 pi 0];
    end
    move(binPose);

    % drop
    gripGoal=packGripGoal(0,gripGoal);
    sendGoal(gripAct,gripGoal);
    pause(10)
end
```

The process of grabbing and placing

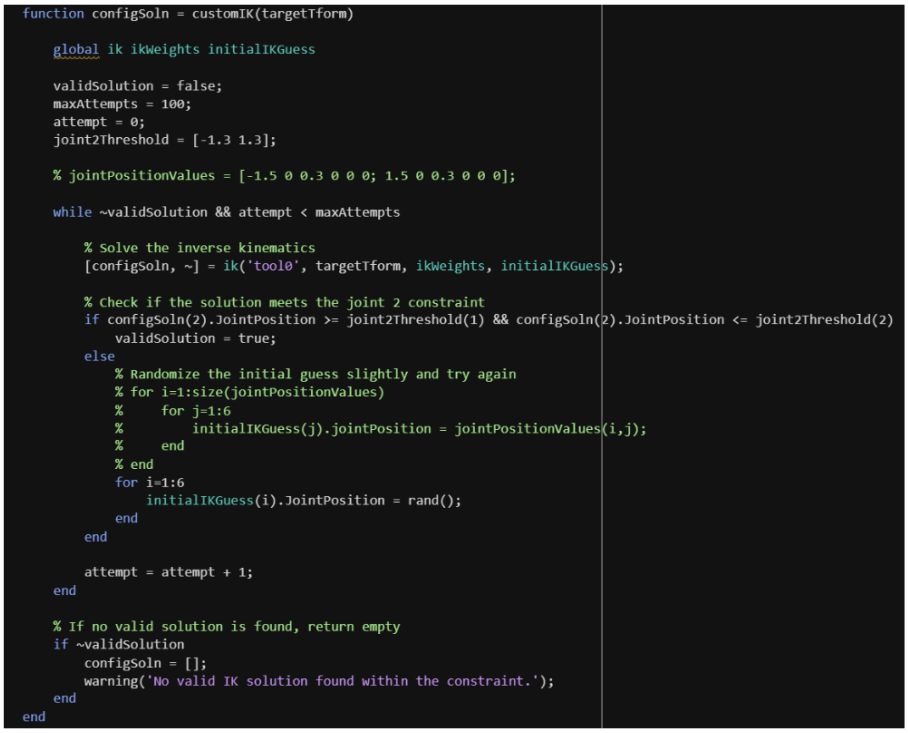
```
% for pose - x: right, y: straight, z: up. enter as [x y z rotz roty rotx]

function move(targetPose)
    global trajAct trajGoal

    % if targetOrientation == "x"
    %     ori = [0 1 0 0];
    % elseif targetOrientation == "y" && targetLocation(1) >= 0
    %     ori = [0 0.71 -0.71 0];
    % elseif targetOrientation == "y" && targetLocation(1) < 0
    %     ori = [0 0.71 0.71 0];
    % else
    %     ori = [0.71 -0.71 0 0];
    % end

    tform = eye(4);
    tform(1:3, 1:3) = eul2rotm(targetPose(4:6));
    tform(1:3,4) = targetPose(1:3)';
    %[configSoln, ~] =ik('tool0',tform,ikWeights,initialIKGuess);
    configSoln = customIK(tform);
    trajGoal = packTrajGoal(configSoln,trajGoal);
    sendGoal(trajAct,trajGoal);
    pause(15)

end
```



```
function configSoln = customIK(targetTform)

    global ik ikWeights initialIKGuess

    validSolution = false;
    maxAttempts = 100;
    attempt = 0;
    joint2Threshold = [-1.3 1.3];

    % jointPositionValues = [-1.5 0 0.3 0 0 0; 1.5 0 0.3 0 0 0];

    while ~validSolution && attempt < maxAttempts

        % Solve the inverse kinematics
        [configSoln, ~] = ik('tool0', targetTform, ikWeights, initialIKGuess);

        % Check if the solution meets the joint 2 constraint
        if configSoln(2).JointPosition >= joint2Threshold(1) && configSoln(2).JointPosition <= joint2Threshold(2)
            validSolution = true;
        else
            % Randomize the initial guess slightly and try again
            % for i=1:size(jointPositionValues)
            %     for j=1:6
            %         initialIKGuess(j).jointPosition = jointPositionValues(i,j);
            %     end
            % end
            for i=1:6
                initialIKGuess(i).JointPosition = rand();
            end
        end

        attempt = attempt + 1;
    end

    % If no valid solution is found, return empty
    if ~validSolution
        configSoln = [];
        warning('No valid IK solution found within the constraint.');
    end
end
```

Figure 12: Appendix Image 2

Inverse kinematics solution with constraints

```
function xyz_uv = get_xyz_uv_from_depthimg()
    global depthSub tfTree

    depthImageRaw = receive(depthSub);
    depthImageMat = rosReadImage(depthImageRaw);
    K = [554.3827128226441 0.0 320.5; 0.0 554.3827128226441 240.5; 0.0 0.0 1.0];
    xyz_uv = zeros(480,640,3);

    for u=1:size(depthImageMat,2)
        for v=1:size(depthImageMat,1)
            pixel = [u,v,1];
            xy1 = K\pixel';
            xyz = xy1*(depthImageMat(v,u));
            xyz_uv(v,u,:)=xyz;
        end
    end

    % Above xyz_uv needs to be transformed
    tfTree = rostf;
    tform = getTransform(tfTree, 'base', 'camera_depth_link');
    translation = [tform.Transform.Translation.X, ...
                   tform.Transform.Translation.Y, ...
                   tform.Transform.Translation.Z];
    quaternion = [tform.Transform.Rotation.W, ...
                  tform.Transform.Rotation.X, ...
                  tform.Transform.Rotation.Y, ...
                  tform.Transform.Rotation.Z];
    rotMatrix = quat2rotm(quaternion);
    tform1 = [rotMatrix, translation'; 0 0 0 1];
    tform2 = [0 1 0 0; -1 0 0 0; 0 0 1 0; 0 0 0 1];
    tform = tform2 * tform1;
    rot = tform(1:3, 1:3);
    trans = tform(1:3,4)';
    geometricTform = rigidtform3d(rot, trans);

    xyz_uv = reshape(xyz_uv,[],3);
    xyz_uv = transformPointsForward(geometricTform,xyz_uv);
    xyz_uv = reshape(xyz_uv,480,640,3);
```

Convert depth maps to 3D coordinates

Figure 13: Appendix Image 3