International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Robotic Waste Detection and Segregation System Using Computer Vision and ROS2

Ayushi Gupta[1], Richa Sunil Dixit[2], Atif Shah[3], Janhavi Prashant Bartakke[4], Prof. Puja Johari[5]

[1,2,3,4]Bachelors of Electronic and Telecommunications Engineering, Savitribai Phule Pune University, Pune, India
[5]Prof. Alard College of Engineering and Management

Abstract
The municipal solid waste generated by world annually is 2.01 billion tonnes, averaging 0.74 kg per person per day. India produces between 159,000 and 170,338 tonnes of solid waste, which is expected to double by 2025. Moreover, approximately 35%–40% of urban municipal solid waste consists of dry recyclable materials. There are many developed countries that use advanced robotic machinery for waste segregation, while many other regions still rely on human workers. These traditional manual methods could be extremely slow, error-prone and hazardous to workers, who are consistently exposed to toxic, biomedical, and sharp materials. To address these problems, this paper presents a low-cost autonomous robotic waste detection and segregation system that use integrates Robot Operating System 2 (ROS 2), a Raspberry Pi, a Raspberry Camera Module, and a 5-degree-of-freedom robotic arm. Furthermore, a well-tuned YOLOv8 deep learning model is used for real-time, multi-class waste detection and classification. A ROS2 nodes manage multiple component such as perception, actuation and decision making in real time. The proposed system provides a practical, affordable, and scalable solution for intelligent waste management, suitable for smart city infrastructure and resource-constrained urban environments.

Keywords: Waste Segregation, ROS2, YOLOv8, Deep Learning

1. Introduction

The generation of solid waste is increasing at an extraordinary rate. According to World Bank predictions, solid waste is expected to rise from approximately 2.01 billion tonnes annually to nearly 3.40 billion tonnes by 2050 due to rapid urbanisation, population growth, and industrial expansion. This rapid increase leads to several environmental problems, including groundwater contamination, greenhouse gas emissions, ecosystem imbalance, and severe long-term public health hazards such as respiratory disorders, cholera, and vector-borne diseases.

The problem is particularly severe in densely populated urban areas of developing nations, where population growth and expanding settlements often outpace the development of municipal infrastructure. Waste commonly accumulates in public spaces such as parks, residential streets, markets, and busy transportation areas. In many cases, waste is generated faster than it can be collected. Additionally, the lack of proper waste segregation at the source leads to mixed waste streams, which reduces recycling efficiency and increases dependence on landfills.

Traditional waste management systems are mostly manual, involving human-dependent collection, sorting, transportation, and disposal. Although these methods have been useful for decades, they are limited by critical inefficiencies such as inconsistent segregation due to human error, high operational costs, slow throughput, and significant health risks. Workers are frequently exposed to toxic pollutants, biomedical waste, sharp materials, and biohazards, leading to respiratory disorders, skin diseases, infections, and other long-term health complications. Furthermore, manual methods are fundamentally incapable of scaling effectively with the increasing amount of waste generated in rapidly growing urban areas.

To address these challenges, there is a critical need for an automated and intelligent waste management solution that improves accuracy, efficiency, and worker safety. This paper presents the implementation and evaluation of a robotic waste detection and segregation system built using ROS2, a Raspberry Pi, a Raspberry Pi Camera Module, and a 5-DOF robotic arm. The primary goal is to replace the traditional manual systems with the automated waste detection and sorting system, which can not only handle the volume and variety of waste but also ensuring faster segregation than human workers.

Unlike previous systems that are often expensive, limited to a single waste category, or dependent on high-end computing hardware, the proposed system focuses on affordability, modularity, and multi-class waste handling. By deploying a trained YOLOv8 object detection model on a Raspberry Pi 5 and coordinating system components through ROS2 nodes, the system achieves real-time performance without requiring cloud connectivity or specialized processing units. The robotic arm provides the flexibility required to grasp objects of different shapes and sizes and place them into designated bins based on their material classification.

2. Literature Review

As populations continue to grow, waste management has become an critical challenge. This lead to a rise in research on automation and intelligent systems. This section reviews three key areas relevant to the proposed work: deep learning for waste detection, robotic manipulation for sorting, and the use of ROS2 in robotic systems.

Deep Learning for Waste Detection

Deep learning techniques, particularly the YOLO (You Only Look Once) family of object detection models, have become widely used for real-time waste detection. YOLO's single-stage architecture allows to process images in a single pass, making it faster than traditional multi-stage models such as Faster R-CNN or Mask R-CNN. This makes it more suitable for real-time applications on embedded systems.

Dongare and Thombare (2025) showed that even lightweight models like YOLOv5n (nano) can handle real-time multi-class waste detection on low-power hardware. Similarly, Dube et al. (2025) compared different YOLO versions using the COCO dataset and found that newer models such as YOLOv11 achieve better accuracy and faster inference compared to earlier versions. When deployed on a Raspberry Pi using ONNX conversion, YOLOv11 achieved strong performance, illustrating that modern deep learning models can be effectively used on affordable embedded devices.

Robotic Manipulation for Waste Sorting

While object detection is one of the key aspect, the physical handling of waste introduces additional challenges. Waste objects vary in shape, size, and material, making it difficult to reliably grasp objects in real-world conditions.

The LitterBot system (Almanzor et al., 2022) addressed this problem using soft Fin Ray grippers combined with visual servoing. The system achieved a high grasp success rate of 98.1% in cluttered environments

while relying only on a standard 2D camera, avoiding the need for expensive depth sensors. Dube et al. (2025) developed a low-cost 4-DOF robotic arm controlled by a Raspberry Pi, which was able to sort waste into three categories. However, the limited degrees of freedom reduced its ability to handle objects at different orientations. Liu et al. (2021) proposed a more advanced system that combined deep learning-based segmentation with robotic grasping on a mobile platform. Although effective, the system relied heavily on 3D point-cloud data, making it computationally expensive and less suitable for low-cost applications.

ROS2 in Robotic Systems

The Robot Operating System 2 (ROS2) has become a widely used middleware platform for modern robotics development. Compared to ROS1, ROS2 provides several important improvements, including real-time communication through the Data Distribution Service (DDS), better security features, improved support for multi-robot systems, and compatibility with resource-constrained embedded hardware. For waste management robots operating in dynamic, unstructured environments, ROS2's modular node-based architecture allows the system to be divided into separate components for perception, planning, and actuation. These components communicate through standardized message-passing interfaces, making the system easier to manage and expand.

Despite these advantages, the use of ROS2 in waste management robotics is still limited in existing research. Most existing systems connect hardware components directly through GPIO libraries or custom communication protocols, which lack the modularity and scalability provided by ROS2. This gap creates a significant opportunity: by building on ROS2, waste detection and sorting systems can be more readily upgraded, extended to support multiple robots, and integrated with smart city infrastructure.

Identified Research Gaps

A review of the existing literature highlights several important research gaps that motivate this work:

- Absence of ROS2-based architectures in waste detection and sorting systems. EMost setups are still quite rigid and not easily scalable.
- Limited multi-class sorting capability in low-cost systems. Many affordable systems are designed to collect a single type of waste or operate without material classification entirely, which reduces their effectiveness for recycling-based waste management.
- Inadequate manipulation capability for different waste objects. Most systems use 3-DOF or 4-DOF arms that cannot reliably handle objects placed at different orientations. A 5-DOF robotic arm provides an additional degree of freedom for wrist orientation control.
- Dependence on expensive hardware. Many research prototypes rely on industrial robotic arms, depth cameras, and GPU-based computing platforms. Such components are often too expensive for large-scale deployment in resource-constrained environments.

The proposed system directly addresses each of these gaps by combining a ROS2 middleware architecture, a Raspberry Pi 5 with Raspberry Pi Camera Module 3, a fine-tuned YOLOv8 model for multi-class real-time detection, and a 5-DOF robotic arm. This integrated approach aims to provide an affordable and practical solution for automated waste detection and segregation.

3. Methodology

The proposed system is made up of three-stage autonomous waste detection and sorting pipeline, integrating deep learning, ROS2 as a middleware, and embedded hardware. Unlike conventional approaches that focus only on detection, this system emphasizes end-to-end execution—from perception to actuation.

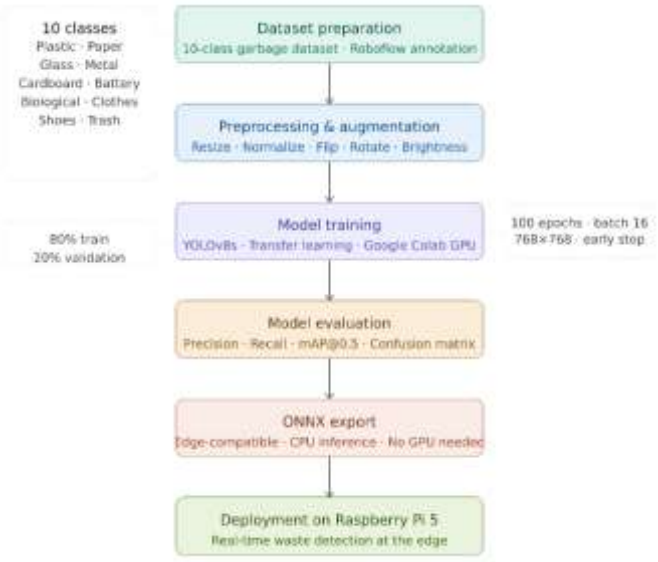
The architecture is divided into:

1. Vision-based waste detection using YOLOv8
2. Middleware-based communication using ROS2
3. Hardware-level actuation for physical waste segregation

This modular design enables scalability, real-time processing, and future integration with autonomous navigation systems.

Stage 1 — Software: YOLO-Based Waste Detection

Fig 1: YOLO Flowchart



Dataset Preparation and Annotation

A multi-class waste dataset was used, sourced from publicly available repositories of garbage dataset by Suman Kunwar. The dataset includes ten categories: plastic, paper, biological, glass, metal, cardboard, battery, clothes, shoes, and trash. An open-source platform Roboflow, has been used for the pre-processing and annotation of data. Bounding box annotation was performed with labels of different color code, which are given as follow:

Table 1: Labels

Color Code	Class Name	Image count
#FF00FF	Battery	491
#0E7AFE	Biological	1659
#00B7EB	Cardboard	352
#FF8000	Cloths	306
#FFFF00	Glass	572
#8622FF	Metal	366

Color Code	Class Name	Image count
#FFABAB	Paper	685
#FE0056	Plastic	759
#0000FF	Shoes	495
#A0522D	Trash	601

Furthermore, the dataset was divided into the training set (80%) and validation set (20%). Images are then resized to 640×640 pixels and pixel values are normalized. Class balancing is applied to handle minority categories.

Data Pre-processing and Augmentation

To improve generalization, the following augmentation techniques are applied:

- Image resizing to 640×640 resolution
- Augmentations including:
- Horizontal flipping

Model Selection and Training

The YOLOv8 (You Only Look Once version 8) model was selected due to its superior balance between speed, accuracy, and deployment efficiency compared to earlier versions such as YOLOv5 .

A pretrained YOLOv8s model was fine-tuned using transfer learning on Google Colab with GPU acceleration.

Training configuration

- Epochs: 100 (early stopping applied)
- Batch size: 16
- Image size: 768×768
- Optimizer: default Ultralytics optimizer

Early stopping was triggered at 85 epochs, when validation performance plateaued, preventing overfitting and reducing unnecessary computation.

Model Evaluation

The trained model was evaluated using standard object detection metrics:

- Precision – correctness of detections
- Recall – ability to detect all objects
- mAP@0.5 and mAP@0.5:0.95 – overall detection performance

Additionally, a confusion matrix was analyzed to identify class-level performance variations. It was observed that structured objects such as metal and clothes achieved higher accuracy, while irregular classes like biological waste showed comparatively lower performance due to variability in shape and texture.

Furthermore, similar objects such as plastic and glass were difficult to classify correctly.

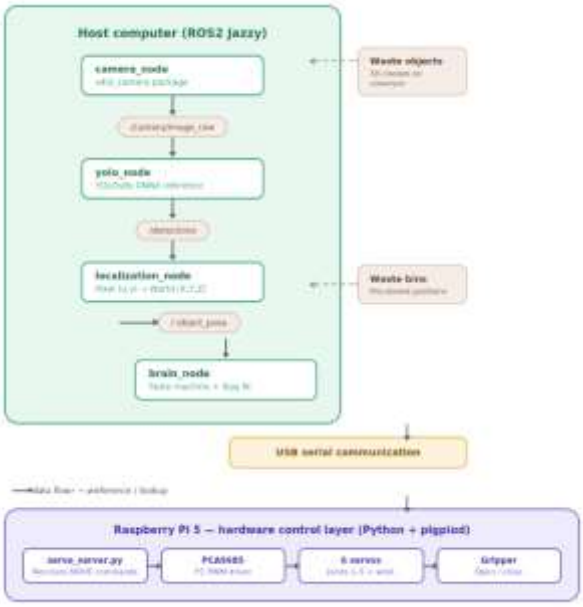
Model Deployment

To enable deployment on edge devices, the trained model was exported to ONNX(Open Neural Network Exchange) format, allowing efficient inference on CPU-based systems such as the Raspberry Pi 5.
This conversion ensures:

- hardware compatibility
- reduced latency
- independence from GPU requirements

Stage 2 — ROS2 Architecture

Fig 2: Ros2 Architecture



The proposed system is built using a distributed, two-layer architecture. At the top level, there's a host computer running the Robot Operating System 2 (ROS2 Jazzy), which handles most of the high-level processing. Alongside it, a Raspberry Pi 5 acts as a dedicated hardware control layer, mainly responsible for directly driving the physical components.

ROS2 Node Architecture

The perception and decision pipeline is implemented using four seperate ROS2 nodes. These nodes are communicating asynchronously via the ROS2 publish-subscribe mechanism.
The camera_node uses the v4l2_camera package to connect with the USB camera and continuously publishes raw image frames on the /camera/image_raw topic at 15 frames per second.
The yolo_node subscribes to /camera/image_raw and performs real-time object detection using a YOLOv8s model exported to the ONNX format. Detections are filtered with a confidence threshold of 0.6 and non-maximum suppression (NMS) IoU of 0.45. After that, the results are packaged into structured JSON messages—containing class IDs, confidence scores, and bounding box details—and published on the */detections* topic.

The **localization_node** then takes these detections and converts the bounding box center coordinates (u, v) from image space into real-world robot coordinates (X, Y, Z). This is done using the camera's intrinsic parameters along with an assumed workspace height. Once calculated, the object's position and its assigned bin are published to the */object_pose* topic.

Finally, the **brain_node** acts as the main controller. It runs a finite state machine with four states: IDLE, DETECTING, PICKING, and PLACING. When it receives an object pose, it calculates the required joint angles using inverse kinematics via the ikpy library, which is loaded with the robot's URDF model. It then generates smooth motion by interpolating waypoints for approaching, grasping, and lifting the object, sending them step by step.

Hardware Communication Layer

Since direct GPIO access is required for controlling the servos and gripper, this part is handled separately on the Raspberry Pi 5. A standalone Python script, servo_server.py, runs on the Pi and manages all low-level actuation.

The brain_node communicates with this script over a USB serial connection between the host computer and the Raspberry Pi 5, operating at a baud rate of 115200. The commands are sent as comma-separated ASCII strings. For example, messages with a MOVE prefix include six joint angles (in degrees), while messages with a GRIPPER prefix tell the gripper to either open or close.

On the Raspberry Pi side, the servo_server.py script continuously reads these incoming serial messages, parses them, and converts them into actual control signals. It uses the pigpiod daemon to generate hardware PWM signals, which are then sent to five GPIO pins controlling the arm joints, plus one additional pin for the gripper.

RViz2 Visualisation

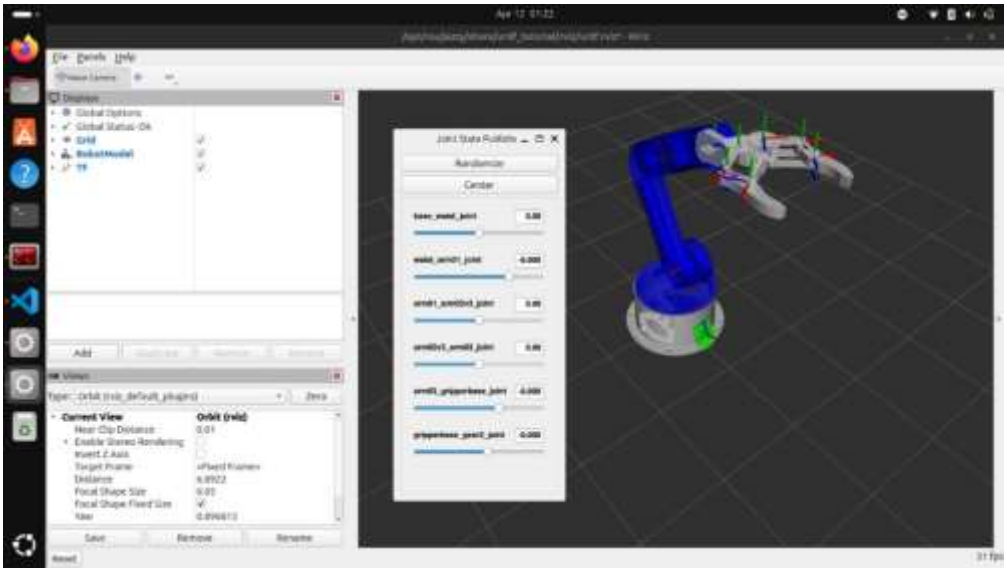


Fig 3: RViz Visualisation

RViz was mainly used during development and testing to visualize what the system was actually detecting in real time. Instead of directly testing everything on the physical robot, the annotated detection output from the yolo_node—published on the */detections* topic—was streamed into RViz for easier inspection.

System Launch and Configuration

All four ROS2 nodes—camera_node, yolo_node, localization_node, and brain_node—are started together using a single launch file called *waste_sorter.launch.py*. This launch file also loads system parameters from a YAML configuration file (*config/params.yaml*), so everything is initialized in one go.

On the hardware side, the servo_server.py script running on the Raspberry Pi 5 is started separately via SSH before launching the main system. It listens for incoming commands over the specified USB serial port at a baud rate of 115200.

Most of the important system settings are defined inside the YAML file. These include things like the ONNX model path, detection confidence threshold, camera frame rate, serial port details, joint angle limits for the servos, and mappings between detected classes and their corresponding bins.

This setup makes the system quite flexible. Instead of changing the source code every time something needs to be adjusted—like switching hardware or updating waste categories—you can just modify the configuration file. That makes it easier to reuse, adapt, and reproduce the system across different setups or test environments.

Stage 3 — Hardware Platform

Table 2: Hardware platform table

Component	Role
Raspberry Pi 5	The Raspberry Pi 5 acts as the main control unit of the system. It runs the **servo_server.py** script, which receives joint angle commands from the host system over a USB serial connection.
Raspberry Pi Camera Module 3	The Camera Module 3 is used for capturing visual input for the perception pipeline. It connects through the CSI-2 interface and works with the v4l2_camera ROS2 package to publish image frames to the */camera/image_raw* topic at around 15 fps.
Servo Motors (MG996R, SG90)	Three MG996R servos are used for the base, shoulder, and elbow joints, since these parts of the arm handle most of the load. Their higher torque helps maintain stable movement, especially when the arm is extended or lifting objects. For the wrist (roll and pitch) and gripper, three SG90 micro servos are used. These joints don't require as much torque, so lighter servos are sufficient. Using them helps reduce overall weight and inertia, which makes the arm's movement a bit more efficient and easier to control.

Component	Role
PCA9685 Servo Driver	The PCA9685 module handles PWM signal generation through the I²C interface and supports up to 16 channels. It provides stable, hardware-based PWM output, which helps avoid jitter and ensures smoother coordination between joints.
3D Printed Structure	All structural components of the system—including the arm links, joints, and gripper are based on an open-source design obtained from the *HowToMechatronics* platform.
Power Supply	The system uses separate power supplies for computation and actuation to avoid interference. The Raspberry Pi is powered using a 5V/3A supply, while the servos and PCA9685 are powered from a 6V rail.

Physical Operation Flow

The system runs in a continuous loop, starting with the camera capturing frames at around 15 fps and publishing them to the */camera/image_raw* topic. These frames are picked up by the yolo_node, which performs object detection using the YOLOv8s model (via ONNX). From each frame, it selects the most confident detection and publishes it to the */detections* topic.

Next, the localization_node takes this detection and applies a small debounce filter to avoid reacting to brief false positives. It then converts the detected object into a 3D position in the robot's coordinate frame and publishes this information on the */object_pose* topic.

Once the brain_node receives the object pose, it calculates the required joint angles using inverse kinematics. These angles are then packaged into a structured MOVE command and sent over USB serial to the Raspberry Pi.

On the Raspberry Pi side, the servo_server.py script reads the command, converts the joint angles into PWM pulse widths, and sends them to the PCA9685 driver via I²C. The robotic arm then executes the motion sequence—it moves above the object, closes the gripper to pick it up, lifts it, rotates toward the correct bin, and releases it. After that, it returns to its home position, ready for the next cycle.

Power and Electrical Integration

To keep the system stable, the power setup separates computation and actuation. This is important because servo motors can draw high current—especially during sudden movements—which could otherwise affect the Raspberry Pi.

The Raspberry Pi is powered using a dedicated 5V/3A USB-C supply, while the servos run on a separate 6V regulated power source. The PCA9685 driver gets its logic power from the Pi's 3.3V GPIO pins and communicates over I²C (at address 0x40), operating at a standard 50 Hz PWM frequency suitable for hobby servos.

Even though the power supplies are separate, they share a common ground to maintain proper signal reference. To handle sudden current spikes—like when multiple joints move at once—1000 µF capacitors are placed near the PCA9685. These help smooth out voltage fluctuations, especially during high-load actions like base rotation, where the MG996R servos can draw significant current.

Mechanical Design and 3D Printed Structure

All the structural parts of the system—including the arm links, joints, and gripper are based on an open-source design obtained from the *HowToMechatronics* platform.

The robotic arm has a 5-DOF configuration: base rotation, shoulder, elbow, wrist (roll and pitch), and gripper control. This setup gives enough flexibility to cover the working area and reach all designated bins.

For the heavier joints (base, shoulder, and elbow), MG996R servos are used because they can handle higher loads, especially when lifting objects or extending the arm. For the wrist and gripper, lighter SG90 servos are sufficient, which helps reduce overall weight and inertia.

The camera is mounted at a fixed 45-degree angle using a dedicated bracket, positioned so it can clearly see the workspace without being blocked by the arm during operation.

Another advantage of the design is its modularity—if a part breaks or wears out, it can be reprinted and replaced individually without having to rebuild the entire system.

3. Results

Model Performance

Table 3: Model Performance table

Metric	Values
mAP@0.5	0.770
mAP@0.5:0.95	0.550
Precision	0.785
Recall	0.706
F1 Score	0.74
Training Epochs	50
Training Platform	Google Colab

Per-Class Detection Performance

Table 4: Per Class Performance

Class	Precision (P)	Recall (R)	mAP50	mAP50-95
All	0.785	0.706	0.770	0.550
Battey	0.866	0.952	0.974	0.844
Biological	0.568	0.301	0.354	0.179

Class	Precision (P)	Recall (R)	mAP50	mAP50-95
Cardboard	0.853	0.678	0.790	0.669
Clothes	0.979	0.901	0.984	0.716
Glass	0.762	0.661	0.700	0.461
Metal	0.723	0.790	0.807	0.721
Paper	0.787	0.620	0.699	0.454
Plastic	0.757	0.701	0.777	0.551
Shoes	0.804	0.784	0.867	0.514
Trash	0.755	0.674	0.746	0.387

Graphs and Images

Training Curves

The model was trained for 100 epochs, with losses and metrics tracked throughout. As shown in Figure 4, all training losses decrease steadily, and validation metrics converge, confirming stable model training.

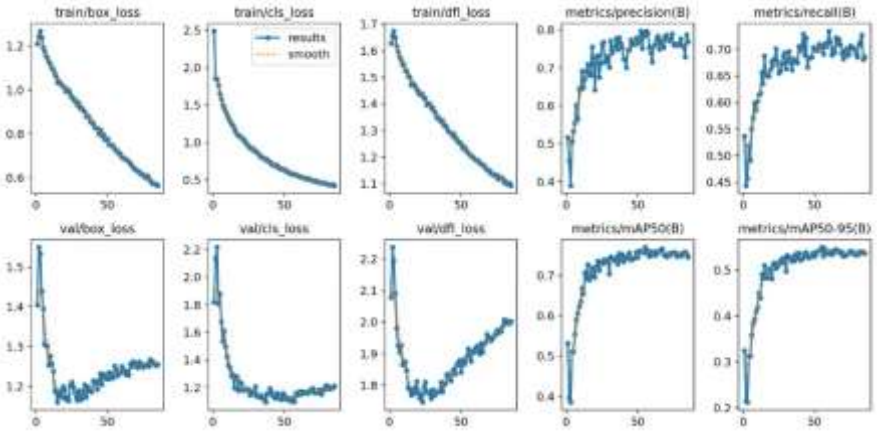
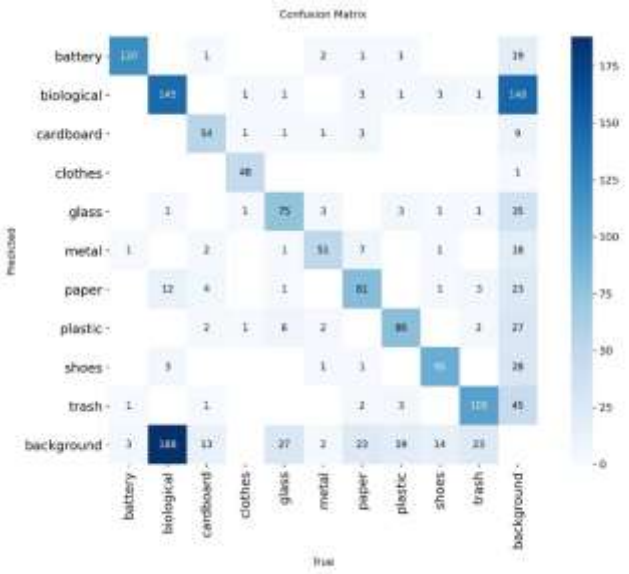


Fig 4: Training and validation loss curves along with key performance metrics

Confusion Matrix

Figure 5 present the per-class classification results on the validation set. Most classes show strong diagonal values, though *biological* and *glass* exhibit some confusion with neighbouring categories due to visual similarity.

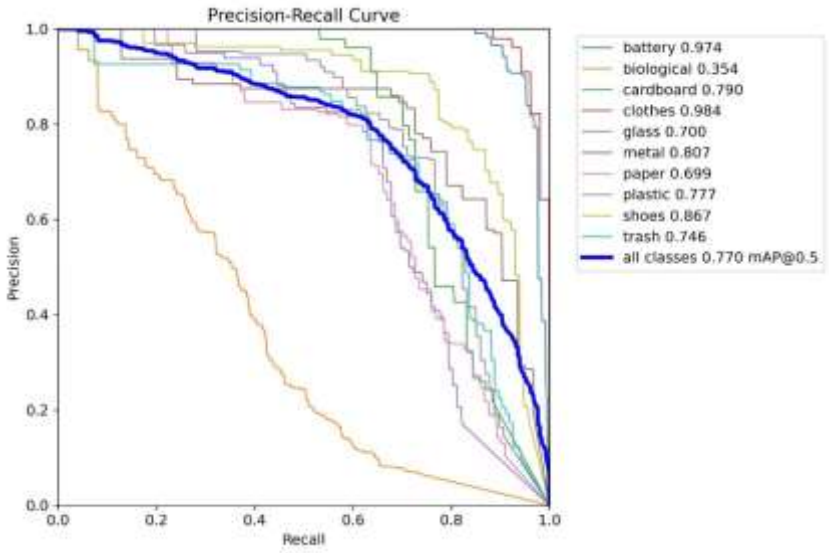
Fig 5: Confusion matrix of predicted versus true labels.



Precision-Recall Curve

Figure 6 shows the Precision-Recall curves per class. The model achieves an overall mAP@0.5 of 0.770, with *clothes* and *battery* performing best, while biological and *paper* remain the most challenging classes.

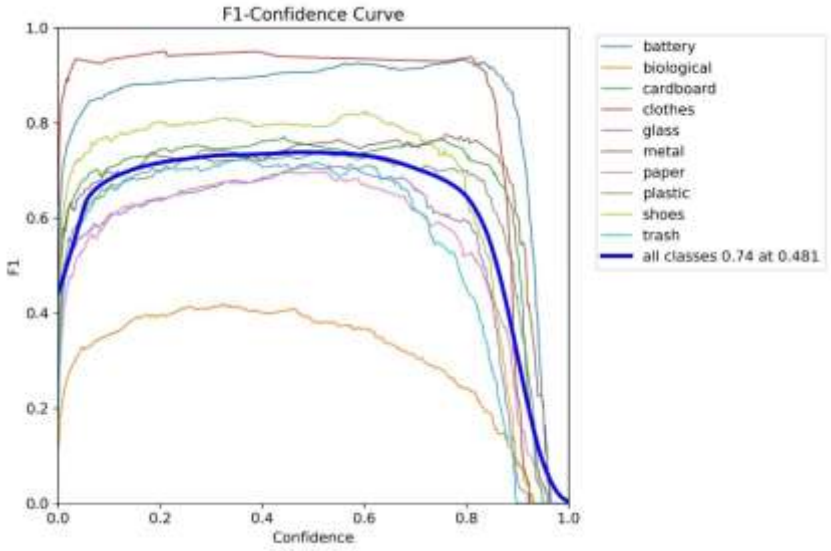
Fig 6: Precision-Recall curves per class and for all classes combined (mAP@0.5 = 0.770).



F1-Confidence Curve

Figure 4 shows the F1-score across confidence thresholds. The optimal threshold of 0.481 yields a macro-average F1 of 0.74, with *clothes* and *battery* being the most reliable classes.

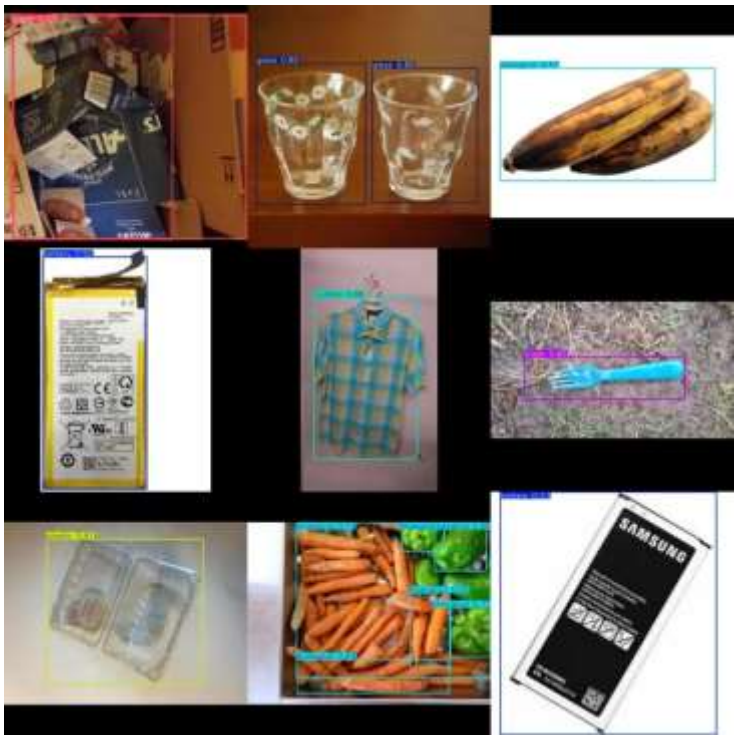
Fig 7. F1-Confidence curves threshold of 0.481.



Sample Detection Results on Validation Set

Figure 5 presents sample detection outputs from the model evaluated on the validation set of approximately 5,000 images. The model demonstrates reliable localization and classification across diverse waste categories, including battery, glass, clothes, plastic, trash, and biological waste. Confidence scores generally range above 0.80 for most classes, while biological waste exhibits lower and more fragmented detections, which aligns with its weaker performance metrics reported earlier.

Fig 8: Representative detection results from the validation set, showing predicted bounding boxes and confidence scores across multiple waste categories.



The trained YOLOv8s model achieved an overall mAP@0.5 of 0.770 and mAP@0.5:0.95 of 0.550 across ten waste categories, with an overall precision of 0.785 and recall of 0.706. Training completed in 85 epochs over approximately 3.67 hours on Google Colab (Tesla T4 GPU). The best F1 score across all classes was 0.74, achieved at a confidence threshold of 0.481. Structured categories such as clothes (mAP@0.5: 0.984) and battery (mAP@0.5: 0.974) performed best, while biological waste showed the lowest performance (mAP@0.5: 0.354) due to high variability in shape and appearance. On the Raspberry Pi 5, the average inference time was approximately 150–300 ms per frame, enabling near real-time detection. The overall system is expected to achieve an accuracy of 80–85% in a controlled environment, based on model evaluation results.

4. Future Scope

The current system has several limitations that present opportunities for future improvement. For one, it has only been tested in a controlled indoor setup. In real-world conditions, things like changing lighting, shadows, or objects partially blocking each other could affect how reliably the system detects waste.

On the manipulation side, the robotic arm also has its limits. It sometimes struggles with irregularly shaped or overlapping objects, which can lead to failed grasp attempts. The detection model itself is another constraint—it currently supports ten waste categories, but doesn't yet include important real-world types like e-waste or hazardous materials. Finally, the system lacks autonomous navigation, meaning waste must be placed within the camera's field of view manually.

- **Mobile Base with Autonomous Navigation** The most impactful upgradation is replacing the static base with a mobile ground robot equipped with wheel encoders and motor controllers. This would allow the system to move in defined areas autonomously, approaching detected waste rather than requiring waste to be placed in front of the camera.
- **SLAM and Localization** Simultaneous Localization and Mapping (SLAM) can be integrated using ROS2-compatible packages such as Nav2 and SLAM Toolbox. Combined with a LiDAR sensor or depth camera (e.g., Intel RealSense), the robot could map its environment, localize itself, and plan collision-free paths directly addressing the navigation gap identified in prior work such as Mărgăritescu et al. (2020) and the future recommendation of Dube et al. (2025).
- **Expanded Waste Classes** The current model covers a limited number of material categories. Adding classes like e-waste, hazardous materials, or even multi-material objects (for example, a plastic bottle with a metal cap) would make sorting more accurate and useful for real recycling processes.
- **Solar or Battery Power Integration** For outdoor deployment, integrating a solar panel and LiPo battery management system would make the robot operationally autonomous, allowing the robot to operate without frequent manual charging.
- **Edge AI Optimization** Techniques like quantization or pruning—such as using INT8 precision with TensorRT or ONNX Runtime—could make the YOLO model run faster on the Raspberry Pi 5. This would lead to quicker inference and more responsive sorting overall.
- **IoT Dashboard and Smart City Integration** Detection logs published via ROS2 can be forwarded to a cloud-based analytics platform, providing municipal authorities with real-time data on waste composition, collection frequency, and bin fill levels.
- **Multi-Robot Coordination** In large public spaces, multiple robots could operate collaboratively, with one robot detecting and flagging waste and another collecting it.

5. Conclusion

This paper presented the implementation of a stationary autonomous waste detection and sorting system integrating a YOLOv8s deep learning model, a ROS2 Jazzy middleware architecture, and a Raspberry Pi 5 hardware platform. The system addresses a very real and growing challenge in urban waste management, manual segregation is not only inefficient and inconsistent, but also exposes workers to health risks. By automating both detection and physical sorting, the system is able to classify and handle ten different types of waste in real time.

The proposed architecture was designed using a three-stage structure, that proved to be quite effective in separating system concerns. The perception stage runs on a host laptop and handles camera input along with object detection using ONNX-based inference through the yolo_node and localization_node. The coordination stage, implemented as a ROS2 finite state machine in the brain_node, manages detection debouncing, inverse kinematics computation, and command dispatch. Finally, the hardware actuation stage runs on the Raspberry Pi 5, where a script called servo_server.py converts incoming joint commands (sent over USB serial) into PWM signals using the PCA9685 driver to control the 5-DOF robotic arm. This modularity represents one of the system's principal strengths: each stage can be independently upgraded as superior sensors, models, or actuators become available, without requiring architectural redesign.

The YOLOv8s model used in this system was trained on a ten-class garbage dataset compiled by Suman Kunwar using publicly available data sources. It achieved an overall mAP@0.5 of 0.770 and a precision of 0.785 across categories like plastic, paper, biological waste, glass, and others. Some categories—like clothes and batteries—performed really well, with mAP values close to 1.0, mainly because they're visually distinct. On the other hand, biological waste proved more difficult to detect accurately, scoring around 0.354, likely due to how varied it can look. the model was exported to ONNX format, allowing it to run on low-power hardware instead of relying on cloud-based processing, a deliberate design decision ensuring the system remains viable in resource-constrained environments where cloud inference is impractical or unavailable.

The system is expected to achieve a pick-and-place success rate of around 80–85%, with each sorting cycle taking roughly 3 to 5 seconds. Some common failure cases were observed—like confusion between visually similar objects, small errors in mapping camera coordinates to robot space, and difficulty gripping oddly shaped items. These issues point to clear areas where the system can be improved in future versions. Rather than being a finished product, this system is meant to serve as a strong starting point. Because it's built on ROS2, it can be extended in several interesting ways. For instance, adding a depth camera could improve localization accuracy, while mounting the system on a mobile base with SLAM could enable it to move around and collect waste autonomously. Integrating IoT features could also allow real-time monitoring of bin levels and support remote management of multiple robots. With these additions, the system could evolve from a stationary setup into a fully autonomous waste collection solution for public spaces like parks or campuses.

Overall, this work presents a practical and extensible approach to AI-driven waste management. It shows that effective autonomous sorting systems don't necessarily require expensive industrial hardware—they can be built using accessible, low-cost components while still delivering solid performance. By combining modern deep learning, a standardized robotics framework, and compact computing hardware, this approach opens up a realistic path toward scalable, real-world deployment.

6.

7. References

1. Almanzor, E., Anvo, N. R., Thuruthel, T. G., and Iida, F. (2022). Autonomous detection and sorting of litter using deep learning and soft robotic grippers. Frontiers in Robotics and AI, 9, 1064853. https://doi.org/10.3389/frobt.2022.1064853
2. Dongare, K. and Thombare, B. (2025). A Machine Vision-Based Autonomous Waste Detection System Using Deep Learning. International Journal for Multidisciplinary Research (IJFMR), 7(6).
3. Dube, S. S., Magotsi, M. N., and Dube, T. P. (2025). Autonomous Litter Collecting Robot with Integrated Detection and Sorting Capabilities. International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS), XIV(VIII).
4. Kulshreshtha, M., Chandra, S. S., Randhawa, P., Tsaramirsis, G., Khadidos, A., and Khadidos, A. O. (2021). OATCR: Outdoor autonomous trash-collecting robot design using YOLOv4-tiny. Electronics (Switzerland), 10(18). https://doi.org/10.3390/electronics10182292
5. Liu, J., Balatti, P., Ellis, K., Hadjivelichkov, D., Stoyanov, D., Ajoudani, A., et al. (2021). Garbage collection and sorting with a mobile manipulator using deep learning and whole-body control. IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids), pp. 408–414.
6. Mushiri, T. and Gocheki, E. (2019). Design of a Garbage Collection Robot. https://doi.org/10.4018/978-1-5225-9924-1.ch004
7. Proença, P. F. and Simões, P. (2020). TACO: Trash Annotations in Context for Litter Detection. ArXiv preprint arXiv:2003.06975.
8. World Bank. (2018). What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050. World Bank Group, Washington DC.
9. https://howtomechatronics.com/tutorials/arduino/diy-arduino-robot-arm-with-smartphone-control/
10. https://www.kaggle.com/datasets/sumn2u/garbage-classification-v2/data