TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

KATHMANDU ENGINEERING COLLEGE
KALIMATI, KATHMANDU

Major Project Report On

# OBJECT DETECTION BASED AUTOMATED MOBILE ROBOT

**Submitted By:**
Roshan Pandey (KAT076BEI020)

Submitted To:

DEPARTMENT OF ELECTRONICS, COMMUNICATION AND
INFORMATION ENGINEERING

KATHMANDU, NEPAL
September-2024

# ACKNOWLEDGEMENT

We feel immense pleasure and thankful for getting an opportunity to work on this major project. We would like to sincerely thank **Er. Anmol Ratna Bajracharya** and **Er. Sarina Barahi** our project supervisors for the generous and continuous support for helping us in the different aspects of the way of performing this project.

We owe our profound gratitude to our project coordinator **Er. Sujan Shrestha** and **Er. Dipen Manandhar**, who took keen interest on our project and guided us all along providing the necessary information for developing a good system. We would also like to acknowledge, thank **Er. Rajan Lama**, Head of department of Electronics, Communication, and Information Engineering for providing us with the opportunity to perform this project.

We are thankful and fortunate enough to get constant encouragement, support, constructive criticism, and guidance from all teaching staffs of Department of Electronics, Communication, and Information Engineering.

# ABSTRACT

This project proposes the control design and implementation of an object detection based automated mobile robot. It can be roughly divided into three main directions i.e. the design of vehicle, path planning of mobile vehicle, position control of robot arm system and appropriate image processing technique. The proposed system leverages computer vision algorithms to analyze visual information captured by cameras mounted on the top. The images are processed in real-time to extract relevant features such as object detection, tracking, and pose estimation. These features are then used to generate control signals, enabling precise and intuitive manipulation of the mobile robot. Our system integrates a set of state-of-the-art techniques in computer vision, machine-learning, trajectory optimization, visual serving skill that can be composed to perform robotic task.

**Keywords:** Robotics, Machine Learning, Automation, Optimization

# Contents

# List of Figures

# List of Tables

# LIST OF ABBREVIATIONS

| | |
|---|---|
| AI | Artificial Intelligence |
| CNN | Convolutional Neural Network |
| CVAT | Computer Vision Annotation Tool |
| DC | Direct Current |
| GHz | Gigahertz |
| GPIO | General Purpose Input/Output |
| GPU | Graphics Processing Unit |
| IDE | Integrated Development Environment |
| IEEE | Institute of Electrical and Electronics Engineers |
| IoT | Internet of Things |
| IP | Internet Protocol |
| LiDAR | Light Detection And Ranging |
| PWM | Pulse Width Modulation |
| RPM | Revolutions Per Minute |
| SoC | System On Chip |
| USB | Universal Serial Bus |
| VNC | Virtual Network Computing |
| YOLO | You Only Look Once |

# Chapter 1

# INTRODUCTION

## 1.1 Background Theory

Several studies have been done these last few decades to localize the robot, to avoid obstacles and to navigate safely in map or map-less environments. The Simultaneous Localization And Mapping (SLAM) technique was studied by several researchers and widely used in recent years. With technology development, the use of several advanced sensors was possible such as infra-red (IR), sonar, optical, accelerometer, gyroscope and magnetometer, but the major problem when using many sensors to achieve the localization and the navigation is the consolidation and the fusion of data to compute the robot position in real time. Generally, a Kalman filter is used to achieve this target, but the synchronization between all sensors to receive data at the same time in order to have a precise position is the major problem of this method. An alternative solution for this problem is the use of vision with suitable image processing algorithms to localize the robot in its environment with all static and dynamic obstacles.

Previous research has explored different approaches to object detection and recognition, leveraging computer vision techniques like deep learning and image processing algorithms. These methods enable the system to analyze visual information from cameras and identify objects based on their visual features, such as shape, color, and texture. By leveraging this existing knowledge, our project aims to implement state-of-the-art object detection and recognition algorithms tailored to the specific requirements of the autonomous pick and place system.

Traditionally, manual pick and place operations have been widely practiced. However, these manual processes suffer from several limitations and drawbacks. Manual pick and place tasks are labor-intensive and time-consuming, relying heavily on human workers who may experience fatigue, inconsistencies in performance, and potential errors.

Additionally, prior work has demonstrated the effectiveness of ultrasonic sensors in distance measurement for robotics applications. These sensors emit sound waves and

measure the time it takes for the waves to bounce back, providing information about the distance to objects in their vicinity. By utilizing this established theory, our project incorporates ultrasonic sensors to accurately measure the distances between objects and the robotic arm. This enables precise positioning and manipulation during the pick and place operations, ensuring efficient and reliable performance.

Furthermore, previous implementations have demonstrated the successful integration of robotic arms with autonomous systems for object manipulation. By considering principles of robotic arm kinematics, such as joint angles and joint limits, researchers have achieved precise and controlled movements required for pick and place tasks. Drawing on this background knowledge, our project aims to select an appropriate robotic arm with a gripper mechanism suitable for handling a variety of objects. We will design and implement control algorithms that coordinate the movements of the robotic arm, leveraging the information obtained from computer vision and distance measurement components.

In this project, we are building upon the existing theories and methodologies to create a comprehensive and efficient autonomous object pick and place system. By integrating the Raspberry Pi as the central processing unit, we leverage its computational capabilities, GPIO interfaces, and programming flexibility to control and coordinate the system components. Through the implementation of custom algorithms and software, we will combine the functionalities of object detection, recognition, distance measurement, and robotic arm control.

By understanding and utilizing the background theory, we aim to develop an innovative system that optimizes industrial operations by automating the pick and place process. Our project will contribute to the existing body of research and practical applications in robotics, automation, and computer vision, enabling more efficient, accurate, and safe object handling in various industries.

## 1.2 Problem Statement

In industrial settings, the manual pick and place of objects can be time-consuming, labor-intensive, and prone to errors. This leads to reduced efficiency, increased production costs, and potential safety hazards. The problem is to automate the pick and place operations for different objects to enhance productivity, accuracy, and safety in industrial operations.

## 1.3 Objectives

- To pick and place the object at a desired place.
- To sort the objects based on color.

## 1.4 Scope and Applications

- **Logistics:** The vehicle could be used to load and unload cargo from trucks, ships, and airplanes. It could also be used to pick and place objects in warehouses and distribution centers.
- **Construction:** The vehicle could be used to repair or inspect buildings and infrastructure. It could also be used to load and unload materials from construction sites.
- **Manufacturing:** The vehicle could be used to assemble products, weld parts, and paint surfaces. It could also be used to load and unload materials from manufacturing plants.
- **Military:** The vehicle could be used for a variety of military applications, such as loading and unloading weapons, repairing or inspecting equipment, and disarming bombs.

## 1.5 Organization of Project Report

The material presented in this report is organized into five chapters: Chapter 1 consists of the introduction, objective, and background of the project. The scope and application of the project are also discussed. Chapter 2 deals with the literature review that describes the past works and research that were done related to this project and the methodology that was used in those projects. Chapter 3 discusses the conceptual theories about various related aspects, and components used. Chapter 4 describes the methodology, basic design, outline, and process of the project, and Chapter 5 consists of outputs/results and analysis and finally the references.

# Chapter 2

# LITERATURE REVIEW

The integration of computer vision and robotics has led to significant advancements in various fields, including autonomous vehicles and robotic manipulation. Perception, control, and coordination issues are made worse by the addition of a robotic arm to a vehicle. Through the provision of real-time object detection, tracking, and recognition capabilities, computer vision is essential in overcoming these difficulties. For a vehicle to effectively manipulate its environment, identification and localization of items inside that environment depend on accurate perception. This literature review aims to explore the recent advancements and key research contributions in this area.

Many studies have focused on developing computer vision algorithms for robust and efficient object detection and recognition. For instance, Corke et al. (2013) proposed a computer vision-based object recognition and grasping system for autonomous robotic manipulation in unstructured environments. Their approach combined visual feature extraction, machine learning techniques, and geometric reasoning to accurately identify objects for manipulation tasks [1].

Successful interactions between the robotic arm and objects depend on effective gripping and manipulation techniques. A real-time robotic arm control system that used computer vision for object grasping and manipulation was demonstrated by Wan et al. (2016). In order to precisely grip and manipulate items, their method used visual servoing techniques to track and control the robotic arm's movements [2].

Integration of the robotic arm with the vehicle requires careful navigation and path planning to ensure obstacle avoidance and safe manipulation. Achtelik et al. (2012) proposed a visual servoing approach for a robotic arm in cluttered and uncertain environments. Their system employed computer vision techniques to guide the arm while considering obstacles and environmental uncertainties, allowing effective navigation and manipulation [3].

Numerous fields have found use for the integration of a robotic arm onto a vehicle employing computer vision. A vision-based autonomous robotic manipulation system for industrial assembly tasks was presented by Du et al. (2019). Their solution showed

the promise of this technology in industrial automation by combining motion planning and computer vision algorithms to enable precise and effective assembly operations. The areas of delivery and logistics are another use. A robotic arm control system based on the visual recognition and tracking of moving objects was proposed by Gokasan et al. (2015). They developed a method that allowed the robotic arm to handle and move things on its own, demonstrating the promise of computer vision-based control for self-driving delivery trucks. The application of vehicles with robotic arms extends to industrial assembly tasks. Du et al. (2019) propose a vision-based autonomous robotic manipulation system specifically designed for industrial assembly operations. The system employs computer vision techniques for object detection, pose estimation, and trajectory planning. The work demonstrates the feasibility of utilizing robotic arms on vehicles for complex assembly tasks [4].

Navigation systems allow mobile robots to move freely across their work environments, achieving goals and avoiding obstacles. According to [5], the tasks performed by such systems can be divided into two categories: robot localization and obstacle avoidance. Being able to know where a mobile robot is implies getting its position and orientation accurately in a specific time step. Thus, the ability to estimate its position is of great importance, since it provides a robot great autonomy and the necessary means to achieve multiple and important tasks. A widespread research area is the developing of mobile robots that help people inside their houses as well as in their everyday chores. An example of this is applying this technology inside our houses or at our work, as a complementary help or as a way of improving security. If the robot moves, it needs to navigate, consequently the processing must be done by the robot or some external agent that does all the calculations and sends all the localization information that the robot needs. Nowadays, mobile robots find a wide range of industry applications, such as cleaning floors of buildings and factories, mobile surveillance systems, transporting of items at factories, harvesting and selection of fruits, in medicine, at military, guiding people inside museums and etc. Feature extraction of images captured by a moving camera has been used to estimate its position and orientation and consequently the robot position in cases where the camera is mounted on the top of robot. This process is known as Visual Odometry [6]. It uses an image sequence to process the information needed, being very useful in robot and vehicle navigation, where other techniques cannot be used or the error rates are too high and thus more precise data is necessary. This technique is very promising since it does not produce errors related to wheel slippage, axis length, etc.

As a general method of controlling an autonomous mobile robot, there is a method of estimating the robot's self-position and controlling the robot based on using it. Nakajima et al. succeeded in detecting an obstacle, generating a path and avoiding the obstacle by measuring the distance through a laser range finder. However, using the laser range finder has the problem that the cost increases. In addition, Satake et al. [7] tracked a person

by obtaining depth information of the image using a stereo camera. The advantage of using a camera is that the cost is considerably lower than using the laser range finder. Note however that the distance measurement with a stereo camera poses a problem of expensive computational cost. On the other hand, there is a method using a monocular camera as a sensor. Watanabe et al. realized obstacle avoidance using image-based visual servoing (IBVS), together with using edge detection and fuzzy control. The advantage of using IBVS is that it is reduced to less computational effort because it need not calculate self-localization, and it can also be designed by human-like methods. Note however that when multiple obstacles appeared in the camera image, the obstacle avoidance accuracy decreased because depth information was impossible to be obtained. Watanabe et al. realized a path generation method including the depth information by taking account of the optical flow of obstacles. However, there remain several problems, e.g., it is difficult to deal with moving obstacle, there is no guarantee for approaching a target point while avoiding obstacles, etc. In this paper, a method is proposed for approaching a target point while avoiding obstacles, by setting additional potentials around the target when constructing the potential field. In addition, a collision prediction system is constructed using a convolutional neural network (CNN) and a method of stopping the robot in the vicinity of the target point while avoiding moving obstacle is proposed by switching control based on such a system.

For the design of robot arm motion, literature for kinematic analysis and modeling of robots have been presented numerously. The paper proposed a system which extracts image line features from stereo camera as landmarks, and use stereo property to obtain the 3D vertical line landmarks. The control theory of backstepping is also used for designing stabilizing controls of nonlinear dynamical systems [8, 9]. The images are captured by the camera installed on the clip of robot arm. These images must be analyzed effectively to obtain the position, shape and color of the object. The image analysis applies on the control of vehicle, object grabbing and placing. The part of vehicle is to find the desired angle to track objects. Literatures for image recognition have been presented in [10, 11].

# Chapter 3

# RELATED THEORY

## 3.1 Hardware

### 3.1.1 Raspberry Pi

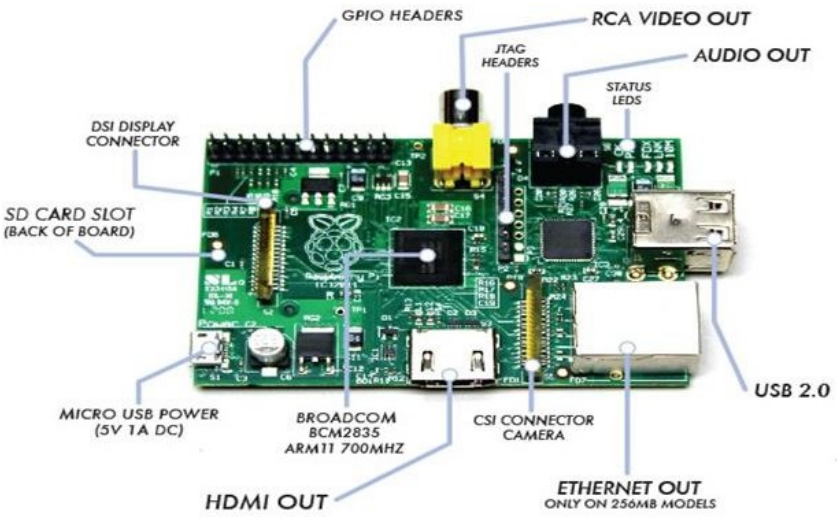



Figure 3.1: Raspberry Pi

The Raspberry Pi is a powerful, small single-board computer. It consists of a Broadcom system on a chip (SoC) with an integrated ARM-compatible CPU and on-chip graphics processing unit (GPU). It is a single-core with processor speed of device ranges from 700 MHz to 1.5GHz and a memory range from 256 MB to 8 GB RAM.

**Specification of Raspberry Pi:**

- Processor: Quad-core 1.2GHz Broadcom
- Installed RAM: 4 GB
- System type: 64 bits

- Operating voltage: 5V
- Digital I/O pins: 17
- Frequency (Clock speed): 900 MHz
- OpenGL ES 3.1, Vulkan 1.0
- Micro-SD card slot for loading operating system and data storage
- 5V DC via USB-C connector (minimum 3A)
- 5V DC via GPIO header (minimum 3A)
- Power over Ethernet (PoE) enabled (requires separate PoE HAT)

### 3.1.2 DC Motors

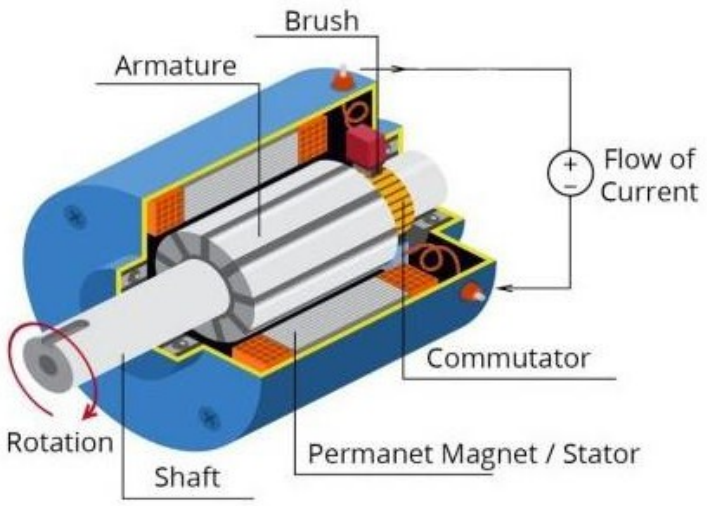



Figure 3.2: DC Motor

Direct Current or DC motor is an electric machine that helps the conversion of electrical energy into mechanical energy. DC motors include two key components: a stator and an armature. The stator is the stationary part of a motor, while the armature rotates. DC motors are used in toys, small tools, electric vehicle propulsion, elevators, and steel rolling mills.

### 3.1.3 Servo Motor

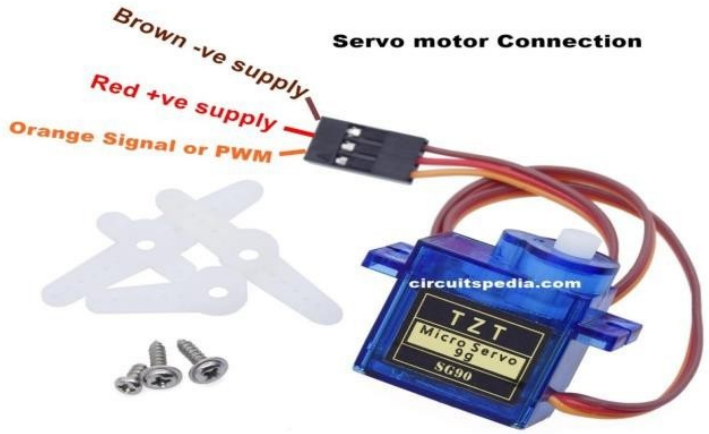



Figure 3.3: Servo Motor

A servomotor (or servo motor) is a rotary actuator or linear actuator that allows for precise control of angular or linear position, velocity and acceleration. A servomotor is a type of motor that can rotate with great precision. Normally this type of motor consists of a control circuit that provides feedback on the current position of the motor shaft; this feedback allows the servo motors to rotate with great precision. It usually comes with a gear arrangement that allows us to get a very high torque servo motor in small and lightweight packages.

**Specification of Servo Motor:**

- Operating voltage range: 4.8 V to 7.2 V
- Stall torque: 9.4kg/cm (4.8V); 11kg/cm (6V)
- Operating speed: 0.2 s/60° (4.8 V), 0.16 s/60° (6 V)
- Rotational degree: 180°
- Operating temperature range: 0°C to +55°C
- Current draw at idle: 10mA
- No load operating current draw: 170mA
- Current at maximum load: 1200mA

### 3.1.4 Li-Po Battery


Figure 3.4: Li Po Battery

Lithium-Ion polymer batteries, or LiPos are rechargeable batteries using lithium ion technology that can be recharged repeatedly. LiPo batteries offer high discharge rates and optimum energy storage/weight ratio. Today this powerful, but yet simple technology is used in a number of applications – starting years ago with mobile telephones and laptops and now moving on to new wearable technologies such as smartwatches, headsets or fitness trackers, small robots.

### 3.1.5 L298N Motor Driver

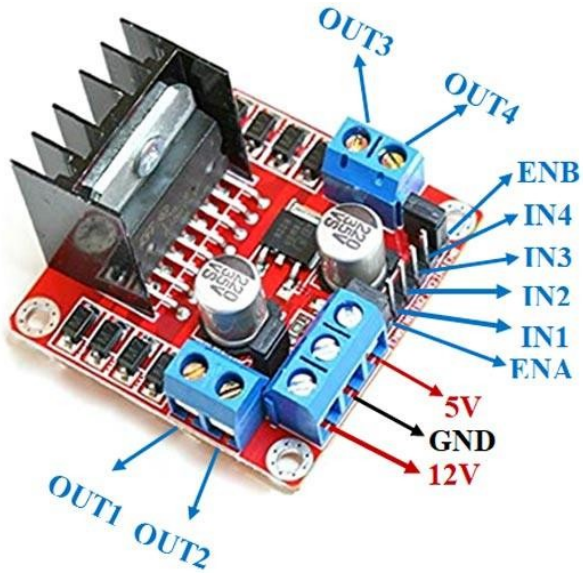



Figure 3.5: L298N Motor Driver

This L298N Motor Driver Module is a high power motor driver module for driving DC and Stepper Motors. This module consists of an L298 motor driver IC and a 78M05 5V regulator. L298N Module can control up to 4 DC motors, or 2 DC motors with directional and speed control.

### 3.1.6 Arduino Mega

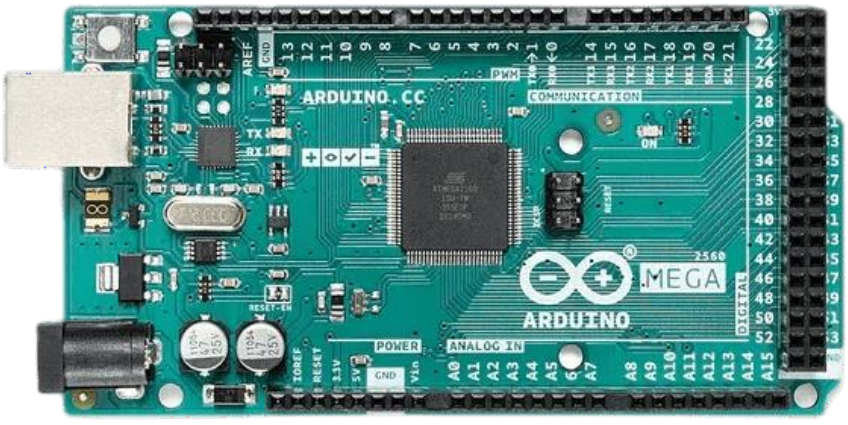



Figure 3.6: Arduino Mega

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power

jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

**Specification of Arduino Mega:**

| Parameter | Value |
|---|---|
| Microcontroller | ATmega2560 |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12V |
| Input Voltage (limit) | 6-20V |
| Digital I/O Pins | 54 (of which 15 provide PWM output) |
| Analog Input Pins | 16 |
| DC Current per I/O Pin | 20 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 256 KB (8 KB used by bootloader) |
| SRAM | 8 KB |
| EEPROM | 4 KB |
| Clock Speed | 16 MHz |

Table 3.1: Arduino Mega Specifications

### 3.1.7 Ultrasonic Sensor

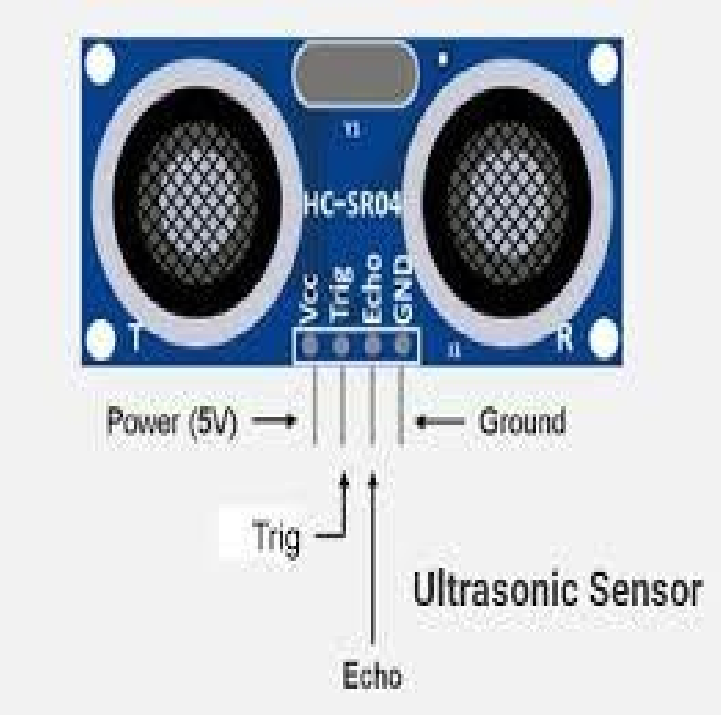



Figure 3.7: Ultrasonic Sensor

An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves. Ultrasonic sensors work by sending out a sound wave at a frequency above the range of human hearing. The transducer of the sensor acts as a microphone to receive and send the ultrasonic sound. Our ultrasonic sensor, like many others, uses a single transducer to send a pulse and to receive the echo. The sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse.

### 3.1.8 Wheels

Figure 3.8: Wheels

Wheels allow light or heavy objects to be moved easily facilitating movement or transportation while supporting a load, or performing labor in machines. A wheel reduces friction by facilitating motion by rolling together with the use of axles.

### 3.1.9 Raspberry Pi-Camera


Figure 3.9: Raspberry Pi-Camera

The 5MP camera module is perfect for small Raspberry Pi projects which have very little space allowance just boot up the latest version of Raspbian. The high-definition 5MP camera delivers outstanding photos but can also shoot video, ideal for drones or a CCTV project. The camera overcomes the disadvantages offered by our other Raspberry Pi Cameras as it has provision for night surveillance too.

### 3.1.10 Web Camera


Figure 3.10: Web Camera

Webcams are video cameras that are used to capture and transmit video in real time. They are commonly used for videoconferencing, live streaming, and other applications that require real-time video communication. Webcams have a number of advantages, including their convenience, portability, and ability to enable real-time video communication. However, they also have some limitations, such as variations in quality, reliance on a stable internet connection, and potential privacy concerns. Overall, webcams are a useful tool for enabling real-time video communication.

**Specifications:**

- Video resolution: 2560×1440
- Camera: 4 MP
- Field of view: 106 degrees
- Frame rate: 2k/25fps
- Cord length: 1.4mm
- Weight: 87gm

## 3.2 Software

### 3.2.1 Python


Figure 3.11: Python

Python is a high-level, interpreted programming language that is widely used for general-purpose programming, scientific computing, data analysis, artificial intelligence, web development, and many other applications. Python is known for its clear and concise syntax, which makes it easy to read and write, and its extensive library of modules, which allow developers to quickly and easily build complex applications. Python is open-source, meaning it is freely available and can be used and modified by anyone. It runs on all major operating systems and is often used as a scripting language for automating tasks or as a glue language for integrating different software components.

### 3.2.2 Raspberry Pi Imager

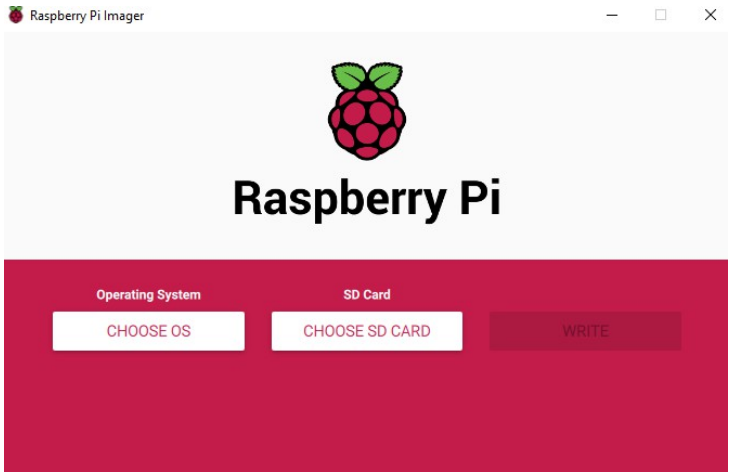



Figure 3.12: Raspberry Pi Imager

Raspberry Pi Imager is a free and open-source tool that allows users to create and flash operating system images onto the SD card of a Raspberry Pi. Raspberry Pi Imager also supports custom user-defined OS images, automatic download of Raspberry Pi OS image updates, and the ability to write to multiple SD cards at the same time, making it useful for large-scale Raspberry Pi deployments. The Raspberry Pi Foundation created and maintains the Raspberry Pi Imager, a free and open-source tool. It is widely used in the Raspberry Pi community by educators and professionals for a variety of applications such as home automation, robotics, and Internet of Things (IoT) projects.

### 3.2.3 MobaXterm


Figure 3.13: MobaXterm

MobaXterm is a remote computing toolkit. It offers a wide range of functions in a single Windows application that are tailored for programmers, webmasters, IT administrators, and pretty much all users who need to handle their remote jobs more simply. MobaXterm brings all of the essential remote network tools (SSH, VNC, and Unix commands (bash)) to the Windows desktop in a single portable exe file that works right away.

### 3.2.4 VNC Viewer

Figure 3.14: VNC Viewer

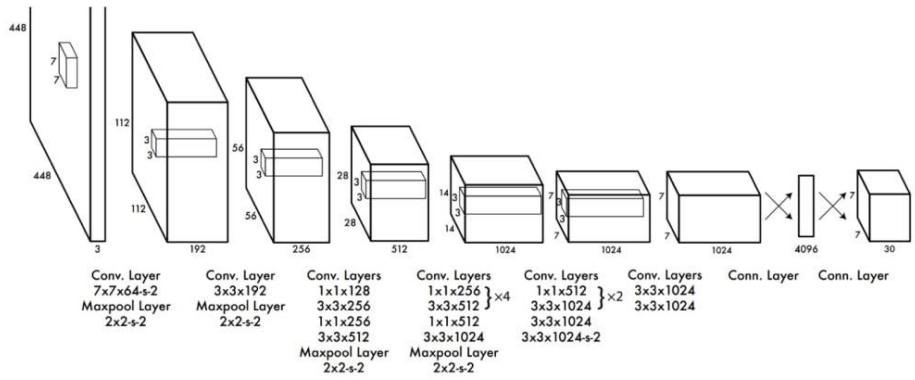
VNC Viewer is a remote desktop application that enables users to remotely access and control a computer over a network by utilizing the Virtual Network Computing (VNC) protocol. Users can use VNC Viewer to view and interact with a remote computer's graphical desktop as if they were sitting in front of it, making it useful for remote support, collaboration, and access to remote resources. VNC Viewer employs a client-server architecture, with the server installed on the computer to be remotely accessed and the client installed on the computer from which the remote access will be performed.

### 3.2.5 YOLO

You Only Look Once (YOLO) proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform detection. Following a fundamentally different approach to object detection, YOLO achieved state-of-the-art results, beating other real-time object detection algorithms by a large margin.

While algorithms like Faster RCNN work by detecting possible regions of interest using the Region Proposal Network and then performing recognition on those regions separately, YOLO performs all of its predictions with the help of a single fully connected layer.

The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image. The architecture of the CNN model that forms the backbone of YOLO is shown below.



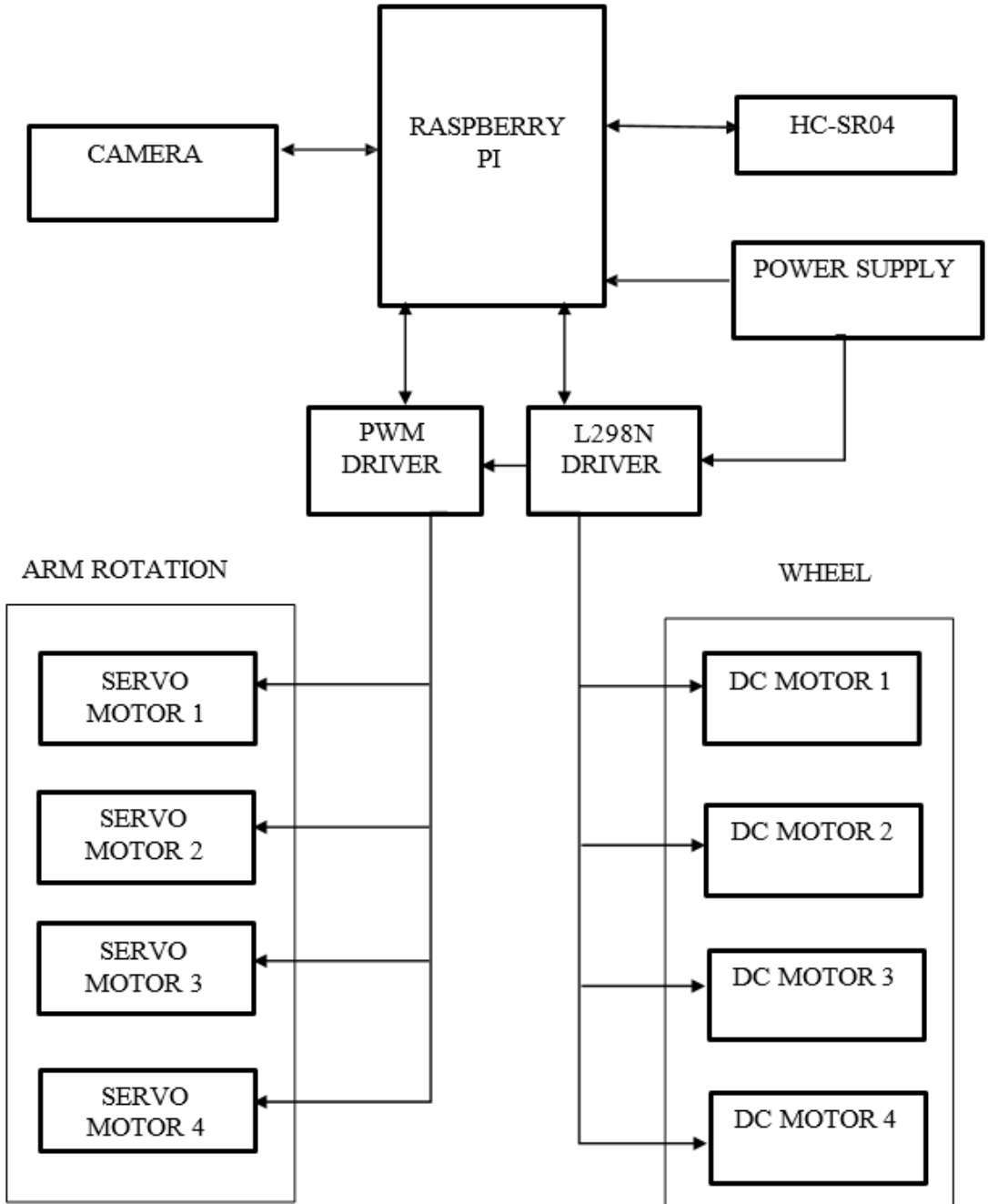

**The Architecture.** Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating $1 \times 1$ convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution ($224 \times 224$ input image) and then double the resolution for detection.

Figure 3.15: YOLO Architecture

# Chapter 4

# METHODOLOGY

## 4.1 Block Diagram



**Raspberry Pi Camera:** The Raspberry Pi Camera module is a small camera attachment that connects to the Raspberry Pi board. It captures the visual data of the surrounding environment and provides it to the computer vision system for processing. The camera module is typically controlled and configured through the Raspberry Pi board.

**Raspberry Pi Board:** The Raspberry Pi is a compact and powerful single-board computer. It serves as the central processing unit for the system, running the necessary software and algorithms to control the vehicle and robotic arm. The Raspberry Pi board receives the image data from the camera and processes it using computer vision algorithms.

**Servo Motor for Robotic Arm:** The servo motor is responsible for actuating the robotic arm. It receives control signals from the Raspberry Pi board, which specify the desired position or movement of the arm. Servo motors are commonly used for their precise control over the arm's position and ability to hold specific angles.

**DC Motor for Vehicle Wheel:** The DC motor is the driving force behind the vehicle's mobility. It receives commands from the Raspberry Pi board to control the speed and direction of the vehicle. The motor's rotation is converted into the forward or backward movement of the wheels, enabling the vehicle to navigate its environment.

**Power Supply:** Power management ensures the appropriate power supply to the Raspberry Pi board, camera module, servo motor, and DC motor.

## 4.2 Algorithms

### 4.2.1 Algorithm for the System

**Data:** Camera input, detected objects
**Result:** Object picked and placed
Initialize the robot and sensors;
**repeat**
  Capture an image using the camera;
  Process the image to detect the objects;
  **if** *objects detected* **then**
    Calculate the position of the object based on the detected image;
    Move the robot towards the detected object;
    Move the arm to the object's position;
    Lower the arm to a suitable height for picking up the object;
    **repeat**
      Measure the distance to the object using distance measuring algorithm;
      **if** *object not at desired distance* **then**
        Adjust the arm position to align with the object more accurately;
      **end**
    **until** *object is at desired distance*;
    Lower the arm to pick up the object;
    **if** *object not securely held* **then**
      Adjust the grip or repeat the picking process;
    **end**
    Lift the arm with the object;
    Move the robot to the destination;
    Move the arm to the desired placement location;
    Lower the arm to release the object at the desired location;
  **end**
**until** *all objects successfully placed*;
Stop;

**Algorithm 1:** System Algorithm

### 4.2.2 Algorithm for Object Detection and Tracking

**Data:** Camera input
**Result:** Object detected and tracked
Start;
Initialize the camera;
**repeat**
    Capture an image using camera;
    Convert RGB to grayscale;
    Apply Canny edge detection;
    Apply region of interest to the image;
    Pass the image to the trained CNN module;
**until** *object detected*;
Make a boundary box to the identified object;
Apply Kalman filter for object tracking;
  - Initialization;
  - Prediction;
  - Correction;
End;

**Algorithm 2:** Object Detection and Tracking Algorithm

### 4.2.3 CNN Algorithm

**Data:** Input image
**Result:** Detected objects with bounding boxes
Start;
Input Image;
Select Region Proposal Method (e.g., Selective Search);
Generate Region Proposals;
Preprocess Region Proposals;
Warp Regions to a Fixed Size;
Apply Preprocessing (subtract mean, normalize);
**CNN Feature Extraction:**;
  Pass Region Proposals through CNN Layers;
  Extract Convolutional Features for each Region;
**Object Classification:**;
  Train an SVM (Support Vector Machine) on Extracted Features;
  Classify each Region Proposal as Object or Background;
**Region Refinement:**;
  Train a Bounding Box Regressor on Extracted Features;
  Adjust Region Proposals by Applying Regression;
**Non-maximum Suppression:**;
  Filter out Overlapping Region Proposals;
**Output Detected Objects:**;
  Generate Final Bounding Box Coordinates;
  Assign Class Labels to Detected Objects;
End;

**Algorithm 3:** CNN-Based Object Detection Algorithm

### 4.2.4 Algorithm for Color Detection

**Data:** Video capture from camera
**Result:** Colored objects detected
Import necessary libraries;
Initialize camera for video capture;
Define frame for the capture;
**while** *camera is capturing image* **do**
    Convert RGB to HSV;
    Declare HSV range for Red and Blue;
    Create mask for Red and Blue;
    /* *mask shows the declared color only on another frame and erodes the rest* */;
    Define contour for the colors;
    Bind the colored objects with contour according to the mask;
    Detect the colored object;
**end**

**Algorithm 4:** Color Detection Algorithm

## 4.3 Robotic Arm Movement Control

Kinematics is the study of motion without considering the cause of the motion, such as forces and torques. Inverse kinematics is the use of kinematic equations to determine the motion of a robot to reach a desired position. For example, to perform automated bin picking, a robotic arm used in a manufacturing line needs precise motion from an initial position to a desired position between bins and manufacturing machines. The grasping end of a robot arm is designated as the end-effector. The robot configuration is a list of joint positions that are within the position limits of the robot model and do not violate any constraints the robot has.

Given the desired robot's end-effector positions, inverse kinematics (IK) can determine an appropriate joint configuration for which the end-effectors move to the target pose.

### 4.3.1 Forward Kinematics

Forward kinematics calculates the position and orientation of the end-effector with respect to the given reference frame, given the set of joint-link parameters. There are two different methods used: Denavit-Hartenberg (D-H) parameter and successive screw displacements. D-H method and geometric methods are used in this paper for implementing forward kinematics. D-H parameters work using four parameters: twist angle, link length, link offset and joint angle $(\alpha_n, a_n, d_n, \theta_n)$. Orthonormal coordinate frames are attached to each

of the links of the robotic arm as shown in Figure 4.2. Transformation matrix describes the position and orientation of the frames which are assigned for each link.

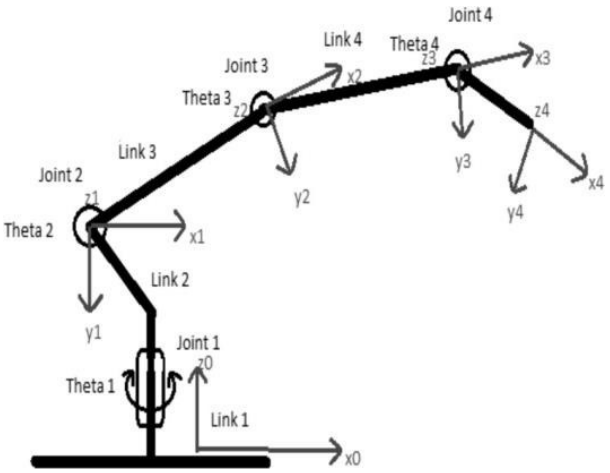



Figure 4.2: Coordinate Frame of 4 DOF Arm Robot

| **Joint** | $\theta_n$ | $\alpha_n$ | $a_n$ | $d_n$ |
|---|---|---|---|---|
| 1 | $\theta_1$ | 0 | L1 | D1 |
| 2 | $\theta_2$ | 0 | L2 | D2 |
| 3 | $\theta_3$ | 0 | L3 | D3 |
| 4 | $\theta_4$ | 0 | L4 | D4 |

Table 4.1: DH Parameters

**Homogeneous Transformation Matrix Equation:**

$$^{i-1}A_i = \text{Trans}(0,0,d_i)\text{Rot}(z,\theta_i)\text{Trans}(a_i,0,0)\text{Rot}(x,\alpha_i)$$

$$^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\alpha_i\cos\theta_i & -\sin\alpha_i\cos\theta_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_1 = \begin{bmatrix} \cos\theta_1 & 0 & -\sin\theta_1 & 50\cos\theta_1 \\ \sin\theta_1 & 0 & \cos\theta_1 & 50\sin\theta_1 \\ 0 & -1 & 0 & 145 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2 = \begin{bmatrix} \cos\theta_2 & 0 & -\sin\theta_2 & 90\cos\theta_2 \\ \sin\theta_2 & 0 & \cos\theta_2 & 90\sin\theta_2 \\ 0 & -1 & 0 & 145 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3 = \begin{bmatrix} \cos\theta_3 & 0 & -\sin\theta_3 & 180\cos\theta_3 \\ \sin\theta_3 & 0 & \cos\theta_3 & 180\sin\theta_3 \\ 0 & -1 & 0 & 145 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4 = \begin{bmatrix} \cos\theta_4 & 0 & -\sin\theta_4 & 65\cos\theta_4 \\ \sin\theta_4 & 0 & \cos\theta_4 & 65\sin\theta_4 \\ 0 & -1 & 0 & 145 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T = T_1 \times T_2 \times T_3 \times T_4$$

The forward kinematics equation was derived by the multiplication of the transformation matrices. Forward Kinematics Equation:

$$T = \begin{bmatrix} r_{11} & r_{12} & r_{13} & P_x \\ r_{21} & r_{22} & r_{23} & P_y \\ r_{31} & r_{32} & r_{33} & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$P_x = \frac{\cos(\theta_1) \times [130\cos(\theta_2+\theta_3+\theta_4) + 216\cos(\theta_2+\theta_3) + 180\cos(\theta_2) + 100]}{2}$$

$$P_y = \frac{\sin(\theta_1) \times [130\cos(\theta_2+\theta_3+\theta_4) + 216\cos(\theta_2+\theta_3) + 180\cos(\theta_2) + 100]}{2}$$

$$P_z = 145 - 108\sin(\theta_2+\theta_3) - 90\sin(\theta_2) - \frac{130\sin(\theta_2+\theta_3+\theta_4)}{2}$$

The result from the transformation matrix obtained the position of end effector of the robot in Cartesian coordinates $(P_x, P_y, P_z)$.

### 4.3.2 Inverse Kinematics

Inverse Kinematics is the procedure in which the joints are controlled in order to achieve the end position, given the position and orientation of the robotic arm. Solving Inverse kinematics is important compared to forward kinematics, as it can move the gripper to the target position, which would be helpful in grasping any object at the target location. Inverse kinematics of the robot can be obtained by using many methods, one of them is using algebraic approach. This technique is based on the output of forward kinematics,

specifically the values of $P_x$, $P_y$, and $P_z$.

$$\theta_1 = \text{atan2}(P_y, P_x)$$

Let:

$$R = \sqrt{x_d^2 + y_d^2} - L_1$$

$$S = \sqrt{R^2 + (L_3 + L_4 \times \cos(\theta_4))^2}$$

$$T = \text{atan2}(L_3 + L_4 \times \cos(\theta_4), R)$$

$$\theta_2 = \arccos\left(\frac{L_2^2 + S^2 - L_4^2}{2 \times L_2 \times S}\right) - T$$

Let:

$$U = \arccos\left(\frac{L_2^2 + L_4^2 - S^2}{2 \times L_2 \times L_4}\right)$$

$$\theta_3 = \pi - U$$

$$\theta_4 = \theta_3 - \theta_2$$

The equations above represent the ultimate outcome of the robot's inverse kinematics calculation.

## 4.4 Flow Charts

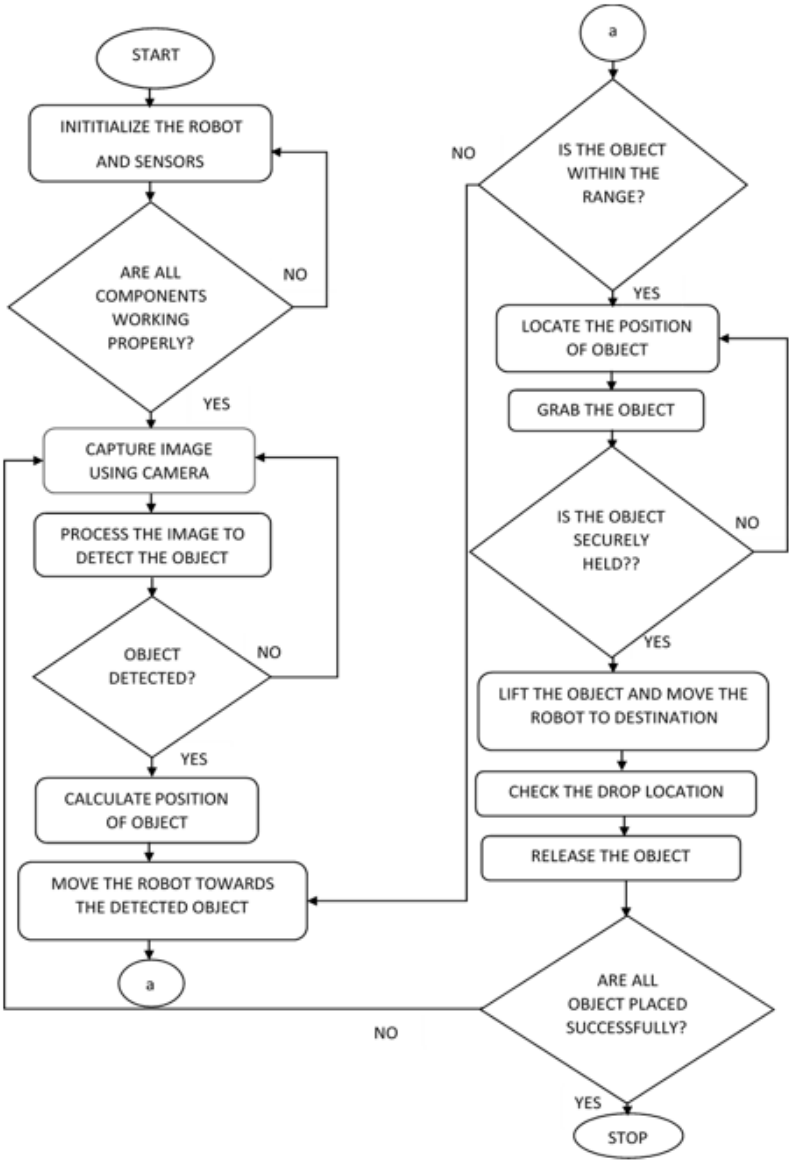



Figure 4.3: Flow Chart of Proposed System

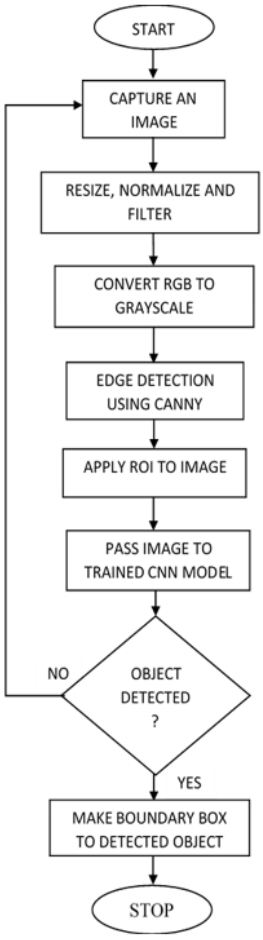



Figure 4.4: Flow Chart of Object Detection

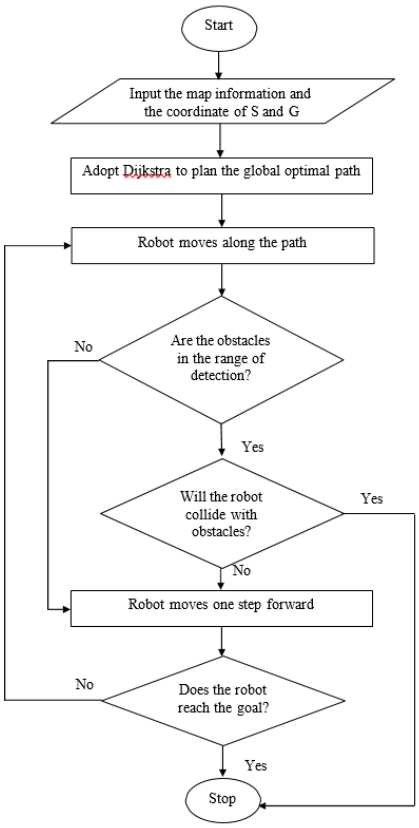



Figure 4.5: Flow Chart for Motion Planning

# Chapter 5

# RESULTS AND ANALYSIS

## 5.1 Hardware Integration

Figure 5.1: Hardware Assembled

In the first phase of hardware integration, we collected every hardware component required for the project from the project department. Then we started assembling every hardware part. First, we took the vehicle kit and connected every wheel with a different motor; every wheel has its own motor, totaling four motors. Then the motors are connected to a motor driver. The Motor Driver Module is a high-power motor driver module for driving DC and Stepper Motors. The motor driver is connected to the Raspberry Pi which is powered by a LiPo battery, and to regulate the constant voltage of 5V we have used a buck converter which provides a constant 5V.

The motor driver we are using is the L298N motor driver. The L298N Motor Driver module consists of an L298 IC Dual H-bridge, 5V voltage regulator, resistors, capacitors, Power LED, 5V jumper, 2 DC motor output pins, 12-volt external motor power supply, motor direction control pins (IN1, IN2, IN3, IN4), motor output enable pins (ENA, ENB), and a heat sink. We work every component on the Raspberry Pi. It provides a set of GPIO pins, allowing you to control electronic components for physical computing and explore the Internet of Things (IoT). The Raspberry Pi controls every component in the hardware section.

In the second phase of hardware integration, we meticulously designed and modeled a 4 degree-of-freedom (DOF) robotic arm using Computer-Aided Design (CAD) software. The arm was then brought to life through 3D printing, utilizing green filament. Our printing settings were optimized for efficiency and precision: a print speed of 80mm/s, hot-end temperature set to 200°C, bed temperature at 60°C, layer height of 0.2mm, shell thickness of 0.8mm, and an infill density of 20%. Following the successful printing process, we seamlessly integrated the arm with four MG90S servo motors, ensuring smooth functionality and precise control.

After completing the individual phases of hardware integration, we seamlessly combined both the vehicle kit and the robotic arm into a unified system. This integration marks the culmination of our efforts, resulting in a versatile and capable robotic platform. With the vehicle's mobility and the arm's manipulation capabilities, our system is poised to tackle a wide range of tasks and scenarios with efficiency and precision.

## 5.2 Object Detection

### 5.2.1 Data Collection

Firstly, we collected the image data for the various classes i.e., red box, yellow box, robot picture, destination and filtered out the unwanted and unclear images.

### 5.2.2 Data Augmentation

Then, we augmented the original image data to create new, modified samples of images by applying various transformations. The reason for the augmentation in our case is the small size of our dataset as it artificially increases the diversity and size of a training dataset by applying various transformations to the existing data.

Different transformations such as flipping, rotation, zoom, crop and resize, brightness and contrast adjustment, noise addition and many more can be done to the image. We used such transformations in our dataset with the help of a Python library named Albumentations.

Figure 5.2: Augmentation

This way we created the dataset for our object detection task to be fed into the YOLO v5 pre-trained model. The dataset was then split into train (70%), validation (20%) and test (10%) sets with each folder containing different sets of images.

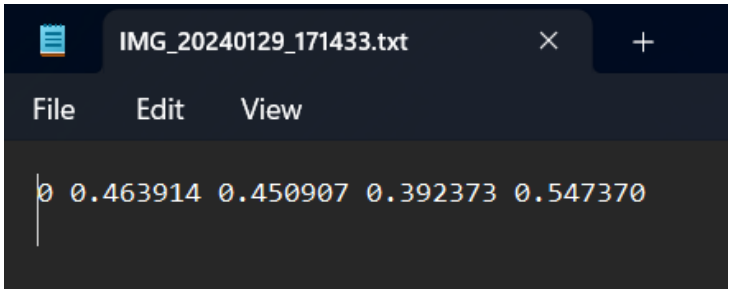



Figure 5.3: YOLO Annotation Value

Then, we labeled the objects to be detected in every image using CVAT which is free and open source, and easy to use. Annotation files of YOLO format were generated for each image and stored in the labels folder.

### 5.2.3 Training and Results

After the dataset was formed, we trained the pretrained model of YOLO v8 using only images of the desired classes. By using the Ultralytics package, we loaded the pretrained model after modification in its YAML file and trained it for 50 epochs.

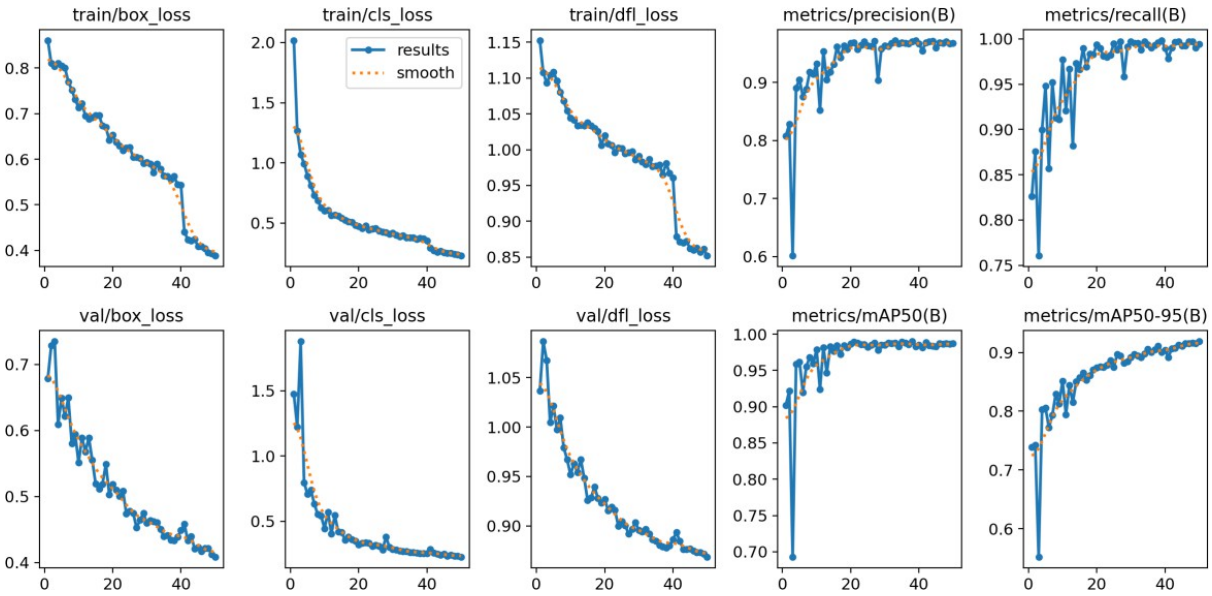



Figure 5.4: Performance Matrix Graphs

As seen in Figure 5.4, the losses are gradually decreasing and the precision and recall are gradually increasing as the epoch increases.

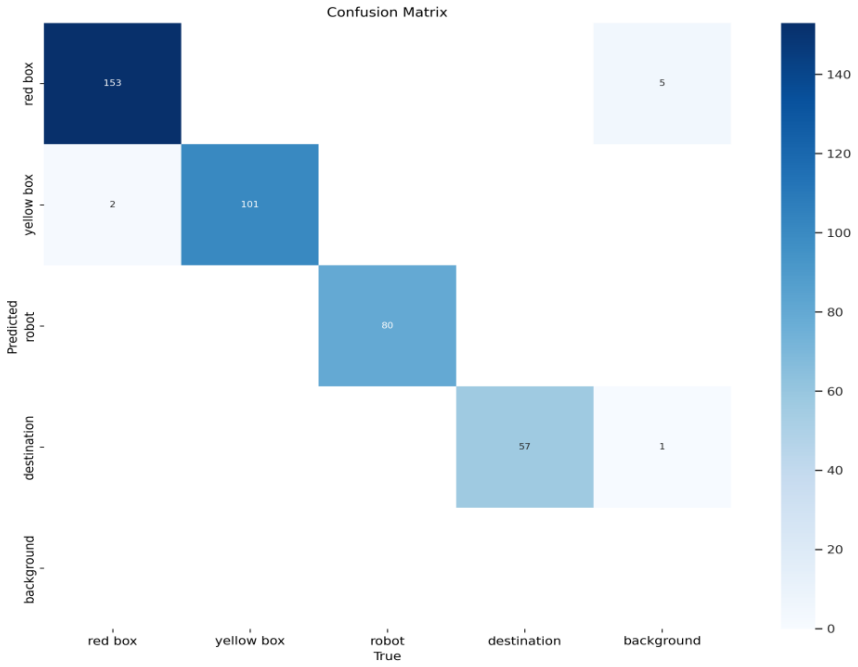



Figure 5.5: Confusion Matrix

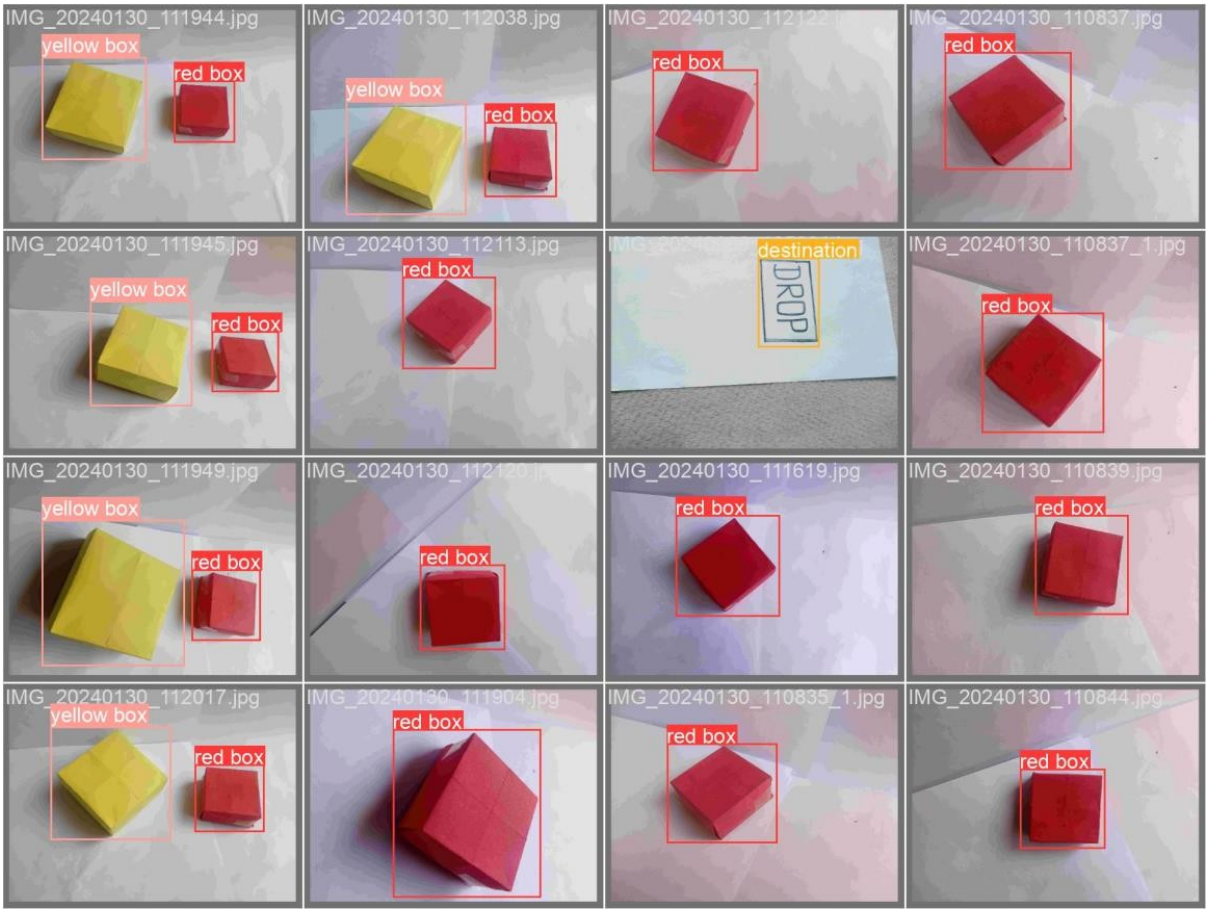



Figure 5.6: Labeled Image Data

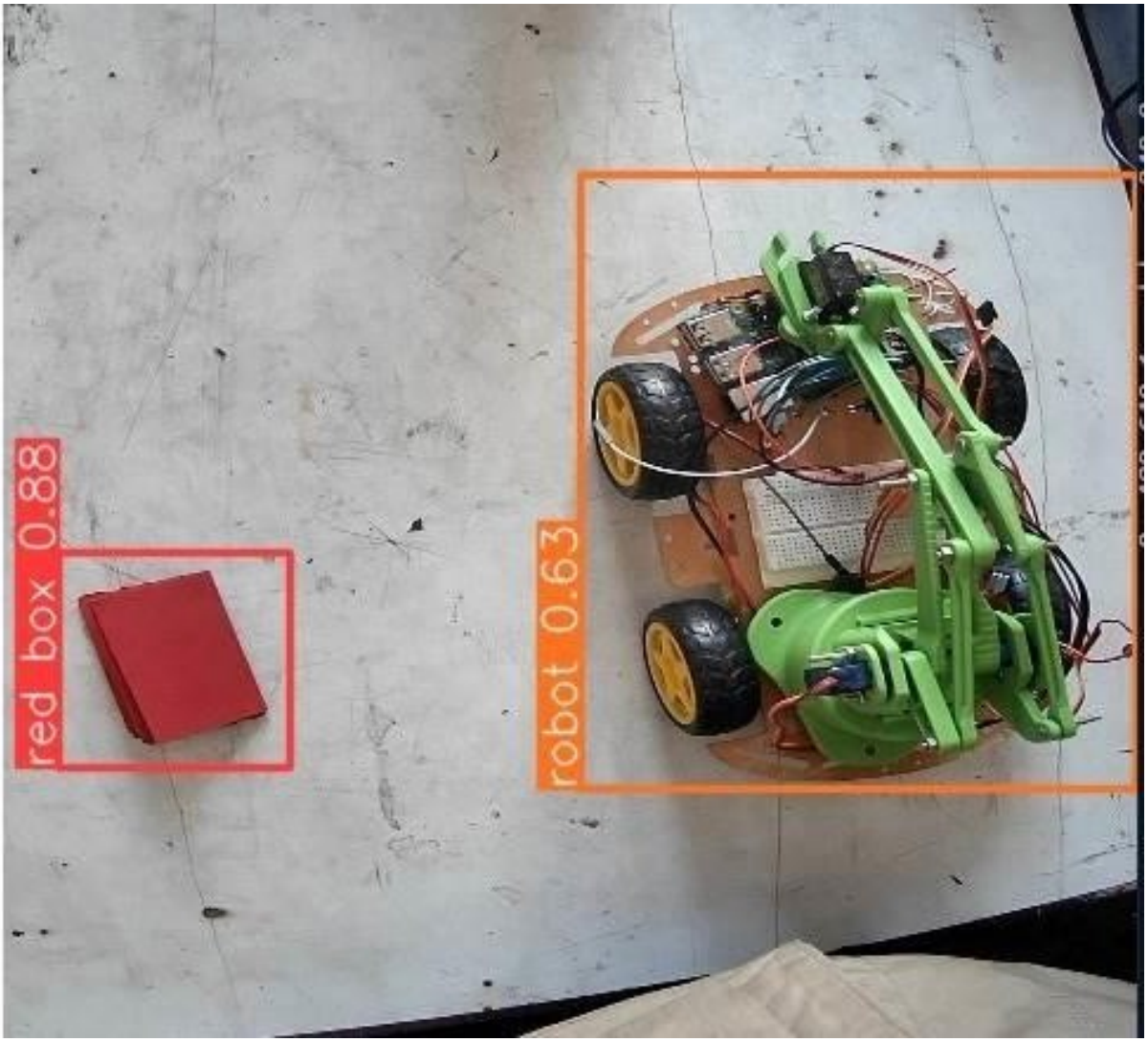



Figure 5.7: Predicted Image

The labeled images that were fed to the YOLO model are shown in Figure 5.6. These images were stored in the train folder. The images in the validation folder were given as input to the trained model and the result is shown in Figure 5.7 with bounding boxes and confidence values. Model accuracy on a validation set is shown in Figure 5.8 and the differences between predicted and true boxes is shown in Figure 5.9.

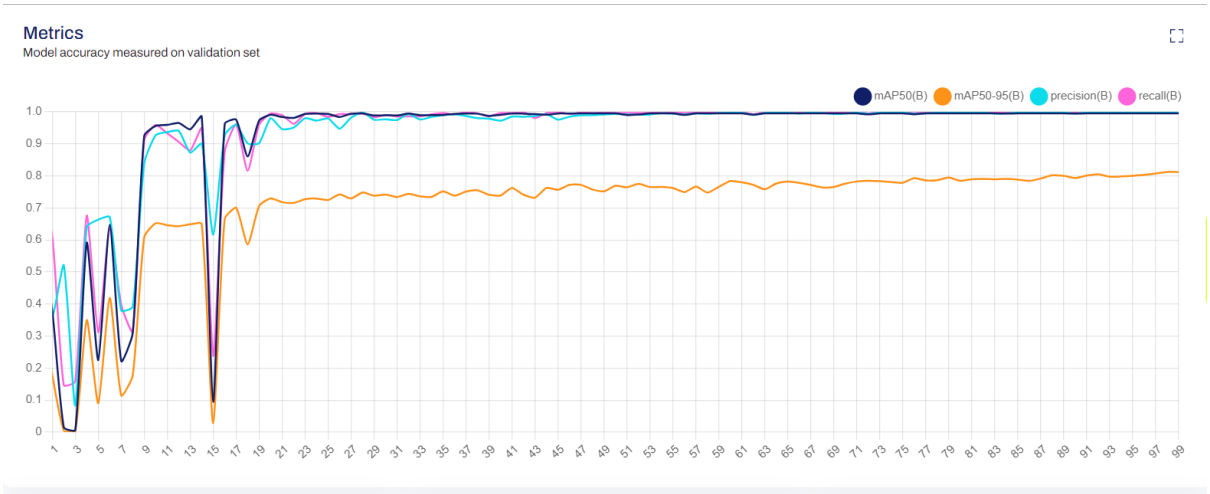



Figure 5.8: Model Accuracy on Validation Set

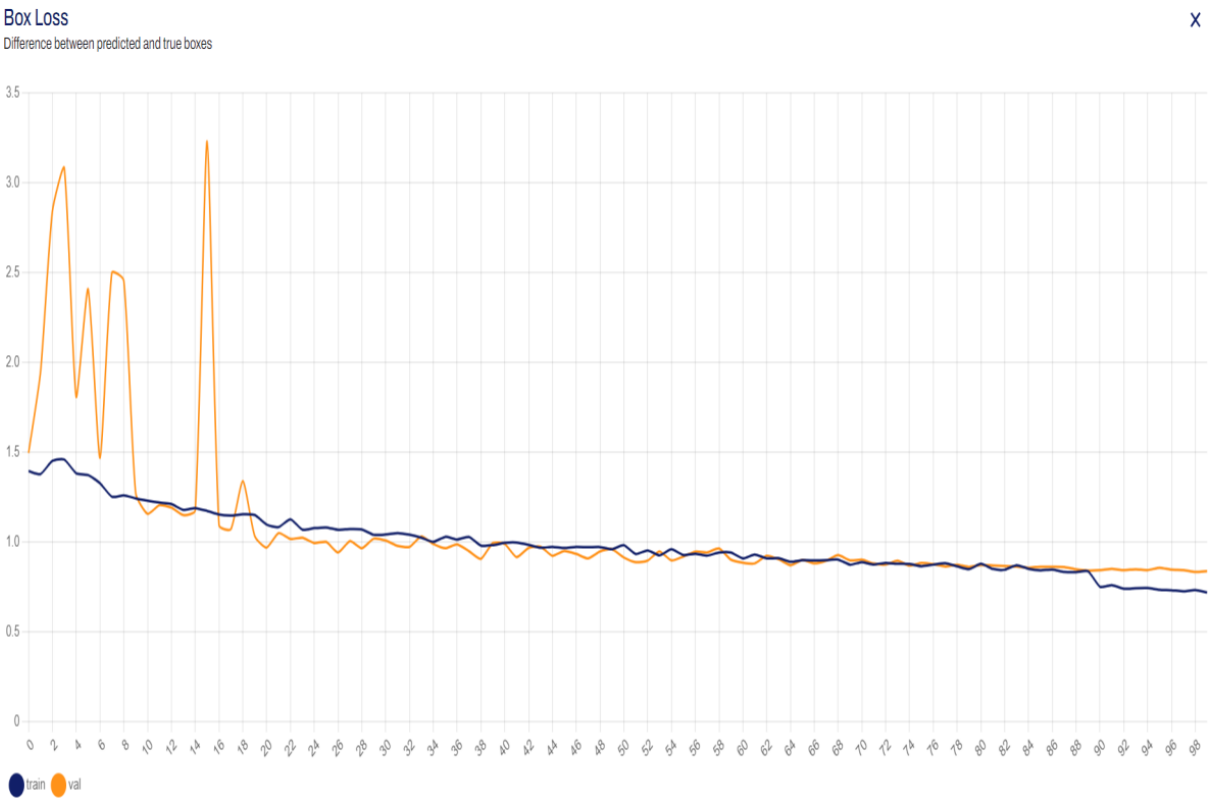



Figure 5.9: Difference between Predicted and True Boxes

## 5.3 Integration and Path Detection

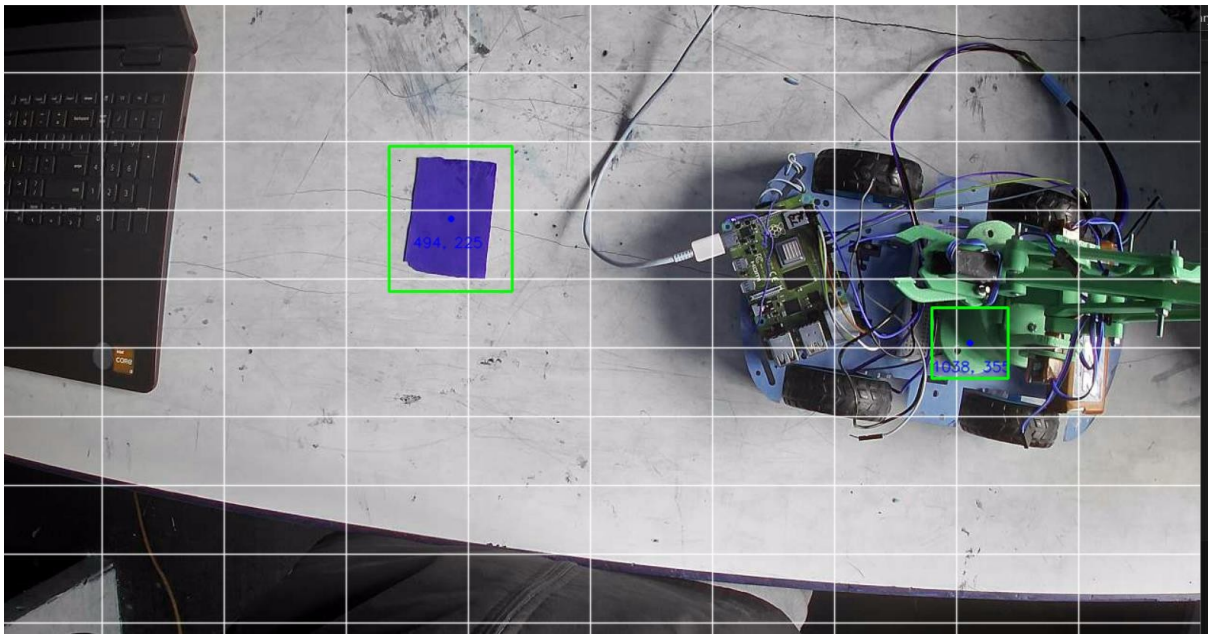



Figure 5.10: Predicted Image with Grid and Coordinates

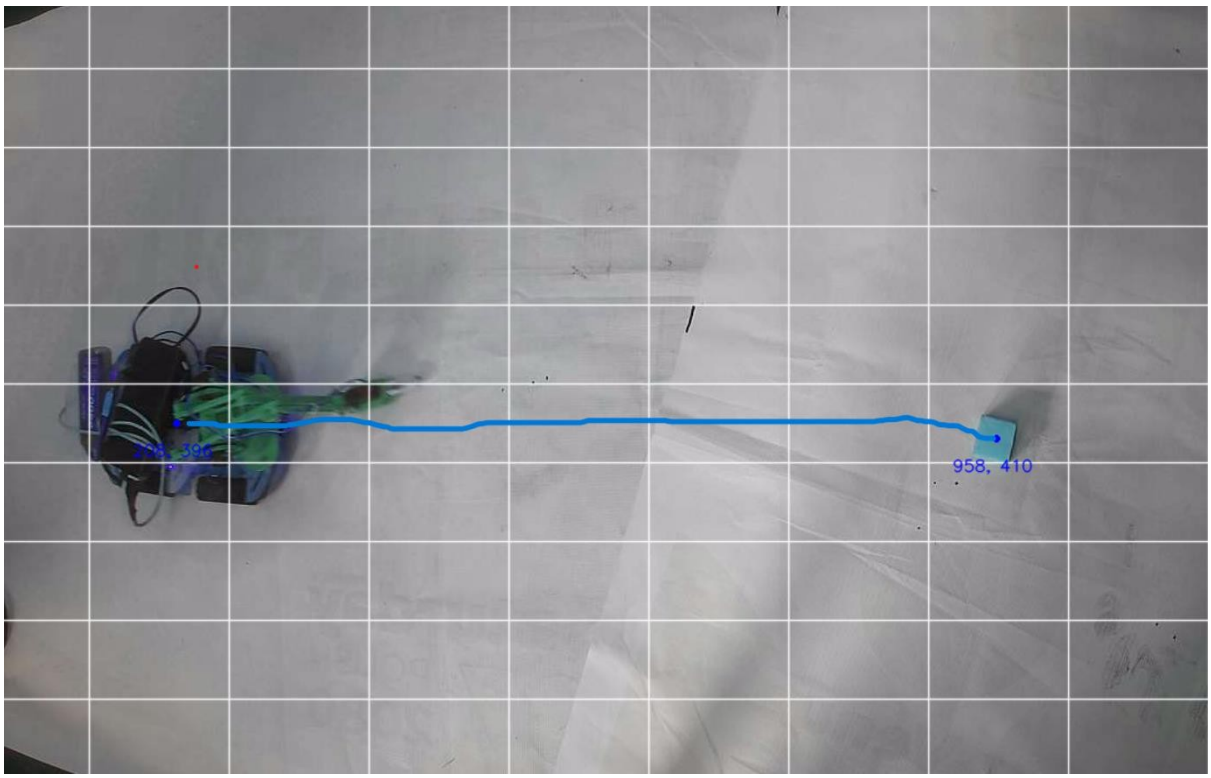



Figure 5.11: Predicted Image with Coordinates and Path

Grid system is used for the path of our vehicle. This means we break down the area into smaller squares like a checkerboard. It figures out exact starting and ending points for the

vehicle within this grid as shown in Figure 5.10 and creates a path as shown in Figure 5.11. We actually get the center points of both the robot and the object. Then, the vehicle follows a route from the starting point to the ending point, kind of like connecting the dots.

In this system, we have utilized YOLOv5 model for training and have built a custom dataset to detect the robot and objects within the camera frame. YOLOv5 provides bounding box coordinates and class probabilities for each detected object, enabling the identification of the robot and the target object. It extracts the coordinates of the detected robot and object from the bounding boxes. These coordinates represent the positions of the robot and the object in the camera frame. The environment is divided into a grid, where each cell represents a discrete location. The coordinate of the robot as the start node and the coordinate of the object as the goal node. Then applied Dijkstra's algorithm to find the shortest path from the start node (robot position) to the goal node (object position) in the map representation. Dijkstra's algorithm iteratively explores nodes in the graph, updating the shortest path to each node until the goal node is reached or all reachable nodes have been explored. It retrieves the shortest path from the robot to the object. The robot follows the shortest path that consists of a sequence of nodes or coordinates to reach the objects. The controller adjusts the robot's movement based on the planned path and camera feedback to navigate towards the object. This method helps the vehicle move smoothly towards its target to pick up an object.

## 5.4 Problems Encountered

The problems encountered in this project are listed as follows:

- **Hardware Compatibility Issue:** The Raspberry Pi 2GB faced hardware capability challenges due to its limited processing power for training, preprocessing, and model implementation. As a consequence, significant delays between consecutive frames were experienced. Moreover, during implementation, the disparity in processing speeds between the camera and motor resulted in inaccurate decisions for the robot.

## 5.5 Future Enhancements

We recommend these enhancements for the future:

- Use stereo vision camera instead of USB camera as it has more depth perception, 3D reconstruction for mapping and robustness to lighting conditions.

- Use LiDAR for object detection as it has 360-degree coverage, highly accurate distance measurements, etc.

# Bibliography


[1] P. L. J. Corke and D. A. J., "Computer Vision-based Object Recognition and Grasping for Autonomous Robotic Manipulation in Unstructured Environments," *IEEE Transactions on Robotics*, vol. II, pp. 15–28, 2013.

[2] S. A. Wan, C. G., and B. J., "Real-Time Robotic Arm Control Using Computer Vision for Object Grasping and Manipulation," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. III, no. 10, pp. 50–55, 2016.

[3] M. Achtelik, K. S., and B. W., "Visual Servoing of a Robotic Arm for Grasping in Cluttered and Uncertain Environments," *International Journal of Robotics Research*, vol. IV, no. 34, pp. 433–451, 2012.

[4] H. Du, X. L., Z., and Z. A., "Vision-based Autonomous Robotic Manipulation for Industrial Assembly Tasks," *IEEE Robotics and Automation Letters*, vol. III, no. 2, pp. 3710–3717, 2019.

[5] M. Gokasan, M. M., and E. O., "Robotic Arm Control Based on Visual Detection and Tracking of Moving Objects," *International Conference on Recent Advances in Space Technologies (RAST)*, pp. 1517–1581, 2015.

[6] L. Sciavicco and B. Siciliano, *Modeling and Control of Robot Manipulators*, New York: McGraw-Hill, 1996.

[7] C. H.-H. Chang, S. Y., L., and Y. C., *SLAM for Indoor Environment*, Wuhan: Hubei, 2010.

[8] F.-S. Chen and J.-S. Lin, "Nonlinear Control Design of Robotic Manipulators," Prague, 2005.

[9] M. Krstic, I. K., and P. V. K., *Nonlinear and Adaptive Control Design*, New York: John Wiley and Sons, 1995.

[10] R. M. Haralick and G. S. Linda, *Computer and Robot Vision*, Washington: Addison-Wesley Pub. C., 1992.

[11] A. R. Weeks, *Fundamentals of Electronic Image Processing*, New York: SPIE Optical Engineering Press, 1996.

[12] Y. Nakashima, N. W., T. Y., N. A., and M. Y., "Path generation of mobile robot in polar coordinate using laser range finder," Japan, 2009.

[13] T. K. Watanabe, S. M., and K., "Obstacle Avoidance for Mobile robots using an image based controller," *Annual Conference on the IEEE Industrial Electronics Society*, Japan, 2013.

[14] S. Shojaeipour, S. M., H. E., G. A., and S., "Webcam-based Mobile Robot Path Planning using Voronoi Diagrams and Image processing," Wisconsin USA, 2010.