TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PASCHIMANCHAL CAMPUS

A Major Project Final Report On

Delta Arm Waste Classifier

Submitted by:

Prateek Paudel (076/BEI/024)

Samman Shrestha (076/BEI/029)

Shiva Shrestha (076/BEI/038)

Sudarshan Gurung (076/BEI/044)

Submitted to:

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

PASCHIMANCHAL CAMPUS

LAMACHAUR, POKHARA

FALGUN, 2080

COPYRIGHT

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to our dedicated project supervisor, Ms. Smita Adhikari, for her unwavering support, valuable insights and continuous encouragement that enabled us in bringing up this project in the present form. We solely take the responsibility of any possible mistakes that may have occured in preparing this report and we would like to welcome comments and queries during the submission of this report. We would also like to extend our heartfelt thanks to Mr.Pujan Pariyar (BAM076) and his team for their invaluable assistance in frame construction for this project. Your skill and help made this project possible.

Finally, we would like to thank all the staffs of Institute of Engineering Pashchimanchal Campus who helped us throughout our project.

ABSTRACT

The Delta Arm waste classifier presented in this project leverages deep learning, specifically the YOLO (You Only Look Once) algorithm, combined with a Delta Arm for efficient waste sorting. The system aims to streamline recycling processes by automating the identification and separation of different waste materials. The core components of the project include a camera for image capture, a control unit for waste classification using deep learning models trained on custom datasets, and a Delta Arm with a vacuum suction pump for waste pickup and placement. The waste items are classified into categories such as paper, plastic, metal, and biodegradable waste. The YOLO algorithm, implemented using Convolutional Neural Networks (CNN), enables real-time object detection and classification. The coordinates of the waste items are calculated by the control unit, and the Delta Arm is controlled by Arduino, guided by these coordinates. By combining the power of deep learning, object detection, and robotics, this system aims to enhance waste recycling efficiency and contribute to sustainable waste management practices.

Keywords: Deep Learning, Delta Arm, Object detection, Waste classification, Waste sorting, YOLO

TABLE OF CONTENTS

List Of Figures

List Of Tables

LIST OF ABBREVIATIONS

AI	*Artificial Intelligence*
CNN	*Convolutional Neural Network*
CV	*Computer Vision*
YOLO	*You Only Look Once*

CHAPTER 1: INTRODUCTION

1.1 Background

In today's world, waste management and recycling have become critical issues for sustainable development. The increasing volume of waste generated by human activities calls for efficient and effective waste sorting systems to minimize environmental impact and maximize resource recovery. According to the World Bank, waste generation could nearly double by 2050, with generation per capita expected to increase by 40% in low- and middle-income countries[1]. Traditional waste sorting methods often rely on manual labor, leading to time-consuming processes and inconsistencies in categorization. To address these challenges, advancements in robotics and artificial intelligence (AI) offer promising solutions.

The "Delta Arm Waste Classifier" project aims to revolutionize waste management by developing an automated system capable of intelligently categorizing and sorting different types of waste. By leveraging robotics and deep learning algorithms, the project seeks to enhance recycling efficiency and contribute to a more sustainable future.

The Delta arm, renowned for its precision and speed, provides a versatile and agile robotic solution for waste manipulation and placement. Through the utilization of a vacuum suction pump, it enables accurate waste collection and transport, ensuring seamless handling of diverse waste items.

To identify and classify waste items, the project employs state-of-the-art deep learning algorithms such as YOLO (You Only Look Once). By training a custom dataset, the system can detect and recognize common waste categories, including paper, plastic, glass, metal, and biodegradable waste. This classification capability enables the robot to make informed decisions on the appropriate disposal location for each waste item.

The model processes the captured images from a camera for which we are using an ios device, calculates the waste item coordinates, and coordinates the actions of the Delta arm. The integration of software frameworks and libraries, such as Python, OpenCV, and TensorFlow, facilitates seamless communication and efficient coordination between the various components of the system.

By combining robotics, deep learning, and precise waste categorization, the project aims to create a reliable and efficient waste management solution. The ultimate goal is to significantly reduce manual effort, streamline recycling processes, and promote a sustainable approach to waste disposal.

1.2 Problem Statement

The efficient recycling of waste materials is crucial for environmental sustainability and resource conservation. However, in Nepal the current percentage of waste that is effectively being recycled remains significantly low. According to recent studies, only 4.1% of the total waste generated in Nepal undergoes proper recycling processes. The major causes of low recycling rate being improper separation of waste materials [2]. This indicates a significant gap in waste management practices, leading to environmental pollution and the squandering of valuable resources.
The existing waste management practices face significant challenges in achieving optimal recycling rates and addressing the growing environmental concerns associated with improper waste disposal. Traditional manual waste sorting methods suffer from inefficiencies, inconsistencies, and limited capacity to handle the increasing volume of waste generated worldwide.
One of the key problems lies in the inadequate recycling of waste materials due to inefficient sorting processes. Manual labor-intensive sorting methods often fail to achieve the required level of precision and accuracy, resulting in cross-contamination and reduced recycling efficiency.
The absence of automated technology hinders the ability to segregate paper, plastic, glass, metal, and biodegradable waste effectively. As a result, recyclable materials are often discarded in general waste streams, limiting their potential for reprocessing and reducing the overall recycling rates.

1.3 Objectives

The main objectives of the ”Delta Arm Waste Classifier” project is:
• To build and utilize a Delta arm to pick up waste items and place them in their respective place.

1.4 Feasibility Analysis

The feasibility of our project can be evaluated in terms of technical, economical and operational feasibility.

1.4.1 Technical Feasibility

The utilization of well-established technologies such as robotics, deep learning algorithms, and image processing techniques has been extensively researched and implemented in waste management applications. For instance, research study conducted by Diya et al. (2018) showcase the successful integration of robotics in waste sorting systems, highlighting the technical feasibility of such solutions [3]. Additionally, the availability of open-source libraries and frameworks simplifies the implementation and integration of the required software components.

1.4.2 Economic Feasibility

While there may be initial investment costs associated with hardware components like the stepper motors, suction pump, control unit, the long-term cost benefits are significant. By improving recycling efficiency, reducing manual labor requirements, and increasing resource recovery rates, the project can lead to substantial cost savings in waste disposal.

1.4.3 Operational Feasibility

One of the key advantages of the project lies in its potential to replace traditional manual waste separation processes in companies. The operational feasibility of the project is evident in its ability to streamline and automate waste classification and sorting operations. By employing robotics and AI-based algorithms, the system can accurately detect and classify different types of waste items, eliminating the need for manual labor-intensive sorting.

1.5 Scope of Project

The scope of the ”Delta Arm Waste Classifier” project encompasses the development and implementation of a robust system for the automated classification and sorting of waste materials. The project aims to create a sustainable solution that addresses the challenges of manual waste separation methods and enhances recycling practices. The key aspects of the project scope include:

1.5.1 Hardware Development

The project involves the design and assembly of the necessary hardware components, such as the Delta arm, camera, suction pump, and control unit. The hardware components will be integrated to form a cohesive system capable of efficiently picking up and sorting waste items.

1.5.2 Software Development

The project includes the development of software components that enable waste classification and sorting. This encompasses the implementation of deep learning algorithms, such as YOLO, for accurate waste detection and categorization. Custom software modules will be created to handle image processing, data analysis, and communication between the hardware components. Using an open-source python library called Streamlit an interactive and customizable web interface that allows operators to interact with the delta arm and view the waste detection results in real time is done.

1.5.3 Training of Deep Learning Models

The project involves the collection and preparation of a comprehensive dataset for training the deep learning models. This dataset will include samples of various waste materials, such as paper, plastic, metal, and biodegradable waste. The deep learning models will be trained using this dataset to accurately classify waste items.

1.5.4 Integration of Robotics and AI

The project aims to seamlessly integrate robotics and AI technologies to enable automated waste classification and sorting. The robotic arm, controlled by the system, will pick up waste items based on the classification results obtained from the AI algorithms. The system will provide precise coordinates to guide the robotic arm in placing the waste items into their respective containers.

1.5.5 Performance Evaluation and Optimization

The project includes a comprehensive evaluation of the system's performance in terms of waste classification accuracy, sorting speed, and overall efficiency. Iterative testing and optimization will be conducted to enhance the system's performance, fine-tune the deep learning models, and ensure reliable operation in real-world scenarios.

CHAPTER 2: LITERATURE REVIEW

The use of deep learning and delta arm for waste classification has been the subject of numerous studies in the literature. The review covers key aspects such as waste classification methods, robotic systems in waste management, and AI algorithms for object detection and classification.

Research by Zhang et al. explores the application of deep learning algorithms, including YOLO, in waste detection and classification. Their work demonstrates the effectiveness of such algorithms in accurately identifying different waste categories [4].

Research conducted by Satav et al. (2023) showcases the successful implementation of robotic systems for waste sorting, emphasizing their potential to improve recycling rates and reduce human labor requirements. The article highlighted the implantation of robotics and AI in sorting of materials such as glass, paper, plastic, metals, etc., from other waste [5].

Robotic arms, including delta arms, have proven to be highly versatile and efficient in manipulating objects. They offer advantages such as speed, precision, and scalability, making them suitable for waste handling and sorting tasks.

Another research conducted by Pierrot et al. (1990) highlights the use of delta arm in various applications. Their work demonstrates the efficiency of delta arm in comparison to other robotics arm. The main advantage of delta robots compared to traditional robotic arms is that delta robots have motors attached to the main body instead of the arms. This allows delta's arms to move at high speed which makes the delta robot ideal for lightweight pick and place operations [6].

Artificial intelligence, particularly deep learning algorithms, has revolutionized object detection and classification tasks. Convolutional neural networks (CNNs), in particular, have shown remarkable performance in accurately detecting and categorizing objects in images and videos. Aishwarya et al. had shown the successfully implementation of The You Only Look Once (YOLO) algorithm for waste classification, achieving high accuracy and efficiency in identifying and categorizing different waste materials [7].

. .

CHAPTER 3: METHODOLOGY

The methodology for implementing the project ”Delta Arm Waste Classifier” involves several key steps, including data collection, annotation, model training, hardware integration, and deployment. These steps are essential to ensure the successful development and implementation of an automated waste classification system.

3.1 Product Development Lifecycle

The Agile model for building a waste classifier with a delta arm is similar to Agile software development. The process begins with defining the project’s vision and goals and identifying the software and hardware requirement of the project. A product backlog is then created, outlining all of the necessary features and functionalities. The development process is divided into short iterations known as sprints. Continuous integration and testing are critical to ensuring the waste classifier’s stability. The iterative approach enables continuous refinement and expansion of the waste classifier in response to changing requirements and feedback. The iterative development process involves planning, designing, developing, testing, deploying, and reviewing the waste classifier during each iteration until it is ready for launch.

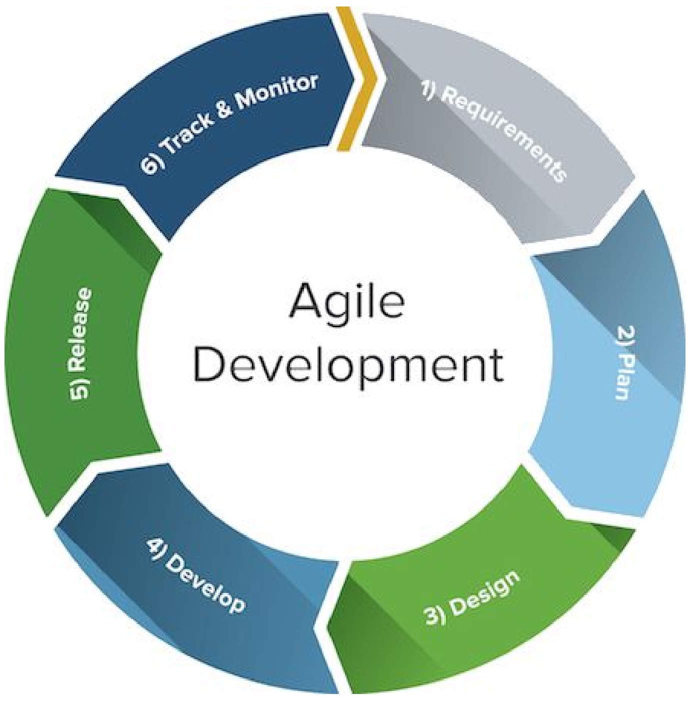


Figure 1: Agile Development Model

3.2 Data Collection and Annotation

To prepare datasets for training a waste classification model, the initial step involves collecting existing datasets from sources like Kaggle and creating custom datasets by capturing images of various waste materials. These datasets should be diverse, containing samples representing different waste categories, lighting conditions, and orientations to enable the model to generalize effectively. Subsequently, each image needs to be annotated with labels indicating the waste category it belongs to, such as plastic, paper, metal, biodegradable, etc. This annotation step is crucial for supervised learning, providing the model with labeled examples to learn from.

During the dataset preparation process, bounding boxes were created to accurately label objects in images. Initially, 2100 original images were collected from Kaggle for the waste classification project's dataset.The dataset also contains 250 custom datasets. Common augmentation techniques include rotating, flipping, scaling, adjusting brightness and contrast, adding noise, and cropping the images. The images were then annotated, yielding a total of 5872 annotated images. The average image size was 0.20 megapixels, ranging from 0.03 megapixels to 16.04 megapixels, with a median image ratio of 512x384, predominantly wide. The dataset was balanced across four classes, with 890 biodegradable, 633 plastic, 556 paper, and 477 metal instances. The maximum number of images containing a single object in a single image was 1879, however images with only 2-3 objects were uncommon.

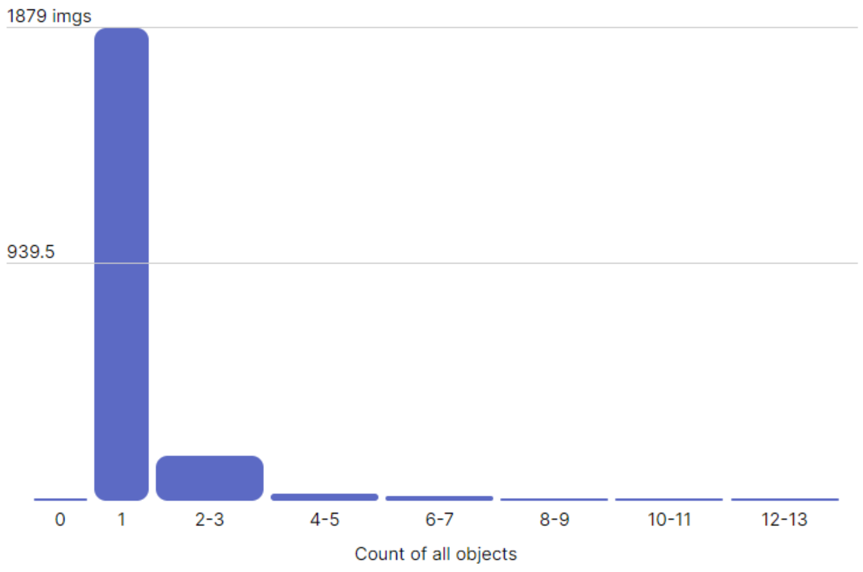


Figure 2: Histogram of Object Count by Image

Overall, the combination of annotation and augmentation ensures that the dataset is well-suited to training a waste classification model.

3.3 Model Training

To train the waste classification model, we'll use the Ultralytics yolov8(You Only Look Once) model, which is well-known for its real-time object detection capabilities. Specifically a version of the YOLO architecture, such as YOLOv8 will be implemented, that uses deep learning frameworks like TensorFlow and OpenCV for image processing. The model will be trained on our annotated datasets, iteratively adjusting its parameters to improve its accuracy in identifying and classifying various waste materials. Using YOLO's image processing efficiency and real-time object detection capabilities, this approach aims to develop a model that can accurately and quickly classify waste materials, thereby contributing to more effective waste management solutions. The iterative training process will fine-tune the model's ability to distinguish between various types of waste, eventually improving.

3.4 Hardware Integration

Simultaneously, the hardware components required has been developed and integrated. The Delta arm is responsible for picking up waste items based on the classification results and coordinates obtained from the trained model.

The Delta robot's kinematics describe the relationship between the movement of the stepper motors and the end effector's position and orientation. The Delta robot employs a parallel kinematic structure, with the end effector's position determined by the lengths of the robot's arms and the angles at which the arms are positioned. Each stepper motor controls the position of one of the arms, and by coordinating the movement of all three motors, the robot can move the end effector to any desired position in three-dimensional space.

In the project workflow, the camera captures images of waste items, and the YOLO algorithm detects these items and provides coordinates for their positions. These coordinates are initially in the camera's frame of reference and then converted into the Delta arm's coordinate system. This transformation involves accounting for the relative position and orientation of the camera about the Delta arm. The system uses inverse kinematics to calculate the precise stepper motor movements needed to position the Delta arm's end effector at the coordinates of the detected waste items. This procedure ensures that the Delta arm can accurately reach and collect each waste item.

Finally, the calculated stepper motor steps are fed into the Delta arm's stepper motor controllers. These controllers translate the steps into stepper motor movements for the Delta arm, allowing it to move precisely and pick up waste items as intended. The integration of image processing, coordinate transformation, and inverse kinematics allows for efficient and accurate waste sorting with the Delta arm.

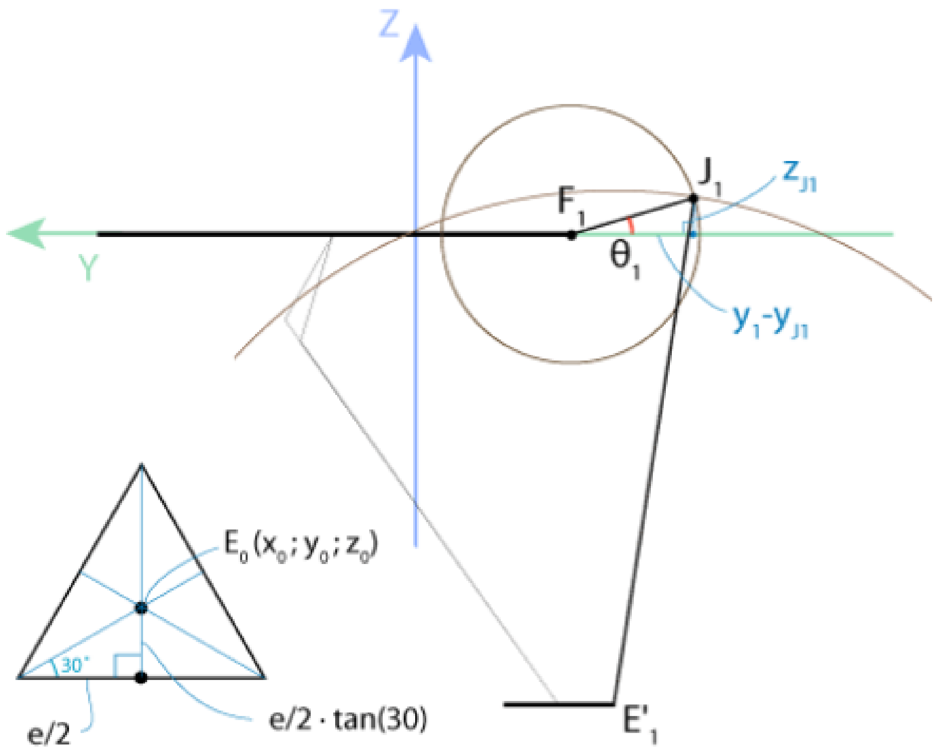


Figure 3: Inverse Delta Kinematics and its Parameters

The formula for one of the arm's angles using inverse delta-kinematics is given below:

$\theta_1 = \arctan\left(\frac{z_{J1}}{y_{F1} - y_{J1}}\right)$

Likewise, we can calculate the respective angles θ_2 and θ_3 for the corresponding arms.

3.5 System Assembly

During the integration phase, the project aims to combine the software and hardware components to create a unified system. This involves combining the trained waste classification model with the delta arm, allowing waste items to be processed in real-time and segregated. The system will work as follows: the camera will take images of waste items, which will then be processed by the YOLO model to identify and categorize them as paper, plastic, metal, and biodegradable waste. The classified waste items are given precise coordinates based on their position in the image. The control unit employs inverse kinematics to determine the stepper motor movements required to position the Delta arm's end effector at the coordinates of each waste item. The Delta arm then picks up the waste items and places them in the appropriate containers based on the classification results.

3.6 User Interface

In addition, a Streamlit web application is developed to allow manual control of the Delta arm and real-time object detection. This application will include a user-friendly interface for controlling the Delta arm's movements and monitoring the object detection process in real time. This integration and development phase will be followed by extensive testing and fine-tuning to ensure the system's accuracy, dependability, and efficiency in waste classification and sorting.

3.7 System Block Diagram

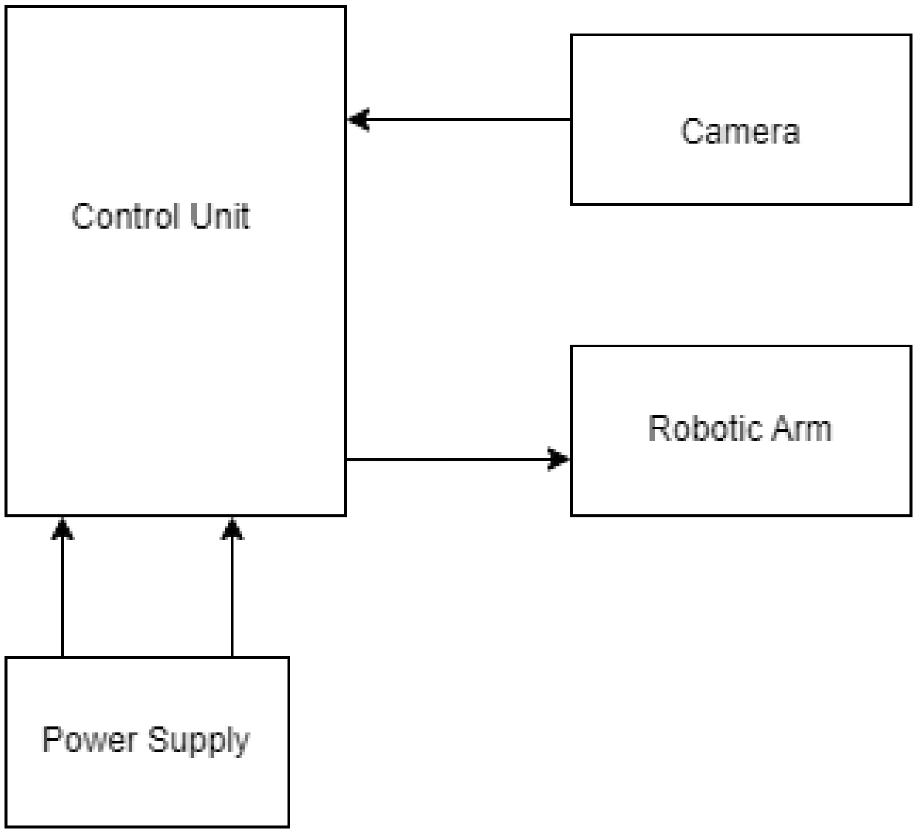


Figure 4: Block diagram

3.8 System Schematic Diagram

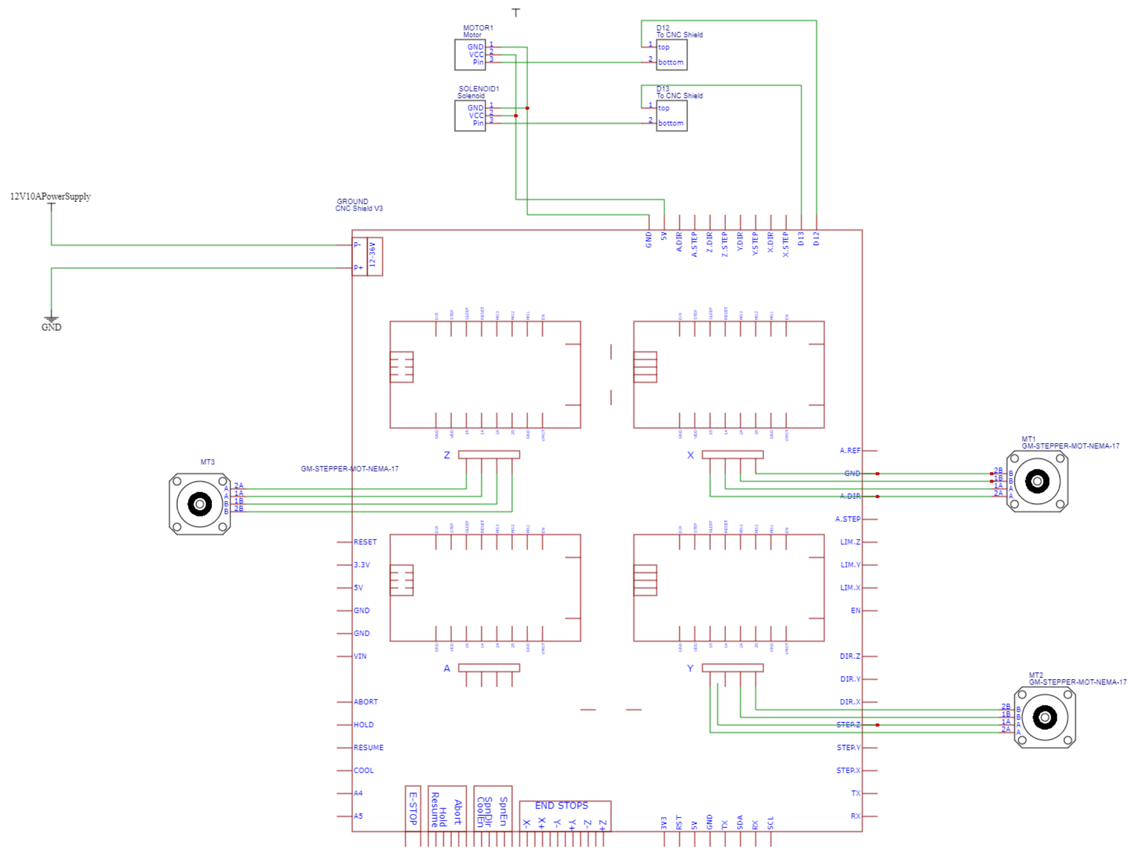


Figure 5: Schematic diagram

3.9 System Workflow

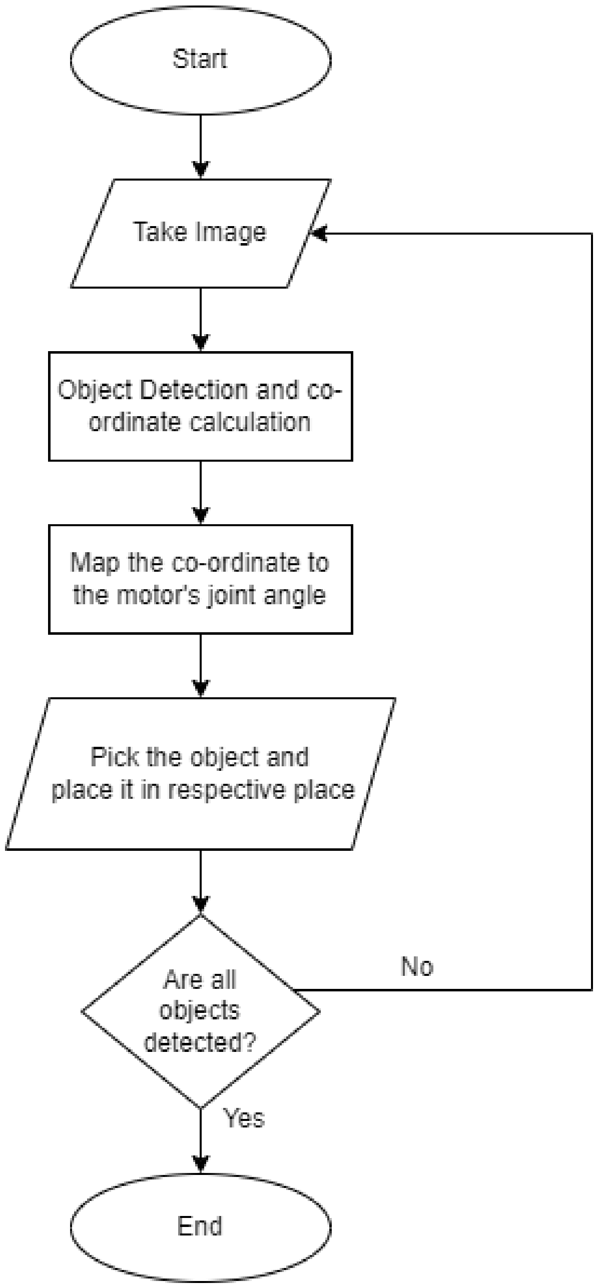

Start
Take Image
Object Detection and co-ordinate calculation
Map the co-ordinate to the motor's joint angle
Pick the object and place it in respective place
Are all objects detected?
No
Yes
End

Figure 6: System Workflow

CHAPTER 4: SYSTEM DESIGN

4.1 Requirement Specification

The functional and non-functional requirements are:

4.1.1 Functional requirements

- Image Acquisition: The system should be able to acquire images of waste items using the camera.
- Object Detection and Classification: The system should accurately detect and classify waste items into predefined categories such as paper, plastic, metal, and biodegradable waste.
- Robotic Arm Control: The system should control the movements of the Delta arm, enabling it to pick up waste items based on their coordinates and place them into their respective containers.
- Real-time Operation: The system should process images and perform waste classification in real-time to ensure efficient waste sorting.

4.1.2 Non-functional requirements

- Accuracy: The waste classification system should exhibit a high level of accuracy in identifying and classifying waste items, minimizing mis-classifications and errors.
- Speed and Efficiency: The system should perform waste classification and robotic arm movements quickly and efficiently to achieve optimal throughput and productivity.
- Reliability and Robustness: The system should be reliable and robust, capable of handling variations in waste items, lighting conditions, and environmental factors.
- Scalability: The system should be scalable, capable of handling a large volume of waste items for classification and sorting.
- Maintainability: The system should be designed in a modular and maintainable manner, allowing for future updates, enhancements, and repairs if needed.
- Safety: The system should adhere to safety standards to ensure the protection of users and efficient operation in the waste sorting environment.

4.2 Use Case Diagram

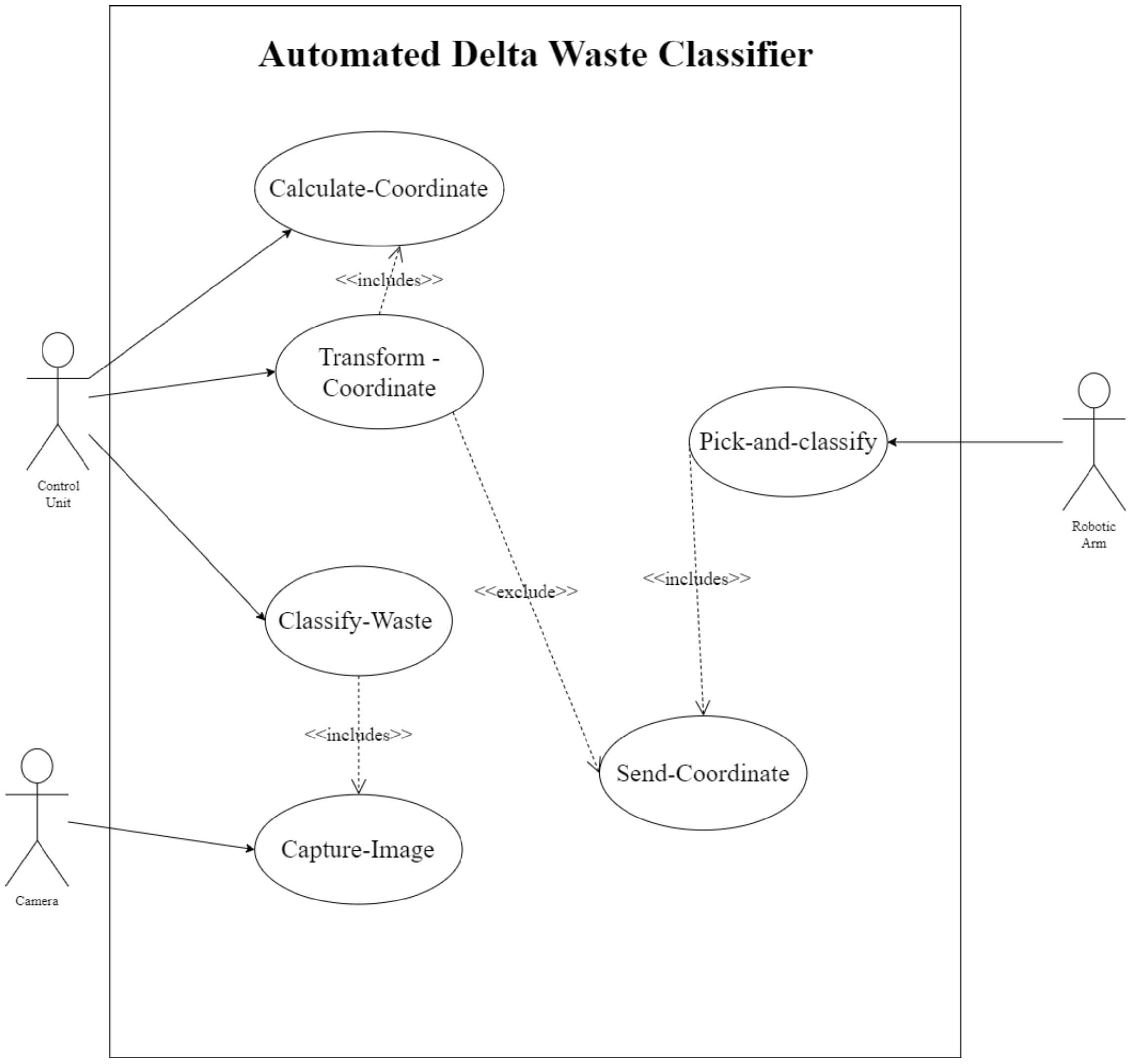


Figure 7: Use-case diagram

CHAPTER 5: TOOLS AND TECHONOLOGIES

5.1 Hardware Components

5.1.1 Arduino Uno

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller (MCU). In this project, the Arduino Uno functions as the main controller, managing the Delta Arm's movements, controlling the vacuum suction pump, and communicating with the camera for image acquisition. It executes the object detection and classification algorithms using YOLOv8. Additionally, the Arduino Uno interprets commands from the user interface, translating them into instructions for the Delta Arm, facilitating effective waste sorting and classification.

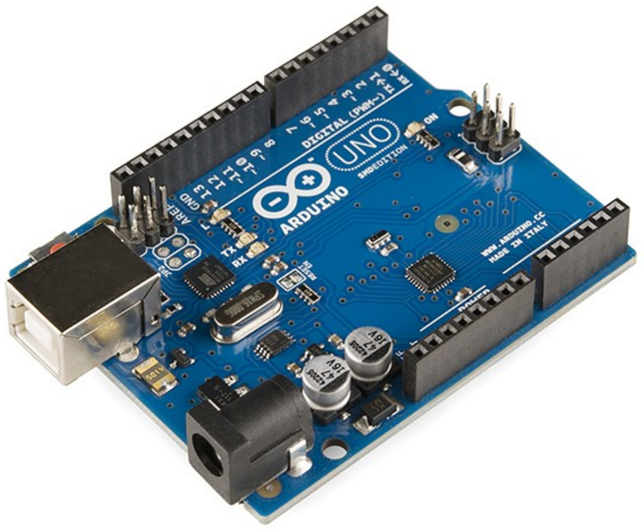


Figure 8: Arduino Uno

5.1.2 Stepper motor Nema 17

The NEMA-17 stepper motor is widely recognized for its compact size and precise control, making it a popular choice for various applications. Its 1.8-degree step angle per step ensures accurate positioning, making it suitable for applications that require precision. In the Delta arm project, NEMA-17 stepper motors are employed to drive the robotic arm's joints and movements.

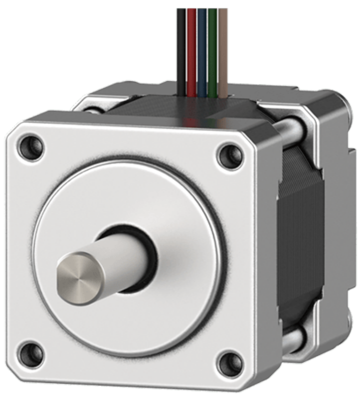


Figure 9: Stepper Motor

5.1.3 Stepper motor driver A4988

The A4988 stepper motor driver is widely used to control NEMA-17 and similar stepper motors, offering reliability and flexibility. It communicates with the Arduino Uno, receiving digital signals to regulate the motor's rotation and position. Through adjustment of the current in the motor windings, the A4988 enables precise and smooth motor movement.

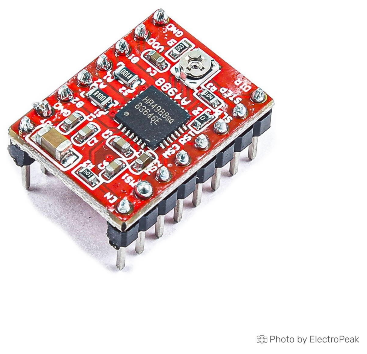


Figure 10: Stepper Motor Driver a4988

5.1.4 Power Supply-12V 10A

The 12V/10A power supply unit provides electrical power to the various components of the Delta arm project. It efficiently converts AC mains voltage to a 12V DC output, ensuring a reliable and steady power source for the Arduino Uno, CNC Shield V3, stepper motor drivers, and other peripherals. A stable power supply is essential for the robotic arm to operate smoothly and reliably.

Figure 11: Power Supply-12V 10A

5.1.5 CNC Shield V3

An expansion board called the CNC Shield V3 provides convenient connections for stepper motor drivers and other peripherals, and it easily plugs into an Arduino Uno. It makes controlling stepper motors for precise movement in the Delta arm project easier.

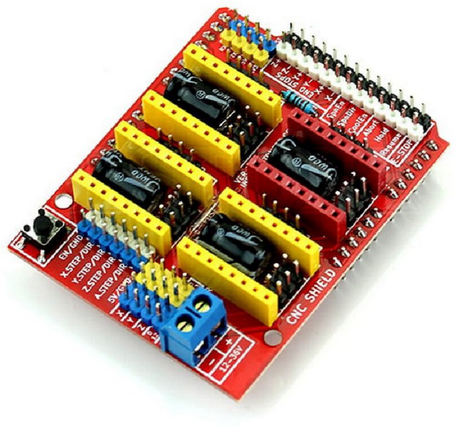


Figure 12: CNC shield V3

5.1.6 Air Suction Pump

An air suction pump is integrated into the Delta arm to create a vacuum suction mechanism. This suction power facilitates the firm grip and lifting of waste items. It consists of a DC motor to generate vacuum and a solenoid valve with a suction cup.

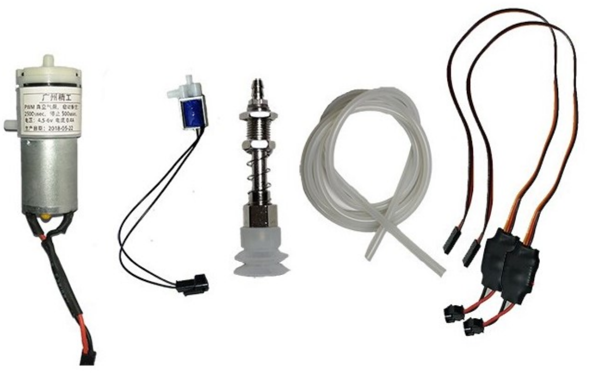


Figure 13: Vacuum Air Suction Pump

5.2 Software

5.2.1 Python

Python is a popular programming language used for developing various applications, including machine learning and computer vision. It serves as the primary programming language for implementing the waste classification and sorting system. Python provides a wide range of libraries and frameworks that facilitate image processing, deep learning, and communication with hardware components.

5.2.2 TensorFlow

TensorFlow is an open-source machine learning framework widely used for building and training deep learning models. It provides a comprehensive set of tools and functionalities for creating and deploying neural networks. In this project, TensorFlow is utilized for training and implementing the YOLO (You Only Look Once) model.

5.2.3 OpenCV

OpenCV is the library that provides various computer vision functions and algorithms. It is used to capture live video feeds and images needed for waste classification with the Delta arm. It supports camera access, image capture, and processing.

5.2.4 Pyserial

PySerial is a Python library that allows for serial communication. It allows us to interact with serial ports using Python code. In our project, PySerial is used to communicate with the Arduino board, which controls the Delta arm and other hardware components. PySerial sends commands from Python code to the Arduino board via a serial connection, instructing the Delta arm to take specific actions based on the commands received. This enables our Python code to control the movement of the Delta arm, activate and deactivate the vacuum, and coordinate the waste classification process in real time.

5.2.5 Ultralytics Yolov8

YOLO is a Convolutional Neural Network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. It is widely used for applications such as object recognition, autonomous driving, and surveillance systems. In this project, YOLO is employed for waste classification, allowing the system to identify and categorize different types of waste items, such as paper, plastic, metal, and biodegradable waste. Different models of YOLO v8 are available. We will be using yolov8m model.

5.2.6 Streamlit

Streamlit is a powerful tool for creating interactive data science web applications with minimal effort. Streamlit is used to create a user interface that allows operators to interact with the delta arm while viewing waste detection results in real time. This makes it easier for operators to control the delta arm and monitor the waste detection process.

5.2.7 Roboflow

Roboflow is a powerful platform that offers a variety of annotation tools designed specifically for computer vision projects, including efficient and accurate labeling capabilities. In our project, Roboflow's features most likely aided the annotation process for the YOLO algorithm, ensuring that the dataset was meticulously labeled, which is critical for training the waste classification model to correctly identify and categorize waste items.

CHAPTER 6: EPILOGUE

6.1 Result and Discussion

In this project, we developed and evaluated a waste classification system utilizing a delta arm robot controlled by an Arduino, featuring a vacuum suction pump, and accompanied by a camera to capture image. The Delta Arm executes precise handling by correctly picking up and placing waste items to their respective class. This setup seamlessly interacts with the YOLOv8 algorithm for immediate object detection and classification, guaranteeing precise identification of waste items. In essence, the hardware system boosts waste sorting efficiency and advocates for sustainable waste management practices.

6.1.1 Results on YOLO v8 model

The YOLOv8m model was used for training. The model was trained for 70 epochs with a batch size of 32. Following are the results obtained:

Box loss	0.30762
Classification loss	0.23631
Distribution focal loss	0.92997

Table 1: Training losses

Box loss	0.34798
Classification loss	0.27666
Distribution focal loss	0.97204

Table 2: Validation losses

Precision	0.94188
Recall	0.96622
mAP@50	0.98252
mAP@50-95	0.92723

Table 3: Metrics

The project achieved high precision (0.94118) and recall (0.96622), indicating accurate and comprehensive detection of waste items. The mAP@50 score of 0.98252 shows strong performance in detecting objects with moderate overlap. However, the slightly lower mAP@50-90 score of 0.92723 suggests difficulties in detecting objects with higher overlap. Overall, the model effectively detects and categorizes waste items, with the potential for improvement in handling objects with greater overlap.

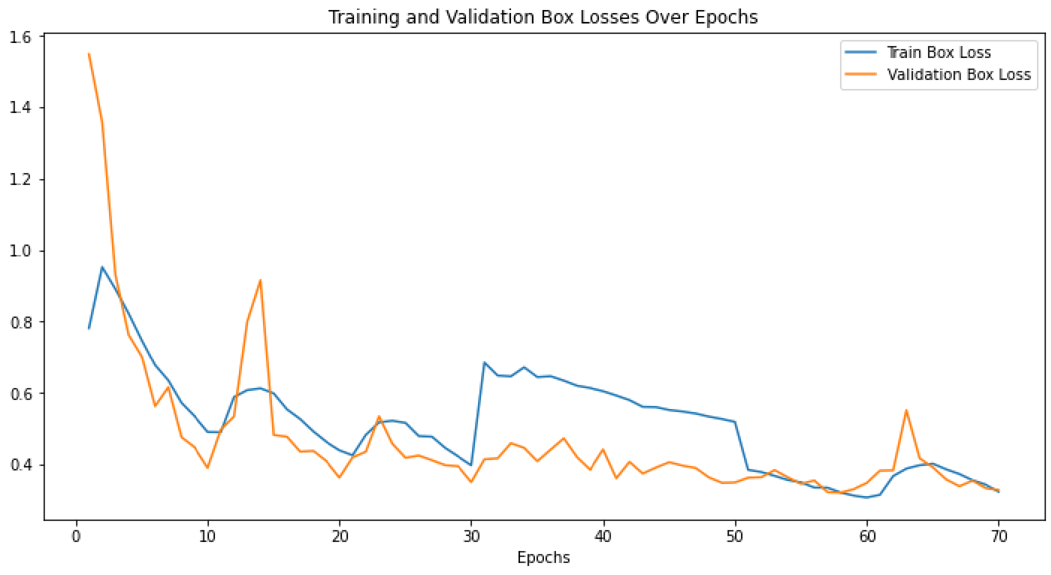


Figure 14: Box loss over epochs

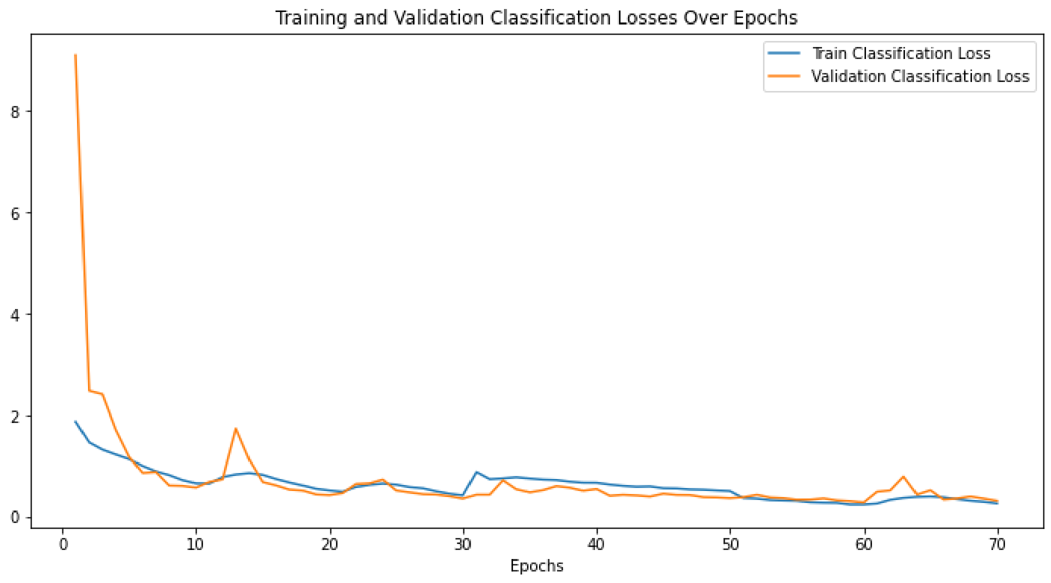


Figure 15: Classification loss over epochs

The spiking behaviour observed in our model training graph is primarily due to the use of small epochs during the training process. Using large epochs results in too much constraint in our computers and may end up crashing.

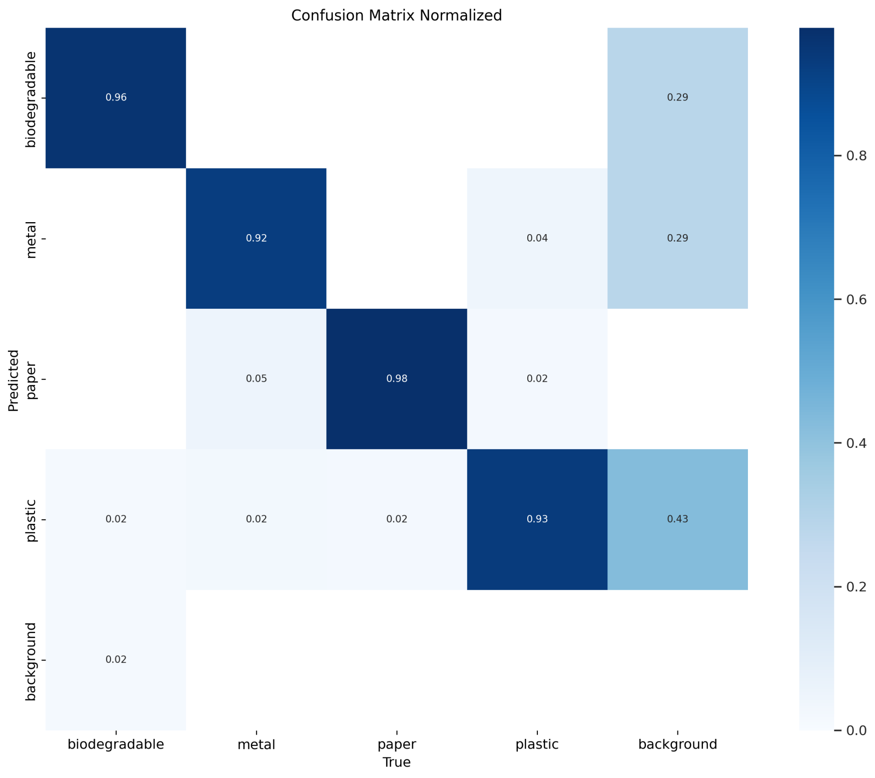


Figure 16: Normalized confusion matrix

6.2 Conclusion

In conclusion, the delta arm waste classifier presents a promising solution for enhancing waste sorting and recycling efforts. Its high accuracy, rapid processing speed, and adaptability make it a valuable tool for improving waste management practices and advancing sustainability goals. However, further research and development are necessary to address challenges and optimize the system for practical implementation in various industrial and municipal waste management settings.

With its precise sorting capabilities, the delta arm waste classifier streamlines operations, reduces waste contamination, and promotes recycling initiatives. Yet, to fully realize its potential, ongoing efforts are required to refine the system's performance and address technical complexities. Through continuous refinement and innovation, the delta arm waste classifier holds the promise of becoming a cornerstone in the quest for sustainable waste management solutions, paving the way for a greener future.

6.3 Challenges

During the completion period of our project, we faced several challenges that needed to be tackled for succesfully implementing the system.

1. One significant challenge faced in our project is the issue of torque inadequacy in the stepper motors used for the delta arm waste classifier.
2. Another challenge encountered in our project is the considerable time consumption associated with 3D printing all the parts for the arm.
3. The inability to purchase better equipment, such as Raspberry Pi, due to insufficient and limited budget.

6.4 Future Enhancements

1. Enhanced Sensor Technologies: Integrating advanced sensor technologies such as 3D scanning can improve the system's ability to detect and classify waste materials with higher accuracy and efficiency. These sensors can provide more detailed information about the composition, shape, and texture of waste items, enabling better decision-making during the sorting process.
2. Multi-stage Sorting: Introducing multi-stage sorting capabilities can enhance the system's versatility and efficiency. By incorporating additional sorting modules or robotic arms specialized for specific types of waste materials, the system can achieve finer classification and separation, leading to higher recycling rates and reduced waste contamination.
3. Integration with Waste Management Infrastructure: Enhancing the integration of the delta arm waste classifier with existing waste management infrastructure, such as recycling facilities , will streamline the overall waste processing workflow. This includes seamless communication and data exchange between the sorting system and downstream processes, optimizing resource allocation and waste diversion strategies.
4. Modular Design for Scalability: Designing the system with a modular architecture will facilitate scalability and customization according to specific application requirements. Modular components such as conveyor belts, sorting modules, and robotic arms can be easily added or modified to accommodate varying waste volumes, types, and processing needs in different settings.

5. Energy Efficiency and Sustainability: Implementing energy-efficient components and sustainable materials in the system's design will reduce its environmental footprint and operating costs. This includes optimizing power consumption, utilizing renewable energy sources, and incorporating recyclable materials in the construction of the sorting infrastructure.

6. Remote Monitoring and Maintenance: Implementing remote monitoring and predictive maintenance capabilities will enable proactive maintenance and troubleshooting of the system. By continuously monitoring performance metrics and detecting potential issues in real-time, operators can minimize downtime and maximize the system's operational efficiency.

7. Collaborative Robotics and Human-Robot Interaction: Exploring collaborative robotics solutions and enhancing human-robot interaction capabilities can improve the overall safety and usability of the system. This includes implementing safety features such as collision detection and emergency stop mechanisms, as well as integrating intuitive user interfaces for intuitive control and monitoring of the sorting process.

REFERENCES

[1] "Re-assessing global municipal solid waste generation," 2022.

[2] A. Khatoon, "Waste management—a case study in nepal," *Solid Waste Policies and Strategies: Issues, Challenges and Case Studies*, pp. 185–196, 2020.

[3] S. Z. Diya, R. A. Proma, M. N. Islam, T. T. Anannya, A. Al Mamun, R. Arefeen, S. Al Mamun, I. I. Rahman, and M. F. Rabbi, "Developing an intelligent waste sorting system with robotic arm: A step towards green environment," in *2018 International Conference on Innovation in Engineering and Technology (ICIET)*. IEEE, 2018, pp. 1–6.

[4] Q. Zhang, X. Zhang, X. Mu, Z. Wang, R. Tian, X. Wang, and X. Liu, "Recyclable waste image recognition based on deep learning," *Resources, Conservation and Recycling*, vol. 171, p. 105636, 2021.

[5] A. G. Satav, S. Kubade, C. Amrutkar, G. Arya, and A. Pawar, "A state-of-the-art review on robotics in waste sorting: scope and challenges," *International Journal on Interactive Design and Manufacturing (IJIDeM)*, pp. 1–18, 2023.

[6] F. Pierrot, C. Reynaud, and A. Fournier, "Delta: a simple and efficient parallel robot," *Robotica*, vol. 8, no. 2, pp. 105–109, 1990.

[7] A. Aishwarya, P. Wadhwa, O. Owais, and V. Vashisht, "A waste management technique to detect and separate non-biodegradable waste using machine learning and yolo algorithm," in *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, 2021, pp. 443–447.

Appendix A

Listing 1: Web Application

```
import streamlit as st
import cv2
from utils import *
import warnings
warnings.filterwarnings('ignore')


def main():
    st.title(”Delta■Arm■Waste■Classifier”)

    # Sidebar panel for coordinates and actions
    st.sidebar.title(”Control■Panel”)
    x_coord = st.sidebar.number_input(”X■Coordinate”, value
        =0)
    y_coord = st.sidebar.number_input(”Y■Coordinate”, value
        =0)
    z_coord = st.sidebar.number_input(”Z■Coordinate”, value
        =-290)

    if st.sidebar.button(”Send■Coordinates”):
        send_coordinates([x_coord, y_coord, z_coord], ”none”)

    if st.sidebar.button(”Return■to■Home■Position”):
        send_coordinates([0, 0, -290,], ”none”)

    # Confidence level input
    confidence_level = st.slider(”Confidence■Level”,
        min_value=0.0, max_value=1.0, value=0.5, step=0.01)
    # Live camera feed
    st.subheader(”Live■Feed”)

    # use 1 to use iphone camera else 0 to use laptop camera
    capture = cv2.VideoCapture(1)

    frame_placeholder = st.empty()
    capture_button_pressed = st.button('Capture■Image■and■
        Predict')

    while capture.isOpened():
        ret, frame = capture.read()

        if not ret:
            st.write("Video■capture■has■ended.")
            break

        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        frame_with_polygon = draw_polygon(frame)
        frame_placeholder.image(frame_with_polygon, channels=
            'RGB')

        if capture_button_pressed:
            capture_button_pressed, result = capture_image(
                frame, confidence_level)
            print(result)
            send_coordinates([0, 0, -380,] ,result)


    capture.release()



if __name__ == "__main__":
    main()
```

Listing 2: Utilities

```
import streamlit as st
import cv2
import os
import numpy as np
import shutil
from ultralytics import YOLO
import json
import serial

MODEL = YOLO("weights\last.pt")
ser = serial.Serial("COM6", 115200)  # Change 'COM3' to the
   appropriate port

def draw_polygon(frame):

    """Draw a polygon zone on the live video frame"""
    # Get frame dimensions
    height, width, _ = frame.shape

    # Define vertices of the polygon
    vertices = np.array([[(width // 3, height // 3),
                          (2 * width // 3, height // 3),
                          (2 * width // 3, 2 * height // 3),
                          (width // 3, 2 * height // 3)]],
                             dtype=np.int32)

    # Draw the polygon on the frame
    frame_with_polygon = cv2.polylines(frame.copy(), [
       vertices], isClosed=True, color=(0, 255, 0), thickness
       =2)

    return frame_with_polygon

def make_prediction(img_path, confidence):
    """Make prediction on the given image and return
       prediction results"""
    if os.path.isdir("app/predict"):
        shutil.rmtree("app/predict")

    # names: {0: 'biodegradable', 1: 'metal', 2: 'paper', 3:
       'plastic'}
    results = MODEL.predict(source=img_path, save = True,
       conf = confidence,save_txt = True, project = 'app/' )
    detected_objects = []
    try:
        for result in results:
            detected_label = int(result.boxes.cls[0].item())
            detected_objects.append(detected_label)
        return detected_objects[0]
    except:
        st.write("Object not detected. Try decreasing the 
           confidence level")
        return 'none'

def send_coordinates(coordinates, waste_label):
    """Take the list of cordinates in list form and send to
       serial port"""
    st.write("Sending coordinates:", coordinates)
    data = f"{coordinates[0]},{coordinates[1]},{coordinates
       [2]},{waste_label}"
    data=data+"\r"
    # Send JSON data to Arduino via serial port
    send_to_arduino(data)


def send_to_arduino(data):
    """Send JSON data to Arduino via serial port"""
    try:
        ser.write(data.encode())
    except Exception as e:
        st.write(f"Error: {str(e)}")
```

```
def capture_image(frame, confidence_level):
    """Capture the image from live feed and make prediction
       """
    img_name = "capture_image.jpg"
    path = 'app'
    img_path = os.path.join(path , img_name)
    frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    cv2.imwrite(img_path, frame)
    results = make_prediction(img_path, confidence_level)
    # show the predicted image with bounding box
    st.image("app/predict/capture_image.jpg", channels="RGB",
        caption="Captured■Image", use_column_width=True)
    return False, results



def main():
    pass

if __name__ == "__main__":
    main()
```

Listing 3: Model Training

```
import locale
locale.getpreferredencoding = lambda: "UTF-8"

import os
HOME = os.getcwd()

# Installing ultralytics for yoloV8
!pip install ultralytics

from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()
```

```
from ultralytics import YOLO
from IPython.display import display, Image

# Using google drive
from google.colab import drive
drive._mount('/content/drive')

# Initializing the model
model = YOLO('yolov8m.pt')

# Model training
results = model.train(data="/content/drive/MyDrive/Colab■
    Notebooks/custom■waste■dataset/data.yaml", epochs=70,
    pretrained=True, visualize=True,project = "/content/drive/
    MyDrive/Colab■Notebooks/waste■model,■patience■=■10")
```

Listing 4: Delta-Arm Hardware Code

```
#include <ArduinoJson.h>
#include <DeltaKinematics.h>
#include <FlexyStepper.h>
#include <Servo.h>

#define STEPS 400
#define SPEED 800
#define ACCELERATE 400

DeltaKinematics DK(91, 310, 40, 100);
FlexyStepper stepper1;
FlexyStepper stepper2;
FlexyStepper stepper3;

Servo motor;
Servo solenoid;

String cmd;
float cor[3];
String waste_class;
char *token;

void suction_on() {
  motor.write(180);
  solenoid.write(0);
}
void suction_off() {
  motor.write(0);
  solenoid.write(180);
}

void setup() {
  Serial.begin(115200);
  motor.attach(13);
  solenoid.attach(12);
  stepper1.connectToPins(2, 5);
  stepper2.connectToPins(3, 6);
  stepper3.connectToPins(4, 7);
  stepper1.setSpeedInStepsPerSecond(SPEED);
  stepper1.setAccelerationInStepsPerSecondPerSecond(
    ACCELERATE);
  stepper2.setSpeedInStepsPerSecond(SPEED);
  stepper2.setAccelerationInStepsPerSecondPerSecond(
    ACCELERATE);
  stepper3.setSpeedInStepsPerSecond(SPEED);
  stepper3.setAccelerationInStepsPerSecondPerSecond(
    ACCELERATE);
}

void loop() {
  if (Serial.available() > 0) {
    cmd = Serial.readStringUntil('\r');
    token = strtok(cmd.c_str(), ",");
    for (int i = 0; i < 3; i++) {
      cor[i] = atof(token);
      token = strtok(NULL, ",");
    }
    waste_class = token;
    DK.x = cor[0];
    DK.y = cor[1];
    DK.z = cor[2];
    DK.inverse();
    stepper_movement(DK.a, DK.b, DK.c);
    delay(1000);
    if (waste_class == "glass") {
      suction_on();
      delay(1000);
      DK.x = 0;
      DK.y = 140;
      DK.z = -290;
      DK.inverse();
      stepper_movement(DK.a, DK.b, DK.c);
      delay(1000);
      suction_off();
      delay(500);
      DK.x = 0;
      DK.y = 0;
      DK.z = -290;
      DK.inverse();
      stepper_movement(DK.a, DK.b, DK.c);
      delay(1000);
    } else if (waste_class == "plastic") {
      suction_on();
      delay(1000);
      DK.x = 0;
      DK.y = -140;
      DK.z = -290;
      DK.inverse();
      stepper_movement(DK.a, DK.b, DK.c);
      delay(1000);
      suction_off();
      delay(500);
      DK.x = 0;
      DK.y = 0;
      DK.z = -290;
      DK.inverse();
      stepper_movement(DK.a, DK.b, DK.c);
      delay(1000);
    } else if (waste_class == "metal") {
      suction_on();
      delay(1000);
      DK.x = -140;
      DK.y = 0;
      DK.z = -290;
      DK.inverse();
      stepper_movement(DK.a, DK.b, DK.c);
      delay(1000);
      suction_off();
      delay(500);
      DK.x = 0;
      DK.y = 0;
      DK.z = -290;
      DK.inverse();
```

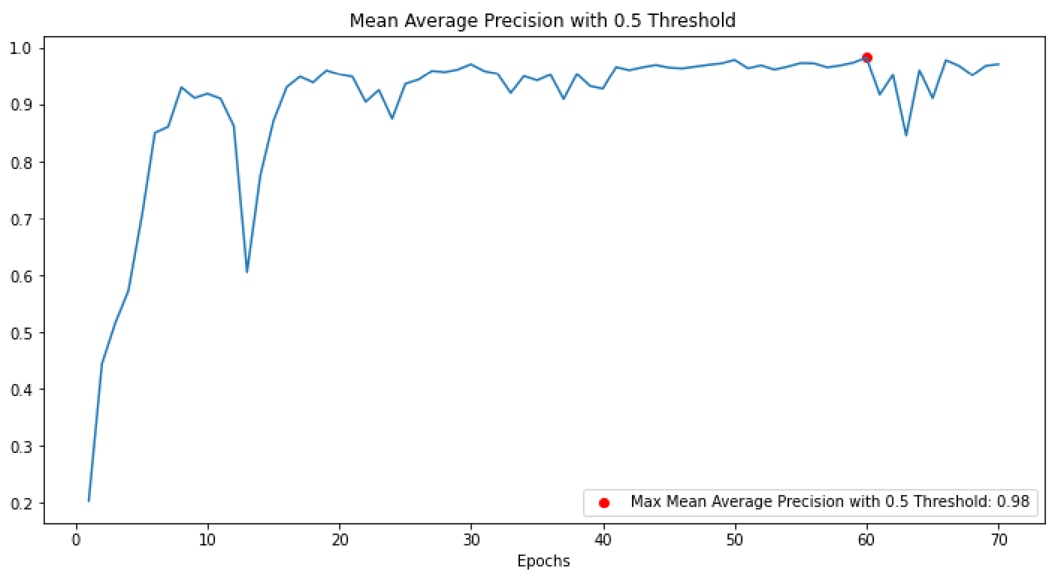
```
        stepper_movement(DK.a, DK.b, DK.c);
        delay(1000);
      } else if (waste_class == "bio") {
        suction_on();
        delay(1000);
        DK.x = 140;
        DK.y = 0;
        DK.z = -290;
        DK.inverse();
        stepper_movement(DK.a, DK.b, DK.c);
        delay(1000);
        suction_off();
        delay(500);
        DK.x = 0;
        DK.y = 0;
        DK.z = -290;
        DK.inverse();
        stepper_movement(DK.a, DK.b, DK.c);
        delay(1000);
      }
    }
}

void stepper_movement(float theta1, float theta2, float
   theta3) {
  int steps1 = (theta1 * STEPS) / 360;
  int steps2 = (theta2 * STEPS) / 360;
  int steps3 = (theta3 * STEPS) / 360;
  stepper1.setTargetPositionInSteps(steps1);
  stepper2.setTargetPositionInSteps(steps2);
  stepper3.setTargetPositionInSteps(steps3);
  while ((!stepper1.motionComplete()) || (!stepper2.
     motionComplete()) || (!stepper3.motionComplete())) {
    stepper1.processMovement();
    stepper2.processMovement();
    stepper3.processMovement();
  }
}
```

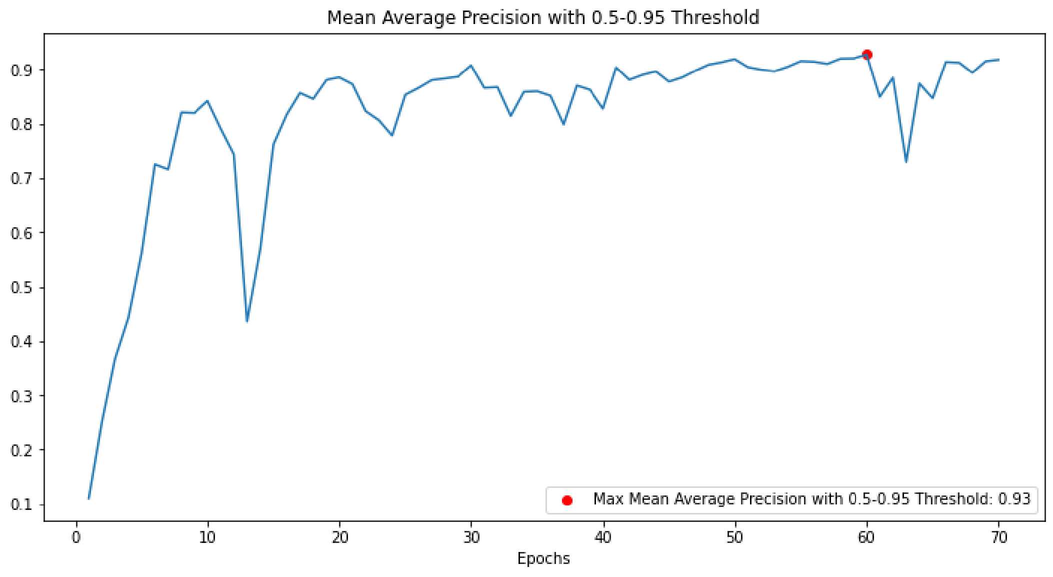
Listing 5: Delta-Kinematics C++ Implementation

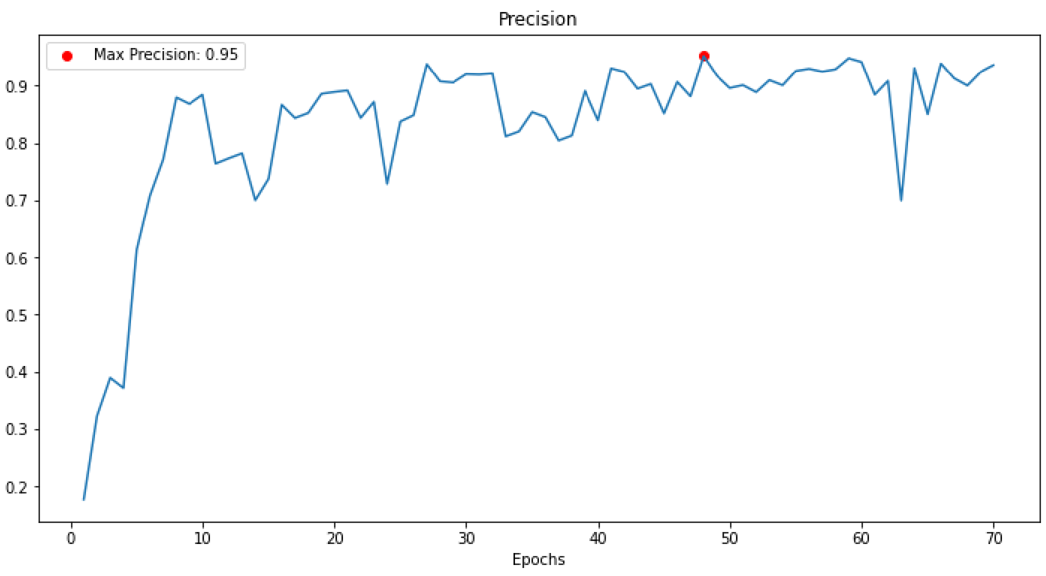
```
#ifndef DeltaKinematics_h
#define DeltaKinematics_h

#define sqrt3 1.7320508075688772935274463415059
#define pi  3.141592653589793238462643383279 5      // PI
#define sin120   sqrt3/2.0
#define cos120   -0.5
#define tan60   sqrt3
#define sin30   0.5
#define tan30   1.0/sqrt3

#define non_existing_povar_error -2
#define no_error 1

#if ARDUINO >= 100
  #include "Arduino.h"
#else
  #include "WProgram.h"
#endif

#include <math.h>


DeltaKinematics::DeltaKinematics(double _ArmLength,double
   _RodLength,double _BassTri,double _PlatformTri)
{
  PlatformTri = _PlatformTri;             // Platform
  BassTri = _BassTri;             // End Effector
  RodLength = _RodLength;
  ArmLength = _ArmLength;
  x = y = z = a = b = c = 0.0;
}

// forward kinematics: (thetaA, thetaB, thetaC) -> (x0, y0,
   z0)
int DeltaKinematics::forward()
{
  return forward(a, b, c);
}

int DeltaKinematics::forward(double thetaA, double thetaB,
   double thetaC)
{
  x=0.0;
  y=0.0;
  z=0.0;

  double t = (PlatformTri-BassTri)*tan30/2.0;
  double dtr = pi/180.0;

  thetaA *= dtr;
  thetaB *= dtr;
  thetaC *= dtr;

  double y1 = -(t + ArmLength*cos(thetaA));
  double z1 = -ArmLength*sin(thetaA);

  double y2 = (t + ArmLength*cos(thetaB))*sin30;
  double x2 = y2*tan60;
  double z2 = -ArmLength*sin(thetaB);

  double y3 = (t + ArmLength*cos(thetaC))*sin30;
  double x3 = -y3*tan60;
  double z3 = -ArmLength*sin(thetaC);

  double dnm = (y2-y1)*x3-(y3-y1)*x2;

  double w1 = y1*y1 + z1*z1;
  double w2 = x2*x2 + y2*y2 + z2*z2;
  double w3 = x3*x3 + y3*y3 + z3*z3;

  // x = (a1*z + b1)/dnm
  double a1 = (z2-z1)*(y3-y1)-(z3-z1)*(y2-y1);
  double b1 = -((w2-w1)*(y3-y1)-(w3-w1)*(y2-y1))/2.0;
  // y = (a2*z + b2)/dnm;
  double a2 = -(z2-z1)*x3+(z3-z1)*x2;
  double b2 = ((w2-w1)*x3 - (w3-w1)*x2)/2.0;

  // a*z^2 + b*z + c = 0
  double aV = a1*a1 + a2*a2 + dnm*dnm;
  double bV = 2.0*(a1*b1 + a2*(b2-y1*dnm) - z1*dnm*dnm);
  double cV = (b2-y1*dnm)*(b2-y1*dnm) + b1*b1 + dnm*dnm*(z1*
     z1 - RodLength*RodLength);

  // discriminant
  double dV = bV*bV - 4.0*aV*cV;
  if (dV < 0.0)
  {
    return non_existing_povar_error; // non-existing povar.
       return error,x,y,z
  }

  z = -0.5*(bV+sqrt(dV))/aV;
  x = (a1*z + b1)/dnm;
  y = (a2*z + b2)/dnm;

  return no_error;
}


// inverse kinematics
// helper functions calculate angle thetaA (for YZ-pane)
int DeltaKinematics::delta_calcAngleYZ(double *Angle, double
   x0, double y0, double z0)
{
  double y1 = -0.5 * tan30 * PlatformTri;  // f/2 * tan(30
     deg)
      y0 -= 0.5 * tan30 * BassTri;  // shift center to edge

  // z = a + b*y
  double aV = (x0*x0 + y0*y0 + z0*z0 +ArmLength*ArmLength -
     RodLength*RodLength - y1*y1)/(2.0*z0);
  double bV = (y1-y0)/z0;

  // discriminant
  double dV = -(aV+bV*y1)*(aV+bV*y1)+ArmLength*(bV*bV*
     ArmLength+ArmLength);
  if (dV < 0)
  {
    return non_existing_povar_error; // non-existing povar.
       return error, theta
  }

  double yj = (y1 - aV*bV - sqrt(dV))/(bV*bV + 1); //
     choosing outer povar
  double zj = aV + bV*yj;
  *Angle = atan2(-zj,(y1 - yj)) * 180.0/pi;

  return no_error;  // return error, theta
}

// inverse kinematics: (x0, y0, z0) -> (thetaA, thetaB,
    thetaC)
int DeltaKinematics::inverse()
{
  return inverse(x, y, z);
}

int DeltaKinematics::inverse(double x0, double y0, double z0)
{
  a = 0;
  b = 0;
  c = 0;
  int error = delta_calcAngleYZ(&a, x0, y0, z0);
  if(error != no_error)
    return error;
  error = delta_calcAngleYZ(&b, x0*cos120 + y0*sin120, y0*
     cos120-x0*sin120, z0);
  if(error != no_error)
    return error;
  error = delta_calcAngleYZ(&c, x0*cos120 - y0*sin120, y0*
     cos120+x0*sin120, z0);

  return error;
}


class DeltaKinematics
{
    public:
        // SETUP
        DeltaKinematics(double _ArmLength,double _RodLength,
           double _BassTri,double _PlatformTri);

        int forward();
        int forward(double thetaA, double thetaB, double
           thetaC);
        int inverse();
        int inverse(double x0, double y0, double z0);

        double x;
        double y;
        double z;

        double a;
        double b;
        double c;

    private:

        int delta_calcAngleYZ(double *Angle, double x0,
           double y0, double z0);

        double ArmLength;
        double RodLength;
        double BassTri;
        double PlatformTri;
```

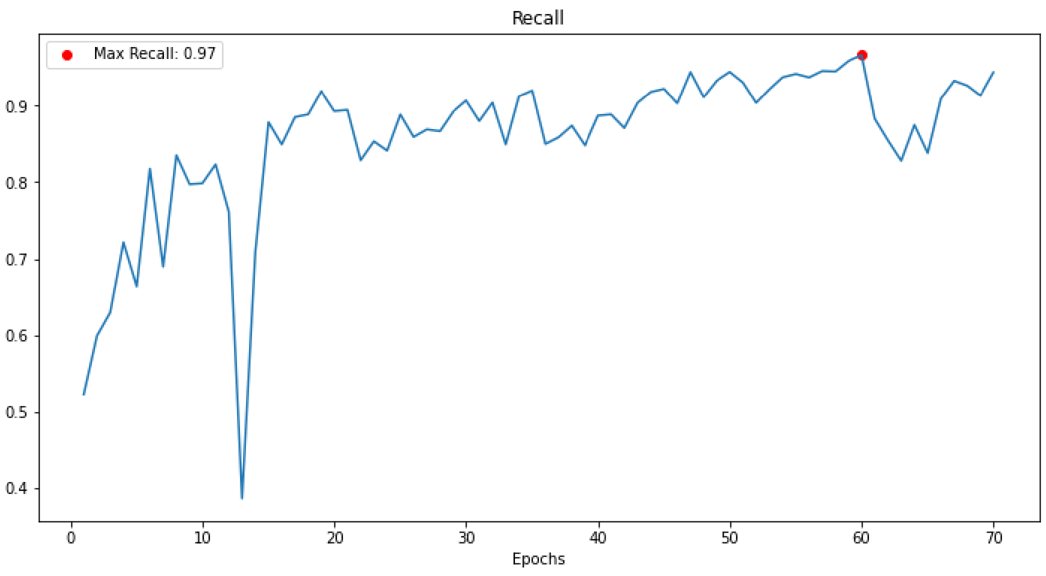
```
};

#endif
```

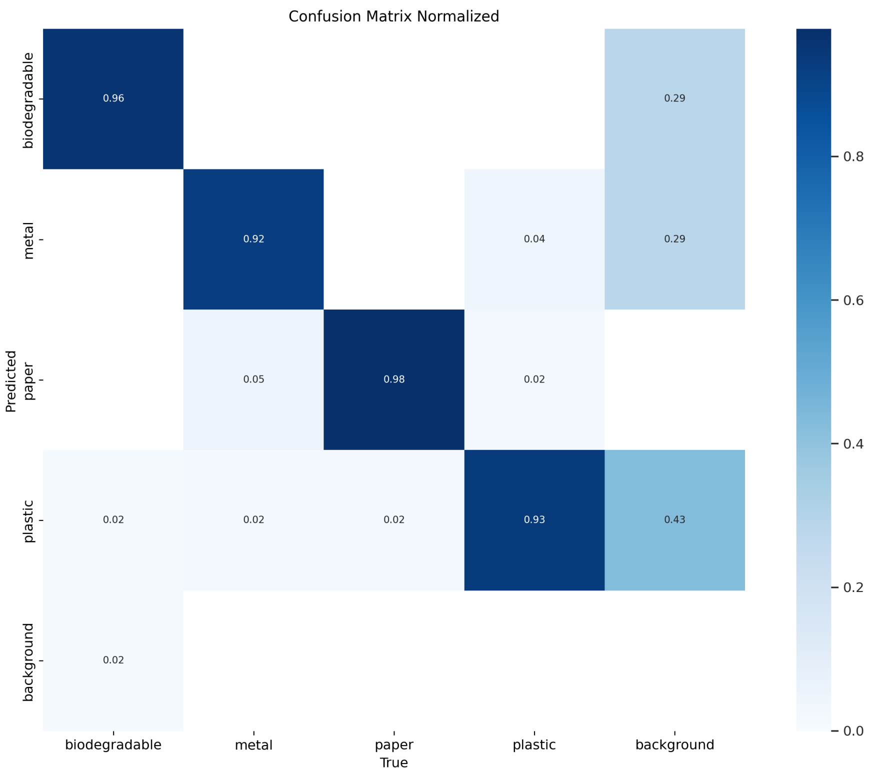
Appendix B

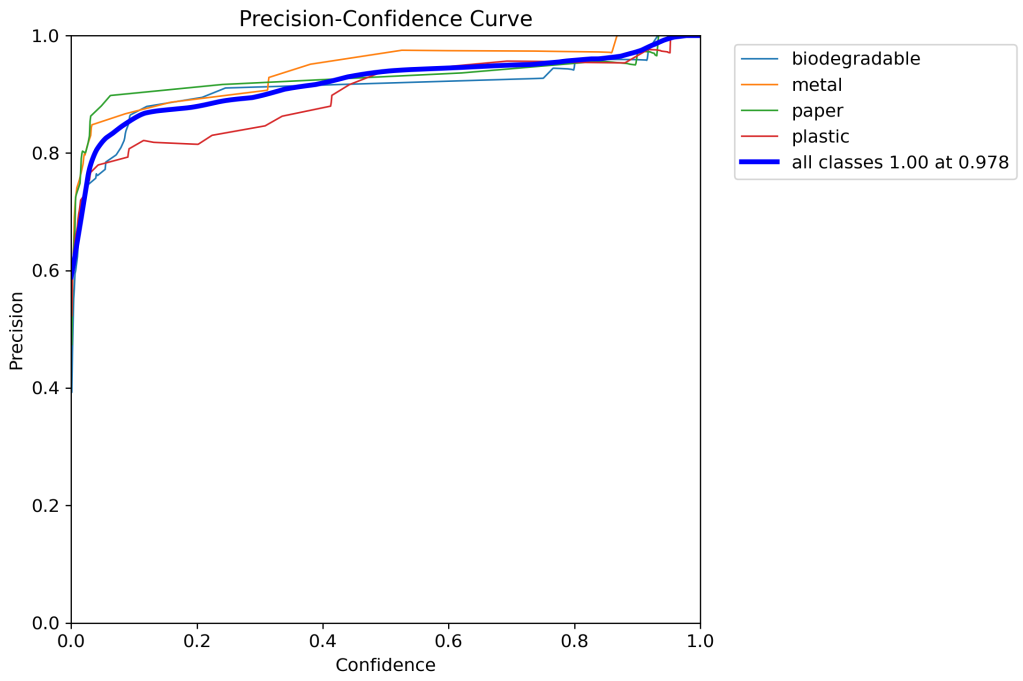

Mean Average Precision with 0.5 Threshold
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
● Max Mean Average Precision with 0.5 Threshold: 0.98
0
10
20
30
40
50
60
70
Epochs

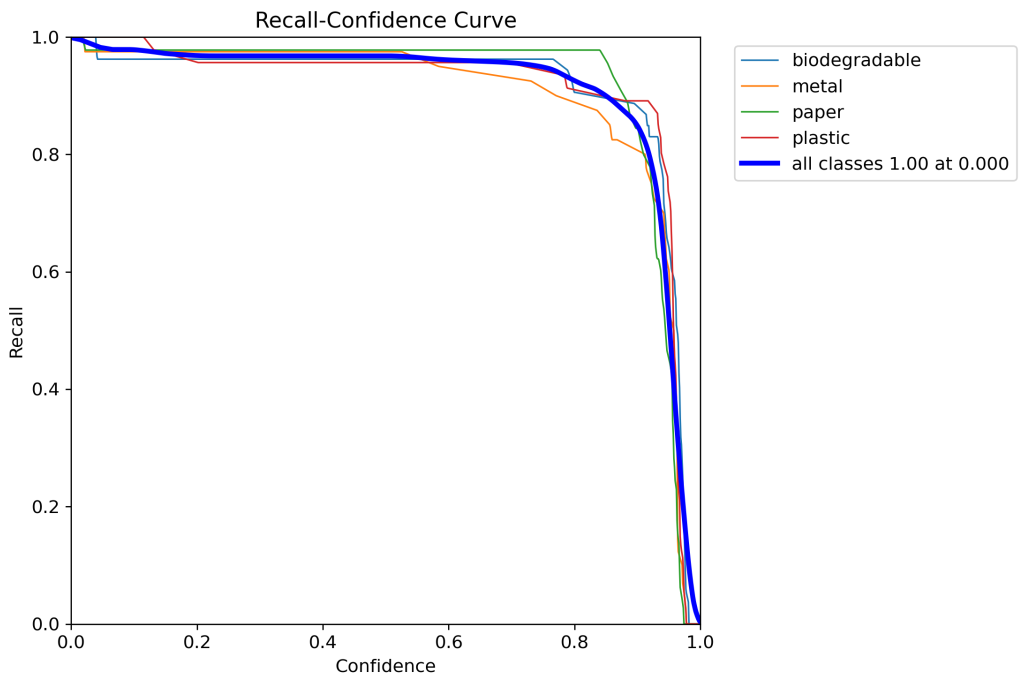

Mean Average Precision with 0.5-0.95 Threshold
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
● Max Mean Average Precision with 0.5-0.95 Threshold: 0.93
0
10
20
30
40
50
60
70
Epochs

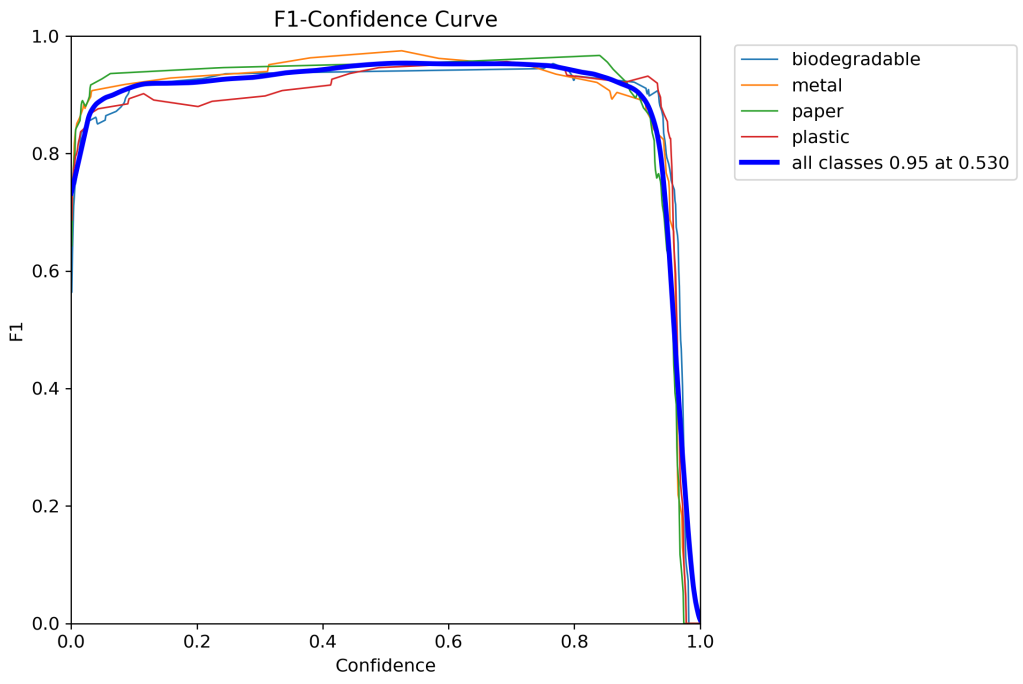

Precision
● Max Precision: 0.95
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0
10
20
30
40
50
60
70
Epochs

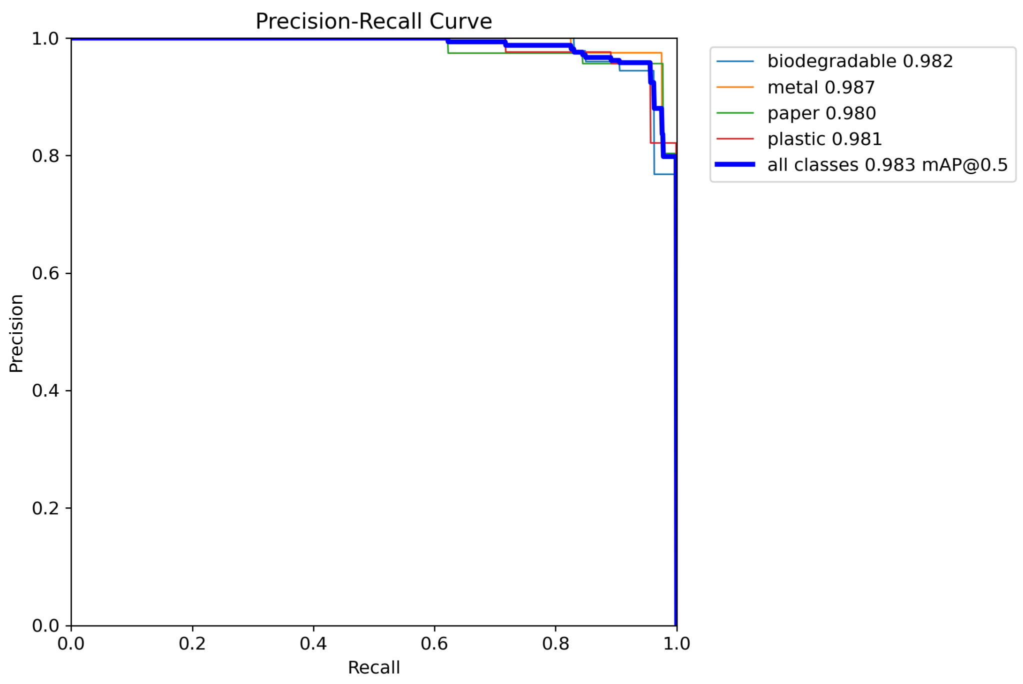

Recall
● Max Recall: 0.97
0.9
0.8
0.7
0.6
0.5
0.4
0
10
20
30
40
50
60
70
Epochs

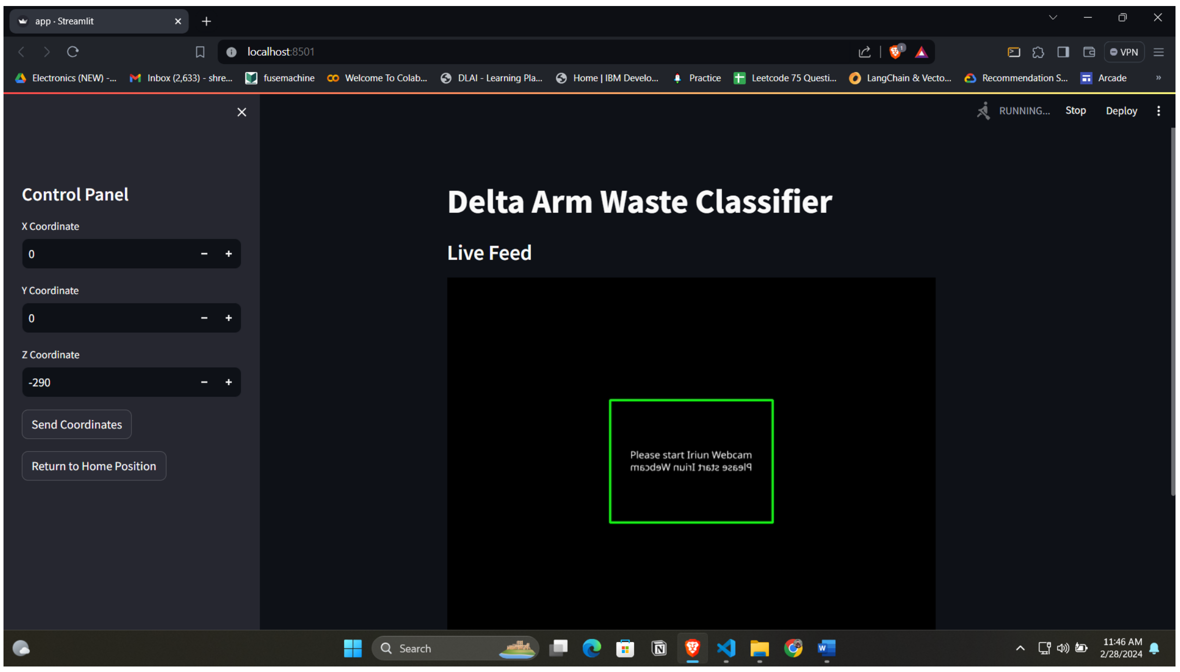

Confusion Matrix Normalized
Predicted
biodegradable
metal
paper
plastic
background
True
biodegradable
metal
paper
plastic
background
0.96
0.29
0.92
0.04
0.29
0.05
0.98
0.02
0.02
0.02
0.02
0.93
0.43
0.02
0.8
0.6
0.4
0.2
0.0


Precision-Confidence Curve
Precision
Confidence
biodegradable
metal
paper
plastic
all classes 1.00 at 0.978
0.0
0.2
0.4
0.6
0.8
1.0


Recall-Confidence Curve
Recall
Confidence
biodegradable
metal
paper
plastic
all classes 1.00 at 0.000


F1-Confidence Curve
F1
Confidence
biodegradable
metal
paper
plastic
all classes 0.95 at 0.530





Control Panel

X Coordinate: 0

Y Coordinate: 0

Z Coordinate: -290

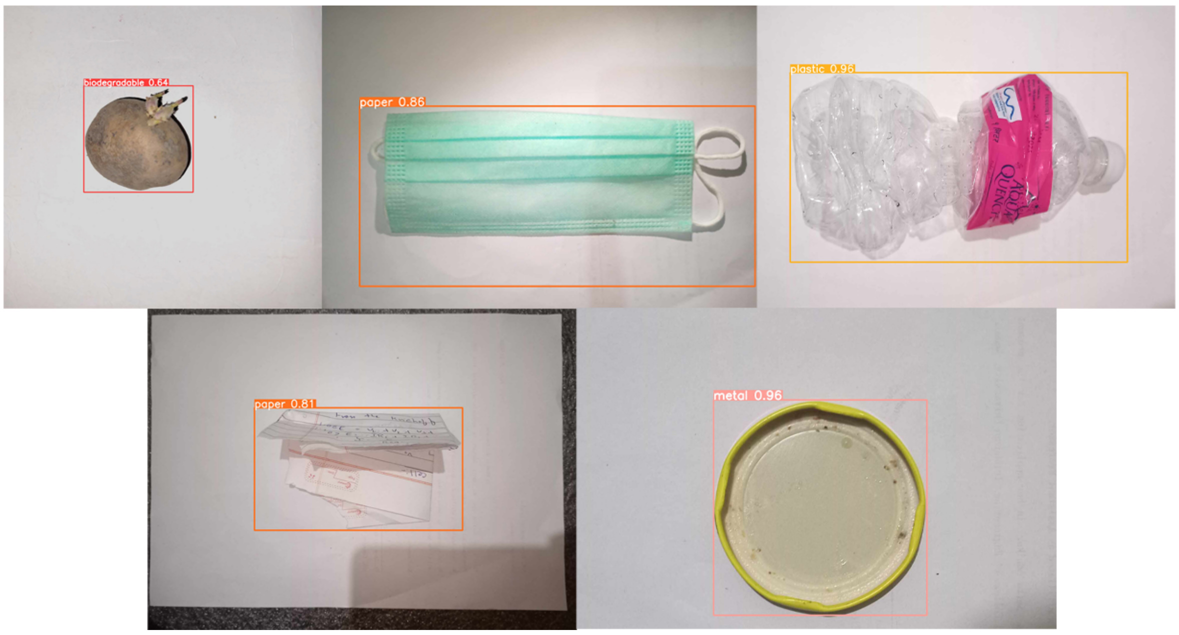
Send Coordinates

Return to Home Position

Delta Arm Waste Classifier

Live Feed

Please start Iriun Webcam


biodegradable 0.64
paper 0.86
plastic 0.96
paper 0.81
metal 0.96