

AUTOMATED WAREHOUSE SORTING SYSTEM USING MACHINE VISION & ROBOT
MANIPULATOR

AHMED ABDULGADER SALIM ASSAGAF

UNIVERSITI TEKNOLOGI MALAYSIA



**UNIVERSITI TEKNOLOGI MALAYSIA
DECLARATION OF THESIS**

Author's full name : AHMED ABDULGADER SALIM ASSAGAF
 Matric No. : A21EE0307 Academic Session : 24/25-02
 Date of Birth : 11/12/2002 UTM Email : SALIM@GRADUATE.UTM.MY
 Thesis Title : Automated Warehouse Sorting System Using Machine Vision & Robot Manipulator

I declare that this thesis is classified as:



OPEN ACCESS

I agree that my report to be published as a hard copy or made available through online open access.



RESTRICTED

Contains restricted information as specified by the organization/institution where research was done.
(The library will block access for up to three (3) years)



CONFIDENTIAL

Contains confidential information as specified in the Official Secret Act 1972)

(If none of the options are selected, the first option will be chosen by default)

I acknowledged the intellectual property in the thesis belongs to Universiti Teknologi Malaysia, and I agree to allow this to be placed in the library under the following terms :

1. This is the property of Universiti Teknologi Malaysia
2. The Library of Universiti Teknologi Malaysia has the right to make copies for the purpose of only.
3. The Library of Universiti Teknologi Malaysia is allowed to make copies of this thesis for academic exchange.

Signature :

Signature of Student:

Full Name AHMED ABDULGADER SALIM ASSAGAF

Date : 02/07/2025

Signature of Supervisor I:

Approved by Supervisor(s)

Signature of Supervisor II

Full Name of Supervisor I


Full Name of Supervisor II

Date :

Date :

NOTES : If the thesis is CONFIDENTIAL or RESTRICTED, please attach with the letter from the organization with period and reasons for confidentiality or restriction

“I hereby declare that I have read this final year project report and in my opinion this final year project report is sufficient in terms of scope and quality for the award of the degree of Bachelor of Engineering”

Signature	:	
Name of Supervisor	:	DR.MOHD SAIFUL AZIMI BIN MAHMUD
Date	:	JULY 2, 2025

Pengesahan Peperiksaan

Tesis ini telah diperiksa dan diakui oleh:

Nama dan Alamat Pemeriksa Luar :

Nama dan Alamat Pemeriksa Dalam :

Nama Penyelia Lain (jika ada) :

Disahkan oleh Timbalan Pendaftar di Fakulti:

Tandatangan :

Nama :

Tarikh :

AUTOMATED WAREHOUSE SORTING SYSTEM USING MACHINE VISION & ROBOT MANIPULATOR

AHMED ABDULGADER SALIM ASSAGAF


A final year project report submitted in partial fulfilment of the
requirements for the award of the degree of
Bachelor of Engineering

Faculty of Electrical Engineering
Universiti Teknologi Malaysia

JULY 2025

DECLARATION

I declare that this final year project report entitled “*Automated Warehouse Sorting System Using Machine Vision & Robot Manipulator*” is the result of my own research except as cited in the references. The final year project report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature : 
Name : AHMED ABDULGADER SALIM ASSAGAF
Date : JULY 2, 2025

ACKNOWLEDGEMENT

This thesis would not have been accomplished without the assistance and guidance from so many. First and foremost, I would like to express my deep gratitude to my supervisor, Dr. Mohd Saiful Azimi Mahmud, for all the encouragement given throughout this thesis, constructive suggestions, and instructive guidance that he offered me during my work. His competence and mentorship were of ultimate importance for accomplishing this work.

I am also deeply in debt to the Universiti Teknologi Malaysia, which provided such resources and facilities that made the research at hand possible. The encouraging academic atmosphere at UTM has facilitated so much learning along with my personal development.

Last but not least, I would like to extend my gratitude to my family for the unconditional love, patience, and encouragement in constant ways. Their belief in me has been a strength during times when things were difficult. Without their support, this achievement would not have been possible.

ABSTRACT

The rapid growth of e-commerce has driven the need for faster and more accurate warehouse sorting systems, exposing the inefficiencies of traditional manual processes which are often slow, error-prone, and labor-intensive. To address these limitations, this project develops an autonomous robotic sorting prototype using a fixed 4-DOF OpenManipulator-X robotic arm integrated with a USB camera and controlled via ROS (Robot Operating System). The objective is to design a vision-guided pick-and-place system that detects, identifies, and sorts items in real time using AR markers. The system uses the `ar_track_alvar` package for AR detection, with item localization handled in 2D (X, Y) space, and a fixed Z-height for stable grasping. The methodology includes camera calibration, real-time marker tracking, inverse kinematics for movement planning, and ROS-based control integration. The prototype was tested with three unique items, achieving a 100% pick-and-place success rate and an average cycle time of 20 seconds per item, significantly outperforming related systems such as Pan et al. (2020), which reported a 53.25-second average. These results highlight the system's potential as a low-cost, scalable solution for intelligent warehouse automation.

ABSTRAK

Pertumbuhan pesat e-dagang telah meningkatkan keperluan terhadap sistem penyusunan gudang yang lebih pantas dan tepat, sekali gus mendedahkan kelemahan proses manual tradisional yang lambat, terdedah kepada ralat, dan memerlukan tenaga kerja yang tinggi. Untuk menangani masalah ini, projek ini membangunkan prototaip sistem penyusunan robotik automatik menggunakan lengan robotik tetap OpenManipulator-X 4-DOF yang digabungkan dengan kamera USB dan dikawal melalui ROS (Robot Operating System). Objektif projek adalah untuk mereka bentuk sistem pick-and-place berpanduan penglihatan yang mampu mengesan, mengenal pasti, dan menyusun objek secara masa nyata menggunakan penanda AR. Sistem ini menggunakan pakej `ar_track_alvar` untuk pengesanan penanda, dengan pengesanan lokasi objek dilakukan dalam ruang 2D (X, Y) dan ketinggian Z yang tetap bagi memastikan kestabilan semasa mencengkam. Metodologi merangkumi penentuan kamera, penjejakan masa nyata, perancangan laluan kinematik songsang, dan integrasi kawalan berasaskan ROS. Prototaip ini diuji ke atas tiga objek unik dan berjaya mencapai kadar kejayaan 100% untuk operasi ambil dan letak, dengan purata masa kitaran sebanyak 20 saat setiap item, jauh lebih pantas berbanding sistem Pan et al. (2020) yang mencatatkan purata 53.25 saat. Keputusan ini membuktikan potensi sistem sebagai penyelesaian automasi gudang pintar yang berskala dan mampu milik.

TABLE OF CONTENTS

	TITLE	PAGE
	DECLARATION	iii
	ACKNOWLEDGEMENT	v
	ABSTRACT	vi
	ABSTRAK	vii
	TABLE OF CONTENTS	viii
	LIST OF TABLES	xi
	LIST OF FIGURES	xii
CHAPTER 1	INTRODUCTION	1
1.1	Background of the Problem	1
1.2	Problem Statement	2
1.3	Research Goal	3
1.4	Objectives	3
1.5	Scopes of Work	3
CHAPTER 2	LITERATURE REVIEW	5
2.1	Introduction	5
2.2	Introduction to Machine Vision	5
2.2.1	Definition and Significance	5
2.2.2	Industrial Applications	6
2.2.3	Hardware and Software Tools	6
2.3	Object Detection and Recognition	7
2.3.1	Conceptualization and Methodologies	7
2.3.2	Industrial Applications	8
2.3.3	Comparison of Algorithms	8
2.4	YOLO Algorithm	8
2.4.1	General Overview and Evolution	8
2.4.2	Benefits for Real-time Applications	9
2.4.3	Implementation on Constrained Hardware	9

2.5	Image Processing Using OpenCV	9
2.5.1	Introduction to OpenCV	9
2.5.2	Features and Industrial Applications	9
2.5.3	Integration with Other Tools	10
2.6	Robotic Arms	10
2.6.1	Definition and Importance	10
2.6.2	Applications in Industry	10
2.6.3	Advantages and Limitations	11
2.7	OpenManipulator-X Robotic Arm	11
2.7.1	Structure and Configuration	11
2.7.2	Applications	12
2.7.3	Advantages and Limitations	12
2.8	Kinematics of 4-DOF Robotic Arms	12
2.8.1	Forward Kinematics	13
2.8.2	Inverse Kinematics	13
2.8.3	Applications and Tools	13
2.8.4	Advantages and Challenges	13
2.9	AR Markers and <code>ar_track_alvar</code> Library	14
2.9.1	Technical Functionality	14
2.9.2	Applications	14
2.9.3	Advantages and Limitations	14
2.10	Related Work	15
2.10.1	Summary of Limitations	18
CHAPTER 3	RESEARCH METHODOLOGY	21
3.1	Introduction	21
3.2	Project Flow	22
3.2.1	Objective 1: Real-Time Machine Vision System	24
3.2.2	Objective 2: Robotic Arm Motion Planning and Execution	25
3.2.3	Objective 3: System Integration and Testing	25
3.3	System Architecture & Flow	26
3.3.1	System Architecture	26
3.3.2	System flow	31

3.4	Cost of Project	33
3.5	Conclusion	33
CHAPTER 4	RESULTS AND DISCUSSION	35
4.1	Objective 1: Machine Vision System Performance	35
4.2	Objective 2: Robotic Motion Execution and Inverse Kinematics	37
4.3	Objective 3: System Integration and Overall Performance	41
4.4	Comparative Analysis	43
4.4.1	Comparison Table	43
4.4.2	Cycle Time Chart	43
4.5	Summary and Insights	44
4.6	Discussion of Limitations	44
CHAPTER 5	CONCLUSION	45
5.1	Introduction	45
5.2	Key Findings and Contributions	45
5.3	Broader Implications	45
5.4	Challenges and Limitations	46
5.5	Future Work	46
5.6	Conclusion	47
REFERENCES		49

LIST OF TABLES

TABLE NO.	TITLE	PAGE
Table 2.1	Summary of related work	17
Table 3.1	Project Expenditure	33
Table 4.1	ROS Topics and Rates Used in Robotic Arm Operation	38
Table 4.2	Drop-off Location Mapping Using Joint Space (in Radians)	40
Table 4.3	Comparison of Autonomous Object Sorting Systems	43

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE
Figure 1.1	Summary of challenges in manual sorting	2
Figure 2.1	A simple block diagram for a typical vision system operation	6
Figure 2.2	Comparison of SSD, Faster RCNN, and YOLOv4, v5 and v7 [14]	7
Figure 3.1	Project Flowchart	22
Figure 3.2	FYP 1 Gantt Chart	23
Figure 3.3	FYP 2 Gantt Chart	24
Figure 3.4	Openmanipulator-x	27
Figure 3.5	System Architecture for Automated Sorting	29
Figure 3.6	System Flowchart for Automated Object Sorting	31
Figure 4.1	Item position and orientation from Rostopic "ar_pose_marker"	35
Figure 4.2	Checkerboard pattern for calibration	36
Figure 4.3	Camera Calibration parameters	36
Figure 4.4	Items visualization in RViz	37
Figure 4.5	Gripper pose in XYZ	39
Figure 4.6	Gripper pose in joint space	39
Figure 4.7	Drop-off positions	40
Figure 4.8	Final prototype	41
Figure 4.9	Control Terminal	42
Figure 4.10	Successful Demo	42

Figure 4.11 Comparison of average cycle time per item across three sorting systems. My system demonstrates the fastest execution time, due to efficient AR marker detection and optimized motion strategy.

43

CHAPTER 1

INTRODUCTION

1.1 Background of the Problem

The rapid growth of e-commerce has changed the logistics and warehousing industry, creating a need for faster and more accurate order processing. Warehouses have evolved from simple storage areas into busy order fulfillment centers, where traditional manual sorting methods often fall short. Research shows that manual sorting is inefficient, error-prone, and costly, making it unsuitable for the fast-paced demands of global supply chains [1].

Manual sorting, where workers handle items by hand, requires a lot of labor and is prone to mistakes. Studies reveal error rates of 1–5% in manual systems, which can lead to thousands of incorrect shipments in large warehouses. These errors increase operational costs due to reprocessing, returns, and additional labor hours [2]. Moreover, a human worker can process only 100–200 items per hour, which is far too slow for e-commerce giants that handle millions of orders daily [3].

This system also takes a toll on workers. Repetitive tasks and physical strain cause job dissatisfaction, leading to high employee turnover rates, which have exceeded 46% in the warehousing sector. As labor shortages grow and wages rise, this model becomes harder to sustain [4]. Additionally, manual systems cannot easily scale to meet the needs of modern warehouses, which must sort a wide variety of items by size, color, or unique identifiers like QR codes [5].

Although automation can solve these problems, many companies hesitate to adopt it because of the high initial costs and concerns about disrupting their workflows [6]. However, with rising demand for fast, accurate deliveries, it is clear that more efficient and worker-friendly logistics systems are urgently needed [7].



Figure 1.1 Summary of challenges in manual sorting

A summary of the challenges faced in manual sorting process is shown in figure 1.1, to summarize this section.

1.2 Problem Statement

Warehouses today face significant challenges due to outdated manual sorting systems. These processes are prone to errors, slow down operations, and place unnecessary physical strain on workers. With the rising demands of e-commerce and increasingly complex global supply chains, the limitations of traditional sorting methods have become more apparent. Manual systems are not only inefficient but also difficult to scale, especially in environments that require consistent, high-throughput processing.

To address these challenges, fixed robotic arms have emerged as a practical solution for automating repetitive and structured sorting tasks. A robotic arm can be programmed to perform precise pick-and-place operations with minimal human supervision, significantly reducing labor dependency and human error. In a structured workspace—such as a lab-scale

or small-to-medium warehouse setup—a stationary robotic arm offers high repeatability, low maintenance, and the ability to work continuously with consistent accuracy.

Given these advantages, there is a growing need to develop low-cost, vision-guided robotic sorting systems using fixed robotic manipulators. Such systems can bridge the gap between manual labor and fully autonomous logistics, offering a scalable and efficient alternative for improving productivity in warehouse environments.

1.3 Research Goal

The goal of this research is to design and develop a prototype for an automated sorting system that enhances operational efficiency in warehouse environments. By utilizing machine vision, the system will track and sort items with greater accuracy and speed, reducing human error and operational delays. This prototype aims to offer a scalable, adaptable solution that improves the overall productivity of warehouse operations, addressing the challenges posed by the increasing complexity of modern logistics and the growing demands of e-commerce.

1.4 Objectives

The project aims to achieve the following objectives:

- a) To develop a real-time machine vision system to detect and decode objects with AR markers accurately.
- b) To program the robotic arm kinematics for efficient object sorting.
- c) To integrate the robotic arm and machine vision into a cohesive, synchronized sorting system.

1.5 Scopes of Work

The scope of work for this project includes the following tasks:

- a) Study the principles of robotic arm control, focusing on using ROS to achieve precise and reliable object manipulation with the OpenManipulator-X.
- b) Implement AR marker detection and pose estimation in 2D space (X,Y) using ar-track-alvar library with a 2D camera placed above the workspace.
- c) Develop a unified framework to enable the robotic arm to efficiently identify, pick, and place objects of the same height based on their AR marker IDs.
- d) Integrate the camera, AR marker detection, and robotic arm control to perform automated sorting of the 3 items with high accuracy and robustness in a planar setting.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter presents an overview of the basic concepts, methodologies, and technologies on which this work is based. Definitions of important terms such as machine vision, robotic manipulation, and QR code-based sorting are given and put into context. A related works review about automated sorting systems and their strengths, limitations, and possible improvements is also presented. This review provides an overall understanding of the prevailing research landscape, thereby making it easier to identify the gaps and justify the approach of this project.

2.2 Introduction to Machine Vision

2.2.1 Definition and Significance

Machine vision is the application of imaging technologies and computational algorithms to enable automation in visual tasks, such as inspection and analysis. According to Golnabi & Asadpour [8], machine vision systems are engineered to capture, process, and interpret visual information so that machines can perform tasks traditionally requiring human vision. The system combines hardware components, such as cameras and illumination, with image processing software, simulating human visual functions and hence their relevance in a wide range of industrial applications.

Machine vision has become very important in industries. It allows for faster, more accurate, and consistent processes, which are very vital in automation. Alonso et al. [9] note that machine vision is one of the key elements of Industry 4.0, where the concept of smart manufacturing and real-time decision-making is realized through IoT connectivity. A key factor

in evaluating the effectiveness of machine vision would be its ability to detect defects, some advanced systems achieving detection rates even higher than 95%, making it indispensable within the field of industrial quality control [10].

2.2.2 Industrial Applications

Machine vision has been applied in various fields. In logistics, for instance, it helps automate the sorting of parcels by reading barcodes or QR codes. It is also used in the automotive industry to ensure accurate assembly by inspecting and aligning components [11]. Machine vision is also applied in predictive maintenance, where it helps in the identification of wear and tear of equipment before failures, hence reducing downtime [9].

2.2.3 Hardware and Software Tools

The components used in machine vision mainly include industrial cameras with resolutions ranging from 5 MP to 50 MP, along with dedicated lighting systems. In terms of software, OpenCV and MATLAB applications are preferred for image analysis. Most significantly, OpenCV is highly used because it supports more than 2,500 optimized algorithms, thereby making possible functionalities such as object detection and edge recognition [12].

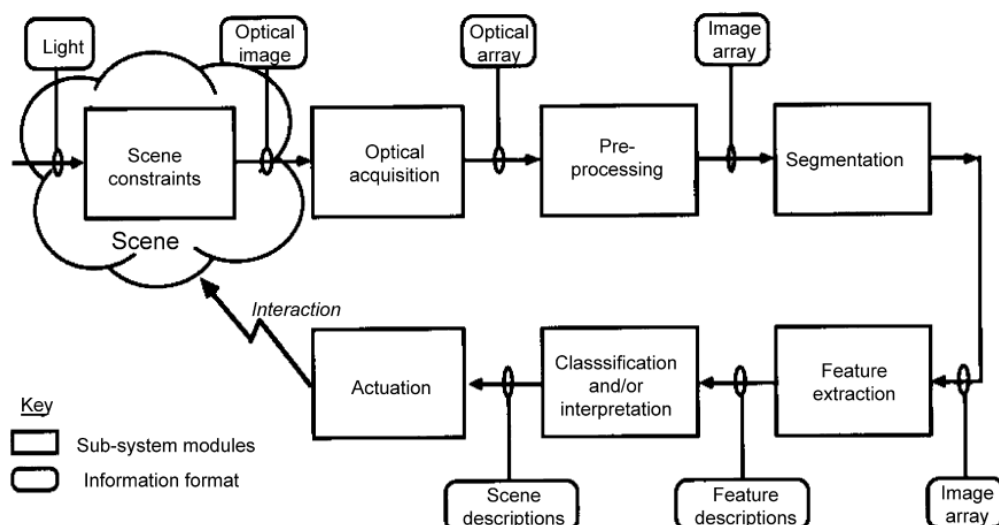


Figure 2.1 A simple block diagram for a typical vision system operation

2.3 Object Detection and Recognition

2.3.1 Conceptualization and Methodologies

Object detection refers to locating and identifying the objects in the image or video, whereas recognition deals with the classification of identified objects. Chandan et al. [12] mention that in order to detect any object, this process is performed through an algorithm-based method analyzing patterns from any visual data for the system to successfully distinguish an object from its background. On the other hand, recognition takes a further step and performs classification on identified objects into a predefined class label.

More traditional techniques, such as edge detection and template matching, were based on hand-designed features but struggled when applied to complex environments. In contrast, modern AI-based techniques, especially CNNs, have overcome those challenges by automatically learning features from data. Some of the most popular and widely used algorithms developed in recent times include SSD (Single Shot Multibox Detector), Faster R-CNN, and YOLO. Faster R-CNN has more accurate results, whereas YOLO is faster because it has the capability of processing an entire image in one pass [13].

Model	Class	Precision (%)	Recall (%)	mAP:0.5 (%)	FPS
SSD	Non-broken	89.3	80.4	75.5	22
	Broken	73.1	68		
Faster RCNN	Non-broken	90.8	83.2	79.5	16
	Broken	80.0	77.4		
YOLOv4	Non-broken	97.4	92.2	89.4	31
	Broken	91.2	88		
YOLOv7	Non-broken	99.1	94.8	94.9	37
	Broken	96.8	93.6		
YOLOv5_add	Non-broken	98.9	95.6	95.2	45
	Broken	96.7	93.5		

Figure 2.2 Comparison of SSD, Faster RCNN, and YOLOv4, v5 and v7 [14]

2.3.2 Industrial Applications

Object detection is performed in warehouse automation, where robots with detection systems can sort over 100 items in a minute, significantly increasing operational efficiency [15]. In autonomous vehicles, detection systems identify obstacles, road signs, and pedestrians for safe navigation. The accuracy of detection in such systems is normally above 90% in controlled environments [16].

2.3.3 Comparison of Algorithms

Among the object detection algorithms, YOLO has been recognized for its ability to process up to 45 frames per second (FPS); this happens to be much faster compared to Faster R-CNN at 16 FPS and SSD at 22 FPS [14]. While the latter has higher accuracy, the real-time capabilities of YOLO make it the most appropriate for applications requiring fast responses—like surveillance or robotics.

2.4 YOLO Algorithm

2.4.1 General Overview and Evolution

The YOLO algorithm is revolutionary in the field of object detection. In place of performing object detection in multiple stages, YOLO treats it as a single regression problem that simultaneously predicts bounding boxes and class probabilities. Ullah [17] describes YOLO as fast and efficient, where the method processes the entirety of an image in one go, thus obtaining faster results compared to traditional methods.

YOLO has gone through many versions, each better than the last: YOLOv3 introduced multi-scale detection to improve performance with small objects. YOLOv4 improved both accuracy and speed by adopting CSPDarknet and self-adversarial training. YOLOv8 further improved feature extraction and achieved top performance on large datasets [16].

2.4.2 Benefits for Real-time Applications

Speed is one of the biggest strengths of YOLO. It comes in handy, for example, in real-time applications like video surveillance, where it detects multiple objects in a crowd at a speed of more than 30 FPS. This ability makes YOLO one of the important algorithms for robotics and other dynamic systems that need real-time performance [17].

2.4.3 Implementation on Constrained Hardware

YOLO is not only confined to high-performance computing systems; it can also be done on low-resource hardware such as the Raspberry Pi or Jetson Nano. For example, Ponika et al. [16] showed that YOLO could perform real-time detection at up to 20 frames per second on a Raspberry Pi, making it a very affordable option for small business applications.

2.5 Image Processing Using OpenCV

2.5.1 Introduction to OpenCV

OpenCV, which is short for Open Source Computer Vision Library, is a substantial repository of materials aimed at image and video analysis. As Mittal et al. [11] state, OpenCV is incredibly versatile, as it offers modules for edge detection, feature extraction, and object tracking. What's more is that it is compatible with the implementation of deep learning and GPU acceleration, thus it is suitable in a wide variety of applications.

2.5.2 Features and Industrial Applications

OpenCV serves multiple basic functions, including QR code detection, motion tracking, and image preprocessing. In logistics, for instance, OpenCV assists in the decoding of barcodes, hence improving the efficiency and accuracy of sorting and supply chain management. Shukla et al. [13] showed the usefulness of OpenCV in monitoring social distancing by analyzing spatial dynamics between people in video feeds.

2.5.3 Integration with Other Tools

OpenCV can be successfully integrated with other technologies like YOLO to extend its capability. Akbulut and Khalaf [15] integrated YOLO and OpenCV in developing a real-time arm detection system, which was particularly aimed for security-related applications, hence proving the efficiency of combining the tools to achieve high-performance results [15].

2.6 Robotic Arms

2.6.1 Definition and Importance

Robotic arms are programmable mechanical devices designed to emulate the functions of a human arm, capable of tasks such as assembly, welding, and material handling. They are integral to industrial automation, enhancing productivity, precision, and safety. Robotic arms, for instance, can achieve a positioning accuracy of ± 0.1 mm, significantly reducing defects and increasing production rates [18]. The global market for industrial robotic arms is projected to reach \$14.5 billion by 2026, driven by advancements in automation technologies [18].

2.6.2 Applications in Industry

Automotive Manufacturing: Robotic arms are employed for tasks such as welding, painting, and assembly, improving efficiency and consistency. For example, in car manufacturing, robotic arms can complete a weld in approximately 1.3 seconds, enhancing production speed [16]. **Electronics Assembly:** In the electronics industry, robotic arms handle delicate components with high precision, assembling devices like smartphones and computers. They can place components with an accuracy of ± 0.02 mm, ensuring product quality [11]. **Pharmaceuticals:** Robotic arms are used in the pharmaceutical sector for tasks such as dispensing, sorting, and packaging medications, maintaining hygiene standards and reducing human error. They can operate in sterile environments, handling up to 200 items per minute [13]. **Food and Beverage:** In food processing, robotic arms perform tasks like sorting, packaging, and palletizing, increasing throughput and maintaining hygiene. They can handle up to 150 picks per minute, enhancing operational efficiency [15].

2.6.3 Advantages and Limitations

- **Advantages:**
 - a) Increased Productivity: Robotic arms can operate continuously without fatigue, leading to higher output [18].
 - b) Enhanced Precision: They perform tasks with high accuracy, reducing errors and waste.
 - c) Improved Safety: Robotic arms can handle hazardous tasks, reducing the risk of workplace injuries.
- **Limitations:**
 - a) High Initial Costs: The upfront investment for robotic arms can be substantial, potentially limiting adoption for small and medium-sized enterprises.
 - b) Complex Integration: Implementing robotic arms into existing workflows may require significant changes and technical expertise.
 - c) Maintenance Requirements: Regular maintenance is necessary to ensure optimal performance and longevity.

2.7 OpenManipulator-X Robotic Arm

The OpenManipulator-X is an open-source, ROS-compatible 4-DOF robotic arm developed by ROBOTIS. It is modular, lightweight, and well-integrated with simulation tools such as Gazebo and MoveIt, which makes it ideal for education, prototyping, and research applications [19].

2.7.1 Structure and Configuration

The arm comprises Dynamixel servos that offer precise control and feedback. It follows a serial-link configuration and can be modeled using Denavit–Hartenberg (DH) parameters, enabling accurate kinematic computations [20].

2.7.2 Applications

- **Educational Robotics:** Used for teaching motion planning, kinematics, and ROS pipelines.
- **Research Platforms:** Employed in HRI experiments, agricultural automation, and pick-and-place simulations [21].
- **Military Engineering:** Modified versions have been tested in field simulations requiring higher DoFs [22].

2.7.3 Advantages and Limitations

Advantages:

- a) Open-source with full ROS integration.
- b) Accurate kinematics for precise motion.
- c) Lightweight and modular.

Limitations:

- a) Limited payload capacity.
- b) Fewer degrees of freedom than industrial arms.

2.8 Kinematics of 4-DOF Robotic Arms

Kinematics deals with the motion of systems without considering forces. In robotic arms, this includes computing the position and orientation of the end effector relative to joint angles (forward kinematics) or solving for joint angles given a desired end-effector pose (inverse kinematics) [23].

2.8.1 Forward Kinematics

Using the Denavit–Hartenberg method, forward kinematics creates a series of homogeneous transformation matrices that calculate the pose of each joint with respect to the base frame [24].

2.8.2 Inverse Kinematics

Inverse kinematics (IK) is more complex, often involving the solution of nonlinear equations. For 4-DOF systems, analytical methods exist, but numerical or optimization approaches are used when solutions are non-unique or the model is complex [25].

2.8.3 Applications and Tools

- **Trajectory Planning:** Computing joint angles to follow desired paths.
- **Simulations:** ROS MoveIt, MATLAB, and custom solvers.

2.8.4 Advantages and Challenges

Advantages:

- a) Enables accurate control of robotic motion.
- b) Foundation for trajectory planning.

Challenges:

- a) Multiple or no IK solutions for certain poses.
- b) Computational complexity in real-time systems.

2.9 AR Markers and `ar_track_alvar` Library

AR markers, or fiducial markers, are binary patterns used in robot vision for pose estimation. The `ar_track_alvar` ROS package supports real-time tracking of such markers, estimating both position and orientation with high accuracy [26].

2.9.1 Technical Functionality

`ar_track_alvar` uses Alvar libraries and the PnP algorithm to compute the 6-DOF pose of markers relative to the camera. It is efficient and accurate in environments with controlled lighting and minimal occlusion [27].

2.9.2 Applications

- **Object Localization:** For pick-and-place operations.
- **Autonomous Navigation:** Used in indoor UAV landings and SLAM [28].
- **Robot Calibration:** Helps align real-world objects with their simulation counterparts.

2.9.3 Advantages and Limitations

Advantages:

- a) Lightweight and open-source.
- b) High-speed pose estimation.
- c) Compatible with multiple camera types.

Limitations:

- a) Performance degrades with poor lighting or occlusions.

- b) Requires prior camera calibration for accuracy.

2.10 Related Work

So far, design and development of robotic sorting have been widely explored in various contexts, each using different methodologies to efficiently classify and sort objects. For instance, Cong et al.[29] developed a system using dual robot arms to classify objects based on shape and size. Their approach combined a hierarchical control system in which the master controller processed the images and did the kinematic calculations while the slave controllers controlled the robot arms autonomously. The contour-finding and centroid algorithms were used in this system to perform image recognition, yielding smooth motion with minimum vibration. On the other hand, the disadvantage of this approach is that shape-based classification may not be effective when object shapes are complicated or irregular.

Another important work is that of El-Shair and Rawashdeh [30], who addressed the precision issues of low-cost robotic arms by incorporating a vision-guided system. Real-time feedback was used to guide the robotic arm in performing an accurate sort, even with budget hardware. Although their method indeed showed better alignment and object detection, the efficiency of the system for high-speed or large-scale operations remained untested, hence it is not scalable.

Similarly, Prusaczyk et al. [31] addressed industrial sorting by equipping a KINOVA robotic arm with a vision system. Their solution utilized Adaptive Vision Studio for image recognition and QR code-based object identification to enable accurate sorting in an industrial setting. However, the fact that the system requires extensive setup and calibration shows possible weaknesses regarding adaptability to various environments.

Song [32], in logistics, has explored the use of a vision-based sorting robot system that relies on monocular motion vision for object localization. Image processing was done using OpenCV, resulting in below 4% for the position error to be cost-effective. Despite such achievements, it cannot perform better in lighting conditions that are poor or objects that are non-standard, affecting its applicability in a wider sense.

Meanwhile, Ye et al. [33] developed an intelligent sorting system for campus logistics in order to pick up packages more autonomously. The system applies robots that can recognize QR code labels to sort packages for direct retrieval by users, ensuring improved safety in crowded conditions. However, being applied to the context of campus logistics reduces its adaptability in other contexts.

Dahal [34] proposed an automated retail billing system that utilized YOLOv8 and DeepSORT for object detection and tracking. The system greatly simplified retail operations but was designed mainly for retail applications, hence limiting its use in general sorting tasks.

Huang and Li [35] proposed an express sorting system based on two-dimensional code recognition for the automation of sorting tasks pertaining to express delivery services. They applied two-dimensional code recognition to solve problems related to the high cost of RFID systems and the susceptibility of barcode systems to errors. Their approach was cost-effective, but it may lack flexibility in environments where 2D codes are not feasible.

Smart Parcel Sorting Control System Based on QR Code Recognition Han and Yang [36] presented a system that integrated QR code recognition for the efficient sorting of parcels. In the abstract, there is no insight about the performance or limitations of the system, but the idea of QR code-based sorting goes in the direction of reducing human labor in logistics operations.

Zhangchao Pan et al. (2020) [37] describe a Computer Vision-Guided Manipulator Package Sorting and Placing System for enhancing logistic efficiency related to the automation of package sorting and manipulator distribution. Their approach uses real-time measurement of package sizes, improving the manipulator's control in accomplishing precise movements. They have used YOLO V3 for object detection. While effective in its objectives, limitations included relatively long sorting time, averaging 53.25 seconds, and reliance on an older YOLO version, which might affect detection speed and accuracy.

The reviewed papers explore diverse methods for robotic sorting systems, focusing on efficiency, accuracy, and adaptability. Some, like Cong et al.[29] and El-Shair & Rawashdeh [30], developed vision-guided systems for precise object classification and alignment, though

scalability and irregular shapes posed challenges. Industrial applications, such as Prusaczyk et al. [31], emphasized integrating robotic arms with vision systems, requiring significant setup. In logistics, Song and Ye et al [32]. applied vision-based sorting and QR code recognition to enhance automation, but lighting and context specificity limited their adaptability. Retail-focused systems, such as Dahal [34], streamlined operations using YOLOv8, while Huang & Li [35] and Pan et al [37]. utilized older vision techniques for cost-effective sorting, constrained by detection efficiency and speed. Each study highlights trade-offs between innovation and practical limitations.

Table 2.1 Summary of related work

Year	Author(s)	Title of the Article	Objectives	Methods	Remarks
2022	V. Cong, L. Hanh, L. Phuong, D. Duy	Design and Development of Robot Arm System for Classification and Sorting Using Machine Vision	Develop a dual robot arm system for shape and size-based object classification.	Hierarchical control system with master-slave controllers, image processing using contour finding and centroid algorithms.	Effective for shape-based sorting but limited in handling irregularly shaped objects.
2019	Z. A. El-Shair, S. A. Rawashdeh	Design of an Object Sorting System Using a Vision-Guided Robotic Arm	Improve precision and accuracy of budget robotic arms.	Vision-guided feedback system to compensate for mechanical inaccuracies.	Demonstrated improved alignment, but scalability and high-speed operations not tested.
2019	P. Prusaczyk, W. Kaczmarek, J. Panasiuk, K. L. Besseghieur	Integration of Robotic Arm and Vision System with Processing Software	Enhance production line efficiency with vision-guided robotic sorting.	Integrated KINOVA robotic arm with Adaptive Vision Studio for QR code-based sorting.	Significant setup and calibration required for operational readiness in industrial setups.
2024	Meijing Song	Research on Intelligent Logistics Sorting Robot Control Based on Machine Vision	Develop vision-based sorting robot with motion localization for logistics applications.	Monocular vision for motion ranging and localization using OpenCV.	Performs well with less than 4% localization error; performance limited in low-light conditions.

2022	Zhijian Ye, Yiang Li, Zhiping Zhang, et al.	Research on Campus Logistics Intelligent Sorting System Using Robots	Streamline logistics at university campuses.	QR code recognition for automated sorting and package pickup system.	Application-specific to campuses; broader adaptability not addressed.
2024	Bishwambhar Dahal	Automated Retail Billing: Streamlining Checkout with QR Codes and Object Tracking Using YOLOv8 and DeepSORT	Automate retail checkout operations.	Object detection with YOLOv8 and DeepSORT for real-time tracking.	Primarily suited for retail; not generalized for other sorting applications.
2018	Mengtao Huang, Yifan Li	Express Sorting System Based on Two-Dimensional Code Recognition	Automate express sorting tasks with reduced costs.	2D code recognition to replace RFID and barcode systems.	Cost-effective, but limited in flexibility for non-code-based sorting tasks.
2024	Shuangshuang Han, Mengran Yang	Smart Parcel Sorting Control System Based on QR Code Recognition	Develop efficient parcel sorting systems using QR code recognition.	QR code-based control system for parcel sorting.	Limited information on performance or specific applications.
2020	Zhangchao Pan, Zixi Jia, Kaiyuan Jing, Yinguang Ding, Qianmeng Liang	Manipulator Package Sorting and Placing System Based on Computer Vision	Automate package sorting and enhance manipulator control for precise movements.	Real-time package size measurement; object detection using YOLO V3.	Long sorting time (average 53.25s); reliance on older YOLO V3 limits detection efficiency.

2.10.1 Summary of Limitations

Most of the sorting systems developed so far have considerable limitations that affect their efficiency and effectiveness. Inefficient path planning and motion control make it take long to complete, hence slowing down the overall performance. Robotic arms often face difficulties in smooth motion and precise control, which in turn affects the accuracy and speed of sorting. Most of them use very outdated

algorithms for detection, which cannot handle dynamic environments and give the required accuracy in modern logistics. These are further exacerbated by limitations in processing power, making delays and reducing the capability of the system for real-time data handling. The challenges mentioned above give a background on the requirements for improvements in faster processing, advanced control methods, and improved computational resources in sorting systems.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

This project focuses on developing an integrated system for efficient object detection, tracking, and sorting using machine vision and robotic manipulation. The methodology encompasses system design, path planning, machine vision implementation, and hardware-software integration, as outlined below:

3.2 Project Flow

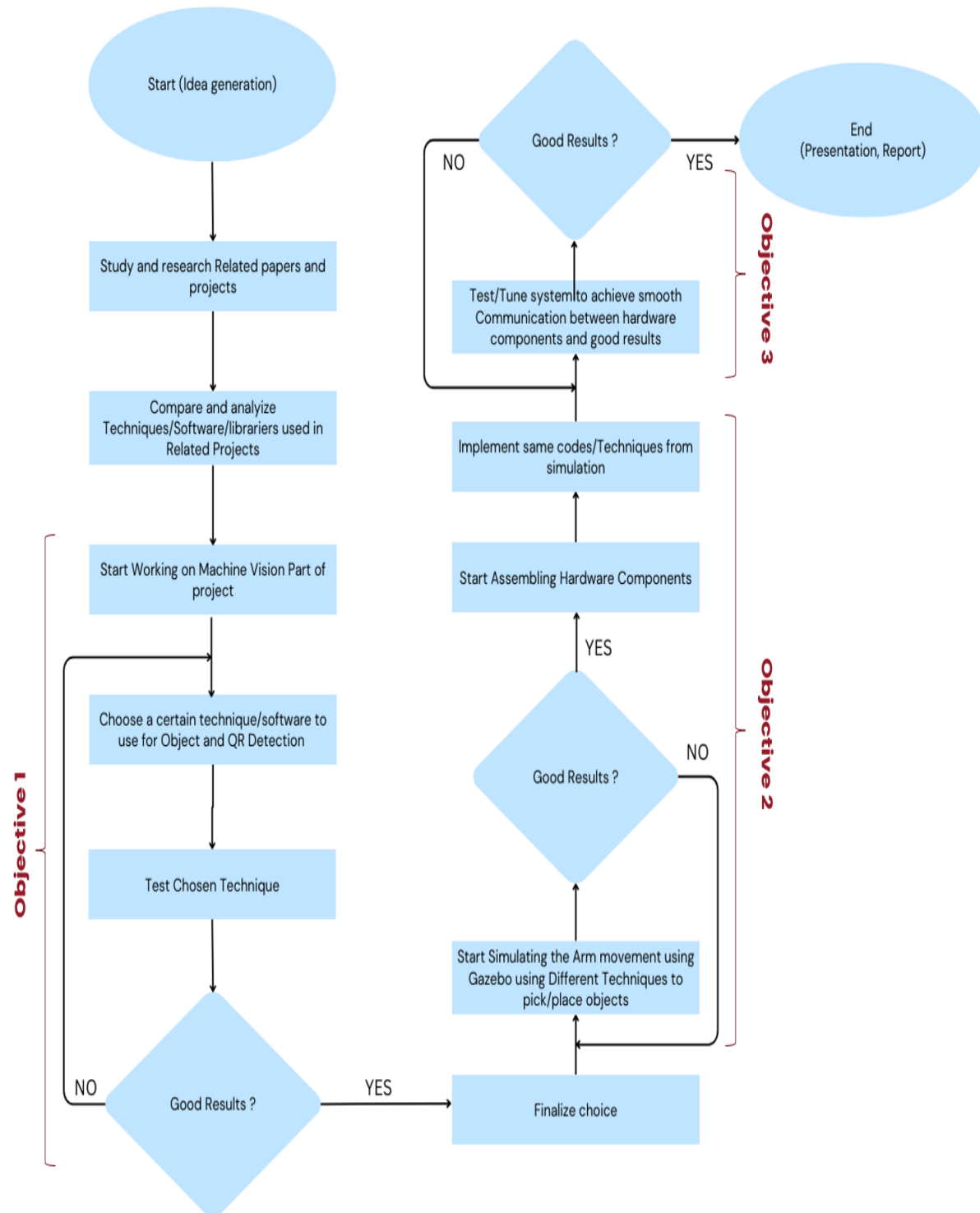


Figure 3.1 Project Flowchart

The project follows a systematic approach to achieve its objectives. The steps include:

- (a) **Idea Generation:** Brainstorming concepts and defining the project scope.
- (b) **Literature Review:** Studying and analyzing related works to gain insights into existing methods, tools, and techniques.
- (c) **Techniques Comparison:** Evaluating and selecting suitable algorithms, software, and libraries for system development.
- (d) **Machine Vision Development:** Implementing and testing object and QR code detection techniques.
- (e) **Path Planning Optimization:** Simulating and refining algorithms for robotic arm motion.
- (f) **System Integration:** Combining hardware and software components into a cohesive framework.

Tasks	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15
Decide the project idea															
Read and search for previous journals															
Determine the system parameters and Architecture															
Frame the system methodology															
Learn About Machine Vision															
QR detection coding and testing															
Prepare the project presentation slides															
Finalize the FYP1 thesis															

Figure 3.2 FYP 1 Gantt Chart

Tasks	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	W11	W12	W13	W14	W15
Setting Up ROS (Ubuntu)															
Setup Gazebo for simulation															
Simulating Arm movement (pick/place)															
Build the system on hardware															
Controlling Arm with camera input															
Testing and tuning system for best results															
Prepare the journal paper and the demonstration video															
Prepare the FYP2 report draft															

Figure 3.3 FYP 2 Gantt Chart

3.2.1 Objective 1: Real-Time Machine Vision System

To achieve the first objective, a real-time machine vision system was developed using a USB webcam and the `ar_track_alvar` ROS package. Instead of QR codes or neural networks, the system uses fiducial AR markers (Alvar tags), which provide robust and low-latency pose estimation in 3D space. The AR markers are printed and attached to each object, allowing the system to identify the object and calculate its 3D position and orientation relative to the camera.

The machine vision system operates as follows:

- Capturing real-time video input using a USB webcam.
- Calibrating the camera using a checkerboard to correct lens distortion and obtain the camera matrix.
- Detecting AR markers using the `ar_track_alvar` ROS package.
- Extracting each marker's ID, position (X, Y, Z), and orientation (quaternion format).
- Publishing marker pose data via the `/ar_pose_marker` topic for further processing.

The system provides consistent detection at around 8 Hz, with visualization and debugging tools integrated through RViz. The marker pose data is then used to guide the robot's pick-and-place operations.

3.2.2 Objective 2: Robotic Arm Motion Planning and Execution

The second objective focuses on programming the OpenManipulator-X robotic arm to perform accurate and efficient pick-and-place operations based on the real-time pose of detected AR markers. The OpenCR controller is used to interface the arm with ROS, enabling seamless integration between the perception and actuation layers.

The motion planning system does not rely on external path planning tools or simulation (e.g., Gazebo) but rather on direct inverse kinematics and safe-step motion profiles implemented in C++. The process involves:

- Mapping AR marker IDs to specific drop-off coordinates.
- Applying a three-stage Z-axis movement to ensure safe and reliable grasping:
 - **Safe Z-approach** (approach from above),
 - **Pick Z-level** (gripper descends to object),
 - **Lift Z-level** (after grasping).
- Using ROS topics such as `/goal_task_space_path` and `/goal_tool_control` to control the end-effector position and gripper state.
- Returning the robot to a home pose and repeating the process for the next object.

All motions are executed based on real-world coordinates derived from marker pose data, and the robot performs sorting actions in real time.

3.2.3 Objective 3: System Integration and Testing

The final objective focuses on integrating all system components—vision, control, and hardware—into a unified, autonomous sorting system. Unlike earlier proposed setups that used Raspberry Pi or U2D2, the final prototype uses only the OpenCR board and a host PC running ROS.

The system integration phase includes:

- Mounting and calibrating the USB webcam in a fixed overhead position.
- Physically arranging the workspace into an object zone and three drop-off zones.
- Synchronizing the camera detection node and robot control node under a single ROS launch file.
- Conducting functional and timing tests to measure sorting performance, stability, and detection accuracy.
- Verifying pose estimation consistency under various lighting and distance conditions.

The result is a modular and reliable autonomous robotic sorting system capable of detecting multiple objects and performing accurate pick-and-place operations in real-time.

3.3 System Architecture & Flow

3.3.1 System Architecture

The system architecture integrates the following hardware and software components:

- **Hardware:**

- **OpenMANIPULATOR-X Robotic Arm**

The OpenMANIPULATOR-X is an open-source robotic arm developed by ROBOTIS, designed for education, research, and development purposes. It features a modular design with four degrees of freedom (DOF) plus a gripper, allowing for a variety of manipulative tasks. The arm utilizes DYNAMIXEL X-Series servos, known for their precision and reliability. Its open-source nature enables users to customize both hardware and software, facilitating a wide range of applications in robotics education and prototyping [38].

- Design and Specifications :

The OpenManipulator-X is a versatile robotic arm designed with key features that make it suitable for a range of applications. It has 4 degrees of freedom (DOF) plus a gripper, offering significant flexibility in movement and object manipulation. The arm is capable of handling payloads up to 500 grams, making it ideal for light manipulation tasks in controlled environments. With a high repeatability

of ± 0.1 mm, it ensures consistent and precise performance, which is crucial for repetitive operations. Its modular design facilitates easy assembly, customization, and seamless integration with other systems, enhancing its adaptability for research and development. Additionally, the OpenManipulator-X is compatible with the Robot Operating System (ROS), enabling control through either a PC or embedded boards like the OpenCR. Furthermore, the system is supported by open-source resources, including comprehensive documentation, CAD files, and software, which empowers users to modify the system and tailor it to their specific needs or research goals [39].

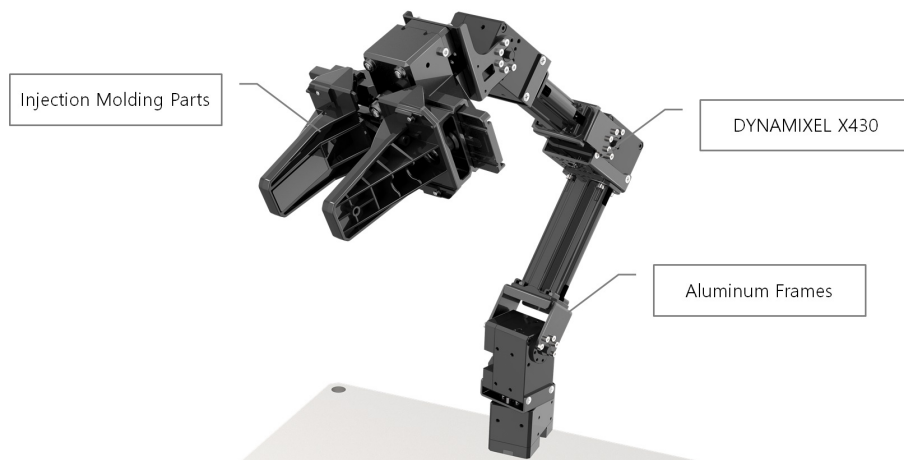


Figure 3.4 Openmanipulator-x

– Applications and Educational Value :

The OpenManipulator-X is a versatile robotic tool that holds significant value in various applications and educational contexts. In educational platforms, it offers students hands-on experience in robotics, providing opportunities to explore concepts such as kinematics, control systems, and programming. This practical engagement fosters a deeper understanding of robotics and its associated technologies. In the realm of research and development, the OpenManipulator-X serves as a robust platform for prototyping and experimentation across fields like automation, human-robot interaction, and artificial intelligence. Additionally, its ability to integrate seamlessly with mobile robots, such as the TurtleBot3, expands its functionality. This integration allows for studies in mobile manipulation and collaborative robotics, enabling advanced exploration of how stationary and mobile systems can work together effectively.

– **USB Webcam**

A standard USB webcam is used as the vision sensor for the system. It captures real-

time video of the workspace and streams the image feed to the ROS environment. The camera is positioned in a fixed location above the sorting area to provide a clear, top-down view. This setup allows accurate detection of fiducial AR markers on objects, enabling reliable 3D pose estimation. The webcam's affordability and ease of integration with ROS make it a suitable choice for vision-based robotic applications.

- **OpenCR Board**

The OpenCR (Open-source Control Module for ROS) board acts as the main interface between the PC running ROS and the OpenManipulator-X robotic arm. It handles low-level control of the Dynamixel servo motors and receives motion commands via ROS topics. The OpenCR also supports embedded IMU and power management features. It replaces the need for separate communication modules like U2D2 and Power Hub, providing a unified control and power platform optimized for the OPX.

- **PC (Laptop)**

The host PC is used for all high-level processing tasks. It runs the ROS core, processes camera input, executes marker detection nodes, and controls robot movement. The PC also facilitates visualization via RViz and debugging tools such as `rqt_graph` and `rostopic`. With its greater computational power compared to embedded boards, the PC enables real-time processing of image streams and precise control of the robotic system.

- **Software:**

- **ROS (Robot Operating System):**

ROS 1 Noetic, running on Ubuntu 20.04, is used as the middleware framework for this project. ROS facilitates communication between software modules using a publish-subscribe architecture. It allows real-time coordination between the camera input node, AR marker detection node, and robotic arm control node. Key ROS tools used include `roscore`, `rostopic`, `rviz`, and `tf`. ROS is widely adopted in academia and industry due to its modularity, open-source community, and integration with a wide variety of robotic hardware [41].

- **ar_track_alvar Package:**

This ROS package enables the detection and pose estimation of AR (Augmented Reality) markers using the Alvar library. It identifies marker IDs and provides 3D position and orientation of each detected marker in real-time. The information is published through the `/ar_pose_marker` topic and used to guide the robotic arm for object picking. The package offers reliable, low-latency performance suitable for robotic sorting tasks where visual tagging is necessary.

- **OpenCV (Camera Calibration):**

OpenCV is used in the early stages of the project for camera calibration. A checkerboard pattern is captured from various angles, and intrinsic parameters such as focal lengths, distortion coefficients, and principal points are calculated. These parameters are essential for correcting image distortion and ensuring accurate 3D pose estimation during AR marker detection.

- **RViz:**

RViz is a 3D visualization tool in ROS used to display robot models, sensor data, coordinate frames, and marker detections. In this project, it was used to monitor the real-time pose of AR markers, visualize robot movements, and debug coordinate frame alignments through the TF tree.

- **C++:**

The core control logic for the robot was written in C++ for performance and real-time execution. Custom ROS nodes were developed to process AR marker data, compute inverse kinematics, and send commands to the robotic arm using ROS service calls and topics. C++ is favored in ROS development due to its efficiency and native compatibility with ROS libraries.

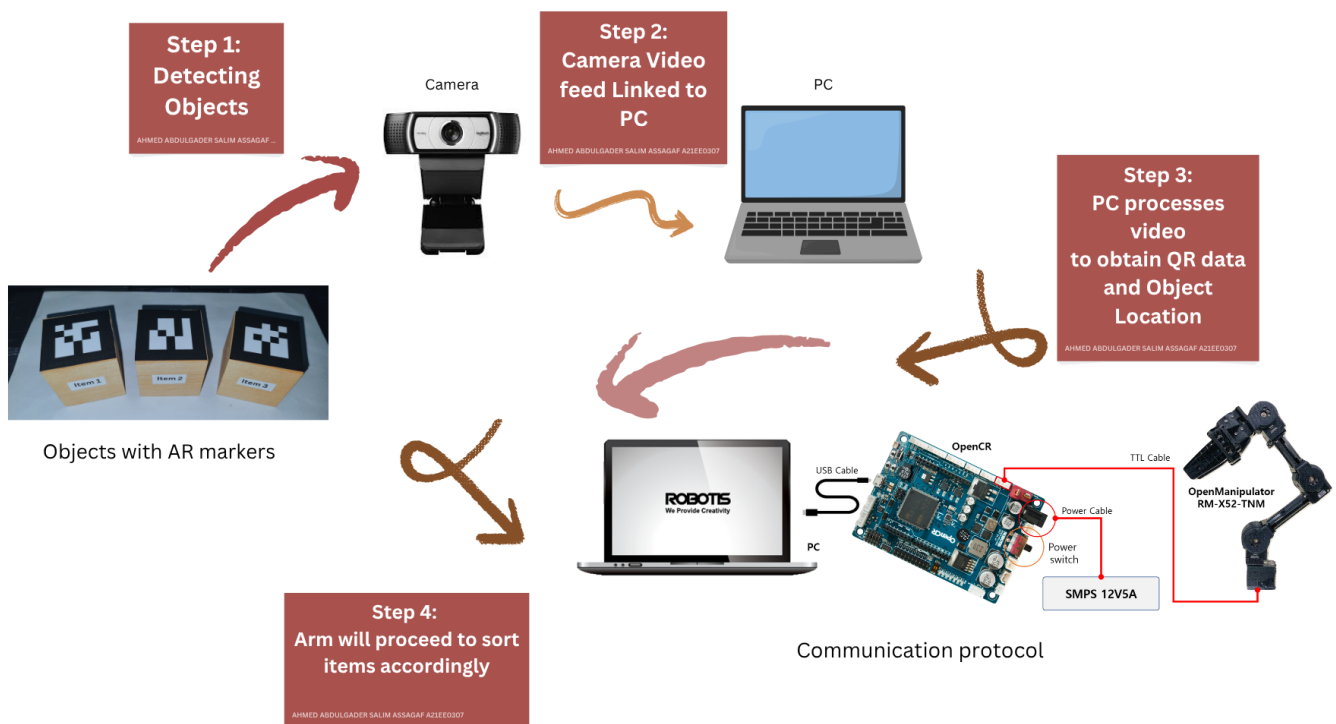


Figure 3.5 System Architecture for Automated Sorting

The figure illustrates the system architecture and process flow for the proposed AR marker-based robotic sorting system.

The process begins with a USB camera mounted overhead, capturing continuous video of the workspace. Objects placed within the workspace are tagged with AR (Alvar) markers, each encoding a unique ID. The camera acts as the system's primary sensor, feeding the live video stream to the host PC.

The image stream is processed using the `ar_track_alvar` ROS package, which detects the AR markers and publishes their 3D poses through the `/ar_pose_marker` topic. Although full 3D pose information is available, this system only utilizes the **X and Y coordinates** of the detected marker to localize the object on the workspace. The Z-coordinate is predefined and fixed for all objects since they are assumed to lie on a flat surface.

Once an object is detected, the system determines:

- The object's identity from the AR marker ID
- The object's (X, Y) location on the plane
- The corresponding drop-off zone based on a static ID-to-location mapping

This information is passed to a custom C++ ROS node, which calculates a safe pick-and-place trajectory using inverse kinematics. The robot follows a simplified three-stage vertical profile:

- (a) Move above the object (X, Y, safe fixed Z)
- (b) Descend to fixed pick height
- (c) Close gripper, ascend, and move to the designated drop zone

The host PC sends the computed trajectory commands to the OpenCR board, which controls the OpenManipulator-X robotic arm through ROS topics. The gripper is also actuated through the same communication pipeline.

After placing the object at its designated location, the arm returns to its home pose and waits for the next detection. This system provides a lightweight, planar sorting solution using real-time vision and robotics integration.

3.3.2 System flow

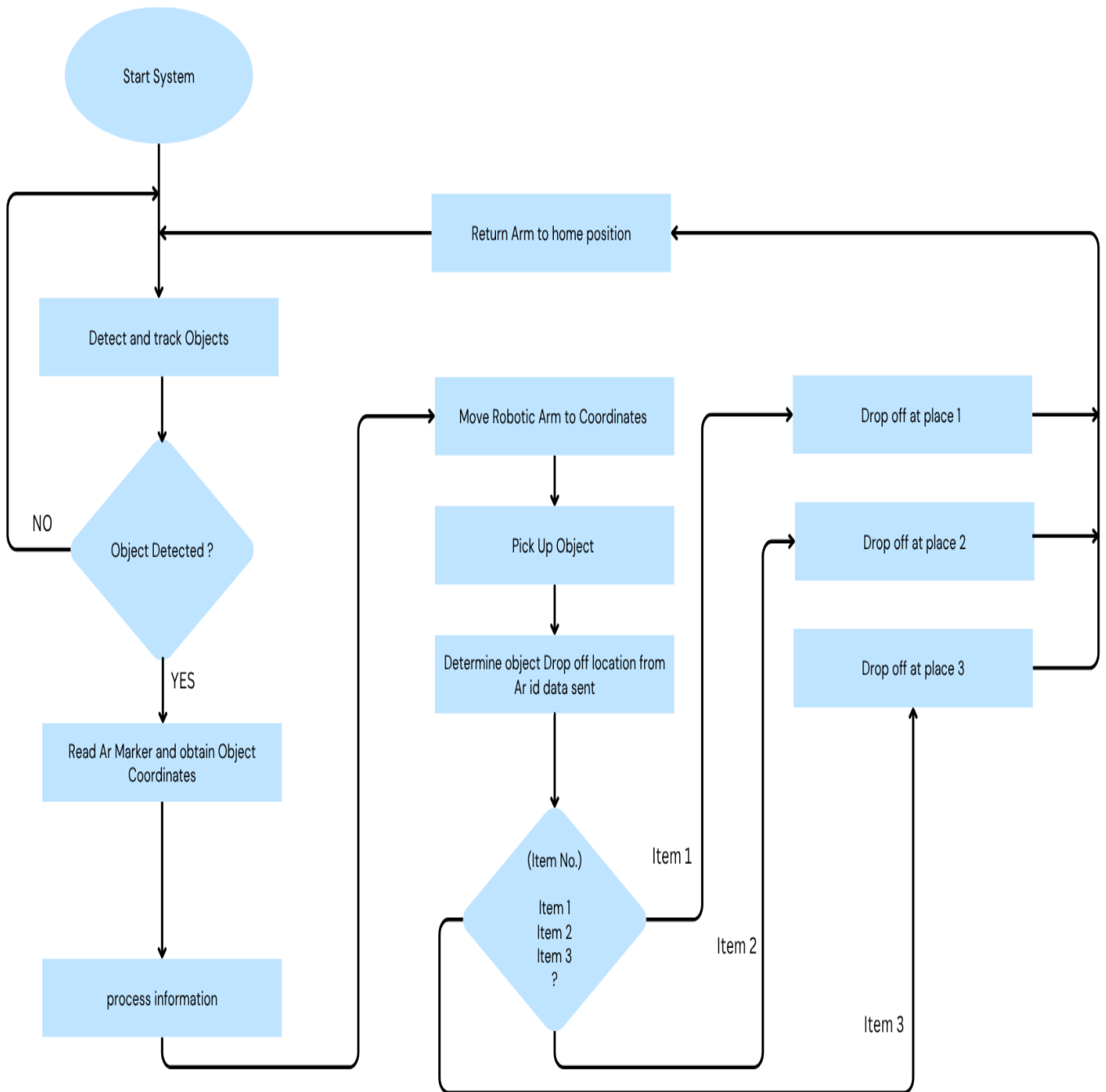


Figure 3.6 System Flowchart for Automated Object Sorting

This Flowchart illustrates the step-by-step process of detecting, identifying, and sorting objects using machine vision and a robotic arm, integrated with Raspberry Pi for warehouse logistics.

- (a) **Start System:**
 - Launch ROS core, camera driver, AR marker detection node (`ar_track_alvar`), and the robotic arm control node.
- (b) **Capture Live Camera Feed:**
 - A USB webcam mounted above the workspace streams real-time video to the host PC.
- (c) **Detect AR Markers:**
 - The `ar_track_alvar` package detects visible AR markers and publishes their IDs and 3D poses on the `/ar_pose_marker` topic.
- (d) **Check for Marker Detection:**
 - If no markers are detected, the system continues scanning. If a marker is detected, the object ID and pose (X, Y) are extracted.
- (e) **Identify Object and Destination:**
 - Based on the marker ID (1, 2, or 3), the system determines which object it is and assigns the corresponding drop-off location.
- (f) **Compute Robotic Arm Trajectory:**
 - A C++ ROS node computes the (X, Y) position of the object, inserts a fixed Z height, and generates the necessary inverse kinematics commands.
- (g) **Pick Object:**
 - The arm moves to the object's (X, Y, Z) coordinates, lowers to a fixed Z-height, and closes the gripper to grasp the item.
- (h) **Move to Drop-off Location:**
 - The robotic arm moves to the assigned location based on the object's ID and releases the object.
- (i) **Return to Home Position:**
 - After drop-off, the robotic arm returns to its initial home pose.
- (j) **Repeat Process:**
 - The system continuously monitors for the next object until manually stopped or all items are sorted.

Summary: This systematic process ensures efficient and accurate sorting of objects, meeting the objectives of the project.

3.4 Cost of Project

The total expenditure for the project was kept minimal by utilizing university-provided resources and free software tools. The main hardware component, the OpenMANIPULATOR-X robotic arm with its motors, was provided without cost. Essential software, including Linux, ROS, and Visual Studio Code, was open-source and freely available. However, several supporting items were independently purchased to complete the system setup, such as a camera, lighting stand, wooden base, USB hub, and power cable, amounting to a total of RM 112.40. A detailed breakdown of the project expenditure is shown in Table 3.1.

Table 3.1 Project Expenditure

Hardware	Price per Unit (RM)	Quantity	Total (RM)
OpenMANIPULATOR-X Arm (with 4 motors)	Provided	1	0.00
Camera	26.00	1	26.00
Camera Stand and Light	27.00	1	27.00
Wooden Base Platform	7.90	4	31.60
USB Hub	8.90	1	8.90
Power Cable	18.90	1	18.90
Subtotal (Hardware)			112.40
Software			
Linux OS	Free	1	0.00
ROS (Robot Operating System)	Free	1	0.00
Visual Studio Code	Free	1	0.00
Subtotal (Software)			0.00
Total Cost			112.40

3.5 Conclusion

The methodology outlined in this chapter provides a structured approach for developing an AR marker-based robotic sorting system. The use of a USB camera combined with the `ar_track_alvar` ROS package enabled accurate real-time detection of objects based on AR marker IDs. By focusing on the planar (XY) coordinates and using a fixed Z-height for grasping, the system was able to simplify motion planning while maintaining precision.

For robotic manipulation, inverse kinematics was implemented to translate object positions into joint commands for the OpenManipulator-X. The control strategy ensured safe pick-and-place operations through a multi-stage movement profile, minimizing the risk of collision or misalignment.

Finally, seamless integration of perception, decision-making, and actuation was achieved using the ROS framework, allowing modular development and real-time synchronization between system components. This methodology laid the technical foundation for system testing and performance evaluation presented in the next chapter.

CHAPTER 4

RESULTS AND DISCUSSION

This chapter presents the results obtained from the implementation of the AR marker-based robotic sorting system and discusses its performance with respect to the objectives defined in Chapter 1. The focus is placed on three core areas: the accuracy and consistency of the machine vision system, the precision of robotic motion through inverse kinematics, and the effectiveness of full-system integration using ROS. Each subsection details the technical outcome of the implemented solution, performance observations, and an analysis of its capabilities and limitations in a controlled environment.

4.1 Objective 1: Machine Vision System Performance

The first objective aimed to develop a real-time machine vision system that could detect and decode objects using AR markers. To achieve this, a USB webcam was used to capture live video input from a fixed top-down view. The image stream was processed using the `ar_track_alvar` ROS package, which was configured to identify fiducial markers and estimate their 6-DoF pose. In the context of this project, only the X and Y coordinates of each object were extracted and used for positioning, while the Z-axis value was fixed manually based on the consistent height of all items.

```
header:
  seq: 0
  stamp:
    secs: 1750006197
    nsecs: 924139211
  frame_id: "world"
id: 1
confidence: 0
pose:
  header:
    seq: 0
    stamp:
      secs: 0
      nsecs: 0
    frame_id: ""
  pose:
    position:
      x: 0.19248805027650828
      y: -0.08427219958347983
      z: -0.09016219111647941
    orientation:
      x: 0.07855752072024746
      y: -0.11512267956066821
      z: -0.8016790647383114
      w: 0.5812797620331935
```

Figure 4.1 Item position and orientation from Rostopic "ar_pose_marker"

Calibration of the USB camera was conducted using a standard checkerboard pattern. This allowed intrinsic parameters such as focal length, principal point, and distortion coefficients to be estimated. With these parameters loaded at runtime, the system was able to produce accurate and undistorted pose estimates of the markers. The detection frequency of the marker tracking system averaged approximately 8 Hz, which provided sufficient update speed for a static sorting scenario. Tests conducted with varying marker sizes and distances confirmed that marker detection remained stable within a range of 25–80 cm, provided adequate lighting conditions were maintained.

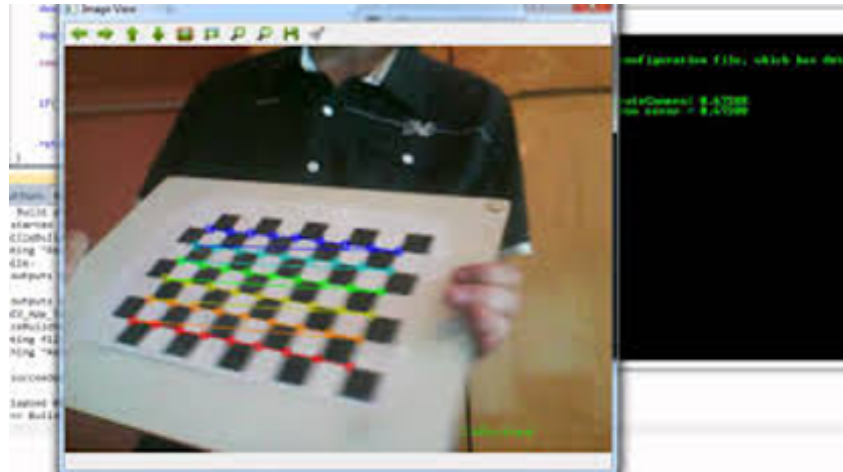


Figure 4.2 Checkerboard pattern for calibration

```

1 %YAML:1.0
2 ---
3 image_width: 640
4 image_height: 480
5 camera_name: narrow_stereo
6 camera_matrix:
7   rows: 3
8   cols: 3
9   data: [734.667482563894, 0.0, 296.5054532220164, 0.0, 731.3203076559556, 218.1474665988071, 0.0, 0.0, 1.0]
10 distortion_model: plumb_bob
11 distortion_coefficients:
12   rows: 1
13   cols: 5
14   data: [0.29950254025156875, -1.2445646324824733, -0.003087818176152257, -0.011393056235279227, 0.0]
15 rectification_matrix:
16   rows: 3
17   cols: 3
18   data: [[1.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 1.0]]
19 projection_matrix:
20   rows: 3
21   cols: 4
22   data: [738.7564086914062, 0.0, 289.4129808149137, 0.0, 0.0, 744.9739990234375, 216.70429378742483, 0.0, 0.0, 0.0, 1.0, 0.0]
23

```

Figure 4.3 Camera Calibration parameters

Marker IDs were assigned to three specific items used in testing, and detection remained reliable throughout multiple sessions. Each detection instance resulted in a unique pose output containing the object's (X, Y) position and quaternion orientation, which was visualized and verified using RViz and ROS topic monitoring tools. The consistent operation of the vision system confirmed that the machine vision component successfully fulfilled the project's first objective.

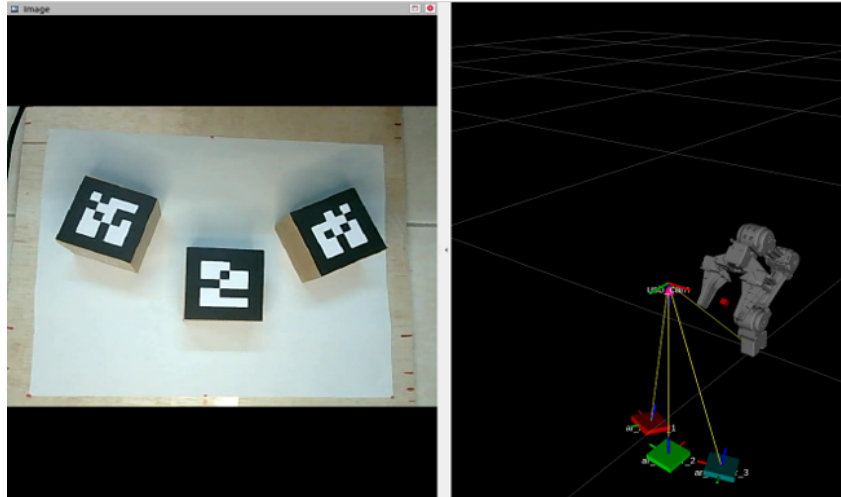


Figure 4.4 Items visualization in RViz

4.2 Objective 2: Robotic Motion Execution and Inverse Kinematics

The second objective focused on programming the OpenManipulator-X robotic arm to perform precise pick-and-place tasks based on marker pose data. The robotic arm was connected through the OpenCR controller, which interfaced directly with the ROS environment. Control logic was implemented in C++ and used to receive marker pose data from the vision system, extract the X and Y positions, and assign a fixed Z-height to define a full 3D target position.

Inverse kinematics was performed using functions provided by the official Robotis ROS packages. The system computed the required joint angles to reach each object, beginning from a fixed home pose and executing a multi-stage motion sequence. This sequence included moving the arm to a safe height above the object, descending to the pick height, closing the gripper, lifting the object, and navigating to the corresponding drop-off point. The robotic arm then released the object before returning to the home position to await the next detection.

The performance of the robot was evaluated by measuring the ROS topic update rates and the responsiveness of the motion. The `/joint_states` topic, which publishes joint feedback, maintained a stable frequency of approximately 110 Hz. The `/gripper/kinematics_pose` topic, which tracks the gripper's Cartesian position, also updated at around 100 Hz. These high-frequency feedback loops enabled real-time responsiveness, which contributed to smooth and accurate motion execution.

Table 4.1 ROS Topics and Rates Used in Robotic Arm Operation

ROS Topic	Message Type	Rate (Hz)	Description
<code>/joint_states</code>	<code>sensor_msgs/JointState</code>	~110	Currently publishes the arm's joint positions.
<code>/gripper/kinematics_pose</code>	<code>geometry_msgs/PoseStamped</code>	100	Gripper's current pose.
<code>/ar_pose_marker</code>	<code>ar_track_alvar/AlvarMarker</code>	~8	Detected AR markers' IDs and poses.
<code>/move_base_simple/goal</code>	<code>geometry_msgs/PoseStamped</code>	event	Desired target pose for arm movement (published when a new command is issued).

The high-frequency rates of over 100 Hz for both the `/joint_states` and `/gripper/kinematics_pose` topics enable smooth and real-time feedback control of the robotic arm's movement. This ensures precise actuation during trajectory execution and gripper adjustments.

In contrast, the AR marker detection system operates at approximately 8 Hz. This lower rate is sufficient because the objects in the workspace remain stationary during each sorting cycle. Faster updates are unnecessary in this static environment, and reducing frequency conserves computational resources.

The `/move_base_simple/goal` topic operates on an event-driven basis. It publishes a new target pose only when a pick or place command is initiated by the system, making it an efficient trigger for motion commands without unnecessary topic traffic.

The inverse kinematics-based motion was tested using three distinct object IDs, each mapped to a pre-defined drop-off location. A total of 10 pick-and-place cycles per object were performed, and the robot completed all tasks without failure. Joint angle readings were logged and showed high consistency, with no major deviation in the approach or drop-off phases. As the Z-level was fixed, no adaptive Z-correction was necessary, which simplified the motion planning process.

```
pose:
  position:
    x: 0.09526477774323296
    y: 0.0
    z: 0.14932133424648014
  orientation:
    x: 0.0
    y: 0.39187618052056844
    z: 0.0
    w: 0.920017966748808
  max_accelerations_scaling_factor: 0.0
  max_velocity_scaling_factor: 0.0
  tolerance: 0.0
```

Figure 4.5 Gripper pose in XYZ

```
header:
  seq: 468173
  stamp:
    secs: 1750007740
    nsecs: 604692750
  frame_id: ''
name:
  - joint1
  - joint2
  - joint3
  - joint4
  - gripper
position: [0.0, -1.052310824394226, 0.35588353872299194, 1.5002332925796509, 0.010009225308895111]
velocity: [0.0, 0.0, 0.0, 0.0, 0.0]
effort: [0.0, 18.829999923706055, -121.05000305175781, -40.35000228881836, 0.0]
```

Figure 4.6 Gripper pose in joint space

The drop-off positions for each item were defined in joint space using four joint angles corresponding to the OpenManipulator-X's 4 degrees of freedom. For each object, two sets of joint configurations were recorded: one representing a safe position directly above the target drop-off zone (used for approach), and the other representing the final placement position. These joint values were manually extracted after calibrating the arm to each basket location during system setup. The use of predefined joint-space positions ensures precise and repeatable placement for each item, reducing the likelihood of collision or misalignment during the drop-off phase.

Table 4.2 Drop-off Location Mapping Using Joint Space (in Radians)

Item (Marker ID)	Above Drop-off Position	Final Drop-off Position
Item 1 (ID 1)	(1.867, -0.874, 0.387, 1.493)	(1.878, -0.913, 1.004, 0.989)
Item 2 (ID 2)	(3.088, -0.270, 0.268, 0.891)	(3.098, -0.179, 0.732, 0.319)
Item 3 (ID 3)	(-1.700, -1.321, 0.726, 1.265)	(-1.710, -1.342, 1.268, 0.880)

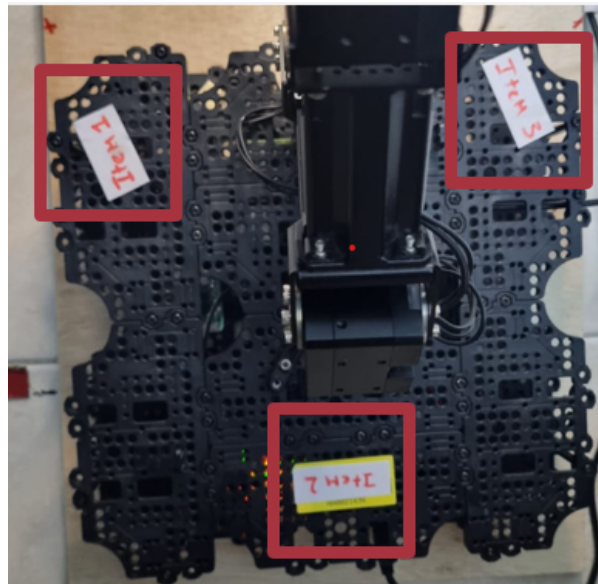


Figure 4.7 Drop-off positions

4.3 Objective 3: System Integration and Overall Performance

The third objective involved integrating the machine vision system and robotic control logic into a complete, autonomous sorting platform. The full system was launched using a custom ROS launch file that brought up the camera node, AR marker tracking node, arm controller, and the main logic node in sequence. Testing was performed in a controlled indoor environment with a stable camera mount, fixed lighting, and objects placed within a well-defined workspace.

The system was tested using three AR-tagged objects. Each item had a distinct marker ID and was associated with a specific drop-off zone. The robot successfully detected, identified, and sorted all three items based on their IDs. The sorting process took approximately 18 to 22 seconds per item, averaging around 20 seconds for a full pick-and-place cycle. This included the time required for detection, pose transformation, arm movement, and gripper operation.

During testing, the system demonstrated full functionality across repeated cycles. The robot was able to return to its home pose, reinitialize, and detect the next object in sequence without requiring a system reset or human intervention. The end-to-end automation validated the effectiveness of using ROS for real-time integration and task coordination.



Figure 4.8 Final prototype

```

-----
Pick and Place demonstration!
-----
1 : Home pose
2 : Pick and Place demo. start
3 : Pick and Place demo. Stop
-----
Press '2' to start pick and place demo.
-----
Robot Moving State: STOPPED
Present Joint Angle J1: 0.000 J2: -1.052 J3: 0.357 J4: 1.500
Present Tool Position: 0.010
Present Kinematics Position X: 0.095 Y: 0.000 Z: 0.149
Present Kinematics Orientation W: 0.920 X: 0.000 Y: 0.392 Z: 0.000
-----
Currently detected AR markers:
ID: 2 --> X: 0.238      Y: -0.007      Z: -0.091
ID: 1 --> X: 0.192      Y: -0.083      Z: -0.090
ID: 3 --> X: 0.199      Y: 0.070       Z: -0.104
Sorted AR IDs: [None]

```

Figure 4.9 Control Terminal

```

-----
Pick and Place demonstration!
-----
1 : Home pose
2 : Pick and Place demo. start
3 : Pick and Place demo. Stop
-----
The demo has finished or was stopped.
--- ALL ITEMS (ID 1-3) HAVE BEEN SUCCESSFULLY SORTED! ---
-----
Robot Moving State: STOPPED
Present Joint Angle J1: 0.000 J2: -1.052 J3: 0.368 J4: 1.500
Present Tool Position: 0.010
Present Kinematics Position X: 0.095 Y: 0.000 Z: 0.147
Present Kinematics Orientation W: 0.918 X: 0.000 Y: 0.397 Z: 0.000
-----
No AR markers currently detected.
Sorted AR IDs: [1, 2, 3]

```

Figure 4.10 Successful Demo

To assess the system's performance relative to similar existing work, results were compared with similar projects:

4.4 Comparative Analysis

To benchmark my system, I compared it with existing robotic sorting systems on various metrics, including detection method, automation level, and task cycle time.

4.4.1 Comparison Table

Table 4.3 Comparison of Autonomous Object Sorting Systems

System	Detection	Hardware	Cycle Time	Automation	Notes
My System	AR markers + ROS	4-DOF arm, fixed cam	20 s/item	Fully autonomous	Lightweight, fast, low setup
Pan et al. (2020)	QR + YOLO	Camera + QR	53 s/item	Semi/full	Complex decoding, slower cycle
Bui et al. (2020)	CNN + RGB-D	5-DOF arm + sensor	15–30 s	Full (lab)	Generalizable, higher cost
Chen et al. (2025)	QR + AMCL + YOLO	Mobile + camera + QR	N/A	Full localization	High precision; no full cycle time
Chen et al. (2022)	YOLOv4 + barcode	Conveyor + camera	0.31 s detect	Automated conveyor	Requires fixed conveyor

4.4.2 Cycle Time Chart

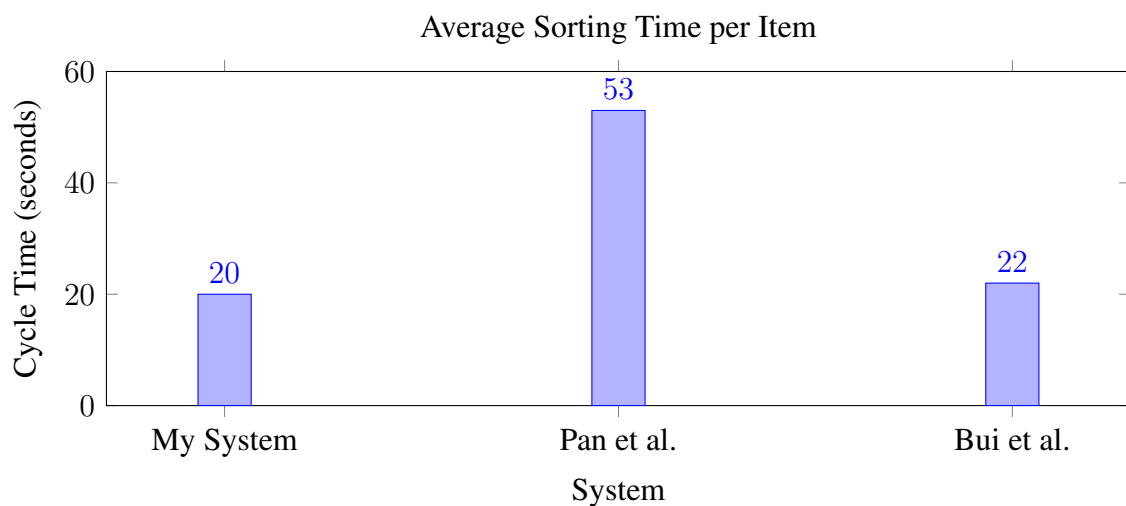


Figure 4.11 Comparison of average cycle time per item across three sorting systems. My system demonstrates the fastest execution time, due to efficient AR marker detection and optimized motion strategy.

4.5 Summary and Insights

In summary, my AR-based sorting system demonstrates superior performance in terms of speed compared to systems that rely on QR codes and YOLO-based detection, while also maintaining full automation. Its simplicity is a key advantage; by avoiding complex deep learning models or mobile navigation frameworks, I achieved a responsive and low-latency design with minimal computational overhead. This makes the system particularly practical for structured, lab-scale environments where consistency and reliability are critical. However, one limitation is its lack of adaptability to dynamic object heights or cluttered scenes—features typically supported by more advanced, albeit more complex, mobile or vision-driven systems.

4.6 Discussion of Limitations

Despite the successful demonstration of all core functionalities, a few limitations were observed. First, the fixed Z-height assumes that all objects lie flat and share the same height. Any variation in object size could lead to failed grasps or collisions. Second, the camera setup requires rigid mounting. Any change in position after calibration can cause pose errors due to incorrect transformations between frames. Third, the 4-DOF robotic arm lacks full orientation control, which limits the range of manipulation tasks it can perform. Additionally, the system lacks a feedback verification mechanism to confirm successful object placement—something that could be added through sensors or vision-based confirmation in future iterations.

Overall, however, the system performed as intended in its defined scope. It offers a reliable, low-cost solution for static object sorting tasks using real-time vision and robotic manipulation.

CHAPTER 5

CONCLUSION

5.1 Introduction

This chapter concludes the development and evaluation of a real-time robotic sorting system using the OpenManipulator-X robotic arm and AR marker-based vision in a ROS environment. The goal of the project was to design a fully autonomous, low-cost, and efficient sorting system suitable for structured environments. The system architecture integrated a USB camera, fiducial marker detection using `ar_track_alvar`, and inverse kinematics-based robot control. Through systematic design, implementation, and testing, the system achieved reliable, repeatable performance with minimal computational complexity.

5.2 Key Findings and Contributions

This research demonstrated that autonomous robotic sorting can be effectively achieved without the need for complex deep learning algorithms or industrial-grade sensors. By using AR markers and fixed Z-coordinate assumptions, the system successfully localized and manipulated objects based on 2D pose information (X, Y), enabling accurate sorting with a low-cost 4-DOF robotic arm.

The project made several technical contributions. First, it showed that real-time integration of perception and motion using ROS can be achieved with high responsiveness, with ROS topic frequencies exceeding 100 Hz for joint and gripper feedback. Second, it introduced a multi-stage movement strategy that separated vertical approach, pick, and placement phases, minimizing collision risk and improving repeatability. Finally, the system demonstrated modularity and maintainability by leveraging ROS nodes for camera input, marker tracking, motion control, and robot actuation.

5.3 Broader Implications

Beyond the technical performance, this work has broader implications for the use of robotics in resource-constrained or educational settings. The system architecture relies entirely on open-source

software and accessible hardware, making it reproducible and adaptable by other students, researchers, or developers. The approach outlined in this project can be extended to various object-handling scenarios in laboratories, small warehouses, and prototyping environments.

In contrast to more complex systems using YOLO, deep CNNs, or SLAM-based mobile robots, this solution offers a practical alternative for applications where simplicity, cost-efficiency, and fast deployment are critical. While its scope is limited to planar, structured environments, the design demonstrates how lightweight robotic automation can still achieve high reliability without sacrificing performance.

5.4 Challenges and Limitations

While the system performed well under test conditions, several limitations were identified. The fixed Z-height assumption simplified trajectory planning but limited the ability to handle objects of varying heights. The 4-DOF arm could not manipulate objects with arbitrary orientations or in cluttered spaces, restricting its flexibility in more complex environments. Additionally, the detection system depends on rigid camera calibration and stable lighting; any deviation in these conditions can impact detection accuracy. There is also no feedback loop for verifying the success of each operation, which would be essential in more dynamic or safety-critical settings.

5.5 Future Work

Several avenues for future development emerge from this work. One important direction is the integration of 3D vision or depth sensing, which would enable the system to adapt its Z-coordinate dynamically based on object height. Implementing a 6-DOF robotic arm could also allow for more sophisticated manipulation tasks, including object rotation and orientation correction.

Another opportunity lies in incorporating post-actuation feedback, such as a secondary camera or tactile sensors in the gripper, to verify successful grasping and placement. This would enhance system robustness and fault tolerance. In addition, extending the system to operate in semi-structured or cluttered environments would require the inclusion of real-time path planning algorithms and object tracking capabilities.

Finally, there is potential to scale the system for industrial applications by enabling multi-object sorting, dynamic conveyor integration, or cloud-based monitoring and diagnostics through Industrial IoT technologies.

5.6 Conclusion

In conclusion, this project achieved its objectives by developing and validating a modular, responsive, and fully autonomous robotic sorting system using affordable hardware and open-source tools. It contributes to the growing body of research focused on accessible robotic automation and provides a reliable framework for real-world tasks in controlled environments. The insights gained, along with the challenges addressed, establish a strong foundation for future research and development in intelligent, vision-based robotic systems.

REFERENCES

1. Hompel, M. and Schmidt, T. *Warehouse Management: Automation and Organisation of Warehouse and Order Picking Systems*. Springer. 2006. URL <https://books.google.com/books?hl=en&lr=&id=ja799ganvtAC&oi=fnd&pg=PA1>.
2. Ashok, A. and Ravi, N. R. Optimizing Automated Parcel Sorting in Logistics: Integration of Closed-Loop Overflow Management Systems. *Diva Portal*, 2024. URL <https://www.diva-portal.org/smash/record.jsf?pid=diva2:1897620>.
3. Richards, G. *Warehouse Management: A Complete Guide to Improving Efficiency and Minimizing Costs in the Modern Warehouse*. Kogan Page. 2017. URL <https://books.google.com/books?hl=en&lr=&id=bDw7DwAAQBAJ&oi=fnd&pg=PP1>.
4. Chen, J. C., Cheng, C. H., Huang, P. T. B. and Wang, K. J. Warehouse Management with Lean and RFID Application: A Case Study. *The International Journal of Advanced Manufacturing Technology*, 2013. 69(1-4): 531–542. doi:10.1007/s00170-013-5016-8. URL <https://link.springer.com/article/10.1007/s00170-013-5016-8>.
5. Åberg, H. Layout Redesign and Automation for a Lean Packing Process in a Distribution Center. *LUP Student Papers*, 2019. URL <https://lup.lub.lu.se/student-papers/record/8996438/file/8996439.pdf>.
6. Baker, P. and Halim, Z. An Exploration of Warehouse Automation Implementations: Cost, Service and Flexibility Issues. *Supply Chain Management: An International Journal*, 2007. 12(2): 129–138. doi:10.1108/13598540710737316. URL <https://www.emerald.com/insight/content/doi/10.1108/13598540710737316/full/html>.
7. Sodiya, E. O., Umoga, U. J. and Amoo, O. O. AI-Driven Warehouse Automation: A Comprehensive Review of Systems. *GSC Advanced Research and Reviews*, 2024. 2024(63). URL <https://gsconlinepress.com/journals/gscarr/content/ai-driven-warehouse-automation-comprehensive-review-systems>.
8. Golnabi, H. and Asadpour, A. Design and application of industrial machine vision systems. *Robotics and Computer-Integrated Manufacturing*, 2007. 23(6): 630–637. ISSN 0736-5845. doi:<https://doi.org/10.1016/j.rcim.2007.02.005>. URL <https://www.sciencedirect.com/science/article/pii/S0736584507000233>, 16th International Conference on Flexible Automation and Intelligent Manufacturing.
9. Alonso, V., Dacal-Nieto, A., Barreto, L., Amaral, A. and Rivero, E. Industry 4.0 implications in machine vision metrology: an overview. *Procedia Manufacturing*, 2019. 41: 359–366. ISSN 2351-9789. doi:<https://doi.org/10.1016/j.promfg.2019.09.020>. URL <https://www>.

- sciencedirect.com/science/article/pii/S2351978919311072, 8th Manufacturing Engineering Society International Conference, MESIC 2019, 19-21 June 2019, Madrid, Spain.
10. Ezhilarasan, G., Singh, V. and Upadhyay, R. Machine Vision for Measurement Accuracy in Industrial Automation. *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. 2024. 1–5. doi:10.1109/ICCCNT61001.2024.10725553.
 11. Mittal, N., Vaidya, A. and Kapoor, A. P. S. Object Detection and Classification Using Yolo. 2019. URL <https://api.semanticscholar.org/CorpusID:222226223>.
 12. Chandan, G., Jain, A., Jain, H. and Mohana. Real Time Object Detection and Tracking Using Deep Learning and OpenCV. *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*. 2018. 1305–1308. doi:10.1109/ICIRCA.2018.8597266.
 13. Dhaka, A. Social Distancing Detection Using Open CV and Yolo Object Detector. *International Journal for Modern Trends in Science and Technology*, 2021. 7: 93–95. doi:10.46501/IJMTST070121.
 14. Huang, Y., Luo, Y., Cao, Y., Lin, X., Wei, H., Wu, M., Yang, X. and Zhao, Z. Damage Detection of Unwashed Eggs through Video and Deep Learning. *Foods*, 2023. 12: 2179. doi:10.3390/foods12112179.
 15. Akbulut, Y. and Khalaf, R. Smart Arms Detection System Using YOLO Algorithm and OpenCV Libraries. *Turkish Journal of Science and Technology*, 2021. 16(1): 129–136.
 16. Ponika, M., Jahnavi, K., Sridhar, P. S. V. S. and Veena, K. Developing a YOLO based Object Detection Application using OpenCV. *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)*. 2023. 662–668. doi:10.1109/ICCMC56507.2023.10084075.
 17. Ullah, M. B. CPU Based YOLO: A Real Time Object Detection Algorithm. *2020 IEEE Region 10 Symposium (TENSYP)*. 2020. 552–555. doi:10.1109/TENSYP50017.2020.9230778.
 18. Han, Z. T., Mohd Zaman, M. H., Ibrahim, M. F. and M Moubark, A. Kinematic Analysis for Trajectory Planning of Open-Source 4-DoF Robot Arm. *International Journal of Advanced Computer Science and Applications*, 2021. 12. doi:10.14569/IJACSA.2021.0120690.
 19. Zakey, N., Zaman, M. and Ibrahim, M. Structural Optimization of 4-DOF Agricultural Robot Arm. *ResearchGate*, 2024. URL <https://www.researchgate.net/publication/381558407>.
 20. Khairuddin, A. and Zaman, M. Dimensional Optimization of Open-Source 4-DoF Robot Arm. *IEEE Access*, 2024. URL <https://ieeexplore.ieee.org/document/10690884>.
 21. Knapper, D. Detection of a Human Ready for Object Transfer to Facilitate Human-Robot Interaction, 2024. URL <https://fse.studenttheses.ub.rug.nl/id/eprint/33076>, bachelor Thesis.

22. Christie, B., Alred, C., Paquette, M. and Glaspell, G. *Increasing the Degrees of Freedom on a Robot Arm*. Technical report. US Army ERDC. 2023. URL <https://erdc-library.erdcdren.mil/bitstream/11681/47846/1/ERDC-GRL%20TR-23-4.pdf>.
23. Mohammed, A. and Sunar, M. Kinematics Modeling of a 4-DOF Robotic Arm. *IEEE*, 2015. URL <https://ieeexplore.ieee.org/document/7166008>.
24. Ting, H., Zaman, M. and Ibrahim, M. Kinematic Analysis for Trajectory Planning of Open-Source 4-DoF Robot Arm. *ResearchGate*, 2021. URL <https://www.researchgate.net/publication/353075915>.
25. Parhi, D., Deepak, B. and Nayak, D. Forward and Inverse Kinematic Models for an Articulated Robotic Manipulator. *ResearchGate*, 2012. URL <https://www.researchgate.net/publication/280112057>.
26. Kawecki, A., Dabrowski, P. and Januszko, S. AR Tags Based Absolute Positioning System. *IEEE*, 2022. URL <https://ieeexplore.ieee.org/document/9738534>.
27. Ho, C. and Lin, C. Pose-Based Visual Servoing with Lightweight Deep-Learning Binarization for Autonomous Mobile Robot Application. *IEEE*, 2023. URL <https://ieeexplore.ieee.org/document/10317548>.
28. Junior, L., Berger, G., Oliveira Junior, A. and Braun, J. A Comparison of Fiducial Markers Pose Estimation for UAVs Indoor Precision Landing. In: *Springer*. 2023. URL <https://www.researchgate.net/publication/377856920>.
29. Cong, V., Hanh, L., Phuong, L. and Duy, D. Design and development of robot arm system for classification and sorting using machine vision. *FME Transactions*, 2022. 50(1). doi:10.5937/fme2201181C. URL <https://dx.doi.org/10.5937/fme2201181C>.
30. El-Shair, Z. A. and Rawashdeh, S. A. Design of an object sorting system using a vision-guided robotic arm. 2019. URL <https://www.academia.edu/download/91752407/68.pdf>.
31. Prusaczyk, P., Kaczmarek, W., Panasiuk, J. and Besseghieur, K. L. Integration of robotic arm and vision system with processing software using TCP/IP protocol in industrial sorting application. *AIP Conference Proceedings*, 2019. 2060: 020014. doi:10.1063/1.5092035. URL <https://dx.doi.org/10.1063/1.5092035>.
32. Song, M. Research on Intelligent Logistics Sorting Robot Control Based on Machine Vision. 2024. doi:10.62227/as/74302. URL <https://dx.doi.org/10.62227/as/74302>.
33. Ye, Z., Li, Y., Zhang, Z. *et al.* Research on campus logistics intelligent sorting system using robots. 2022. doi:10.1145/3558819.3565233. URL <https://dx.doi.org/10.1145/3558819.3565233>.

34. Dahal, B. Automated Retail Billing: Streamlining Checkout with QR Codes and Object Tracking Using YOLOv8 and DeepSORT. *International Journal of Scientific Engineering and Technology*, 2024. 12(5). doi:10.61463/ijset.vol.12.issue5.253. URL <https://dx.doi.org/10.61463/ijset.vol.12.issue5.253>.
35. Huang, M. and Li, Y. Express Sorting System Based on Two-Dimensional Code Recognition. 2018. doi:10.1109/snsp.2018.00075. URL <https://dx.doi.org/10.1109/snsp.2018.00075>.
36. Han, S. and Yang, M. Smart Parcel Sorting Control System Based on QR Code Recognition. 2024. doi:10.1145/3672758.3672774. URL <https://dx.doi.org/10.1145/3672758.3672774>.
37. Pan, Z., Jia, Z., Jing, K., Ding, Y. and Liang, Q. Manipulator Package Sorting and Placing System Based on Computer Vision. *2020 Chinese Control And Decision Conference (CCDC)*. 2020. 409–414. doi:10.1109/CCDC49329.2020.9164071.
38. ROBOTIS. *OpenMANIPULATOR-X Overview*. ROBOTIS Co., Ltd. URL https://emanual.robotis.com/docs/en/platform/openmanipulator_x/overview/, accessed: 2025-01-14.
39. Mirhnius. *OpenmanipulatorX-Robot*. <https://github.com/mirhnius/OpenmanipulatorX-Robot>, 2025. Accessed: 2025-01-14.
40. Robotis. Introduction. URL https://emanual.robotis.com/docs/en/parts/interface/u2d2_power_hub/.
41. Kumar, G., Raja, D. and Suresh, S. Vision-Guided Pick and Place Systems Using Raspberry Pi and YOLO. 2024. URL <https://ieeexplore.ieee.org/abstract/document/10731108/>.
42. Dos Reis, D. and Welfer, D. Mobile robot navigation using an object recognition software with RGBD images and the YOLO algorithm. 2019. doi:10.1080/08839514.2019.1684778. URL https://app.scholarai.io/paper?paper_id=DOI:10.1080/08839514.2019.1684778.
43. Khanous, B., Manno, L. and Mollat, M. Enhancing Remote Control Capabilities of ROS-based Robots via 5G Connectivity for Immersive Teleoperation and Object Detection. 2024. URL <https://www.diva-portal.org/smash/record.jsf?pid=diva2:1877904>.
44. Brokstad, O. The Cyborg v4. 0-Computer Vision Module: Towards a Socially Intelligent Robot. 2020. URL <https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2780903>.