DYNAMIC PICK-AND-PLACE SYSTEM FOR A MANIPULATOR ON A QUADRUPED USING OBJECT DETECTION

BY

ABHISHEK JAGADEESH KASARAGOD
B.E., Sahyadri College of Engineering and Management, 2021
M.S., University of Illinois Chicago, 2024

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Chicago, 2024

Chicago, Illinois

Defense Committee:

Dr. Pranav Bhounsule, Chair and Advisor
Dr. Michael J. Scott, Department of Mechanical Engineering
Dr. Jonathan Komperda, Department of Mechanical Engineering

ACKNOWLEDGMENT

I wish to convey my sincere gratitude to my advisor and mentor, Professor Pranav Bhounsule, for his unwavering support, wisdom, and encouragement all throughout this journey.

I deeply appreciate my committee members, Professor Michael Scott and Professor Jonathan Komperda, for being a part of my thesis committee and for their valuable feedback and encouragement.

I express my deep gratitude to each member of my laboratory for their valuable contributions and companionship. I gained invaluable knowledge from each of them, participating in profound research discussions and sharing some of the most enjoyable recollections. I am especially grateful to Chun-Ming Yang, Daniel Torres, Ernesto Hernandez, James Bittler, Prashanth Chinthalapati, Safwan Mondal, Salvador Echeveste, Subramanian Ramasamy for their unwavering support and cooperation.

Finally, I would like to thank my family. This journey would not have been possible without the unwavering support of my brother and parents, who have always provided me with guidance throughout my life. Their presence and encouragement have been instrumental in making this journey possible. Thank you all for everything.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The growing demand for versatile and efficient robotic systems in industries such as logistics, manufacturing, and service robotics underscores the importance of advancing mobile manipulation technologies. These systems must navigate and operate within unstructured and dynamic environments, requiring seamless integration of locomotion, manipulation, and perception capabilities. This thesis presents the development of a dynamic pick-and-place system for a manipulator mounted on a quadruped robot. The system employs real-time object detection using a custom-trained model, which leverages synthetic images generated in Blender to enable precise identification and tracking of objects. A key feature is the system's ability to perform real-time detection, sorting of alphabets and numbers, and manipulation of multiple objects, showcasing its capability to dynamically track and handle varying items. The manipulator uses inverse kinematics and closed-loop control algorithms to adjust its movements in real-time, ensuring high precision in complex and changing environments. To verify the system's performance, the sorting results were compared between actual positions and motion capture (mocap) data, while the number of successful real-time object tracking attempts was recorded. These evaluations demonstrated that the system is robust and reliable in dynamic environments, advancing mobile manipulation for automated tasks in logistics, manufacturing, and other applications requiring high adaptability and precision. The combination of advanced computer vision, robust control algorithms, and real-time adaptability positions this system as a significant advancement in the field of mobile manipulation robotics.

CHAPTER 1

INTRODUCTION

This chapter introduces the work developed in this thesis, divided into four sections. Section 1.1 provides context for the thesis topic and an overview of the current research status. Section 1.2 explains the motivations behind defining the thesis. Section 1.3 outlines the dissertation's objectives. Lastly, Section 1.4 details the dissertation's structure.

1.1 Context

The rapid evolution of robotics technology is driving significant advancements in various industrial sectors, including logistics, manufacturing, and service robotics. As industries seek to enhance efficiency and productivity, the demand for versatile and efficient robotic systems has grown substantially. Among the most promising areas of development is mobile manipulation, which combines locomotion and manipulation capabilities to enable robots to navigate and operate within unstructured and dynamic environments. Mobile manipulation technologies are essential for performing complex tasks that require the robot to move and interact with objects simultaneously. These systems must seamlessly integrate locomotion, manipulation, and perception capabilities to function effectively in real-world settings. The ability to dynamically interact with and manipulate objects in real-time opens up new possibilities for automation, making mobile manipulation a critical area of research and development. The image in Figure 1 depicts Boston Dynamics' Spot robot equipped with a manipulator arm, highlighting its ability to navigate and perform complex tasks in industrial environments.

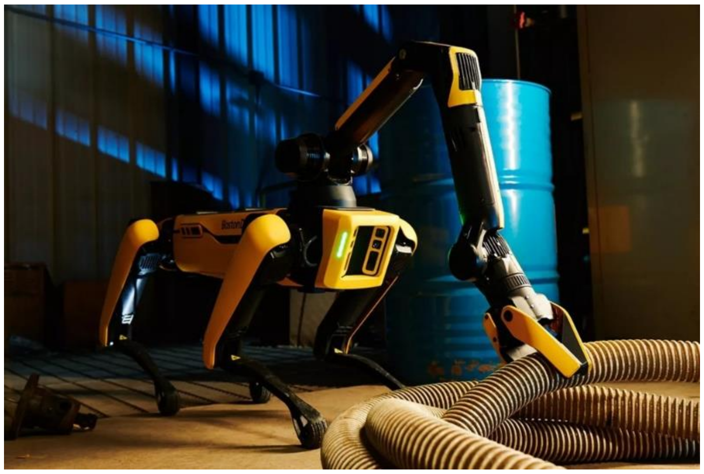


Figure 1. Boston Dynamics' Spot

Despite significant progress in the fields of manipulation and locomotion, mobile manipulation remains a formidable challenge. Traditional robotic systems often focus on either static manipulation or locomotion, lacking the coordination necessary to perform tasks that require both. This lack of integration leads to issues such as compounding errors, delays in decision-making, and a general lack of efficiency and precision. Addressing these challenges requires innovative solutions that can bring together the various aspects of mobile robotics into a cohesive and functional system.

1.2 Motivation

The integration of robotics into industry, defense, agriculture, and construction addresses critical challenges through the combination of quadrupeds and robotic arms. In various sectors, these advanced robotic systems offer tailored solutions to specific problems. The manufacturing sector is currently challenged by labor shortages, the demand for higher precision, and the need to meet increased production demands. Quadruped robots, equipped with robotic arms, can navigate complex factory environments, carrying out tasks such as assembly, quality inspection, and material handling. Their mobility allows them to access areas difficult for fixed robots, and their arms provide the dexterity needed for precise operations, improving efficiency and reducing downtime.

In defense operations, which involve high-risk tasks such as bomb disposal, reconnaissance in hazardous areas, and logistical support in difficult terrains, quadruped robots with robotic arms can perform bomb disposal, handle hazardous materials, and conduct reconnaissance missions safely. Their ability to traverse rough terrains and their manipulative capabilities make them ideal for transporting supplies, setting up equipment, and providing real-time surveillance, thus enhancing soldier safety and mission effectiveness.

The agriculture sector struggles with labor shortages, the need for sustainable practices, and increasing food demand. Quadruped robots equipped with robotic arms can automate labor-intensive tasks such as planting, weeding, and harvesting. Their mobility allows them to move across uneven fields, while their arms can handle delicate crops, apply pesticides precisely, and monitor plant health. This enhances productivity, reduces the reliance on human labor, and promotes sustainable farming practices.

In the construction industry, quadruped robots are being deployed to address issues such as worker safety, efficiency, and the complexity of modern building sites. These robots can navigate uneven terrain, perform inspections, and assist in tasks like material placement and site surveying. The integration of robotic arms further enables them to handle tools, perform precise construction tasks, and reduce the risk of injuries on-site. This combination of mobility and manipulation enhances overall productivity and safety in construction projects.Quadruped robots have emerged as a promising solution to the limitations of traditional wheeled or tracked robots. These robots possess remarkable agility and adaptability, allowing them to traverse rough terrains with minimal impact on the land. Equipping these robots with manipulation capabilities has the potential to greatly enhance the operations and decrease the need for manual labor.

This thesis presents the development of a dynamic pick-and-place system for a manipulator mounted on a Unitree A1 quadruped robot, incorporating advanced object detection using a custom-trained YOLOv5s model, precise control algorithms, trajectory planning, and real-time feedback mechanisms to enable efficient and accurate real-time object tracking and sorting in dynamic and unstructured environments.

1.3 Objectives

Given the context and motivation outlined in the previous sections, the primary objective of this dissertation is to develop a quadruped robot capable of teleoperated navigation and precise task execution using an attached robotic arm. To tackle the earlier mentioned challenges, this approach involves utilizing the Unitree A1 quadruped robot and augmenting its capabilities with a Trossen Robotics WidowX 250S robotic arm. The following steps will be pursued to achieve this goal:

1. Conduct a comprehensive literature review on quadruped manipulators and object detection.
2. Develop a system to analyze and test the arm's behavior, kinematic configurations, and object detection capabilities.
3. Implement control algorithms and trajectory planning.
4. Train and fine-tune the YOLOv5s object detection model using synthetic and real-world datasets.

5. Construct, integrate, and validate the complete system, including teleoperation and autonomous modes.

1.4 Document Structure

In addition to this introductory chapter, this document comprises seven additional chapters:

- Chapter 2 provides an in-depth review of existing literature on quadruped robots equipped with manipulation capabilities and object detection technologies.
- Chapter 3 details the methodologies used in developing the system, including control algorithms, trajectory planning, and synthetic data generation for object detection training.
- Chapter 4 describes the hardware components and experimental setup, including the integration of the Jetson Nano, U2D2 microcontroller, and Unitree A1 quadruped robot.
- Chapter 5 presents the results of the experiments, including object detection accuracy, sorting efficiency, and a comparison of expected vs. actual trajectory tracking for the manipulator.
- Chapter 6 discusses the main conclusions drawn from this research and suggests future work to further enhance the system's capabilities.

CHAPTER 2

LITERATURE REVIEW

This section provides an overview of the existing research and technological advancements in the field of mobile manipulation, object detection, and robotic control systems. Section 2.1 examines the current techniques used to enhance the ability of quadruped robots to manipulate objects. Following that, Section 2.2 discusses the different approaches to generating images and annotations for a custom dataset and Section 2.3 outlines various methods for object detection.

2.1 Quadruped robots with manipulation capabilities

Legged robots outperform wheeled, tracked, and airborne robots when it comes to navigating challenging terrain and other forms of unpredictable settings. Equipping these robots with manipulation capabilities has the potential to greatly enhance the operations and decrease the need for manual labor. Quadruped robots, popularized by companies such as Boston Dynamics and ANYbotics, have garnered significant attention due to their unique capabilities and versatility across various real-world applications. Unlike wheeled or tracked robots, quadrupeds excel at navigating rough and unstructured terrains, thanks to their ability to climb, step, and crawl. Equipped with sophisticated sensors like high-definition cameras, thermal cameras, lidars, and gas sensors, these robots can autonomously navigate and detect objects, making them suitable for numerous practical applications. One of the most documented uses of quadruped robots is in remote inspection, where their capability to traverse difficult terrains allows them to inspect hazardous or repetitive environments, such as buildings for gas leaks and nuclear sites for contamination. In the construction industry, quadruped robots are used for oversight and surveying, moving across unpredictable and multi-leveled work sites, capturing images, and building 3D maps to assess construction progress and maintain project schedules. They also address the limitations of traditional wheeled robots in delivery applications by transporting materials across unstructured terrains, reducing the risk of injury and increasing operational efficiency. Security applications for quadruped robots include acting as automated security guards, patrolling buildings and outdoor areas, and alerting authorities to intruders. In law enforcement, they are utilized for tasks such as bomb disposal and managing dangerous situations like hostage scenarios, reducing risk to personnel. Additionally, quadruped robots are highly effective in search and rescue operations, especially in the aftermath of natural disasters, where they navigate debris, provide situational awareness, and map affected areas, accelerating rescue efforts and saving lives. Despite their advantages, challenges such as battery life and object detection in complex environments remain. The autonomy systems of quadruped robots are advanced but still require human oversight, with remote teleoperation being crucial for their operational and economic viability. Effective robot operations platforms enable one human to manage multiple robots, improving cost efficiency and operational scalability. This literature review highlights the diverse applications of quadruped robots in inspection, construction, delivery, security, law enforcement, and search and rescue, emphasizing the ongoing advancements in autonomy and teleoperation systems to overcome current limitations and enhance their effectiveness across various sectors [1].

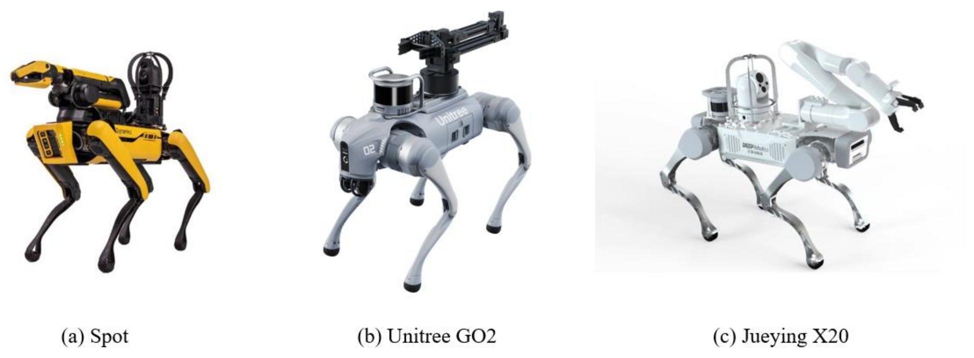


Figure 2. Examples of commercialized quadruped manipulators

Several firms that market quadruped robots have initiated the creation of robotic arm attachments to incorporate with their devices. For instance, Boston Dynamics has developed an arm add-on for their well-known Spot robot. This arm enhances Spot's capabilities, allowing it to perform a variety of tasks such as opening doors, picking up objects, and manipulating tools, thereby expanding its utility in numerous applications from industrial inspections to household chores (Figure 2a). Similarly, Unitree has introduced the GO2 robot equipped with a versatile arm. The GO2's arm enables it to handle complex manipulation tasks, such as precise assembly, material handling, and interaction with various objects in unstructured environments, making it a valuable asset in fields like construction, logistics, and research (Figure 2b). Additionally, DeepRobotics has developed the Jueying X20, which features an advanced robotic arm designed for intricate tasks. The Jueying X20's arm can perform delicate operations, such as electronic component handling, intricate repairs, and detailed inspections, further broadening the scope of what quadruped robots can achieve in specialized industrial settings (Figure 2c).

Sereinig et al. [2] provide a comprehensive review of the challenges and advancements in mobile manipulation, particularly focusing on the integration of manipulator arms with mobile platforms for enhanced flexibility and functionality. The paper discusses various systems and their applications in unstructured environments, highlighting the importance of robust system design for tasks such as disaster response and industrial automation. The authors examine different robotic platforms, including tracked and omnidirectional vehicles, and their respective control mechanisms to navigate and manipulate objects in complex settings. They also explore the development and deployment of mobile rescue robots, such as T.R.U.D.I., and industrial manipulators, such as the KAIROS robot, emphasizing the need for precise control and motion planning to ensure effective operation in diverse scenarios. The review underscores the potential of mobile manipulators to revolutionize fields like search and rescue, agriculture, and manufacturing through advanced robotics and control technologies.

Xin et al. [3] explores advanced control strategies for quadruped robots equipped with robotic arms, focusing on the interaction between the arm and the quadrupedal platform using a whole-body controller (WBC). They present an optimization-based dynamic WBC to manage both the robot's stability and its manipulation capabilities simultaneously. This controller enables the robot to execute multiple tasks by utilizing the system's redundancy, ensuring that both locomotion and manipulation functionalities are maintained effectively.

Ulloa et al. [4] present a mixed-reality tele-operation method for high-level control of legged-manipulator robots, focusing on enhancing operator control and situational awareness using devices like Hololens. The proposed method optimizes the robot's workspace in Matlab and simulates dynamic interactions in Gazebo. The study demonstrates significant improvements in operator efficiency and decision-making during search and rescue operations compared to conventional interfaces.

Fu et al. [5] propose a unified policy for whole-body control of legged robots with manipulators using reinforcement learning (RL). This approach addresses the simultaneous control of locomotion and manipulation, involving high degrees of freedom and dynamic interactions between the robot's legs and arm. The key contributions include a single curriculum parameter that mixes advantage functions for arm (manipulation) and leg (locomotion) actions to streamline learning, and Regularized Online Adaptation to bridge the Sim-to-Real gap by predicting environmental extrinsics from onboard observations. This method was validated on a Unitree Go1 quadruped robot with an Interbotix WidowX 250s arm, showing superior performance in various metrics compared to separate and uncoordinated policies, thus enhancing the robot's efficiency and stability during tasks.

Ferrolho et al. [6] present a robust loco-manipulation framework for quadruped robots equipped with arms, focusing on enhancing motion robustness against disturbances. The study proposes an optimization-based approach that integrates the dynamics of the manipulated object into the robot's centroidal dynamics and full kinematics, allowing for effective exploitation of base-limb coupling. The framework emphasizes proactive robustness by incorporating uncertainties at the planning stage, enabling the generation of robust trajectories that withstand external forces. This work extends previous research by introducing a bilevel trajectory optimization problem to maximize robustness during motion with contact changes, demonstrating significant improvements over traditional reactive robustness methods.

Gai et al. [7] explores continual reinforcement learning (RL) for quadruped robot locomotion, addressing the challenge of maintaining performance across sequential tasks while avoiding catastrophic forgetting and loss of plasticity. The proposed method employs the Piggyback algorithm to protect critical parameters for each task and reinitializes unused parameters to enhance plasticity. The approach encourages policy network exploration by maximizing entropy. Experiments validate the method's effectiveness, demonstrating improved stability and adaptability compared to traditional continual learning algorithms. This research offers significant advancements in enabling quadruped robots to learn multiple tasks continuously in varying environments.

1.2 Synthetic Data Generation and Domain Randomization

Tremblay et al. [8] explore the use of synthetic data to train deep neural networks, aiming to bridge the reality gap through domain randomization. The study proposes generating diverse, randomized synthetic datasets to expose neural networks to a wide range of variations during training. This approach enhances the robustness and generalization capabilities of models when applied to real-world tasks. The authors demonstrate the effectiveness of this method by training object detection models on synthetic data and evaluating their performance on real-world datasets, showing significant improvements. The paper highlights the potential of synthetic data generation as a cost-effective and scalable solution for training deep learning models in various applications.

Borrego et al. [9] present a study on applying domain randomization to synthetic data for object category detection. The authors address the challenge of creating robust object detection models by leveraging synthetic datasets with varied and algorithmically generated patterns. This approach aims to bridge the gap between simulated and real-world domains, making the models perceive the transition as a minor disturbance. The study uses a Single-Shot Detector (SSD) as the base object detector, enhanced with MobileNet for feature extraction. The paper details optimizations to the Gazebo simulation environment, which significantly improve scene composition and texture generation performance. The experiments demonstrate the impact of different synthetic texture patterns on detection accuracy, and the results highlight the effectiveness of domain randomization in enhancing model robustness across multiple classes.

Tobin et al. [10] explore the use of domain randomization for training deep neural networks to bridge the reality gap between simulated environments and real-world applications. By randomizing the rendering parameters in simulations, the authors create a wide variety of training scenarios, enhancing the model's ability to generalize to real-world data. This technique allows neural networks trained on synthetic data to perform effectively in real-world tasks, demonstrating significant potential for applications in robotics.

1.3 Object Detection

YOLOv5 is an open-source real-time object detection system developed by Ultralytics. YOLOv5 leverages the advancements in the YOLO (You Only Look Once) family of models, focusing on speed and accuracy for object detection tasks. This implementation is highly efficient and user-friendly, featuring significant improvements in performance and ease of use compared to its predecessors. The repository provides pre-trained models, training scripts, and deployment solutions, making it accessible for various applications in computer vision. [11]

Ren et al. [12] introduce Faster R-CNN, a groundbreaking object detection framework that integrates Region Proposal Networks (RPNs) with Fast R-CNN. The RPNs generate high-quality region proposals, which are then refined by Fast R-CNN for object detection. This architecture significantly improves detection speed and accuracy by sharing convolutional features. The authors demonstrate that Faster R-CNN outperforms previous state-of-the-art methods on multiple benchmarks, establishing it as a robust solution for real-time object detection tasks.

Redmon and Farhadi [13] present YOLOv3, an incremental improvement to the YOLO object detection framework. YOLOv3 employs a deeper network, Darknet-53, and introduces multi-scale predictions to enhance detection accuracy, particularly for small objects. It also uses logistic regression for class prediction and binary cross-entropy loss for multi-label classification. The model achieves state-of-the-art performance on the COCO dataset, balancing speed and accuracy effectively.

Liu et al. [14] introduce SSD (Single Shot MultiBox Detector), a method for object detection that eliminates proposal generation and subsequent pixel or feature resampling stages. SSD discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. During prediction, SSD scores the presence of each object category in each default box and adjusts the box to better match the object shape. The method achieves competitive accuracy with significantly faster speed.

CHAPTER 3

METHODS

This section details the various methodologies employed in this research. Section 3.1 describes the kinematic modeling of the robotic arm, outlining the mathematical formulations and transformations used. Section 3.2 focuses on trajectory planning, discussing the algorithms and methods for generating optimal paths for the robotic arm. In Section 3.3, the simulation with MuJoCo is elaborated, highlighting the creation and benefits of the MuJoCo model for pre-implementation testing. Section 3.4 covers synthetic image generation, explaining the process of creating and annotating synthetic images for training object detection models. Finally, Section 3.5 discusses object detection method used to enable the robot to identify and locate objects within its environment.

3.1 Kinematic Modelling

Kinematics is a branch of mechanics focused on studying the motion of bodies and structures without considering the forces causing them. In robotics, kinematics analysis focuses on the relationship between the robot arm's links and joints with its position, orientation, and acceleration. This analysis employs geometry to study the movement of multi-DoF kinematic chains which is crucial for planning the robot arm's trajectory. Kinematics is divided into two main types: forward kinematics, which defines the position and orientation of the robot based on its joint parameters, and inverse kinematics, which determines the necessary joint parameters to achieve a desired position and orientation. For this research, the Widow X 250S manipulator by Trossen Robotics is utilized. Although this robotic arm is equipped with six degrees of freedom (DOF), only four DOF are employed for the purposes of this study. This simplification is made to reduce the complexity of control and programming, while still ensuring that the manipulator can effectively perform the required tasks. [15, 16]

3.1.1 Forward kinematics. Forward kinematics involves using the kinematic equations of a robotic arm to determine the position and orientation of the end-effector based on specified joint angles. One of the most common methods used in forward kinematics is the Denavit-Hartenberg (DH) method [17], which represents the relationship between the joint coordinates of two links. Compared to inverse kinematics, which calculates the joint parameters required to achieve a desired end-effector position and is generally more complex, forward kinematics provides a simpler and more straightforward solution.

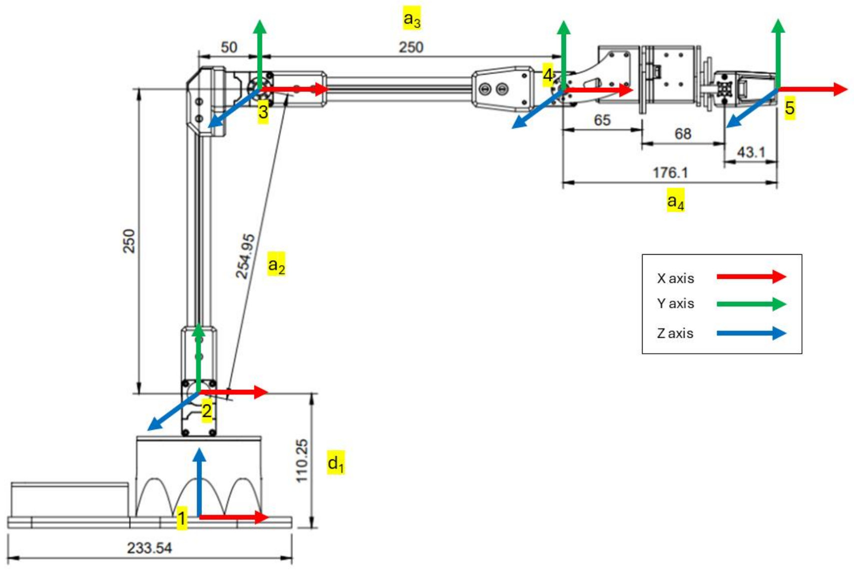


Figure 3. Configuration of the Robot Arm

The dimensions and Denavit-Hartenberg (DH) frames of the WidowX 250s are illustrated in Figure 1. The DH convention is a popular convention to represent the kinematics of robot manipulators. In the Denavit-Hartenberg (DH) convention, link notation is used to describe the spatial relationships between connected joints. The parameters a, α, d, and θ follow the standard Denavit-Hartenberg (DH) convention, where θ_1, θ_2, θ_3, and θ_4 represent the joint revolute angles. The notation $\theta_{i,0}$ refers to the angle of the i^{th} joint in its initial configuration, as depicted in Figure 3. The frame associated with each joint defines its position and geometric relationship relative to the adjacent joint. It is given by:

$$H_i^{i-1} = H_z(\theta_i)\, H_z(d_i)\, H_x(a_i)\, H_x(\alpha_i) \tag{3.1}$$

Where,

1. a_i is called link length which is the distance between z_i and z_{i-1} along x_i.
2. α_i is called link twist which is the angle between z_i and z_{i-1} along x_i.
3. d_i is called link offset and the distance between x_{i-1} and x_i along z_{i-1}.
4. θ_i is called joint angle which is the angle between x_{i-1} and x_i along z_{i-1}.

DH parameters of robot are given in Table 1.

Table 1. DH parameters of WidowX 250s manipulator.

Link	$a_i\ (m)$	$\alpha_i\ (°)$	$d_i\ (m)$	$\theta_i\ (°)$
1	0	90	0.11025	θ_1
2	0.25495	0	0	θ_2
3	0.25	0	0	θ_3
4	0.17415	0	0	θ_4

Once the D-H parameters are determined, they are incorporated into the transformation matrix. Given that 4 DOF are used, the resulting linked matrix is H_0^4. The equations for the transformation matrix can be formulated as follows:

$$H_i^{i-1} = \begin{bmatrix} c\theta_i & -s\theta_i c\alpha_i & s\theta_i s\alpha_i & a_i c\theta_i \\ s\theta_i & c\theta_i c\alpha_i & -c\theta_i s\alpha_i & a_i s\theta_i \\ 0 & s\alpha_i & c\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

Where $s\theta_i = \sin\theta_i$, $c\theta_i = \cos\theta_i$, $s\alpha_i = \sin\alpha_i$, $c\alpha_i = \cos\alpha_i$.

The position and orientation of the end-effector is found using the formula:

$$H_4^0 = H_1^0\, H_2^1\, H_3^2\, H_4^3 = \begin{bmatrix} R_4^0 & d_4^0 \\ 0 & 1 \end{bmatrix} \tag{3.3}$$

The position of the end-effector is d_4^0 and the orientation is R_4^0.

3.1.2 Inverse kinematics. Inverse Kinematics (IK) is used to determine the necessary joint angles for the robotic arm to achieve a specified position and orientation of its end-effector. In this work, a geometric approach was employed to solve the inverse kinematics for the robotic arm [18, 19]. Each joint angle can be calculated by assuming the position given. It is assumed that $\theta_{234} = \theta_2 + \theta_3 + \theta_4$. To keep the end-effector parallel to the ground, θ_{234} is considered to be 0. θ_1 can be calculates as:

$$\theta_1 = \tan^{-1}\left(\frac{p_y}{p_x}\right) \tag{3.4}$$

The angle for θ_1 ranges from -180° and 180°.

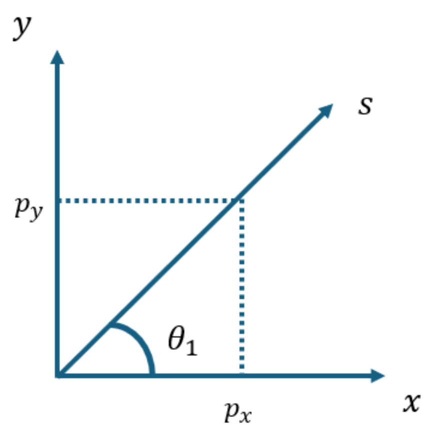


Figure 4. Combination of *x* and *y* axis as *s*-Axis

The *x*-coordinate and *y*-coordinate of the end-effector are combined into the *s*-coordinate using the Pythagorean theorem, as follows:

$$p_s{}^2 = p_x{}^2 + p_y{}^2 \tag{3.5}$$

$$p_s = \sqrt{p_x{}^2 + p_y{}^2} \tag{3.6}$$

The r and z coordinates for joint 3 can be calculated as follows:

$$s_3 = p_r \tag{3.7}$$

$$z_3 = p_z - d_1 \tag{3.8}$$

θ_2, θ_3, and θ_4 can be calculated using the following equations:

$$s_2 = s_3 - a_4 \cos\theta_{234} \tag{3.9}$$

$$z_2 = z_3 - a_4 \sin\theta_{234} \tag{3.10}$$

$$\cos\theta_3 = \left(\frac{s_2{}^2 + z_2{}^2 - (a_2{}^2 + a_3{}^2)}{2a_2a_3}\right) \tag{3.11}$$

$$\theta_3 = \pm\cos^{-1}\left(\frac{s_2{}^2 + z_2{}^2 - (a_2{}^2 + a_3{}^2)}{2a_2a_3}\right) \tag{3.12}$$

$$\cos\theta_2 = \left(\frac{(a_2 + a_3 \cos\theta_3)s_2 + (a_3 \sin\theta_3)z_2}{r_2{}^2 + z_2{}^2}\right) \tag{3.13}$$

$$\sin\theta_2 = \left(\frac{(a_2 + a_3 \cos\theta_3)z_2 + (a_3 \sin\theta_3)s_2}{r_2{}^2 + z_2{}^2}\right) \tag{3.14}$$

$$\theta_2 = \tan^{-1}\left(\frac{\sin\theta_2}{\cos\theta_2}\right) \tag{3.15}$$

$$\theta_4 = \theta_{234} - (\theta_2 + \theta_3) \tag{3.16}$$

Based on the configuration of the robot arm, the angle range for θ_2 is adjusted to between 0° and 180°, and the angle range for θ_3 is adjusted to between -180° and 0° and angle range for θ_4 is between -90° and 90°.

3.2 Trajectory Planning

Trajectory planning is a crucial aspect of robotic arm control that ensures smooth and efficient movement from an initial position to a target position. In this project, trajectory planning involves calculating the optimal path for the robotic arm to follow, considering constraints such as joint limits, and the need for smooth motions[20]. This section outlines the methods and algorithms used for trajectory planning in the context of the WidowX robotic arm. The initial position of the robotic arm is determined based on its current joint angles. The target position is specified as a set of 3D coordinates obtained from the object detection and 3D coordinate transformation process. Intermediate waypoints are generated between the initial and target positions to guide the robotic arm along a smooth path. These waypoints ensure that the arm moves in a controlled manner, avoiding sudden changes in direction or speed. Once the trajectory is planned, the next step is to execute the trajectory by controlling the robotic arm's joints. For each waypoint in the planned trajectory, the inverse kinematics (IK) algorithm calculates the required joint angles to position the end effector at the specified 3D coordinates.

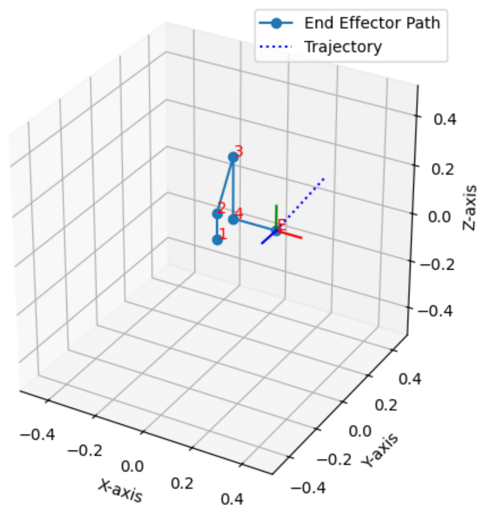


Figure 5. Plot of Trajectory of End Effector

To ensure smooth and controlled movements, velocity and acceleration profiles are applied to the controller. These profiles define the maximum speed and acceleration for each joint, preventing abrupt changes in motion and reducing wear on the motors. The calculated joint angles are sent as commands to the motors of the robotic arm. During trajectory execution, the system continuously monitors the position and status of the robotic arm. Real-time control is essential for the effective operation of the WidowX robotic arm, ensuring that it can respond dynamically to changes in its environment and perform tasks with high precision. The Intel RealSense depth camera provides real-time depth and RGB data, which are crucial for detecting objects and determining their spatial coordinates. To smooth the trajectory and reduce noise in the detected positions, a low-pass filter is applied to the coordinates. A low-pass filter is a signal processing technique used to allow signals with a frequency lower than a certain cutoff frequency to pass through while attenuating signals with frequencies higher than the cutoff frequency. The equation used for the low-pass filter in the code is:

$$coord_{output} = \alpha \cdot coord_{current} + (1 - \alpha) \cdot coord_{previous} \tag{3.17}$$

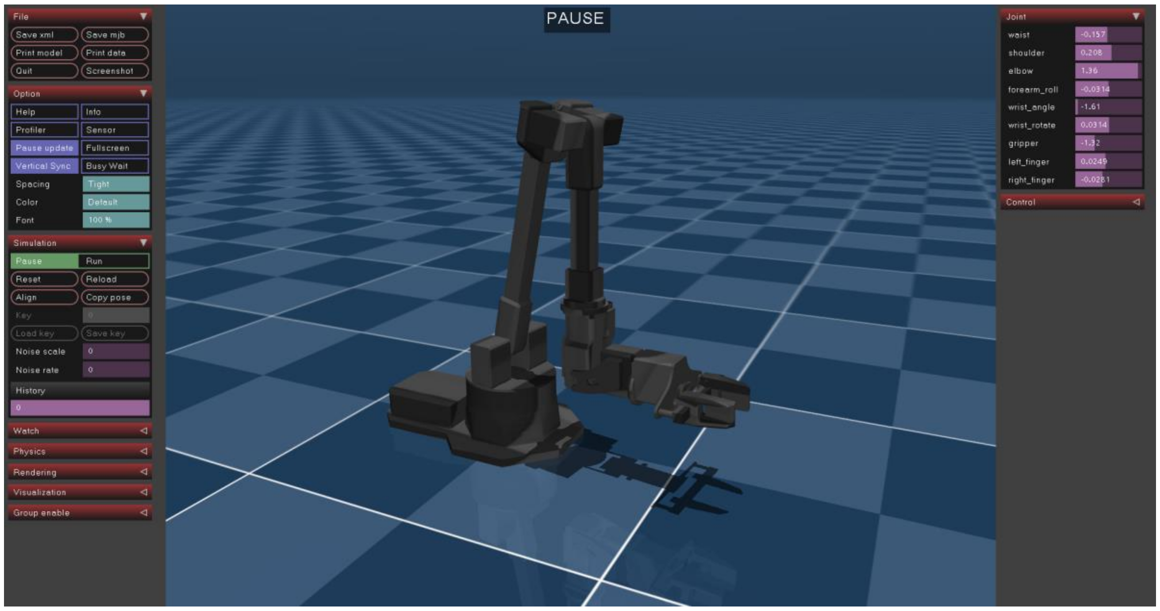
where:

- $coord_{output}$ is the filtered output at the current step.
- $coord_{current}$ is the new input value (the latest 3D coordinate) at the current step.
- $coord_{previous}$ is the filtered output from the previous step.
- α is the smoothing factor, set to 0.1 in the code, which provides a balance between responsiveness and smoothing. A higher α would result in a faster response but less smoothing.

It helps to smooth out the trajectory and reduce noise in the detected positions, ensuring stable and precise movements. This helps in stabilizing the arm's movement by filtering out sudden changes in the detected coordinates, leading to more accurate and stable control. Real-time monitoring detects any deviations from the planned path, allowing the system to adjust joint commands and correct the arm's position promptly.

3.3 Simulation with MuJoCo

To validate the kinematic model and ensure the proper functioning of the robotic arm before implementing it on physical hardware, the MuJoCo (Multi-Joint dynamics with Contact) physics engine was employed. MuJoCo is a high-performance simulation framework that allows for the precise modeling of complex, articulated structures and their interactions with the environment.



Spacing Tight
Color Default
Font 100 %
Noise scale 0
Noise rate 0
History 0

waist	-0.157
shoulder	0.208
elbow	1.36
forearm_roll	-0.0314
wrist_angle	-1.61
wrist_rotate	0.0314
gripper	-1.32
left_finger	0.0249
right_finger	-0.0281

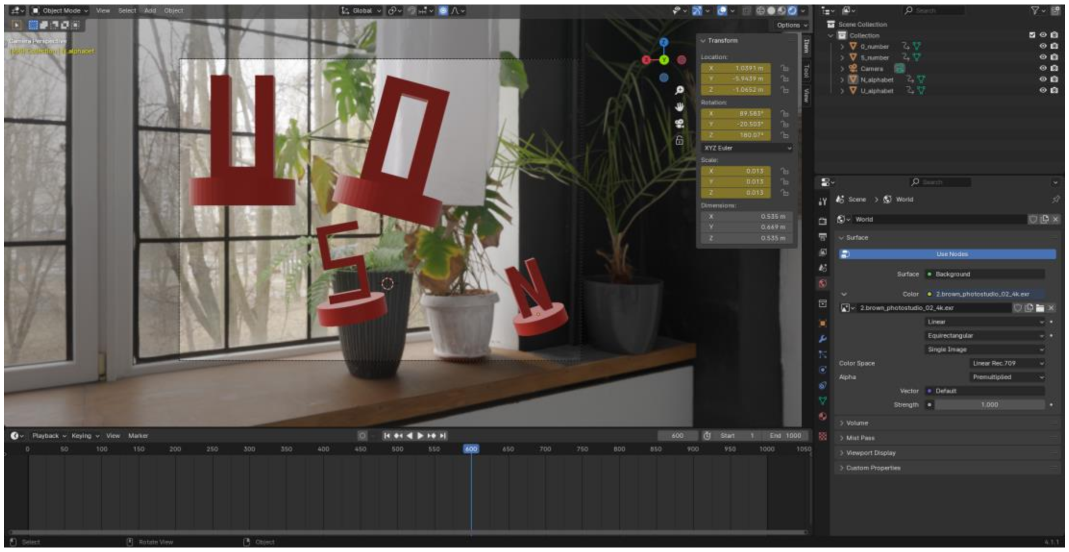
Figure 6. MuJoCo Simulation of the Robotic Arm

Various experiments were conducted in MuJoCo to validate the kinematic model. The simulated end-effector positions were compared with theoretical calculations for different joint angles, ensuring the accuracy of the forward kinematics model. The inverse kinematics algorithm was tested by setting target end-effector positions and verifying that the computed joint angles achieved these positions in the simulation.

3.4 Synthetic Image Generation

Synthetic image generation is a crucial technique used in this project to create a robust dataset for training the YOLOv5s model. By generating synthetic images, we can produce a large and diverse set of training data, which enhances the model's ability to detect and recognize objects in various conditions. This approach is particularly useful when real-world data is limited or difficult to obtain. Generating synthetic images is significantly more cost-effective than collecting and annotating real-world data, especially when large datasets are required. The ability to control every aspect of the synthetic environment allows for the creation of specific scenarios that might be rare or difficult to capture in real life. Synthetic data can be easily augmented to include variations in objects and environments, enhancing the diversity of the training set and improving the model’s performance.

3.4.1 3D Modeling and Scene Creation. The initial 3D models of the objects were created using SolidWorks. These models accurately represent the objects that the robotic arm will interact with, including their shapes, textures, and colors. The 3D models created in SolidWorks were exported as STL files, which were then imported into Blender for further processing and rendering. This workflow ensures that the models maintain high fidelity and accuracy. The objects 3D printed for this project include the letters "U" and "N" and the numbers "0" and "5". These printed objects were used for real-world validation of the model's detection capabilities.



Transform
Location:
X 1.0391 m
Y -5.9439 m
Z -1.0652 m
Rotation:
X 89.583°
Y -20.503°
Z 180.07°
XYZ Euler
Scale:
X 0.013
Y 0.013
Z 0.013
Dimensions:
X 0.535 m
Y 0.669 m
Z 0.535 m

World
Surface
Use Nodes
Surface: Background
Color: 2.brown_photostudio_02_4k.exr
2.brown_photostudio_02_4k.exr
Linear
Equirectangular
Single Image
Color Space: Linear Rec.709
Alpha: Premultiplied
Vector: Default
Strength: 1.000

Figure 7. 3D models created in SolidWorks

3.4.2 Image and Annotation Generation. Blender is an open-source 3D modeling and rendering software, used to create detailed and realistic 3D models of the objects that the robotic arm will interact with. These models are crafted to resemble their real-world counterparts including a range of backgrounds, lighting conditions, and object orientations, ensuring that the training data covers a wide array of scenarios. Alongside the synthetic images, corresponding annotations are generated automatically. These annotations include bounding boxes and class labels for each object in the image, formatted according to the YOLOv5 requirements. Blender and additional scripts are used to automate the annotation process, ensuring precision and uniformity across the entire dataset. Using Blender's powerful rendering engine, the scenes are rendered to produce high-quality images. The engine is capable of simulating realistic lighting, textures, and materials.

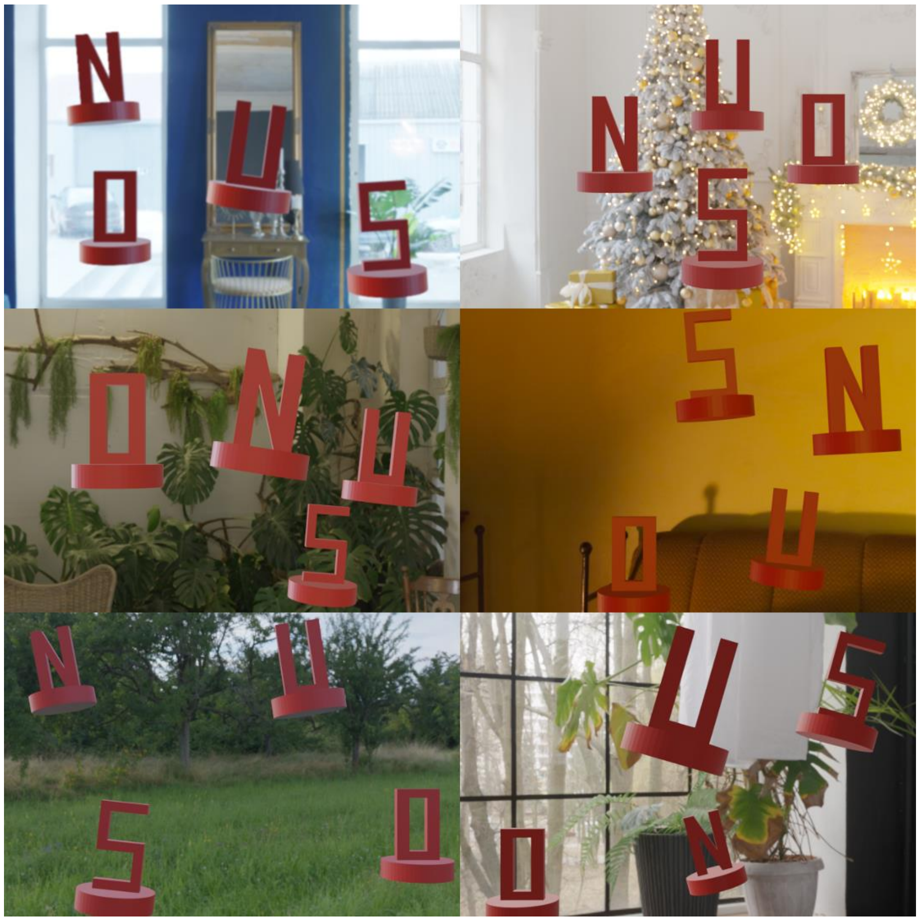
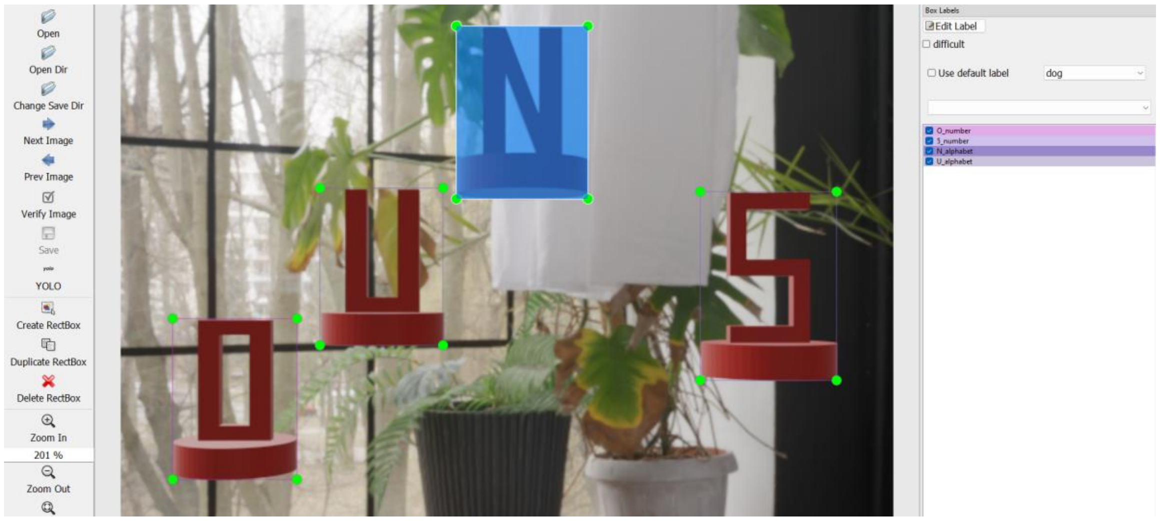


Figure 8. Collage of images generated

The rendered images, along with their annotations, are compiled into a dataset ready for training the YOLOv5 model. This dataset forms the foundation for the training process. The generated annotations are verified using a Python library called LabelImg. This tool allows for manual review and correction of the annotations, ensuring their accuracy and reliability.



- ☐ difficult
- ☐ Use default label: dog
- ☑ O_number
- ☑ 5_number
- ☑ N_alphabet
- ☑ U_alphabet

Figure 9. Annotation Verification using labelImg

3.5 Object Detection

Object detection is a critical component of this project, enabling the robotic arm to identify and locate objects within its operational environment. YOLOv5 is a state-of-the-art object detection model. It was employed for this task due to its high accuracy and real-time performance capabilities. This section details the process of training and utilizing the YOLOv5s model for object detection in the context of the robotic arm's operations. YOLOv5s (You Only Look Once, version 5, small) is a version of the YOLO object detection model optimized for speed and efficiency, making it suitable for real-time applications. YOLOv5s is capable of detecting multiple objects within an image and providing their bounding boxes and class labels.

3.5.1 Dataset Preparation. As described earlier, a large dataset of synthetic images was generated using Blender. This dataset includes various scenarios with the target objects in different positions, scales, and lighting conditions. The objects used for training the YOLOv5s model include the letters "U" and "N" and the numbers "0" and "5". These objects were classified into four distinct classes: 0_number, 5_number, N_alphabet, and U_alphabet. Automatic annotations were generated alongside the synthetic images, detailing the bounding boxes and class labels for each object. These annotations were verified using the labelImg tool to ensure accuracy. The synthetic dataset consisted of 8,000 images. The dataset was split into 80% for training (6,400 images) and 20% for validation (1,600 images). This split allows for effective training while maintaining a robust validation set to evaluate the model's performance.

3.5.2 Data Augmentation. Images were randomly rotated to simulate different viewing angles and orientations of the objects. This helps the model learn to recognize objects regardless of their orientation. Random noise was added to the images to make the model robust against varying image qualities and sensor noise. This ensures the model can perform well even with noisy input data. Gaussian blur was applied to some images to simulate out-of-focus conditions. This helps the model learn to detect objects even when the image quality is not optimal.

3.5.3 Model Training. The YOLOv5s model was trained using the synthetic dataset to generate custom weights tailored to the specific objects of interest. Data augmentation techniques, including random scaling, flipping, and color adjustments, were applied to enhance the model's robustness. The training was conducted with an image size of 640x480 pixels and ran for 100 epochs to ensure thorough learning.

3.5.4 Validation and Testing. A separate validation set, comprising both synthetic and real-world images, was used to evaluate the model's performance during training. This helped in fine-tuning the model and preventing overfitting. After training, the model was tested on a diverse set of real-world images to assess its accuracy and generalization capabilities.

The dataset used for training the model comprises four distinct classes: O_number, 5_number, N_alphabet, and U_alphabet. The distribution of these labels is depicted in Figure 10. Each class represents a unique category that the model is trained to recognize. The bar chart illustrates the number of instances for each class, showing that the dataset is relatively balanced, with each class having a similar number of samples. This balance is crucial for ensuring that the model does not become biased towards any particular class during training. Additionally, understanding the distribution of these labels helps in assessing the potential challenges in detection tasks, such as class imbalance, which can significantly affect the model's performance.

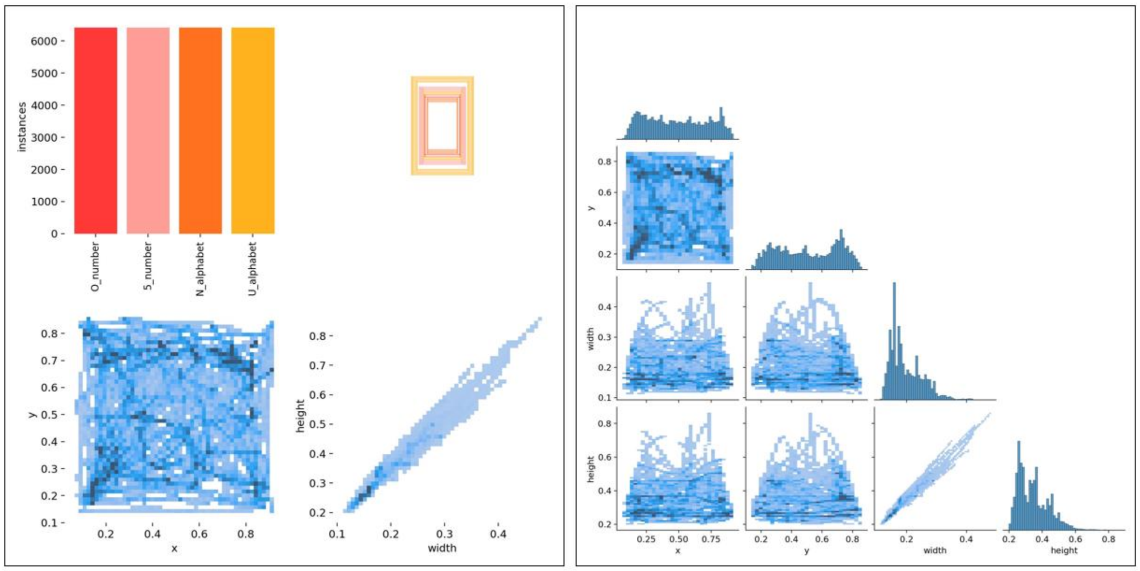


Figure 10. Graphs showing Label Distribution (left) and labels_correlogram (right)

CHAPTER 4

EXPERIMENTAL SETUP

4.1 Robotic Arm Control

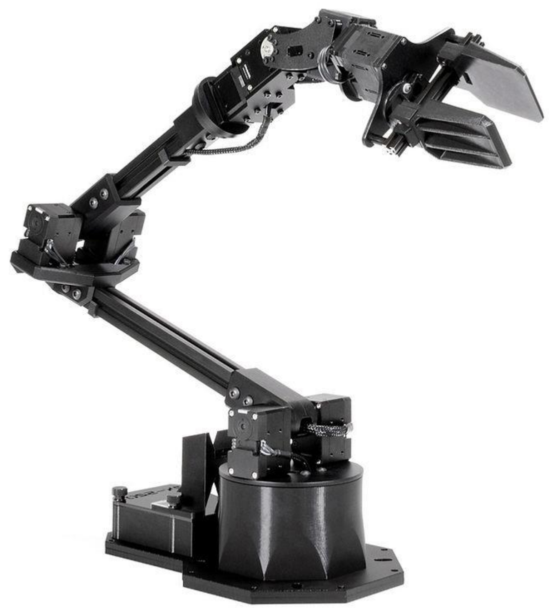


Figure 11. WidowX 250s Robot Arm

To control the WidowX robotic arm, the Dynamixel SDK was employed, which facilitated direct communication with the arm's motors. The control implementation included several critical steps to ensure precise, reliable, and safe operation of the robotic arm in performing complex tasks. The following sections detail the steps and techniques used in the control implementation. To utilize the DYNAMIXEL SDK for controlling the robotic arm, it is essential to properly set up both the Controller and the DYNAMIXEL motors as shown in the Figure 13. The U2D2 controller was used as an interface between the PC and the DYNAMIXEL motors of the robotic arm. The U2D2 was connected to the PC via a USB cable, and a separate power supply was connected to ensure the motors received adequate power. The communication port on the U2D2 was linked to the DYNAMIXEL motors using a dedicated cable. This setup provided a robust communication channel for controlling the motors.

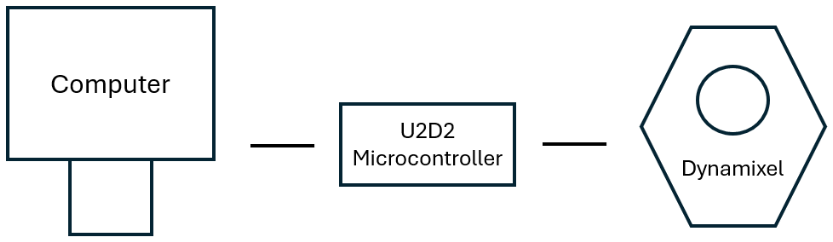
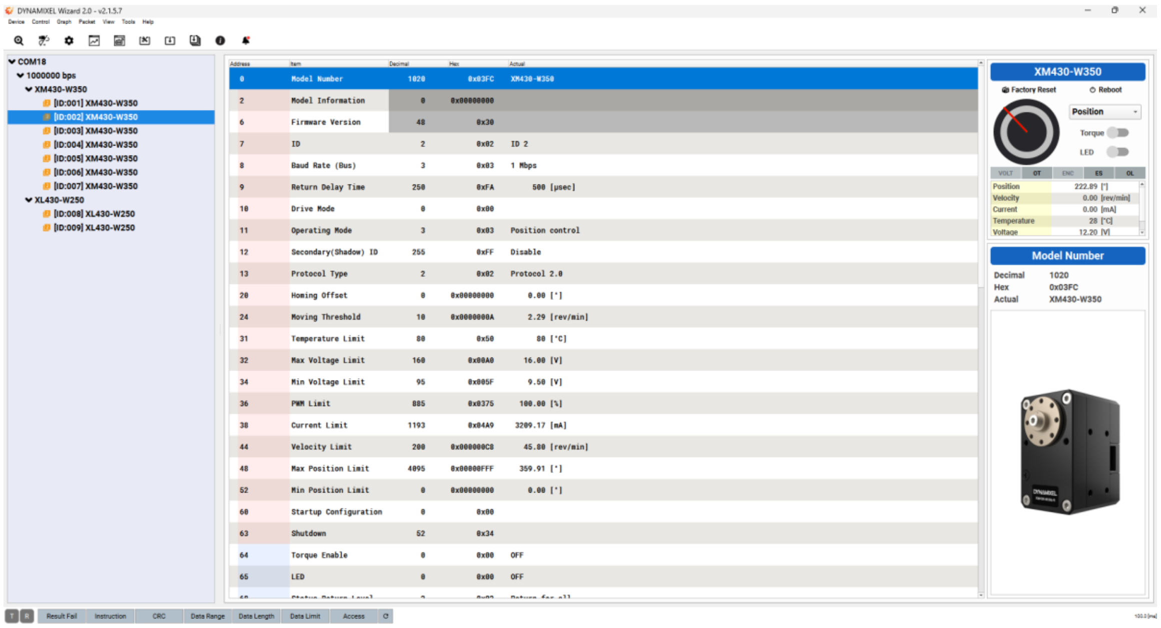


Figure 12. System Setup: Computer to Dynamixel Motor via U2D2 Microcontroller

The WidowX robotic arm utilizes a combination of seven XM430-W350 and two XL430-W250 DYNAMIXEL motors. Both types of motors support the DYNAMIXEL Protocol 2.0, ensuring reliable communication and seamless integration with the control system. The combination of these motors in the WidowX arm allows for smooth and controlled movements, enabling the arm to perform intricate pick-and-place operations efficiently. Additionally, the arm's design incorporates "shadow motors" at the shoulder and elbow joints to enhance stability and precision. These shadow motors mirror the movements of the primary motors, providing additional support and reducing the load on the primary motors. This setup allows for smoother and more controlled movements, particularly during complex tasks that require high accuracy and stability. DYNAMIXEL Wizard 2.0 is a comprehensive software tool designed to facilitate the configuration, monitoring, and maintenance of DYNAMIXEL motors. It provides a user-friendly interface that allows users to perform various tasks such as firmware updates, parameter adjustments, and real-time diagnostics. The tool is essential for ensuring that DYNAMIXEL motors operate at their optimal performance and can be easily integrated into robotic systems. Figure 14 below shows the DYNAMIXEL Wizard 2.0 interface, displaying detailed information about the connected DYNAMIXEL motors, including model numbers, firmware versions, operating modes, and various configuration parameters. [21, 22]



COM18
1000000 bps
XM430-W350
[ID:001] XM430-W350
[ID:002] XM430-W350
[ID:003] XM430-W350
[ID:004] XM430-W350
[ID:005] XM430-W350
[ID:006] XM430-W350
[ID:007] XM430-W350
XL430-W250
[ID:008] XL430-W250
[ID:009] XL430-W250

Address	Item	Decimal	Hex	Actual
0	Model Number	1020	0x03FC	XM430-W350
2	Model Information	0	0x00000000	
6	Firmware Version	48	0x30	
7	ID	2	0x02	ID 2
8	Baud Rate (Bus)	3	0x03	1 Mbps
9	Return Delay Time	250	0xFA	500 [μsec]
10	Drive Mode	0	0x00	
11	Operating Mode	3	0x03	Position control
12	Secondary(Shadow) ID	255	0xFF	Disable
13	Protocol Type	2	0x02	Protocol 2.0
20	Homing Offset	0	0x00000000	0.00 [°]
24	Moving Threshold	10	0x0000000A	2.29 [rev/min]
31	Temperature Limit	80	0x50	80 [°C]
32	Max Voltage Limit	160	0x00A0	16.00 [V]
34	Min Voltage Limit	95	0x005F	9.50 [V]
36	PWM Limit	885	0x0375	100.00 [%]
38	Current Limit	1193	0x04A9	3209.17 [mA]
44	Velocity Limit	200	0x000000C8	45.80 [rev/min]
48	Max Position Limit	4095	0x00000FFF	359.91 [°]
52	Min Position Limit	0	0x00000000	0.00 [°]
60	Startup Configuration	0	0x00	
63	Shutdown	52	0x34	
64	Torque Enable	0	0x00	OFF
65	LED	0	0x00	OFF
68	Status Return Level	2	0x02	Return for all

XM430-W350

Position	222.89 [°]
Velocity	0.00 [rev/min]
Current	0.00 [mA]
Temperature	28 [°C]
Voltage	12.20 [V]

Model Number

Decimal	1020
Hex	0x03FC
Actual	XM430-W350

Figure 13. Dynamixel Wizard 2.0

4.1.1 Port Initialization. The communication port to which the robotic arm was connected was initialized and configured to establish a stable connection. This step involved opening the port using the PortHandler provided by the Dynamixel SDK and setting the baud rate to 1,000,000 bps. This high baud rate was crucial for ensuring reliable and fast data transmission between the control software and the motors, minimizing latency and communication errors.

4.1.2 Velocity and Acceleration Profiles. The velocity and acceleration profiles for the motors were configured to ensure smooth and controlled movements. Setting these profiles involved defining the maximum speed (velocity) and the rate of change of speed (acceleration) for each motor. These parameters were crucial for achieving precise control during pick-and-place operations, reducing the risk of overshooting or jerky movements.

4.1.3 Position Control. Position control of the WidowX robotic arm is a critical aspect of ensuring precise and reliable movement, particularly for tasks such as pick-and-place operations. The control process involves several steps to convert joint angle commands into actual motor positions, taking into account the specific characteristics and requirements of the DYNAMIXEL motors used. The joint angles required for the manipulator's movements are read from a predefined text file. These angles are specified in degrees, representing the desired positions for each joint of the robotic arm. To control the motors accurately, these angles must be converted into the motor's native units, typically encoder counts. The conversion formula used is:

$$position = \left(\frac{4096}{2\pi}\right) \times \left(\frac{angle \times \pi}{180}\right) + \pi \tag{4.1}$$

This formula converts the angle from degrees to radians and then scales it to the range of the motor's encoder counts. The added π ensures that the position value is correctly aligned with the motor's zero position.

Figure 15 is the block diagram describing the position controller in Position Control Mode. When a DYNAMIXEL motor receives an instruction from the user, it undergoes a detailed process to ensure precise control of the horn (output shaft). The user command is transmitted via the DYNAMIXEL bus and registered as the Goal Position. This Goal Position is then converted into a desired position trajectory and a desired velocity trajectory using the Profile Velocity and Profile Acceleration settings. These trajectories are stored at specific addresses for position and velocity. The Feedforward and PID controllers then calculate the Pulse Width Modulation (PWM) output required to achieve these trajectories. The PID control utilizes proportional, integral, and derivative gains to make necessary adjustments for accurate positioning and movement.

To ensure safe operation, the calculated PWM output is limited by the Goal PWM setting, determining the final PWM value applied to the motor through an inverter. This action drives the horn of the DYNAMIXEL motor. Throughout this process, the motor continuously monitors and stores the Present Position, Present Velocity, Present PWM, and Present Current, providing feedback for real-time adjustments and ensuring reliable performance. This comprehensive control mechanism enables the DYNAMIXEL motor to execute user commands with high precision and reliability, crucial for tasks requiring accurate mechanical movements.

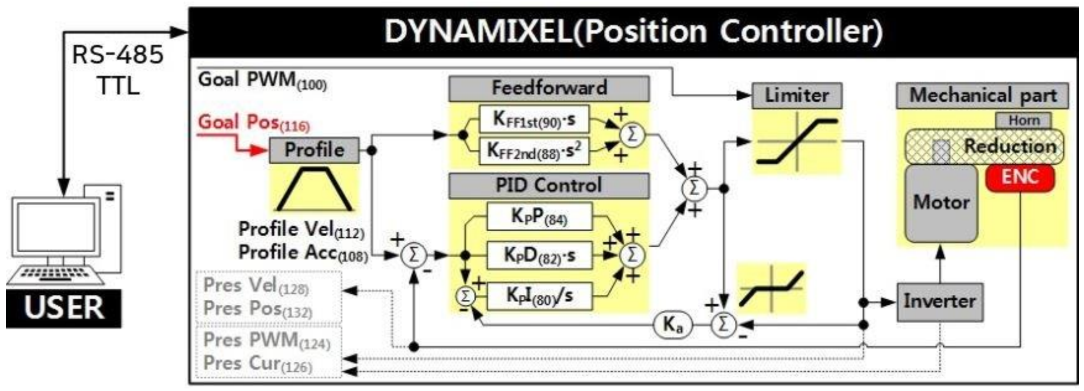


Figure 14. Dynamixel Position Controller

4.2 Depth Camera

Figure 15. Intel RealSense Depth Camera D435

The depth camera used is the Intel RealSense depth camera, which is a critical component for providing high-resolution depth and RGB data. This enables precise object detection and spatial understanding, essential for the effective operation of the robotic arm. The Intel RealSense depth camera is mounted on the end effector of the robotic arm. This positioning allows the camera to capture a clear and unobstructed view of the workspace, ensuring accurate data collection for object detection and manipulation tasks. Once the camera is calibrated, the transformation algorithms convert 2D image coordinates and depth information from the camera into 3D world coordinates. The primary transformation involves converting pixel coordinates (*u*, *v*) and depth (*d*) into 3D coordinates (*X*, *Y*, *Z*) using the intrinsic parameters. The formula for this conversion is:

$$X = (u - c_x) \cdot \frac{d}{f_x} \tag{4.2}$$

$$Y = (v - c_{xy}) \cdot \frac{d}{f_y} \tag{4.3}$$

$$Z = d \tag{4.4}$$

where (c_x, c_y) are the coordinates of the principal point, and (f_x, f_y) are the focal lengths in the *x* and *y* directions. After obtaining the 3D coordinates in the camera frame, the next step is to transform these coordinates into the robot's operational frame. This involves applying the extrinsic transformation matrix obtained during calibration. The transformation matrix T can be represented as:

$$P_{robot} = T \cdot P_{camera} \tag{4.5}$$

where P_{camera} is the 3D point in the camera coordinate system, and P_{robot} is the corresponding point in the robot's coordinate system. To ensure accurate mapping between the depth and RGB data, the depth data must be aligned with the RGB images. This alignment corrects for any discrepancies between the depth and color streams, providing a coherent view of the environment. The Intel RealSense SDK includes tools for aligning the depth to the color frame. This process involves matching the depth data pixels to their corresponding pixels in the RGB image, ensuring both datasets are synchronized spatially.

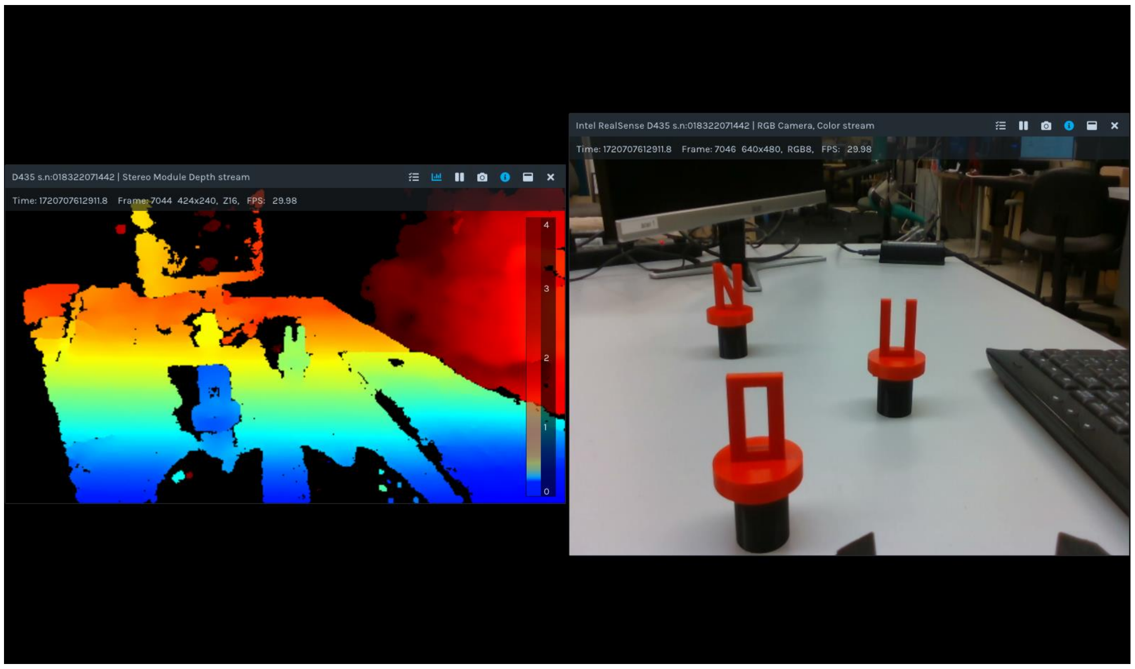

D435 s.n:018322071442 | Stereo Module Depth stream
Time: 1720707612911.8 Frame: 7044 424x240, Z16, FPS: 29.98
4
3
2
1
0
Intel RealSense D435 s.n:018322071442 | RGB Camera, Color stream
Time: 1720707612911.8 Frame: 7046 640x480, RGB8, FPS: 29.98

Figure 16. Depth Stream (left) and Color Stream (right)

The transformed 3D coordinates are sent to the inverse kinematics (IK) module of the robotic arm. The IK module calculates the required joint angles to position the end effector at the specified 3D coordinates. These calculated joint angles are then used to control the motors of the robotic arm, enabling precise and accurate movements to perform tasks such as pick-and-place operations.

4.3 Computing Hardware

The computing hardware utilized in this project is crucial for managing the computational demands of real-time object detection, 3D coordinate transformation, and robotic arm control. The key components of the computing hardware setup include the NVIDIA Jetson Nano and the DYNAMIXEL U2D2 microcontroller. These components work in conjunction to ensure efficient and reliable operation of the robotic arm.

4.3.1 NVIDIA Jetson Nano. The NVIDIA Jetson Nano is a compact, powerful computing device designed specifically for AI and robotics applications. It provides the necessary computational power to run complex AI models and manage real-time data processing tasks, making it ideal for this project. The Jetson Nano is capable of handling intensive machine learning and computer vision workloads, thanks to its integrated GPU and CPU architecture. The Jetson Nano draws power from the Unitree A1 quadruped robot at 19V 2A, which is stepped down to 5V 4A using a buck converter. This ensures that the Jetson Nano receives a stable power supply necessary for its operations.

Specifications:

- CPU: Quad-core ARM® Cortex®-A57 MPCore processor, which handles various tasks, including reading data from the camera, preprocessing images, and running control algorithms for the robotic arm. Its multi-core architecture allows for parallel processing, ensuring smooth and efficient operation.

- GPU: 128-core NVIDIA Maxwell™ architecture GPU, accelerates the YOLOv5s object detection model, enabling real-time inference on the RGB frames captured by the Intel RealSense camera. The parallel processing capabilities of the GPU ensure fast and accurate object detection, which is crucial for dynamic environments.
- Memory: 4GB 64-bit LPDDR4 at 25.6GB/s, is used to load and run the machine learning models, store intermediate data during processing, and manage the real-time data streams from the camera. The 4 GB capacity ensures that the system can handle large datasets and complex models without significant lag.
- Connectivity: Various I/O options, include USB 3.0 and USB 2.0 ports, HDMI output, and Gigabit Ethernet. The USB port connects to the Intel RealSense camera. Gigabit Ethernet provides a fast network connection for remote monitoring and updates.
- Operating System: Linux-based JetPack SDK provides the necessary software tools and libraries, including CUDA, cuDNN, and TensorRT, for optimized AI processing. It also supports popular AI frameworks like TensorFlow and PyTorch, facilitating easy development and deployment of machine learning models.

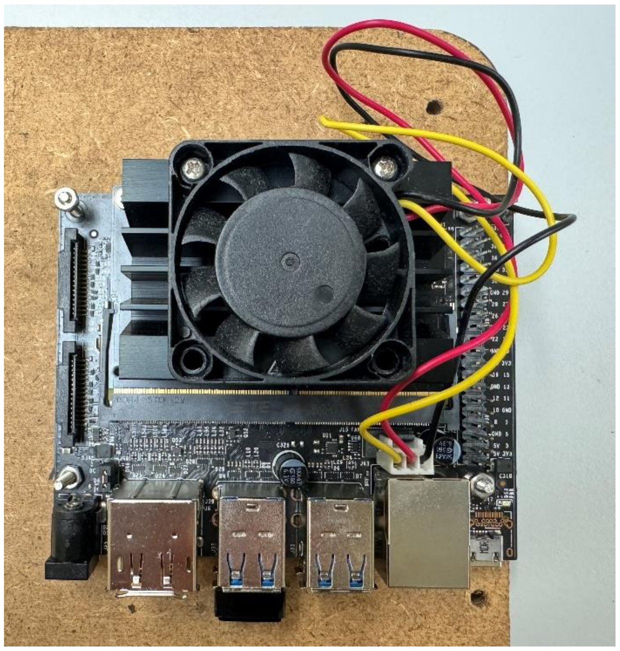


Figure 17. Jetson Nano

The Jetson Nano runs the YOLOv5s object detection model, utilizing its GPU to perform accelerated inference. This allows the system to detect and identify objects in real-time, a critical capability for the robotic arm's operation. It processes the RGB, and depth data captured by the Intel RealSense camera. It handles the alignment of these data streams and extracts the necessary 3D coordinates for object detection and manipulation tasks. The Jetson Nano communicates with the Raspberry Pi computer on Unitree A1 via a Gigabit Ethernet connection. This setup ensures reliable and high-speed data transmission, allowing for real-time data exchange and command transmission.

4.3.2 DYNAMIXEL U2D2 Microcontroller. The DYNAMIXEL U2D2 is a communication interface that bridges the computing hardware and the robotic arm's motors. It converts the high-level commands from the Unitree A1 Raspberry Pi into precise control signals for the DYNAMIXEL servos, ensuring accurate and responsive movements of the robotic arm. The U2D2 interfaces with the Raspberry Pi via USB and communicates with the DYNAMIXEL motors. This setup ensures reliable and fast data transmission, which is essential for real-time control. The U2D2 draws power from the Unitree A1 quadruped using one of its ports that provides 12V 2A, ensuring consistent and reliable power for motor control.

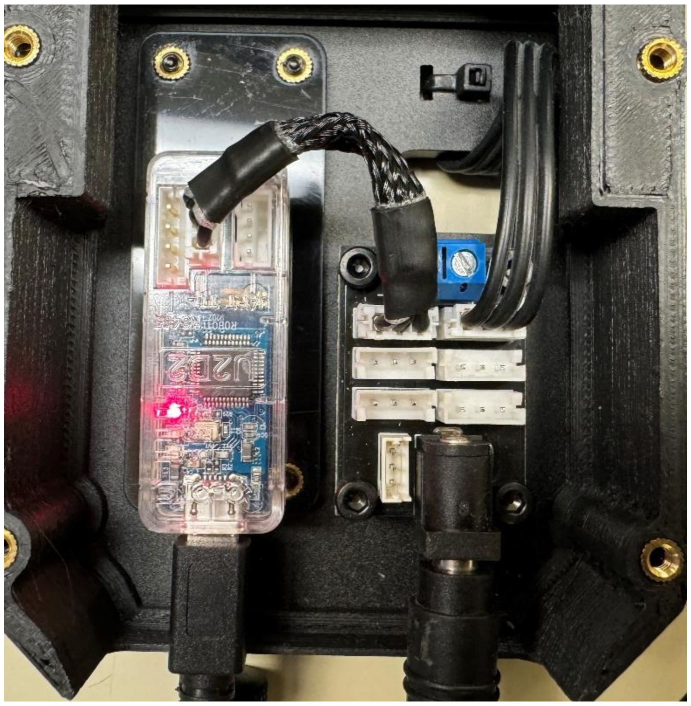


Figure 18. U2D2 Microcontroller

4.3.3 Communication Setup with Unitree A1 Raspberry Pi Computer. The communication setup between the Jetson Nano and the Unitree A1 Raspberry Pi computer is essential for coordinating the operations of the robotic arm mounted on the quadruped robot. This communication is facilitated through a Gigabit Ethernet connection, ensuring reliable and high-speed data transmission.

4.4 Unitree A1 Quadruped

The Unitree A1 quadruped robot serves as the mobile platform for the robotic arm, providing the necessary locomotion capabilities to navigate and operate within dynamic and unstructured environments. The integration of the quadruped with the robotic arm enhances the system's versatility, allowing it to perform complex tasks that require both movement and manipulation. The Unitree A1 is a highly agile and robust quadruped robot designed for various applications, including research, inspection, and service robotics. Its advanced locomotion capabilities and stability make it an ideal platform for mounting the robotic arm, enabling it to move across different terrains and interact with objects in its environment. Its four-legged design ensures stability and agility, enabling it to navigate through various terrains and obstacles. The A1 is designed to carry payloads up to 5 kg, providing sufficient capacity to support the robotic arm and other necessary equipment such as the Intel RealSense camera and the Jetson Nano. The A1 provides power to the Jetson Nano at 19V 2A, which is stepped down to 5V 4A using a buck converter. Additionally, the DYNAMIXEL U2D2 microcontroller draws power from the A1 at 12V 2A, ensuring that all components receive a stable and sufficient power supply for their operations. A Gigabit Ethernet cable connects the Jetson Nano to the Raspberry Pi computer on the A1, ensuring reliable and high-speed data transmission. This setup allows for real-time data exchange and command transmission between the computing hardware and the quadruped robot. The 3D coordinates calculated by the Jetson Nano are sent to the Raspberry Pi via a Python socket connection, facilitating precise control of the robotic arm.

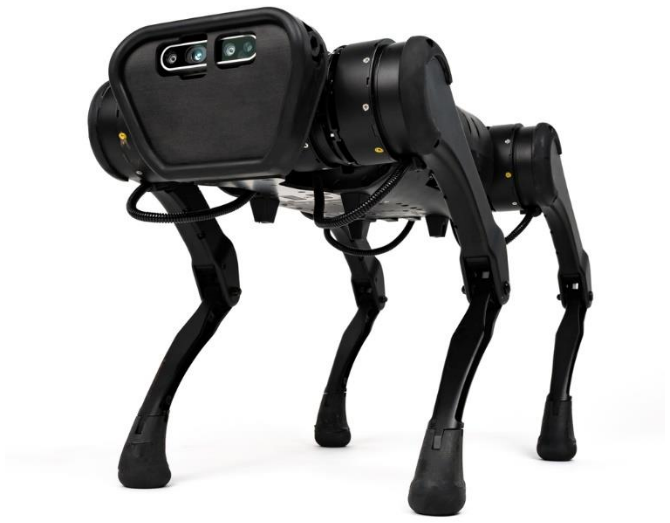


Figure 19. Unitree A1 quadruped robot

4.5 Operational Modes

The robotic system implemented in this project operates in two distinct modes: Autonomous Mode and Teleoperation Mode. These modes allow the system to perform a variety of tasks efficiently and effectively, adapting to different operational requirements and environmental conditions.

4.5.1 Teleoperation Mode. Teleoperation Mode allows for direct human control of the robotic system, providing input and guidance for performing tasks. This mode is particularly useful for complex or non-routine tasks that require human judgment and decision-making. During the experimental phase of this project, Teleoperation Mode was extensively utilized to fine-tune the system's performance and to handle tasks that demanded a high degree of precision and adaptability.

Upon booting up the quadruped robot, the system starts in trotting mode. In this mode, the quadruped can be teleoperated to move in all directions, allowing the operator to navigate the environment effectively. The operator can adjust the pitch and tilt of the quadruped, providing precise control over its orientation. This capability is crucial for positioning the robot correctly before switching to Autonomous Mode. In this mode, the robotic arm's torque is disabled, meaning the arm cannot be used for manipulation tasks. This limitation ensures the safety and stability of the system during navigation. Once the quadruped reaches close proximity to the target object, a button can be pressed to switch the system to Autonomous Mode. This action enables the torque on the robotic arm, preparing it for manipulation tasks.

4.5.2 Autonomous Mode. In Autonomous Mode, the robotic system operates without direct human intervention, relying on pre-programmed instructions, real-time sensor data, and AI algorithms to perform tasks. This mode leverages the full capabilities of the integrated hardware and software components, ensuring high levels of precision and efficiency. When switching to Autonomous Mode, the torque on the robotic arm is enabled. This activation allows the arm to perform manipulation tasks, such as object detection, sorting, and placement. Another button press initiates the object detection and inverse kinematics (IK) processes. The system uses the YOLOv5s object detection model to identify and locate objects. The 3D coordinates of the detected objects are calculated using the depth data from the Intel RealSense camera. The system is trained to detect four custom objects: the letters "U" and "N" and the numbers "0" and "5". These objects are sorted based on their type, with the robotic arm placing them into designated cups. Two cups are used for sorting, each marked with a different AprilTag ID. One cup is designated for alphabet objects and the other for number objects. The arm detects the appropriate cup using the AprilTags and places the objects accordingly. The dual operational modes of Teleoperation and Autonomous Mode provide a comprehensive and versatile approach to robotic control. The ability to switch between these modes enhances the system's functionality and adaptability, making it suitable for a wide range of industrial applications.

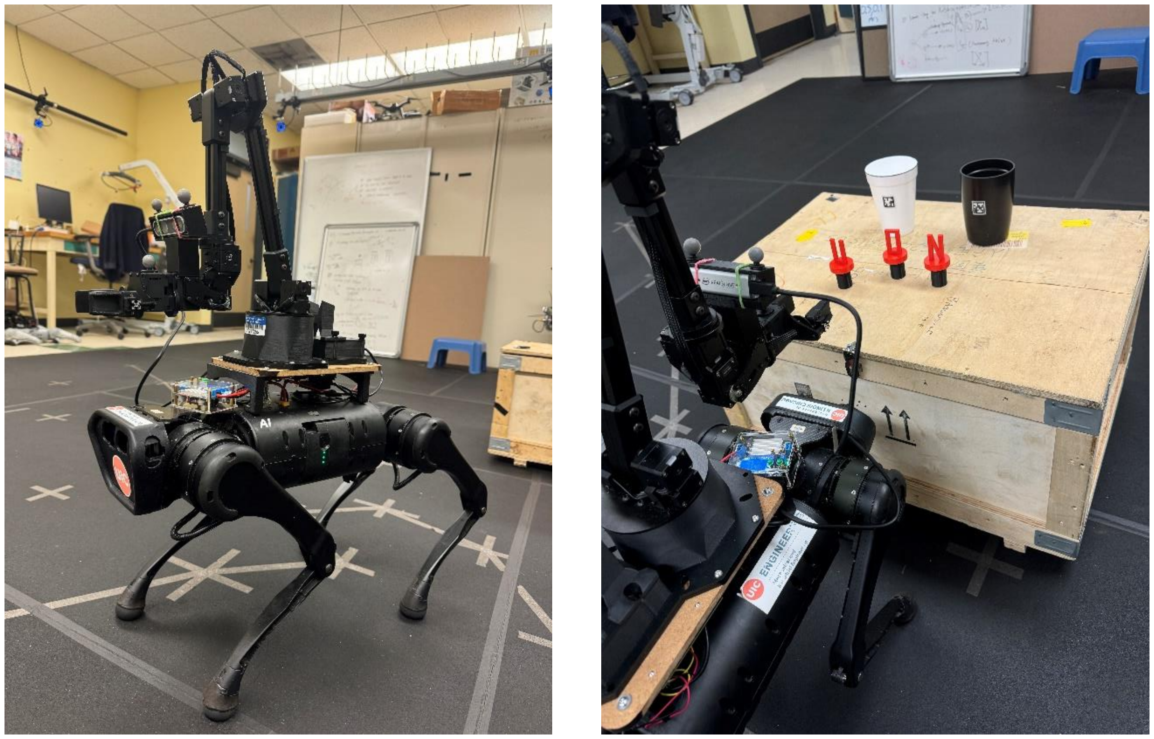


Figure 20. Quadruped Manipulator Pick-and-Place System

CHAPTER 5

RESULTS

This section presents the outcomes of the experimental setup, evaluating the system's performance in both Teleoperation Mode and Autonomous Mode. The key metrics assessed include the accuracy of object detection, the efficiency of object sorting, and the overall system robustness in dynamic and unstructured environments. Specific metrics were also used to evaluate the performance of the manipulator: trajectory tracking comparison and real-time grabbing success rates.

5.1 Object Detection

The object detection capabilities of the system were evaluated using a custom-trained YOLOv5s model. Training was conducted over 100 epochs with a batch size of 16 and an image resolution of 640x480 pixels. The evaluation metrics focused on precision, recall, and mean average precision (mAP). Additionally, the precision, recall, and mean Average Precision (mAP) metrics are plotted, showing high values near 1.0, which signify excellent performance in detecting and classifying objects. These metrics demonstrate that the model has learned to accurately predict object locations and classifications, achieving robust performance across various evaluation criteria. The Figure 22 below illustrates the training and validation performance metrics over 100 epochs, providing a comprehensive view of the model's learning process. The top row displays the training losses for bounding box regression (train/box_loss), objectness score (train/obj_loss), and classification accuracy (train/cls_loss). These losses consistently decrease, indicating effective learning and convergence. The bottom row presents the corresponding validation losses (val/box_loss, val/obj_loss, val/cls_loss), which also show a similar downward trend, confirming the model's ability to generalize well to unseen data.

5.1.1 Evaluation Metrics. The model achieved an average precision of 99.5%, indicating a high level of accuracy in identifying and classifying objects within the images. The recall rate was also 99.5%, demonstrating the model's effectiveness in detecting the majority of relevant objects in the dataset. The mAP score for the four classes was 99.5%, reflecting the model's balanced performance across all object categories.

5.1.2 Qualitative Analysis. The model successfully detected objects under different lighting conditions, angles, and backgrounds, showcasing its robustness and generalization capability. During live testing, the model maintained a high detection speed, processing frames at an average of 15 frames per second (FPS), which is suitable for real-time applications.

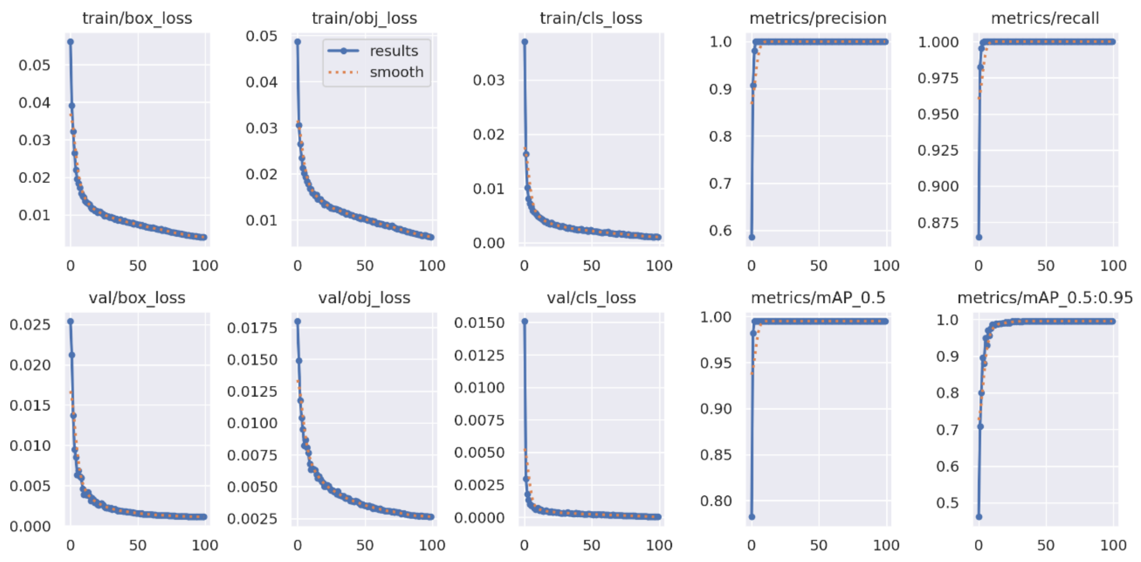


Figure 21. Graphs showing the training and validation loss over epochs, indicating the model's learning progress

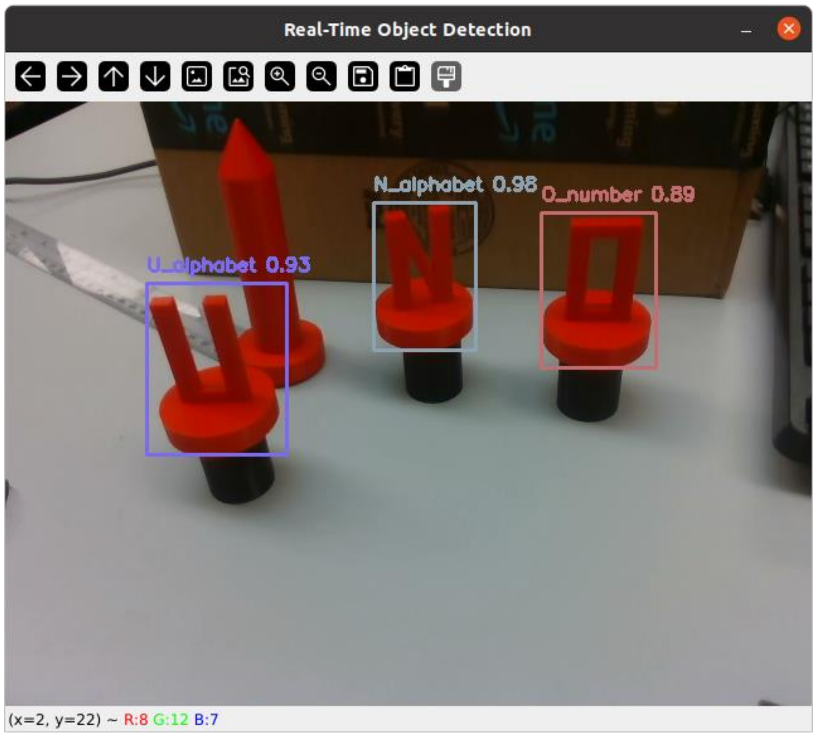


Figure 22. Screenshot of real-world testing

5.2 Trajectory Comparison

The manipulator's performance in following planned trajectories was assessed by comparing the expected to the actual trajectory recorded using a motion capture system. The expected trajectory represented the calculated path for the manipulator to move from its initial position to all pick-and-place locations based on detected object positions. The actual trajectory was tracked in real time to measure deviations and assess accuracy.

5.2.1 Evaluation Metrics. The deviation between the expected and actual paths was measured to evaluate the accuracy of the manipulator's movement.

5.2.2 Qualitative Analysis. The manipulator's ability to follow the planned trajectory closely and perform accurate movements highlights its precision and control, essential for effective operation in industrial applications. The system's adaptive movements, as observed through the motion capture data, demonstrate its capability to handle dynamic changes and adjust its path as needed. The figures 23 - 27 below illustrate a comparison between the expected trajectory and the actual trajectory recorded using a motion capture system, along with the corresponding error analysis. The left side of each figure displays the trajectory plots for the X, Y, and Z coordinates, comparing the desired positions with those captured by the motion capture system. The right side of each figure presents the error in these coordinates over time. The close alignment between the desired and actual trajectories, combined with the error plots, highlights the precision of the system, while the occasional deviations reveal areas where fine-tuning may further enhance accuracy.

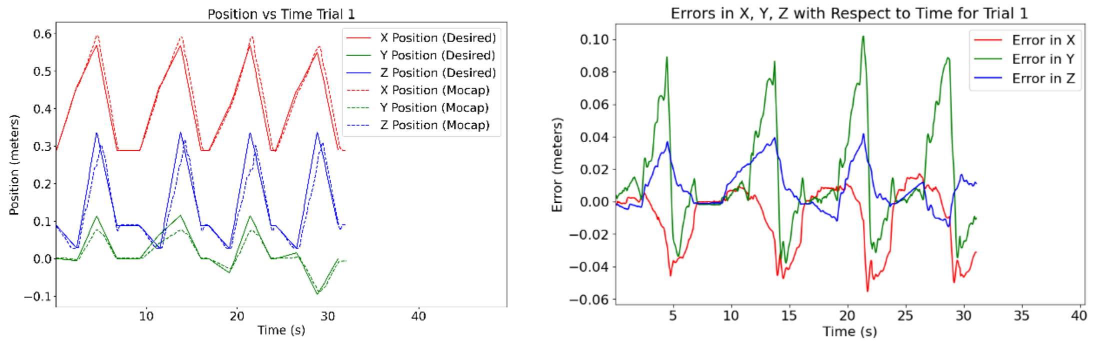


Figure 23. Position and Error Analysis for XYZ Coordinates - Trial 1

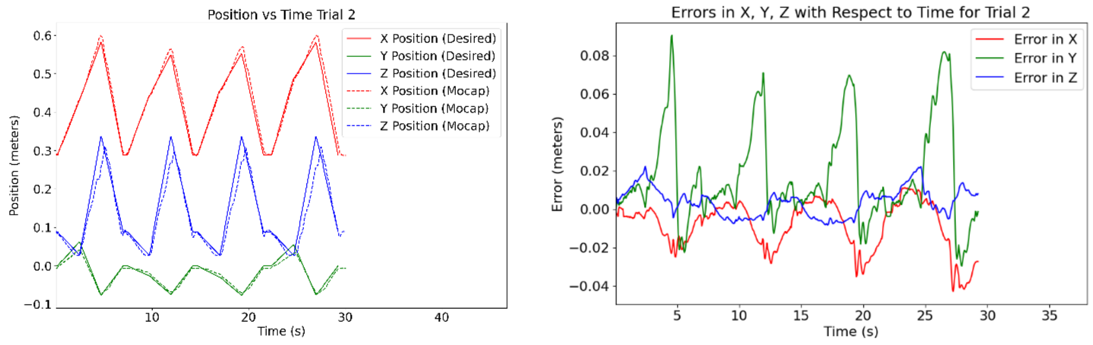


Figure 24. Position and Error Analysis for XYZ Coordinates - Trial 2

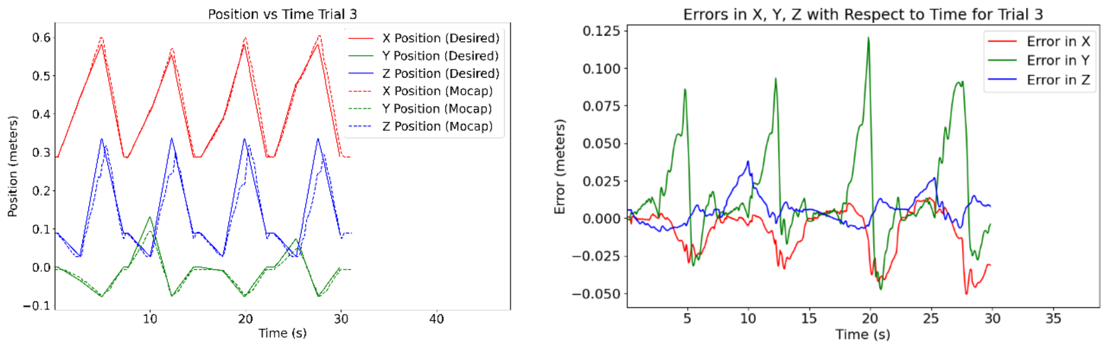


Figure 25. Position and Error Analysis for XYZ Coordinates - Trial 3

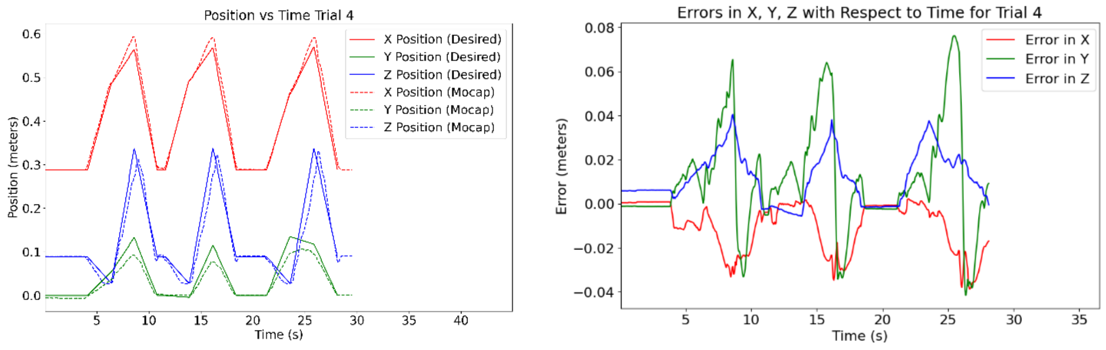


Figure 26. Position and Error Analysis for XYZ Coordinates - Trial 4

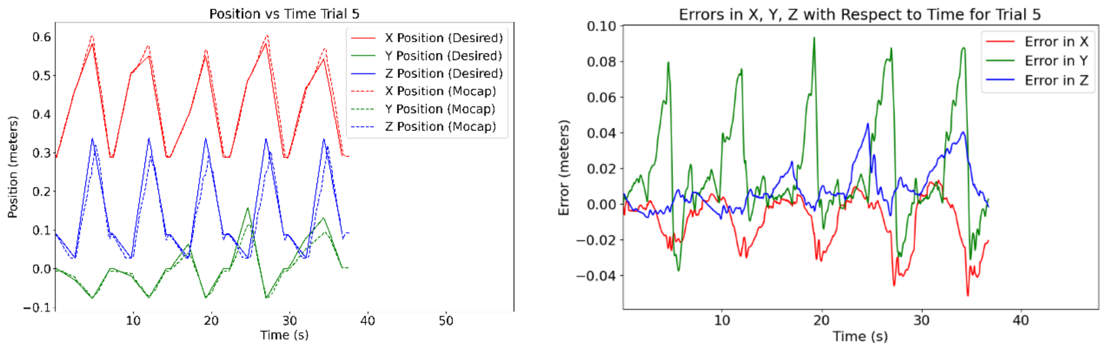


Figure 27. Position and Error Analysis for XYZ Coordinates - Trial 5

5.3 Real-Time Grabbing Success

The success rate of real-time grabbing was evaluated to assess the manipulator's ability to accurately and reliably pick up objects in dynamic environments. The success rate was calculated based on the number of successful grabs out of the total attempts during the testing phase.

5.3.1 Evaluation Metrics. The manipulator achieved a grabbing success rate of 80%, successfully picking up 8 out of 10 objects during the tests. The failure rate was mainly due to the Intel RealSense camera's difficulty in detecting objects that were too close to the sensor, resulting in occasional failures to grab objects accurately.

5.3.2 Qualitative Analysis. The manipulator's high success rate in grabbing objects reflects its accuracy and reliability in dynamic environments. The system's ability to adapt to slight positional changes of objects during the grabbing process indicates robust handling of real-time operations.

CHAPTER 6

CONCLUSIONS

This thesis presents the development and evaluation of a dynamic pick-and-place system for a manipulator mounted on a Unitree A1 quadruped robot. The system integrates advanced object detection using a custom-trained YOLOv5s model, precise control algorithms, trajectory planning, and real-time feedback mechanisms to enable efficient and accurate real-time object tracking and sorting in dynamic and unstructured environments.

The experimental results demonstrate the system's high accuracy in object detection. The YOLOv5s model achieved an impressive average precision and recall rate of 99.5%. This high level of accuracy ensures reliable identification and classification of objects, which is crucial for effective manipulation tasks. The manipulator's performance in following planned trajectories also showed excellent accuracy. This indicates the system's capability to execute precise movements, essential for tasks requiring high precision and coordination.

In terms of real-time operations, the system maintained a grabbing success rate of 80%. The primary source of errors was identified as the Intel RealSense camera's difficulty in detecting objects at close range, which occasionally resulted in failed grabs. Despite this limitation, the system demonstrated robust handling of dynamic tasks, adapting effectively to positional changes of objects during the grabbing process.

6.1 Key Contributions

- The system successfully combines the locomotion capabilities of a quadruped robot with the dexterity of a manipulator, enhanced by advanced object detection. This integration allows the robot to navigate and interact with its environment efficiently, making it suitable for a wide range of applications.
- Sophisticated control algorithms were developed for precise trajectory planning and real-time feedback. These algorithms ensure accurate and efficient movements of the robotic arm, essential for performing complex manipulation tasks.
- A custom YOLOv5s model was trained on a synthetic dataset, achieving high detection accuracy for the targeted objects. This demonstrates the effectiveness of synthetic data in training robust object detection models.
- The system's ability to operate in real-time, maintaining high detection speeds and efficient object sorting, highlights its potential for industrial applications requiring rapid and accurate manipulation.

6.2 Future Scope

While the current system exhibits strong performance, several areas offer opportunities for further enhancement and research. Addressing the limitations of the Intel RealSense camera in detecting objects at close range could further improve the grabbing success rate. Exploring alternative sensors or enhancing the existing setup with additional cameras could mitigate this issue. Developing more sophisticated gripping mechanisms that can handle a wider variety of object shapes and sizes would enhance the system's versatility and effectiveness in diverse scenarios. Expanding the dataset to include more object classes and increasing the variety of objects in training could further improve the model's generalization and robustness, making it more adaptable to different environments. Integrating advanced navigation algorithms to enable the quadruped robot to autonomously navigate complex environments without human intervention would significantly enhance the system's autonomy and operational efficiency. Conducting extensive testing in real-world industrial settings, such as warehouses and manufacturing plants, to validate the system's performance and reliability in practical applications would provide valuable insights into the system's robustness and scalability. Investigating energy-efficient algorithms and hardware optimizations to extend the operational time of the robot, making it more viable for prolonged industrial tasks, would enhance the system's practicality and sustainability in real-world applications.

6.2 Conclusion

In conclusion, this thesis showcases significant advancements in mobile manipulation, object detection, and robotic control. The developed system demonstrates high accuracy, precision, and robustness, highlighting its potential for real-world applications in various industries. By addressing the identified areas for improvement and continuing to build on this foundation, future research can further enhance the capabilities and applications of such robotic systems. This will contribute to greater efficiency, productivity, and sustainability in industrial operations, paving the way for more advanced and versatile robotic solutions.

BIBLIOGRAPHY

[1] John George. 5 real-world applications of quadruped robots, 2022. Accessed on Dec. 28, 2022. URL: https://www.robotics247.com/article/5_real_world_applications_of_quadruped_robots.

[2] Sereinig, M., Werth, W., & Faller, L. M. (2020). A review of the challenges in mobile manipulation: systems design and RoboCup challenges. Elektrotech. Informationstechnik, 137(6), 297-308.

[3] Xin, G., Zeng, F., & Qin, K. (2022). Loco-manipulation control for arm-mounted quadruped robots: Dynamic and kinematic strategies. Machines, 10(8), 719.

[4] Cruz Ulloa, C., Domínguez, D., Del Cerro, J., & Barrientos, A. (2022). A mixed-reality tele-operation method for high-level control of a legged-manipulator robot. Sensors, 22(21), 8146.

[5] Fu, Z., Cheng, X., & Pathak, D. (2023, March). Deep whole-body control: learning a unified policy for manipulation and locomotion. In Conference on Robot Learning (pp. 138-149). PMLR.

[6] Ferrolho, H., Ivan, V., Merkt, W., Havoutis, I., & Vijayakumar, S. (2023). Roloma: Robust loco-manipulation for quadruped robots with arms. Autonomous Robots, 47(8), 1463-1481.

[7] Gai, S., Lyu, S., Zhang, H., & Wang, D. (2024). Continual Reinforcement Learning for Quadruped Robot Locomotion. Entropy, 26(1), 93.

[8] Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., ... & Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 969-977).

[9] Borrego, J., Dehban, A., Figueiredo, R., Moreno, P., Bernardino, A., & Santos-Victor, J. (2018). Applying domain randomization to synthetic data for object category detection. arXiv preprint arXiv:1807.09834.

[10] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017, September). Domain randomization for transferring deep neural networks from simulation to the real world. In 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS) (pp. 23-30). IEEE.

[11] https://github.com/ultralytics/yolov5

[12] Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE transactions on pattern analysis and machine intelligence, 39(6), 1137-1149.

[13] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

[14] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14 (pp. 21-37). Springer International Publishing.

[15] Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2020). Robot modeling and control. John Wiley & Sons.

[16] Angeles, J. (Ed.). (2003). Fundamentals of robotic mechanical systems: theory, methods, and algorithms. New York, NY: Springer New York.

[17] Denavit, J., & Hartenberg, R. S. (1955). A kinematic notation for lower-pair mechanisms based on matrices.

[18] JhA, P. (2015). Inverse kinematic analysis of robot manipulators (Doctoral dissertation).

[19] El-Sherbiny, A., Elhosseini, M. A., & Haikal, A. Y. (2018). A comparative study of soft computing methods to solve inverse kinematics problem. Ain Shams Engineering Journal, 9(4), 2535-2548.

[20] Lynch, K. M., & Park, F. C. (2017). Modern robotics. Cambridge University Press.

[21] https://emanual.robotis.com/docs/en/dxl/

[22] https://docs.trossenrobotics.com/interbotix_xsarms_docs/specifications/wx250.html