LR-Bot: An Autonomous Litter Rescuer

E6692.2024Spring.WOAT.report.tx2237.yq2379

Tianyi Xu, Yuan Qing
Department of Electrical Engineering
Columbia University
New York, USA
{tx2237, y.qing}@columbia.edu

***Abstract*—Trash accumulation around bins is a common issue due to improper disposal habits. In this project, to help alleviate this issue in indoor settings, we developed the litter-rescuing robot (LR-Bot), an autonomous litter-picking robot that leverages the Jetson Nano 2GB for edge computing and a 4 DoF robotic arm with a suction gripper for trash picking. To evaluate our robot, we collected a custom dataset named TrashDet, featuring images from 9 common indoor trash categories, each annotated with bounding boxes. After testing three popular detection algorithms on Jetson Nano, we selected YOLOv5s as our object detector for its low latency and high accuracy. The robot uses a front-mounted camera to identify and position target trash, achieving an inference speed of 12 FPS, complemented by an ultrasonic sensor for distance measurements. It is controlled through a customized task-specific control loop. In our testing environments, our LR-Bot achieved an overall success rate of 78.33% in complete litter rescue tasks across six trash categories over 120 trials.**

***Index Terms*—Robot, Jetson Nano, Object Detection, Control**

I. Introduction

Trash bins are ubiquitous in both outdoor and indoor settings worldwide. Despite their prevalence, improper disposal often leads to trash accumulating around these bins, posing significant environmental risks. This issue requires governments to allocate considerable resources to maintain cleanliness, burdening their budgets. Advancements in robotics and deep learning have introduced new possibilities for addressing such challenges. Robots are now capable of performing tasks that are too difficult or hazardous for humans, excelling in roles that require heavy lifting or involve dangerous conditions [1]–[3]. The development of Embodied AI has significantly enhanced robotic intelligence, driven by advancements in computational hardware and algorithms [4]. Similarly to humans, vision is a fundamental component of how robots perceive and interact with their environment, providing critical information for decision-making processes. Equipped with camera sensors and powerful object detection algorithms, such as YOLO [5], Fast-RCNN [6], and DETR [7], robots can now perceive and interact with their environments effectively. Existing research has demonstrated the successful deployment of litter-picking robots utilizing visual (i.e., RGB+Depth) and tactile inputs in outdoor environments [8]. However, the litter-picking task remains under-explored in indoor environments.

In this project, we propose the development of an innovative litter-picking robot designed specifically for indoor environments. Powered by the NVIDIA Jetson Nano, this robot features a simple yet effective robotic arm with a jaw gripper and a 4WD car base equipped with encoder motors and an ultrasonic distance sensor. It is designed to improve cleanliness by autonomously collecting misplaced trash. It employs a control policy integrated with the YOLOv5 [9] detection algorithm to identify litter efficiently. The deployment of this system aims to not only improve cleanliness but also foster a sustainable and well-ordered indoor environment. We also proposed a custom dataset named TrashDet with 9 categories of common indoor trash with 1035 annotated images to train our detection model. A comparative analysis of three different detection models on the Jetson Nano led us to select the YOLOv5s model as our trash detector, balancing accuracy with processing efficiency. The robot finally achieved an impressive overall success rate of 78.33% in litter rescue task over 6 trash categories.

II. Related Work

The background knowledge of this project can be summarized into three areas: object detection, system control, and inverse kinematics of the robotic arm.

A. Object Detection

With the development of computer vision and machine learning, the field has evolved through various stages, from early, handcrafted feature-based approaches to the state-of-the-art deep learning models that dominate today. In this project, we focus on deep learning methods for object detection. In general, there are single-stage and two-stage object detectors. R-CNN [10] is a representative work of two-stage methods, using selective search to propose regions and a CNN to classify them. Subsequent methods like Fast R-CNN [6] and Faster R-CNN [11] improved this by integrating region proposals and feature extraction into a more efficient pipeline. Single-stage detectors like YOLO [5] achieved faster detection speeds by bypassing the region proposal step, directly predicting

Code is available at https://github.com/eecse6692/e6692-2024Spring-FinalProject-WOAT-yq2379-tx2237/tree/main.

Slides are available at https://docs.google.com/presentation/d/1gLuo6JaGeOJf3PwBzSnUWWYqbtyZsl73AYNQY_3XAGo/edit?usp=sharing.

Demo is available at https://drive.google.com/file/d/1Go_O_1Yr1Tuo00A_9AV2A6Qz8YYZV2YR/view?usp=sharing.

bounding boxes and class scores in a single pass using anchors. It has led to a series of follow-up works, which have been widely used in real-time applications due to their speed, though at the expense of some accuracy compared to two-stage detectors. Building upon its predecessors, YOLOv5 [9] employs a more modular architecture using the CSP network for efficiency, integrates mosaic augmentation and CIoU for enhanced training, and is optimized for deployment across platforms via its PyTorch-based implementation. It is also the first widely applied and deployed YOLO method. YOLOv8 [12] brings significant improvements in detection accuracy and model efficiency by leveraging enhanced network architecture and training techniques. YOLOv9 [13] further refines these aspects by focusing on advanced post-processing methods and novel training schemes to push the limits of accuracy and speed in single-stage detectors. More recently, transformer-based models have influenced object detection with DETR [7], which introduced a novel, end-to-end approach using attention mechanisms, simplifying the detection pipeline. However, transformer-based approaches require extensive computational power, making them less feasible for edge computing due to the large number of parameters. In this project, we chose YOLOv5 [9] for object detection, as it provides a good balance between accuracy and latency.

B. Control

Control systems play a pivotal role in the robot industry, governing the behavior of robotic systems to achieve desired outputs. Multiple control methods have been created in decades to deal with complicated control systems, but in this project, the traditional method is sufficient for our robot. A very aged control method, Proportional – Integral – Derivative (PID) control dates back to the late 19th and early 20th centuries. The earliest form of PID control can be traced to pneumatic and mechanical controllers used in the process industries. The notable contribution to the PID controller was made while working on automatic steering systems for US Navy ships [14], observed the behavior of helmsmen in controlling the ship and formulated the PID principles based on these observations. In recent years, PID control has been enhanced with adaptive and intelligent control elements to handle nonlinearities and uncertainties in more complex systems. These advancements continue to make PID control a relevant and critical component in both traditional and modern control systems across various industries [15].

Another classical control method is linear quadratic regulator (LQR) control. Originating in the 1960s, LQR was formulated as part of the optimal control theory [16].In recent years, adaptations of LQR for nonlinear systems and the incorporation of robust and adaptive control strategies have expanded its applicability. These developments ensure that LQR remains relevant in contemporary control challenges, including those encountered in autonomous vehicles and intelligent manufacturing systems [17].

Fuzzy control is a branch of control theory that utilizes fuzzy logic to handle imprecision and uncertainty in complex systems, a distinct departure from traditional binary logic approaches. The concept of fuzzy control originates from the theory of fuzzy sets, which was introduced in 1965 [18]. Fuzzy sets were proposed as a means to mathematically represent uncertainty and imprecision, providing a systematic way to address problems in systems analysis, control, and decision-making where traditional binary logic was inadequate. Today, fuzzy control is explored in areas such as autonomous vehicles, robotics, and smart grids. Its ability to handle ambiguity and make decisions in real-time continues to make it a valuable tool in the design of intelligent systems [19].

In this project, we need to adjust the motor-driven voltage output according to the speed measured by motor encoders. We finally chose PID as our control method because it can be easily implemented, relatively faster, and can achieve high accuracy with low computational cost without influencing the inference speed of object detection.

C. Inverse Kinematics

In a synthesized review of kinematic analysis in robotics, we explore essential methodologies from prominent scholarly articles. The forward kinematics employs the Denavit-Hartenberg (DH) parameterization to simplify robotic joint configurations, fundamental for robotic design and simulation [20]. Additionally, geometric methods [21] for efficiently computing joint angles for desired end-effector positions emphasize the enhancement of real-time application performance in robotic systems. These pivotal studies underscore the critical role of kinematic modeling in advancing the precision and functionality of robotic manipulators. In our project, we use this method which is originally supported by the robotic arm to compute its end-effector pose.

III. Method

In this section, we present our method, which is divided into several key components: overall architecture, trash detection, control system, LR-Bot body, and environment setup.

A. Overall Architecture

1) Hardware Architecture: Fig. 1 depicts the comprehensive hardware architecture of the LR-Robot. The system utilizes the Jetson Nano and ESP32 as its controllers, with the Jetson Nano serving as the primary controller. The sensors of the robot include two-speed encoders, an ultrasonic sensor, and a camera. The vehicle's propulsion is powered by four TT motors, which are operated by an L298N motor driver module.

The connection between the Jetson Nano and the motor driver module involves two General-Purpose Input/Output (GPIO) pins and two Pulse-Width Modulation (PWM) pins. The motor's power derives from the electrical potential difference; setting the GPIOs to low enables a higher PWM duty cycle to increase motor power. Conversely, setting the GPIOs to high reverses the logic, causing the motors to operate in the opposite direction. All four TT Motors are connected to the output ports of the motor driver module, with the ESP32 supplying power to the driver.

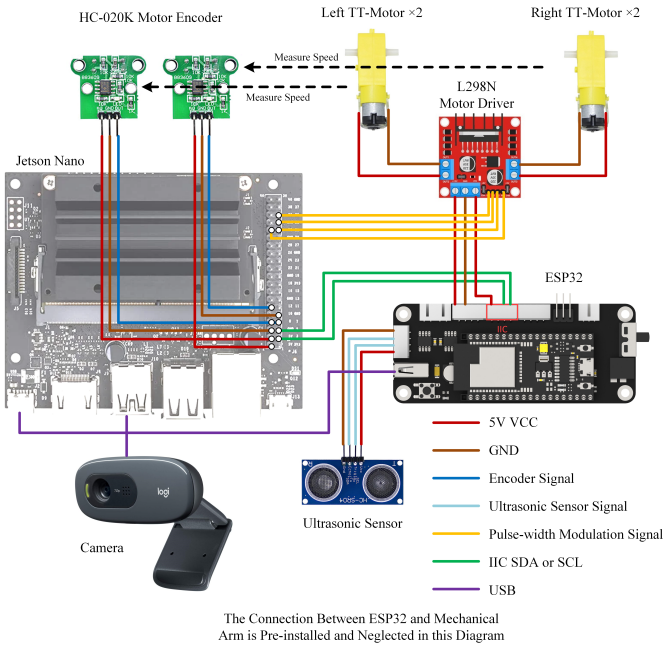

Fig. 1. Hardware architecture of the whole robot system.

The motor speed encoders function by measuring the rotational speed through a slotted disc attached to the motors. As the slots pass the encoders, they detect changes in light and generate pulses on the output pin, which are counted by the Jetson Nano to determine the motor's speed. The encoders receive their power directly from the Jetson Nano.

The ultrasonic sensor is connected to the ESP32, which triggers the sensor via pulses on a trigger pin, prompting the sensor to emit ultrasound. Upon receiving the reflected sound, the sensor sends a pulse back to the echo pin, which is then captured by the ESP32.

Furthermore, the Jetson Nano is linked to a camera via a USB port, and it receives its power supply from the ESP32. Communication between the two controllers is facilitated through I^2C. Additionally, servos and a pump integrated into the robotic arm are connected to the ESP32, although these components are not included in the diagram.

2) Software Architecture: Fig. 2 illustrates the software architecture of the LR-Bot. The primary computing platforms for the robot are the Jetson Nano and ESP32, while other components function as peripheral devices.

The image acquisition process begins with the camera capturing visuals that are subsequently processed by YOLOv5 to detect objects and generate bounding boxes. The positional deviation δ between the center of these bounding boxes represented by a 4-element array $(x_{upperleft}, y_{upperleft}, x_{lowerright}, y_{lowerright})$ and the image (width w, height h) center is calculated using Eq. 1 to determine the necessary angular adjustment for the robot's

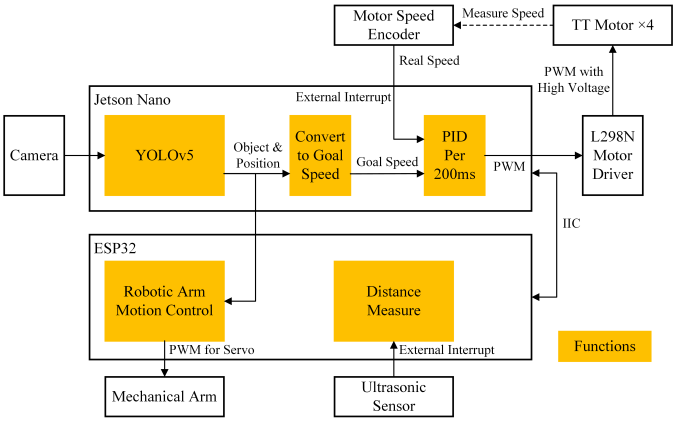

Fig. 2. Software architecture of the whole robot system.

orientation toward the target object.

$$\delta = \frac{x_{upperleft} + x_{lowerright} - w}{w}. \quad (1)$$

This deviation is then translated into differential speeds for the left and right motors to facilitate directional changes.

The speed of the motors is monitored by encoders that produce a pulse each time they detect changes in the light passing through the slotted disc on the motor. The Jetson Nano employs an external interrupt to track these pulses, incrementing a count that quantifies the rotational distance of the motor. The motor's speed is computed by multiplying this count by a coefficient within a specified time frame, after which the count is reset. The desired speed is compared with the actual motor speed using a Proportional-Integral-Derivative (PID) controller to compute the appropriate PWM signal, which adjusts the motor's power output. This PID regulation is crucial for adapting the robot's speed to varying load conditions, such as inclines, preventing stalling or reduced speed under increased load.

The ESP32 controls the robotic arm. It generates PWM signals to adjust the servo angles and operate the pump and valve. Additionally, the ESP32 controls the ultrasonic sensor by emitting pulses to initiate ultrasound emissions and receive the resultant signal through an external interrupt. The duration of the high-level voltage signal returned by the sensor is directly proportional to the measured distance, which the ESP32 calculates by applying a predefined coefficient to the voltage duration. This integrated control and feedback mechanism allows the ESP32 to accurately gauge and respond to environmental distances.

B. Trash Detection

Our detection model is a fine-tuned YOLOv5s model. The overall architecture is shown in Fig. 3. The model consists of three parts: a CSPNet [22] backbone, a PANet [23] neck, and an output YOLO [5] layer head. For technical details of each submodule, we refer readers to the original papers. The model outputs bounding boxes and classification scores for

each image frame, which are then passed to the control system to guide the robot's movements.

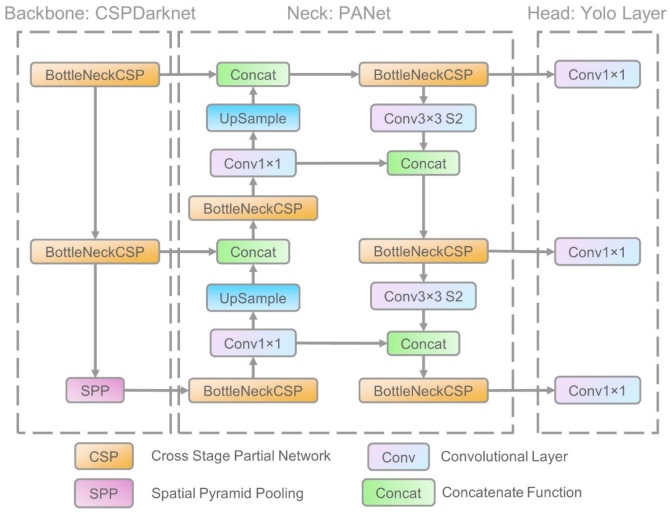

Fig. 3. Model architecture of YOLOv5 [9] from [24].

C. Control System

The litter-picking task is complex and cannot be accomplished in a single step. Therefore, we have divided the task into eight distinct stages to form a task-specific control loop, which is cyclically repeated. This control loop is depicted in Fig. 4 and described as follows:

- Stage 0: The initial stage where the camera continuously monitors for trash. The car remains stationary until a piece of trash is detected. Upon sighting trash, the robot approaches the first detected item and ignores subsequent detections. The transition to the next stage occurs when the ultrasonic sensor measures the distance to the trash as being below a predetermined threshold.
- Stage 1: Upon entering this stage, the ESP32 instructs the Jetson Nano to stop the vehicle. The robotic arm then descends to grasp the trash, with the extent of the arm's reach and depth determined by the type of trash, as communicated by the Jetson Nano. Completion of the grasping task triggers the transition to Stage 2.
- Stage 2: When Jetson Nano knows that ESP32 has finished the grabbing task, it will start turning the car to find the trash bin. Left motors are set to move forward and right motors move backward to turn the car in one place without moving forward. When the trash bin appears in the camera's sight, Jetson Nano will tell the ESP32 that they are in stage 3.
- Stage 3: Similar to Stage 0, but with the target now being the trash bin. When ESP32 finds out that they are close enough to the trash bin through the ultrasonic sensor, the car will enter stage 4.
- Stage 4: This stage mirrors Stage 1 in mechanics but involves the robotic arm extending slightly to ensure the trash is deposited into the bin. The car remains stationary during this process. Completion of the depositing task prompts the ESP32 to advance to Stage 5.
- Stage 5: The car will go backward in stage 5. All the motors will go backward. The condition for the next stage is the distance between the car and the trash bin is smaller than the threshold.
- Stage 6: The car searches for the red cube, designated as the starting point. This stage operates similarly to Stage 2, with the vehicle pivoting to locate the red cube. Spotting the cube advances the system to Stage 7.
- Stage 7: This stage functions similarly to Stage 0, with the red cube as the target. The ultrasonic sensor continues to detect the distance between the car and the red cube. If the distance is smaller than the threshold, the car will enter stage 8.
- Stage 8: The concluding stage of the loop. The vehicle turns and searches for the trash bin, preparing to re-enter Stage 0 for continuous monitoring of the environment. Once the trash bin is located, the vehicle automatically resets to Stage 0, thus completing the control loop cycle.

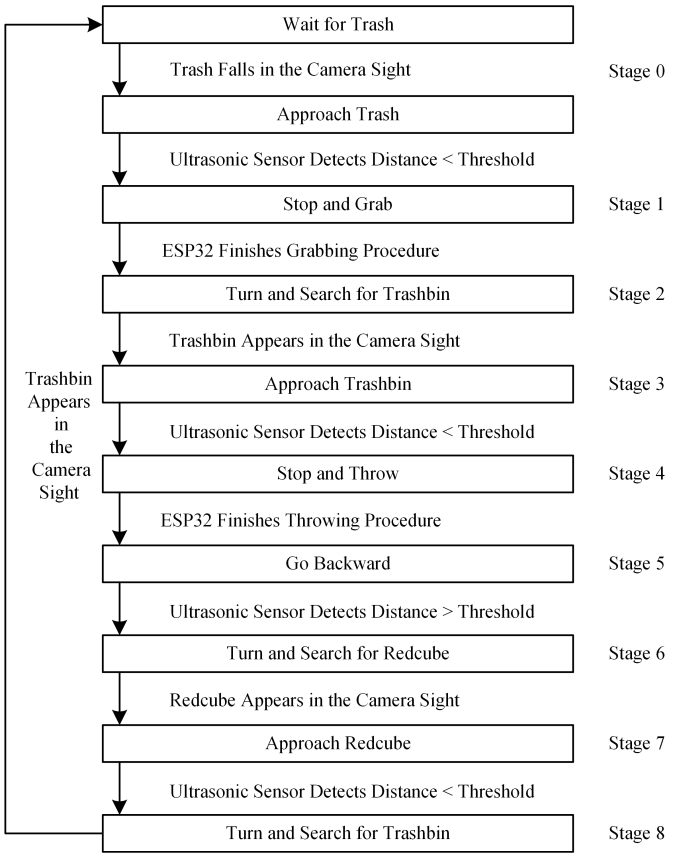

Fig. 4. The control loop of the robot.

D. LR-Bot Body

Fig. 5 illustrates the overall design of the robot and the placement of its hardware components. Fig. 5(a) shows the

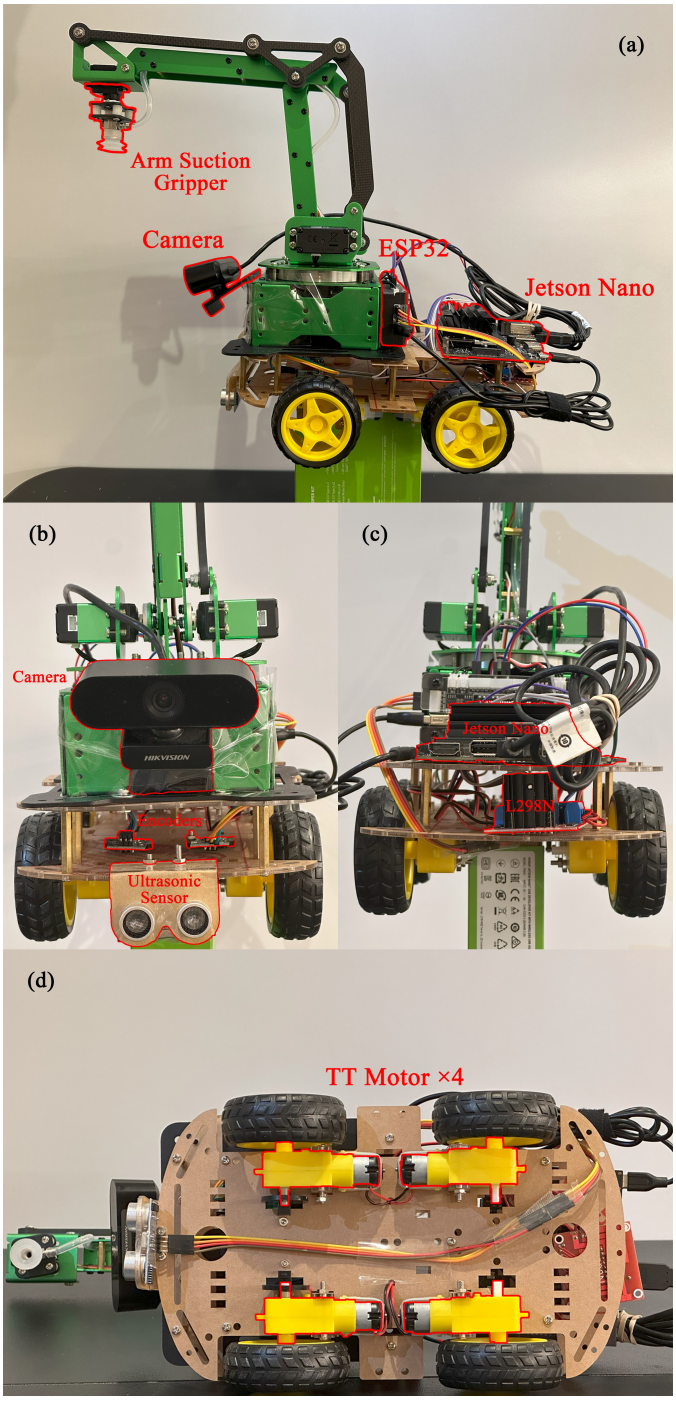

Fig. 5. The LR-Bot.

ESP32, which is mounted on the rear of the robotic arm. Jetson Nano is carried on the back of the car. The suction gripper is also visible, extending from the body of the vehicle. The camera is in front of the robotic arm. (b) presents a front view where the camera is attached to the robotic arm, positioned above an ultrasonic sensor. Additionally, the encoders are affixed to the first panel of the chassis. In (c), the motor driver module is located on the vehicle's first panel, with the Jetson Nano situated directly above it on the second panel. Finally, (d) reveals the underside of the vehicle, where two of the four front motors are equipped with encoders.

E. Environment Setup

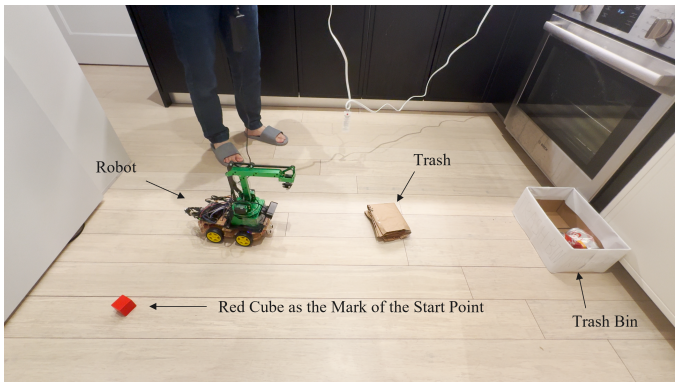

Fig. 6. The environment setup of the experiment.

The environment setup is shown in Fig. 6. The setup includes our LR-Bot collecting trash from the floor and depositing it into a bin located on the opposite side of the testing area. A red cube serves as the designated starting point for the robot. This indicates that after completing each collection task, the robot must return to the red cube, reorient itself towards the trash bin, and monitor the area for any additional trash that may be discarded onto the floor.

IV. Experiments

In this section, we introduce our custom dataset, detail the implementation of both software and hardware, and present the results of our experiments along with relevant analysis.

A. Dataset

To test our robot, we set up the testing environment mentioned above within our living room for both testing and data collection (see Appendix A). For this task, we built a Trash Detection (TrashDet) dataset. Our dataset comprises 11 classes: *envelope*, *fruitpeel*, *paperbag*, *paperball*, *paperbox*, *plasticbag*, *plasticbottle*, *plasticcup*, *redcube*, *tincan*, and *trashbin*. The *trashbin* class identifies the target trash bin, while the *redcube* class indicates the robot's starting point. These classes were selected based on their frequency of occurrence in daily life. The distribution of instances for each class is illustrated in Fig. 10. The total dataset consists of 1035 images and 2632 annotations. The distribution of the training and validation sets is presented in Table I.

B. Implementation Details

1) Object Detection: For this project, we chose YOLOv5s [9] as our trash detector. We tested and deployed YOLOv5 [9], YOLOv8 [12], and DETR [7] on the Jetson Nano, and the results are shown in Table II. Note that the inference time

TABLE I
DISTRIBUTION OF THE TRAINING, TESTING, AND VALIDATION DATASETS.

Split	# Samples
Training	833
Validation	202

was measured using TensorRT engines of all the three models. Although the current state-of-the-art detector is YOLOv9 [13], its authors did not release a small model, and the existing versions are too large to run on the Jetson Nano. Hence, we used YOLOv8 [12] and DETR [7] for comparison. DETR [7] is a transformer-based object detector with more parameters and a larger input size compared to the YOLO models. We included DETR [7] to make our comparison more comprehensive.
Based on the results, YOLOv5 [9] demonstrates the fastest inference speed at approximately 75ms per image, with the smallest number of parameters. In contrast, YOLOv8 [12] requires nearly double the inference time of YOLOv5 [9] for a single image. DETR [7] takes the longest time, significantly exceeding our latency requirements. Despite these differences, YOLOv5 [9] achieves a comparable mean average precision across IoU thresholds from 0.5 to 0.95, relative to the other models. Taking into account the latency and accuracy requirements of our task, we have selected YOLOv5 [9].

The YOLOv5 model was trained for 500 epochs with a starting learning rate of 0.01, a weight decay of 0.0005, and a batch size of 16. We used the Adam optimizer [25] for all experiments. Our model was fine-tuned on our dataset using the YOLOv5s weights pretrained on COCO [26] dataset. The training plot of YOLOv5s is shown in Fig. 7.

After fine-tuning, the model was converted into a TensorRT engine using the API provided by [27]. We also quantized the model to FP16 during the conversion process to the TensorRT engine, as this approach increases inference speed, and our accuracy requirements are not exceedingly stringent. The fine-tuning was conducted on an RTX 4070 Ti, and inference was executed on the Jetson Nano.

For the implementation details of the other two models, please refer to Appendix B.

TABLE II
COMPARISON OF YOLOV5, YOLOV8, AND DETR ON SEVERAL PERFORMANCE METRICS.

Model	**Input Size**	**# of Params**	**Inf. Time**	**mAP50-95**
YOLOv5 [9]	640x640	7.2M	75ms	0.933
YOLOv8 [12]	640x640	11.2M	132ms	0.949
DETR [7]	800x1066	41M	1215ms	0.883

2) Hardware: The hardware list is shown in Table III. The installation can refer to Fig. 5.

The camera should be mounted slightly downward to enhance floor visibility and minimize interference from the surrounding environment. The robotic arm is positioned at the front of the vehicle. Due to the considerable weight of the arm, encoders are installed on the front motors to prevent the rear motors from idling. Dupont wires are routed through the car panel for organized cable management. Each motor is connected with one black and one red wire. Given that the L298N motor driver module contains only two sets of output ports, with each set comprising two output pins, wires of the same color on the same side must be soldered together to secure the connections and prevent detachment. Excess wire length should be either tied or affixed to the vehicle body to avoid entanglement in the wheels.

TABLE III
COMPONENT LIST.

Component	#
Jetson Nano	1
ESP32 Micro Controller	1
Robotic Arm	1
TT Motor	4
Wheel	4
Slotted Disc	4
HIKVISION DS-E12a Camera	1
HC-020K Motor Speed Encoder	2
HC-SR04 Ultrasonic Sensor	1
L298N Motor Driver Module	1
Plastic Panel (Car Body)	2
Micro USB Cable	1
USB-C Cable	1
12V Battery	1
M3 Screw and Nut	Several
Dupont Wire	Several

C. Results

In this section, we report the experimental results of our task both in parts and as a whole. The complete task involves litter detection, picking, and dropping. We also present results on detection and picking task in two subsections.

1) Overall Results: To enhance the comprehensiveness of our experiment, we randomly select 2 objects from each category and conduct 10 runs for each object. Consequently, we have performed a total of 180 experiments for the complete task, encompassing recognition, picking, dropping, and returning to the standby state. The overall experimental results are presented in Table IV.

TABLE IV
EXPERIMENTAL RESULTS OF THE COMPLETE TASK.

Category	# Success	# Failure	Success Rate (%)
Envelope	14	6	70.00
Fruitpeel	0	20	0.00
Paperbag	17	3	85.00
Paperball	2	18	10.00
Paperbox	18	2	**90.00**
Plasticbag	13	7	65.00
Plasticbottle	17	3	85.00
Plasticcup	0	20	0.00
Tincan	15	5	75.00
Overall	96	84	53.33

It is evident from the table that three categories of trash, namely *fruitpeel*, *paperball*, and *Plasticcup*, exhibit an almost

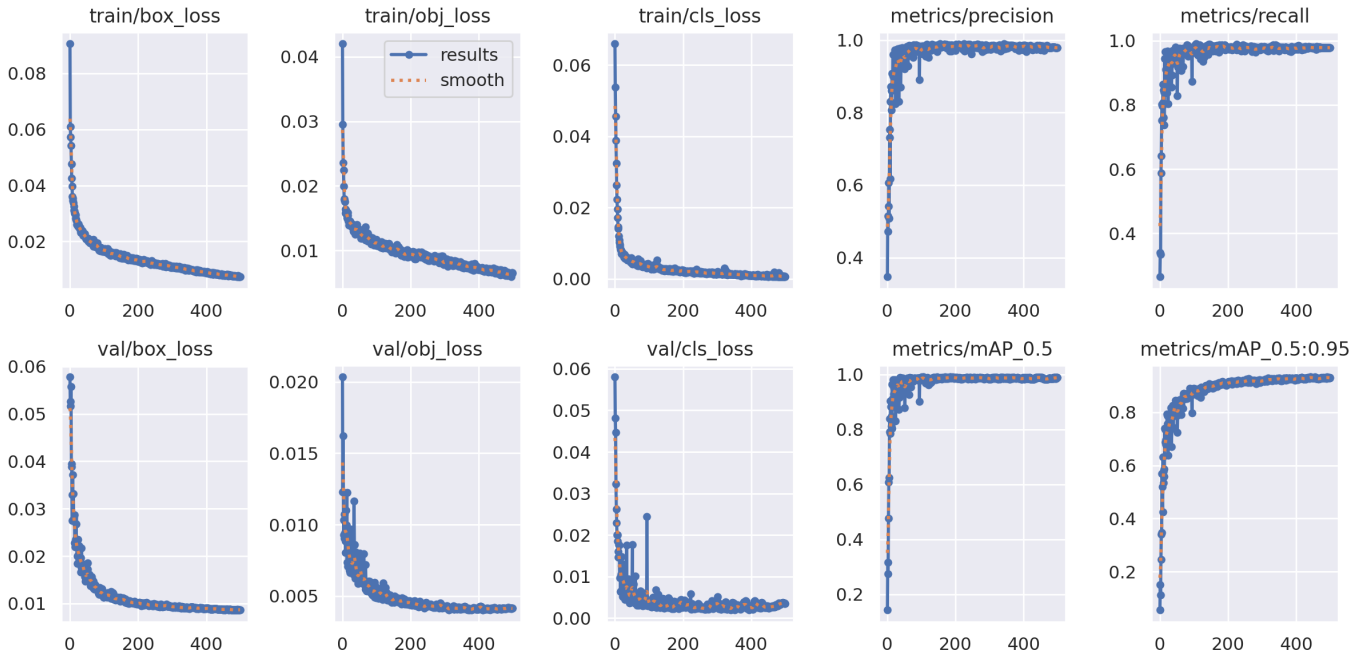

Fig. 7. Training plots of YOLOv5 model during 500 epochs.

0% success rate. This outcome is expected due to the nature of the gripper used, which is a suction gripper. Such a gripper necessitates seamless contact with the target object to execute successful suction. However, objects from these three classes possess uneven surfaces, making them challenging to grasp using this gripper. For this issue, which stems not from our control system or detection algorithm, but rather from a mechanical limitation, we opted to refine the evaluation trash categories to the following six: *envelope*, *paperbag*, *paperbox*, *plasticbag*, *plasticbottle*, and *tincan*. With this adjustment, we present the refined results for further evaluation in Table V. In general, for one complete loop of the litter-picking task (i.e., start from the red cube,..., go back to red cube), the average execution time is approximately 57 seconds.

TABLE V
REFINED EXPERIMENTAL RESULTS OF THE COMPLETE TASK.

Category	# Success	# Failure	Success Rate (%)
Envelope	14	6	70.00
Paperbag	17	3	85.00
Paperbox	18	2	**90.00**
Plasticbag	13	7	65.00
Plasticbottle	17	3	85.00
Tincan	15	5	75.00
Overall	94	26	78.33

From the refined results, it's evident that the overall success rate of our method stands at 78.33%, with the highest success rate of 90.00% observed in the class *paperbox*, and the lowest success rate observed in the class *plasticbag*. This overall success rate aligns with our expectations, demonstrating the robust performance of the system in testing environments. For objects with rigid body properties and smooth surfaces, such as paper boxes, the suction gripper excels. However, for objects like plastic bags, characterized by deformable shapes and uneven surfaces, grasping poses a greater challenge for the gripper. For further insights on grasping, please refer to Section IV-C3.

2) Trash Detection: The detection experiment results of the best model during training are shown in Table VI. The overall detection mAP50 reached 99%, and the mAP50-95 achieved 93.9%, meeting our preset requirements for the detection task. Among all classes, *paperball* had a comparatively low mAP50-95, which was expected since paper balls and white plastic bags were very similar visually. *Trashbin* had the highest mAP50-95, at 0.985, as it contained the largest number of training instances among all classes. A visualization of some detection results is shown in Fig. 8. From the visualization, we can see that the objects are all correctly detected with very high classification scores and accurate bounding boxes. These results, along with the numerical analysis, imply that our detection model is quite robust and accurate. Additionally, the results of detection during real-world experiments are illustrated in Table VII under the refined settings. This result is expected as we collected all our training data in the same testing scene, and the test objects are all inside our training domain.

3) Grasp: We further elaborate on the grasp task to demonstrate its significance in determining the overall success rate. Following the refined categories, we report the results of the grasp task in Table VIII.

TABLE VI
EXPERIMENT RESULTS OF FINETUNED YOLOV5 MODEL ON TRASHDET VALIDATION SET, W.R.T. PRECISION, RECALL, MAP50 AND MAP50-95.

Class	Instances	P	R	mAP50	mAP50-95
All	497	0.984	0.980	0.990	0.933
Envelope	31	0.999	0.968	0.994	0.892
Fruitpeel	39	0.996	1.000	0.995	0.919
Paperbag	16	0.970	1.000	0.995	0.955
Paperball	67	1.000	0.955	0.987	0.898
Paperbox	55	1.000	0.996	0.995	0.961
Plasticbag	54	0.928	0.981	0.981	0.928
Plasticbottle	49	0.976	1.000	0.990	0.925
Plasticcup	22	1.000	0.903	0.991	0.901
Redcube	15	0.987	1.000	0.995	0.964
Tincan	52	0.977	0.981	0.968	0.936
Trashbin	97	0.988	1.000	0.995	0.985

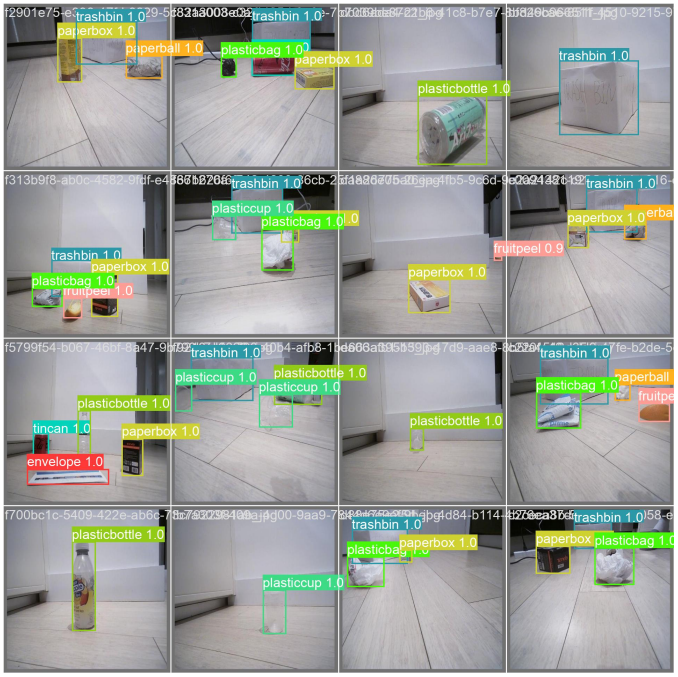

Fig. 8. YOLOv5 detection results on TrashDet dataset.

From the table, it's evident that there's minimal difference compared to the overall task success rate, with a 5% increase. This difference is primarily attributed to some instances of out-of-bin trash drops. However, we refrain from reporting the experimental statistics on the drop task here, as it is highly correlated with the picking task. This correlation stems from the fact that if picking fails, dropping must inevitably fail as well. Upon comparison with Table V and VII, it becomes apparent that the success rate of picking holds the most significant overall impact on the complete task. This observation is supported by the nearly 99% success rate in detection, and few out-of-bin droppings caused by inaccurate detection of the ultrasonic sensor.

TABLE VII
REFINED EXPERIMENTAL RESULTS OF THE DETECTION TASK.

Category	# Success	# Failure	Success Rate (%)
Envelope	18	2	90.00
Paperbag	19	1	95.00
Paperbox	20	0	**100.00**
Plasticbag	20	0	**100.00**
Plasticbottle	20	0	**100.00**
Tincan	20	0	**100.00**
Overall	117	3	97.50

TABLE VIII
REFINED EXPERIMENTAL RESULTS OF THE GRASP TASK.

Category	# Success	# Failure	Success Rate (%)
Envelope	17	3	85.00
Paperbag	17	3	85.00
Paperbox	19	1	**95.00**
Plasticbag	14	6	70.00
Plasticbottle	18	2	90.00
Tincan	15	5	75.00
Overall	100	20	83.33

V. DISCUSSION

In general, we obtained an overall success rate of 53.33% on the 9 classes of trash and a success rate of 78.33% on the refined 6 classes. From our analysis in Section IV-C, we found that the grasping task significantly impacts the overall success of our experiment. This is primarily because the vision algorithm is extremely robust, exhibiting almost no failures that could lead to incorrect or missed detections. Meanwhile, the dropping task, which is correlated to picking, proves to be error-tolerant; we can consistently deposit trash into the bin by extending the robotic arm. In the following parts, we will provide further analysis on runtime and discuss some specific failure cases.

In our project, the average duration of a complete control loop is approximately 57 seconds, with the majority of this time consumed by the car's movement towards the object. This prolonged duration is primarily attributable to the following factors:

- **Battery**: The battery is not stable during the experiment. The output voltage fluctuates as the battery depletes, resulting in inconsistent power delivery to the motors. This instability can impede the vehicle's ability to maintain smooth motion at high speeds.
- **Motor**: The TT motors employed are insufficiently powerful for the vehicle's load requirements. The vehicle was not initially designed to support heavy loads, such as a robotic arm. During experiments, the added weight of the arm significantly lowered the vehicle's front, causing the rear motor to idle and the front motor to stall. Under these conditions, the motors fail to achieve high speeds.
- **Speed Measuring System**: The slotted disc mounted on the motor features only 20 slots per revolution, allowing the encoder to detect a maximum of 40 pulses per revolution. Consequently, this limitation necessitates an

extension of the intervals between PID calculations, as the encoder does not acquire sufficient speed data in a brief period. The lack of precise speed measurements hinders the car's ability to control speed accurately and execute precise maneuvers at elevated speeds.

During testing, there are three major causes of failures:

- **Detection Failure**: There are instances where the trash may either be thrown out of the camera's field of view or detected with confidence below the predetermined threshold. In such cases, the vehicle remains stationary until the trash is repositioned within sight. Additionally, during the approach to the trash, if it becomes close enough to exit the camera's field of view yet remains beyond the detection range of the ultrasonic sensor, the vehicle will halt and remain in Stage 0.
- **Suction Failure**: Suction failure represents the most frequent malfunction encountered in this project. As mentioned above, we use a suction gripper on the robotic arm to pick up the trash. If the trash object cannot make seamless contact with the gripper, it cannot be picked up. Certain types of trash, such as *fruitpeel*, *plasticcup*, and *paperball*, pose significant challenges for suction-based retrieval due to their shapes and materials.
  Additionally, the gripper requires a specific pressure to achieve seamless contact with the object. However, certain materials, such as *tincan* and *plasticbottle*, can easily be displaced by the pressure applied by the gripper. Consequently, precise tuning of the end-effector's pose is essential to ensure effective pickup and prevent displacement of the target objects.
- **Localization Failure**: This type of failure frequently occurs with thin objects, such as envelopes. As previously discussed, an ultrasonic sensor is utilized to measure the distance between the object and the vehicle. Efforts have been made to position the sensor as low as feasible. However, the vehicle's potential to tilt forward during motion occasionally causes the sensor to angle slightly downward. Consequently, if the sensor is installed too low, its detection capabilities may be impeded by the floor. Given the minimal thickness of objects like envelopes, the ultrasonic sensor often fails to recognize them, resulting in the vehicle inadvertently bypassing these items.

Obviously, suction failure is the major cause of the complete task, as not all the objects can be grasped using this simple suction gripper. To accommodate this issue, some other grippers, such as parallel or jaw grippers, could be used along with grasp generators to plan more executable and robust grasps.

VI. Future Work

In our project, we comprehensively explore the feasibility of using raw RGB images as input for an autonomous litter-picking robot. With an advanced detection algorithm and closed-loop control logic, our robot has successfully performed the given tasks in our controlled environment. However, our method's generalizability remains limited. To enhance the generalizability of the detection model, it would be beneficial to gather a larger dataset that encompasses a wider variety of scenes and conditions. Additionally, incorporating depth images would improve the accuracy of measuring the distance between the object and the gripper, thus enabling more precise grasp planning. This feature was not included in our current system due to budget constraints. Extending this concept to public scenarios, a more powerful robotic arm and gripper could be employed to autonomously collect trash, reducing the reliance on human labor.

VII. Conclusion

In this project, we developed an autonomous litter-picking robot, LR-Bot, which integrates a Jetson Nano with a 4 DoF robotic arm for indoor litter-picking task supported by a 4WD car base. We designed a comprehensive task logic and control loop to support its operations. A custom dataset called TrashDet, containing nine common indoor trash categories, was collected to train various detection models, including YOLOv5, YOLOv8, and DETR. These models were converted to TensorRT engines for real-time inference on the Jetson Nano, with YOLOv5 emerging as the optimal solution due to its 12 FPS inference speed and high accuracy. In real-world experiments, the LR-Bot achieved an overall success rate of 53.33% across all nine trash categories and an impressive 78.33% across the refined six graspable trash categories.

In summary, the design, detection algorithm, hardware, and control loop were all rigorously tested both individually and in combination. The results demonstrate that our robot is indeed capable of efficiently handling the autonomous indoor litter-picking task within our testing environment under an efficient workflow.

However, we also identified some limitations in our robot's design. The suction gripper struggles with objects that have uneven surfaces, and the accuracy of distance estimation could be enhanced by incorporating depth information using a depth camera. Furthermore, the potential upgrades to a more robust battery and motor, along with evaluating other grippers with different mechanical structures, could broaden the range of trash categories the robot can effectively handle. These improvements would make the LR-Bot even more versatile and efficient in tackling various litter-picking challenges and generalizing to more use cases.

Acknowledgment

We would like to extend our heartfelt thanks to Prof. Zoran Kostić for his insightful lectures, which provided the foundational knowledge necessary to complete this work. Additionally, we are deeply grateful to William Ho and Bo Yu for their exceptional organization of the labs and their invaluable advice on this project.

REFERENCES

[1] B. Singh, N. Sellappan, and P. Kumaradhas, "Evolution of industrial robots and their applications," *International Journal of emerging technology and advanced engineering*, vol. 3, no. 5, pp. 763–768, 2013.

[2] B. Yan, P. Fan, X. Lei, Z. Liu, and F. Yang, "A real-time apple targets detection method for picking robot based on improved yolov5," *Remote Sensing*, vol. 13, no. 9, p. 1619, 2021.

[3] K. Price, J. Peine, M. Mencattelli, Y. Chitalia, D. Pu, T. Looi, S. Stone, J. Drake, and P. E. Dupont, "Using robotics to move a neurosurgeon's hands to the tip of their endoscope," *Science Robotics*, vol. 8, no. 82, p. eadg6042, 2023.

[4] J. Duan, S. Yu, H. L. Tan, H. Zhu, and C. Tan, "A survey of embodied ai: From simulators to research tasks," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 6, no. 2, pp. 230–244, 2022.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[6] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[7] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[8] I. d. L. Páez-Ubieta, J. Castaño-Amorós, S. T. Puente, and P. Gil, "Vision and tactile robotic system to grasp litter in outdoor environments," *Journal of Intelligent & Robotic Systems*, vol. 109, no. 2, p. 36, 2023.

[9] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, K. Michael, TaoXie, J. Fang, imyhxy, Lorna, Z. Yifu, C. Wong, A. V, D. Montes, Z. Wang, C. Fati, J. Nadar, Laughing, UnglvKitDe, V. Sonck, tkianai, yxNONG, P. Skalski, A. Hogan, D. Nair, M. Strobel, and M. Jain, "ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation," Nov. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7347926

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[12] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," Jan. 2023. [Online]. Available: https://github.com/ultralytics/ultralytics

[13] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "Yolov9: Learning what you want to learn using programmable gradient information," *arXiv preprint arXiv:2402.13616*, 2024.

[14] N. Minorsky., "Directional stability of automatically steered bodies," *Journal of the American Society for Naval Engineers*, vol. 34, no. 2, pp. 280–309, 1922. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1559-3584.1922.tb04958.x

[15] A. O'dwyer, *Handbook of PI and PID controller tuning rules*. World Scientific, 2009.

[16] R. E. Kalman *et al.*, "Contributions to the theory of optimal control," *Bol. soc. mat. mexicana*, vol. 5, no. 2, pp. 102–119, 1960.

[17] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012.

[18] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[19] D. Driankov, H. Hellendoorn, and M. Reinfrank, *An introduction to fuzzy control*. Springer Science & Business Media, 2013.

[20] O. Khatib and B. Siciliano, *Springer handbook of robotics*. Springer International Publishing, 2016.

[21] S. Bruno, S. Lorenzo, V. Luigi, and O. Giuseppe, "Robotics: modelling, planning and control, 2010," *Cited on*, vol. 1, 1994.

[22] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 390–391.

[23] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8759–8768.

[24] R. Xu, H. Lin, K. Lu, L. Cao, and Y. Liu, "A forest fire detection system based on ensemble learning," *Forests*, vol. 12, no. 2, p. 217, 2021.

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[27] e. Xinyu Wang, "tensorrtx," https://github.com/wang-xinyu/tensorrtx, 2021.

[28] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2018.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

APPENDIX

A. Data Collection

To simplify our testing environment, we conducted the experiments in the living room of our apartment. All training samples were captured within this environment to minimize the domain gap between training and testing. Initially, we collected 453 images to train our model, but we found the detection results were vulnerable to environmental changes, such as variations in background and lighting conditions. This was primarily because the training samples were collected with a fixed background and camera angle. To enhance the model's performance and generalizability, we collected an additional 582 images from diverse backgrounds and camera angles. Eventually, we created the TrashDet dataset and achieved excellent detection results with our model. Some instances from our dataset are shown in Fig. X. We used Roboflow for data annotation and dataset export, generating datasets in formats compatible with YOLOv5 and YOLOv8 as well as the standard COCO [26] format for DETR. Our TrashDet dataset is publicly available at https://universe.roboflow.com/trashdetection-ubntx/trashdet Some samples of our TrashDet are depicted in Fig. 9.

B. Implementation Details of YOLOv8 and DETR

1) YOLOv8: For this experiment, we used YOLOv8s model. The experiment stopped early after 365 epochs due to a lack of improvement for 100 consecutive epochs. For training, we used an initial learning rate of 0.01 with a weight decay of 0.0005. The batch size was 16, and AdamW [28] served as the optimizer. The final overall results are presented in Table IX, and the training plot is shown in Fig. 11. The training took approximately 1.5 hours on an RTX 4070 Ti. Notably, the model achieved a mAP50-95 that is 1.6% higher than YOLOv5, which was expected.

2) DETR: We chose DETR with a ResNet50 [29] backbone to accommodate the Jetson Nano. The structure of the transformer is illustrated in Table X. For finetuning this model, we used a learning rate of 0.0001 and a weight decay of 0.0001. AdamW [28] was employed for optimization, and we clipped the gradients at 0.1 during training. The model achieved a mAP50-95 of 0.89, so we stopped training after 300 epochs. The results on the TrashDet validation set are shown in Table XI.

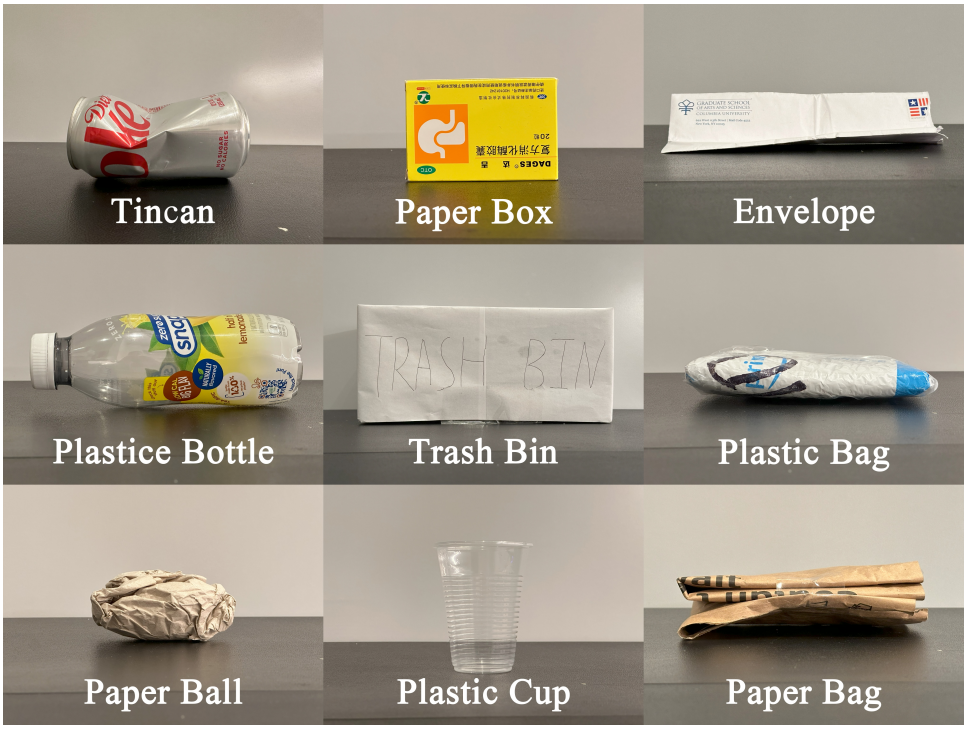

Fig. 9. Some samples in TrashDet dataset.

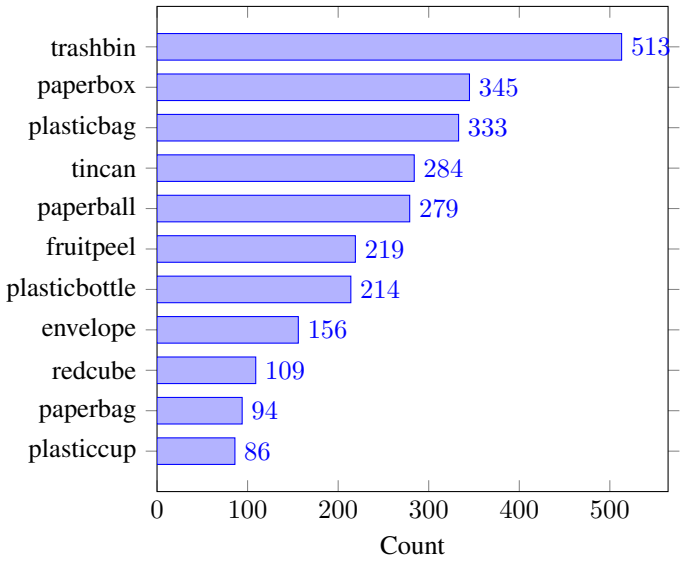

Fig. 10. Distribution of labelled instances in each class.

TABLE IX
EXPERIMENT RESULTS OF FINETUNED YOLOV8 MODEL ON TRASHDET VALIDATION SET, WITH REGARD TO PRECISION, RECALL, MAP50 AND MAP50-95.

Class	Instances	P	R	mAP50	mAP50-95
All	497	0.980	0.983	0.988	0.949
Envelope	31	0.965	0.968	0.976	0.911
Fruitpeel	39	0.994	1.000	0.995	0.946
Paperbag	16	1.000	0.992	0.995	0.969
Paperball	67	1.000	0.970	0.991	0.929
Paperbox	55	1.000	0.998	0.995	0.980
Plasticbag	54	0.908	0.981	0.979	0.941
Plasticbottle	49	0.956	1.000	0.995	0.939
Plasticcup	22	1.000	0.923	0.973	0.901
Redcube	15	1.000	1.000	0.995	0.991
Tincan	52	0.972	0.981	0.982	0.947
Trashbin	97	0.986	1.000	0.993	0.987

TABLE X
TRANSFORMER HYPERPARAMETER SETTING.

Parameter	Value
`enc_layers`	6
`dec_layers`	6
`dim_feedforward`	2048
`hidden_dim`	256
`dropout`	0.1
`nheads`	8
`num_queries`	100

C. Some Improvements for TensorRT Inference

At the outset of our experiment, we initially attempted to craft a TensorRT inference script from scratch. However, the inference speed of our YOLOv5s model was merely 1.8 FPS. We subsequently explored two different GitHub implementations: one by OpenJetson and another that utilizes Docker by alxmamaev. Unfortunately, the implementation from these two methods still has limited inference speed. Ultimately, we adopted a highly-starred repository and successfully integrated our YOLOv5s model using a multi-threading approach pro-

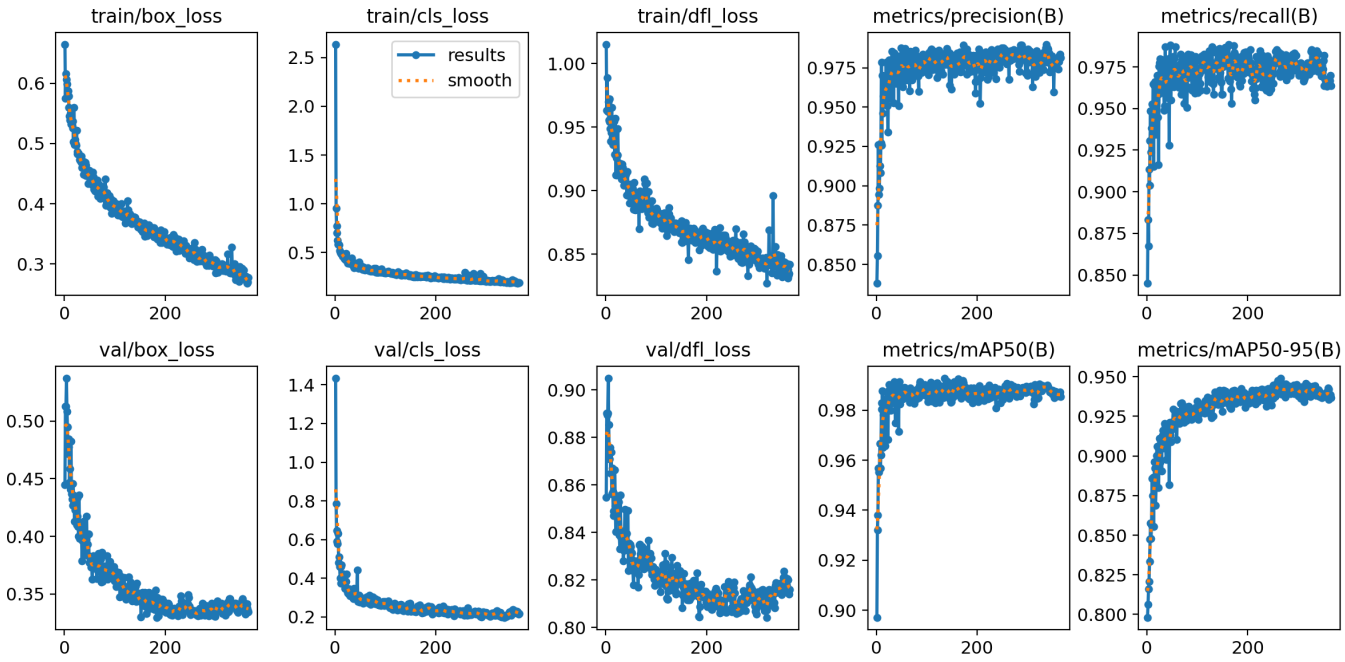

Fig. 11. Training plots of YOLOv8 model during 365 epochs.

TABLE XI
EXPERIMENT RESULTS OF FINETUNED DETR W.R.T. AVERAGE PRECISION AND RECALL OVER DIFFERENT IOU AND AREA CONDITIONS.

Metric	IoU	Area	maxDets	Value
AP	0.50:0.95	all	100	0.898
AP	0.50	all	100	0.977
AP	0.75	all	100	0.970
AP	0.50:0.95	small	100	0.574
AP	0.50:0.95	medium	100	0.829
AP	0.50:0.95	large	100	0.939
AR	0.50:0.95	all	1	0.850
AR	0.50:0.95	all	10	0.924
AR	0.50:0.95	all	100	0.924
AR	0.50:0.95	small	100	0.700
AR	0.50:0.95	medium	100	0.869
AR	0.50:0.95	large	100	0.953

vided by [27].

D. PID Parameter Tuning

TABLE XII
PID PARAMETER SETTINGS.

Parameter	Value	
	Left	Right
Proportional	20	30
Integral	1	
Derivative	1	
	Lower Bound	Higher Bound
Integral Saturation	-20	60
Duty	0	100

PID parameter settings are shown in Table XII. The velocity is computed at intervals of 200 ms. Due to the limited number of pulses received from the slotted disc attached to the motor, the maximum achievable speed ranges between 7 and 8 units. When the proportional gains are too low, the PID controller cannot achieve a 100% duty cycle. Although the integral component can help reach full duty, the imprecision in speed measurement often hampers the ability of the integral term to adequately decrease, leading to excessive motor speeds even when the target velocity is lower, thereby introducing significant system latency. Consequently, it is imperative to set a sufficiently large proportional gain. The higher proportional gain on the right motor compensates for its lower strength compared to the left motor.

Furthermore, limits on the integral term are essential; we have set the upper saturation limit to 60 and the lower limit to -20. This configuration prevents any further increase in the integral term once it has assisted in achieving a 100% duty cycle. The lower limit serves a similar purpose, preventing the integral term from contributing negatively when it is not needed.

We also implement an integral reset mechanism. Whenever the actual speed aligns with the target speed, the integral term is reset to zero. This adjustment ensures that the integral term does not remain elevated after the speed has been appropriately regulated.

E. Motor Speed Tuning

As described in the paper, YOLOv5 generates bounding boxes, whose centers are compared with the center of the

TABLE XIII
SPEED PARAMETER SETTINGS.

Parameter	Value	
	left	right
Upper Bound	0.4	
Goal Speed	5	3
Middle Bound	0.2	
Goal Speed	3	2
Lower Bound	0.1	
Goal Speed	2	2
Minus Upper Bound	-0.4	
Goal Speed	1	4
Minus Middle Bound	-0.2	
Goal Speed	1	5
Minus Lower Bound	-0.1	
Goal Speed	2	5
Straight Goal Speed	2	3

image to determine the angular adjustment required for the vehicle. To enhance the precision of these adjustments and prevent the vehicle from over-rotating and potentially losing sight of the target, we have established six distance thresholds. The corresponding goal speeds are adjusted based on these thresholds, as detailed in Table XIII.

A positive distance indicates that the object is located to the right of the image center, necessitating a rightward turn to align the vehicle's orientation with the object. Conversely, a negative distance implies that the object is to the left, requiring a leftward turn.

F. End-Effector Pose Tuning

TABLE XIV
END-EFFECTOR PARAMETER SETTINGS.

Trash Name	Y Coordinates Adjustment	Z Coordinates Adjustment
Envelope	30	240
Fruitpeel	10	220
Paperbag	50	235
Paperball	5	220
Paperbox	10	220
Plasticbag	10	220
Plasticbottle	6	185
Plasticcup	10	220
Redcube	10	220
Tincan	20	220
Trashbin	10	220

The end-effector pose tuning parameters are shown in Table XIV. The Y coordinates represent how long the arm should stretch to pick the object. The best pick point is the center of the object. But the length of the objects is not the same. For example, the arm needs to fetch farther to reach the center of *paperbag*, but if the arm stretches the same distance when it picks a *tincan*, it may miss the pick. So, this coordinate should be different for different objects.

The Z coordinates are the depth at which the arm picks up an object. The height of the objects is different. If the arm goes down too much, the head of the car may be pushed up, which influences its pose. If the arm goes down too little, it may not reach the object.

G. Contribution

The authors contributed equally to this project. The detailed contributions are summarized in Table XV.

TABLE XV
DETAILS ON CONTRIBUTION.

Task	Tianyi Xu	Yuan Qing
Idea Formulation	✓	✓
Proposal	✓	✓
Robot Design	✓	✓
Data Collection	✓	✓
Data Labeling	✓	✓
Test Environment Setup	✓	✓
Detection Algorithm&Experiments		✓
Robot Assemble	✓	
Control Loop Design	✓	✓
Control Parameter Finetuning	✓	
On-site Experiments	✓	✓
Demo Video	✓	
Final Presentation	✓	✓
Report	✓	✓
Github Repo Structure		✓