University of Southern Denmark
The Maersk Mc-Kinney Moeller Institute
Master of Science in Engineering
Advanced Robot Systems

Master Thesis

Making WALL-E

An autonomous litter collector

Developed by:
Isabella Mia Lin Nielsen 4123993
Nikolas Neathery 1234567

Supervisor:
Aljaz Kramberger,
alk@mmmi.sdu.dk

June 4, 2025

Character count: 105438

Abstract

This thesis presents the development and evaluation of an outdoor autonomous robot designed to detect and collect litter in public environments such as sidewalks, parks, and open fields. The system comprises a mobile robot base, a custom-designed trash collection mechanism, and a vision-based object detection pipeline, all integrated through the Robot Operating System (ROS) to enable modularity and real-time communication between subsystems.

The drivetrain was rigorously tested across various terrain types, including asphalt, grass, pavement, and a 20°incline to evaluate its ability to traverse different terrains. Results demonstrate reliable locomotion performance under moderate environmental variation, supporting the robot's capacity for autonomous navigation in urban outdoor settings, although with a low tolerance for high levels of vibrations.

A dedicated collection system was developed to physically retrieve trash items detected by the vision system. This subsystem was evaluated through controlled tests, validating its ability to consistently capture objects of diverse shapes and materials. The system proved reliable in repeated attempts and ended with a collection success rate of 98.4% among 15 different trash objects.

The perception component is based on a deep learning model trained to detect litter using an onboard camera. The vision performance was assessed using the F1 score across varying confidence thresholds, and the mean average precision score (mAP) at a confidence threshold of 50%. The model achieved a maximum F1 score of 0.88 at a confidence threshold of ~ 0.259, indicating a well-balanced trade-off between precision and recall suitable for real-time deployment. The system ensures high detection rates while maintaining a low false-positive rate, which is essential for collection reliability. The model also achieved a mAP@00.5 of 93.7%, showcasing a strong performance for a model trained in a similar environment as the deployed model.

The integration of these subsystems via ROS provides a scalable and adaptable platform for further development. Overall, the results demonstrate the feasibility of deploying a fully autonomous trash-collecting robot in real-world outdoor environments. Future work includes hardware improvements for an increased robustness under vibrations, a fully autonomous exploration and park upkeep, and a safety program, ensuring safe operations during deployment.

Contents

Chapter 1

Introduction

1.1 Background and Motivation

Litter management has become a significant global challenge affecting urban and natural environments with more than 7.5 million tons of trash ending up as litter each year [1]. Public spaces such as parks, streets, and waterways are frequently contaminated with litter, posing environmental hazards and requiring continuous manual cleanup efforts. Traditional litter collection methods rely heavily on human labor, which is costly and time-consuming.

With advancements in robotics and artificial intelligence, there is a growing opportunity to automate litter collection using autonomous robots. By integrating computer vision, navigation algorithms, and robotic manipulation, an autonomous litter-collecting robot can improve the efficiency of waste management, reduce human workload, contribute to cleaner public spaces, and save taxpayers millions each year.

1.1.1 State of the Art

Manual litter collection remains the predominant method for maintaining cleanliness in urban and park environments. Municipalities deploy sanitation workers equipped with tools such as brooms, grabbers, and mechanical sweepers to remove waste from streets and public spaces. While effective in certain contexts, these methods are labor-intensive, time-consuming, and often limited in coverage, particularly in less accessible areas like park edges and narrow walkways. Additionally, they pose health and safety risks to workers and are less responsive to real-time littering events.

To enhance efficiency, many cities are using semi-automated systems for bigger paved areas. Mechanical street sweepers, equipped with rotating brushes and suction mechanisms, are commonly used to clean road surfaces. However, these machines are typically large, consume significant fuel, and are less effective in areas with obstacles or uneven terrain. Moreover, they often require human operators and are not designed for dynamic litter detection or collection. These machines are usually only used to sweep wide streets, remove leafs from roads and to clean up festival areas.

With operators and manual workers, cities are spending a lot of money cleaning the communal outdoor areas. With the recent advancement within robotics and AI, many governments are starting to look into using these technologies to further improve environment cleaning. This has led to the emergence of autonomous litter collection research. A key challenge here is the integration of computer vision, robotic manipulation, and robot mobility to detect and collect litter with minimal human intervention.

The Outdoor Autonomous Trash-Collecting Robot (OATCR) [2] is one of such research. They built a mobile robot that utilized a YOLOv4-Tiny deep learning model for real-time object detection. Their robot used a mechanical flap to scrape trash into a bucket, although not tested in a real-world setting it showed promising results in simulation.

Similarly, a study by UC Santa Cruz demonstrated the use of the ResNet50 Convolutional Neural Network (CNN) for trash detection in parks, attaining an accuracy of 94.52%. They deployed a

mobile robot with a rake-lift attached that, that successfully picked up 80% of litter in a grass park [3].

Another study by the the University of Alicante [4] used a prebuilt mobile robot BLUE with a UR5e robtics manipulator attached to the mobile platform, creating a mobile manipulator. Their solution used a ROBOTIQ gripper with DIGIT tactile sensors that were used to pick up litter detected using an RGBD camera. They achieved a 94% mAP of detecting litter, and a 80% collection success rate in outdoor environments.

1.1.2 Current Limitations

While all of these studies show successful litter collection, they are all focused on litter around the size of plastic bottles and pizza trays. This leaves a big gap in environmental cleanup capabilities, as smaller trash are also not picked up by manual workers, leaving environmental pollution that potentially harm animals and our planet.

Other limitations on diverse litter collection, challenges remain with deploying autonomous robot solutions for litter cleanup.

Firstly there is the *"Environmental Variability"*, such as changing weather conditions, lighting, and surface types can vastly affect sensor performance and object detection accuracy, making it difficult to create robust autonomous solutions. Additionally, litter appears in all different shapes and sizes, as they get deformed and affected by their environments, making it challenging to accurately detect old litter in natural environments.

Secondly there is *"Power Constraints"*, where battery life remains a limiting factor, especially for robots operating over extended periods in outdoor environments.

And lastly, deploying these solutions at scale for effective environmental cleaning, a key problem remains with *"Cost and Scalability"*. Autonomous robotic solutions have a high development and maintenance cost, which may hinder widespread adoption, particularly in resource-constrained towns and districts. Here a key to commercializing a robotic solution for environmental cleanup is to ensure that the cost of the solution is competitive with the cost of the current manual labor. As robotic solutions always have a higher upfront cost, ensuring that the solution is robust and long lasting is detrimental to keep the long-term costs low.

To address these challenges, this thesis develops and builds a robot solution from scratch, that can detect and collect litter in natural environments, nicknamed WALL-E.

1.2 Problem Statement

This thesis proposes the design and development of an autonomous robot capable of detecting, localizing, and collecting a vast variety of litter in outdoor environments.

1.3 Objectives

The main objective of this thesis is to develop a fully functional autonomous litter-collecting robot. Specifically, the project aims to:

- Design and construct a robotic platform capable of navigating outdoor environments.
- Develop a perception system using computer vision to detect litter.
- Implement autonomous navigation to move efficiently in unstructured terrain while avoiding obstacles.
- Design and integrate a collection mechanism to pick up and store litter.
- Test and evaluate the robot's performance through experiments in real-world conditions.

The robot tests and evaluations will be measured against predefined project criteria, as defined below in subsection 1.3.1.

1.3.1 Project Criteria

The project criteria have been divided into sections based on the subsystems encompassing the robot WALL-E. Each system must be thoroughly tested to ensure the quality required by a commercial product can be delivered. The subsystems of the project consist of (i) the mechanical robot, (ii) the collection system, (iii) trash detection, and (iv) navigation. The capabilities of each of these elements will be measured and must meet the criteria as defined in Table 1.1.

Subsystem	**Success Criteria**
The Mechanical Robot	The robot can successfully (i) drive on a 20% incline, and (ii) drive on asphalt, grass, and pavement.
Collection System	The robot can successfully pick up (i) paper, (ii) cardboard, (iii) soft plastic, (iv) hard plastic, (v) compressed metal, (vi) metal cylinders, (vii) glass shards, (viii) glass bottle, and (ix) thread with a 95% success rate on pavement, asphalt and grass.
Trash Detection	The robot's vision system can recognize (i) 90% of litter in a distance within one to five meters of the robot, and (ii) can run inference in real-time.
Navigation	The robot can (i) path plan in real-time, (ii) avoid collisions (static and dynamic), (iii) fully explore a defined area.

Table 1.1: Success criteria for WALL-E

1.4 Thesis Structure

The thesis is divided into chapters for each development discipline (hardware and software) which is it's own sub-thesis in itself, complete with its own experiments, evaluations, and conclusions. After these chapters, there will be an overall discussion and conclusion, going through the sub-thesis, and discussing and concluding upon the full robot solution.

Chapter 2 is about the hardware of WALL-E and describes the full robot design, the electronics in the system, the trash collection ability, and the robot's durability. Chapter 3 is the robot's software, explaining both about WALL-E's litter detection and on it's ability to drive autonomously. Lastly, chapter 4 and chapter 5 reflects upon the full system in a discussion and conclusion, respectively.

Chapter 2

Hardware

2.1 Drivetrain

The goal of this project is to create a robot that is able to traverse public outdoor areas and remove litter from those areas. To accomplish the task of litter collection, the robot must be able to handle various types of terrain. Terrain that is not only different materials like concrete dirt or grass but also at different incline levels and marked with obstacles such as tree roots and branches. The robot needs to be able to traverse these obstacles in an efficient manner.

A drive train is what provides mobility or motion to the system. It is composed of components like motors and gearboxes that are used to transfer torque to the wheels, resulting in the robot motion. In this case the drive train is responsible for maneuvering a trash manipulator through the environments mentioned.

2.1.1 Design

To design the robot drivetrain, the use case and environment needed to be taken into account; specifically, the terrain. An overview of the drivetrain design can be seen in Figure 2.2.

Frame

Due to the nature of the uneven terrain, omnidirectional methods like mechanum wheels or omni wheels, which are very maneuverable on flat surfaces, are not a viable option for WALL-E. These methods can't handle uneven terrain very well and often get stuck. Methods like tank treads seen in figure 2.3 and larger pneumatic wheels, however, are well suited for uneven and difficult terrain.

Tank treads distribute the robot's weight more evenly across a larger surface area, enhancing traction and stability over soft or uneven ground. Its continuous contact with the terrain also prevents slippage and improves obstacle climbing capabilities. Similarly, large pneumatic wheels offer shock absorption and adaptability over rocks, mud, and debris, reducing the risk of immobilization.

Another critical consideration for all-terrain navigation is **high-centering**, to prevent a situation where the robot's undercarriage becomes stuck on an obstacle, lifting its wheels or tracks off the ground, limiting its movements. Avoiding high-centering requires strategic design considerations, such as increased ground clearance, proper weight distribution, and perhaps the use of articulated suspension systems that allow the drivetrain to conform to the terrain dynamically. In the case of this system, a raised center was chosen.

In order to help prevent getting the center stuck on obstacles like roots or rocks, the center of the robot along the width was raised. This is illustrated in Figure 2.4. The front and back of the drive train are raised by the wheels when passing over obstacles, and the raised center ensures that once the wheels come off the obstacle, the center does not end up getting stuck on the obstacle. However, as our raised center design is only along the width of WALL-E, the robot can still get stuck on the front or the back of the robot, if the obstacle is smaller than the width of WALL-E. To prevent this

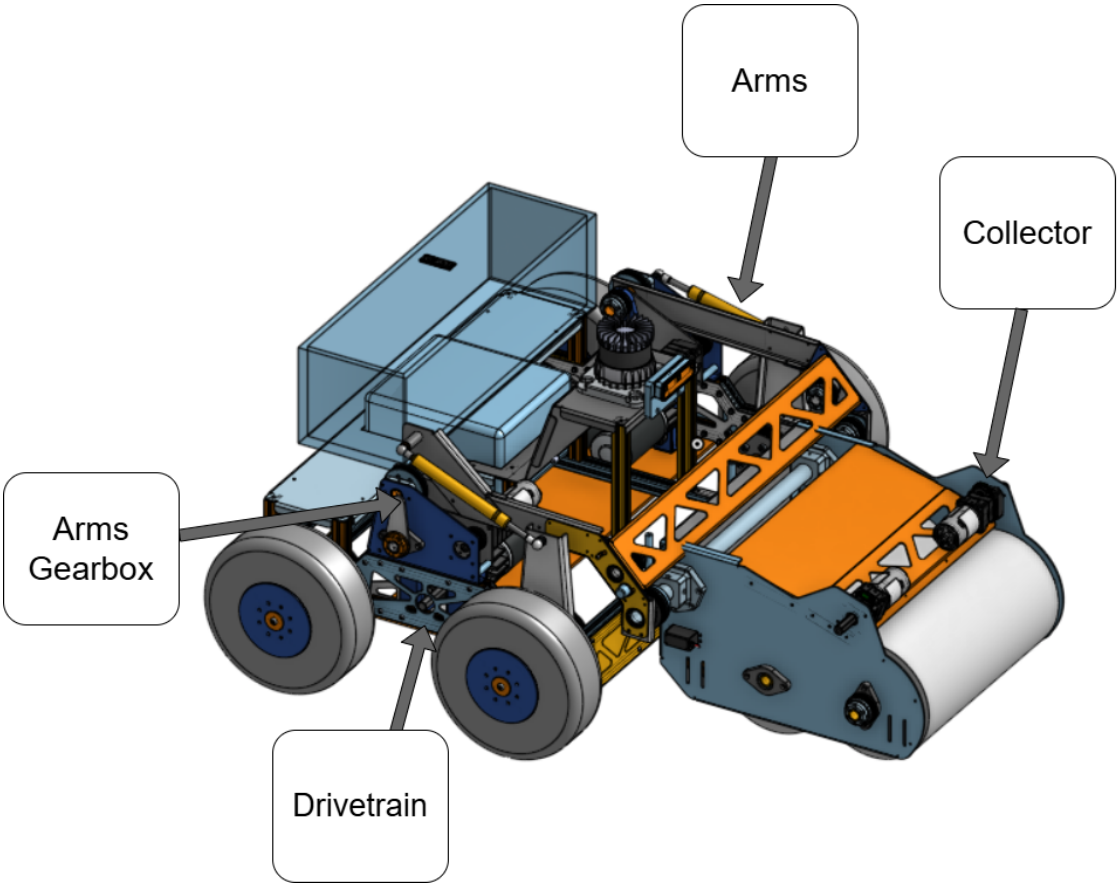


Figure 2.1: Full Robot Model

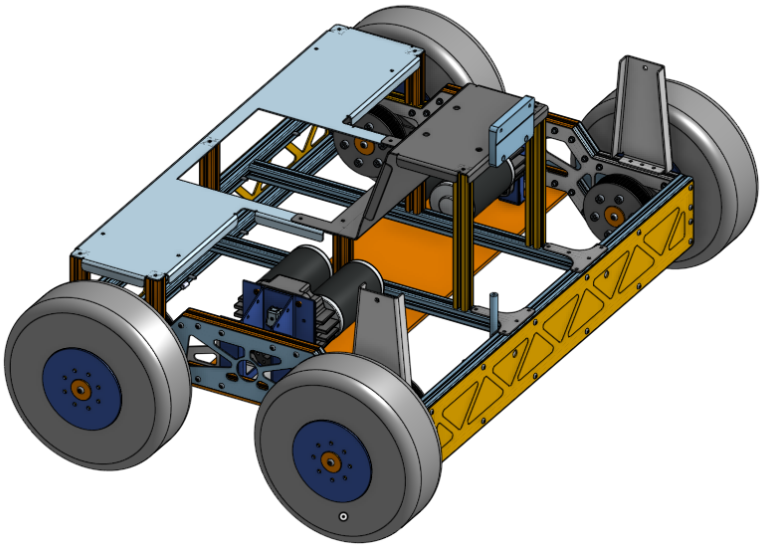


Figure 2.2: Drivetrain Made From Sheet Metal and Extrusion

Figure 2.3: Tank Treads on Differential Drive from Hackaday[5]

in the future, the front and the back could be raised in a similar manner as the left and right rails of the robot.

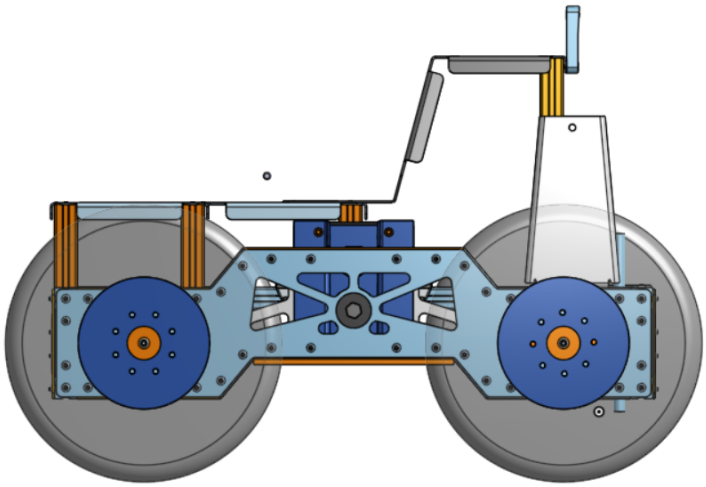


Figure 2.4: Raised Center Drive Train

Torque Transfer

This system leverages a belt and pulley system to transfer the torque output from the two gearboxes to the left and right sides of the robot. HTD 5 profile belts were chosen due to ease of access, design, and ability to handle higher loads than profiles like GT2. The pulleys themselves were 3D printed, making it easy to swap in the case of needing to change something in the future. A 3D-printed pulley by itself would not last in the harsh outdoor environment. To add strength to the pulley 3 millimeter steel plate is mounted to the outer faces of the pulleys via a bolt pattern. These steel plates also prevent the axle from turning the hex profile of the pulley into a round profile by rounding it out, rendering the pulley useless. The axle is 20 millimeter round stock that has precision milled flats to match the pulley plates and the wheel plate hubs as seen in Figure 2.5. The torque is transferred from the axle to the wheels via two additional steel plates and 3D printed inserts that are mounted to the steel plate and fit into the gaps of the wheel itself. The two plates are mounted on either side

of the wheel, sandwiching it, as seen in Figure 2.6.

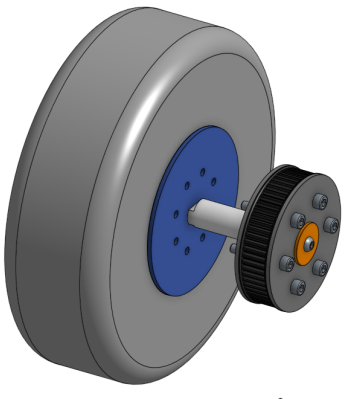


Figure 2.5: Wheel and Pulley Assembly

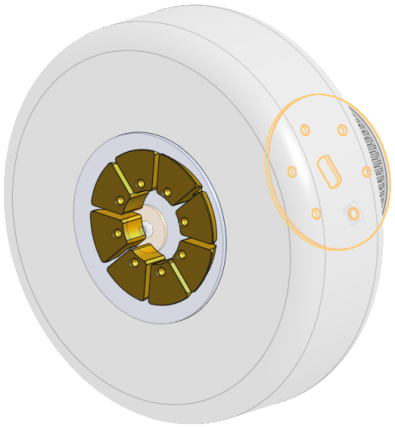


Figure 2.6: Wheel Inserts

Manufacturing and Assembly

The system was designed to leverage the use of a large CNC plasma cutter and several sheet metal bending tools, such as bench breaks and shears. The structural frame of the robot is constructed from 6061 aluminum extrusions, a material selected for its favorable balance of strength, low weight, and ease of cutting. It provides sufficient rigidity for outdoor operation while maintaining a relatively low mass, which is critical for energy efficiency and maneuverability. The use of T-slot profiles enables modularity in the design, allowing for straightforward assembly and future expansion or reconfiguration using standard T-slot fasteners. This modular approach supports iterative development and simplified the integration of additional subsystems as the platform evolved. The extrusion is held together via 2 millimeter anodized steel plates. These plates line the entire frame, adding rigidity and mounting points for the gearboxes and wheel bearing blocks. These plates are then mounted to the extrusion using the t-slot nuts, acting as brackets for the frame.

Using sheet metal also reduced the manufacturing time. Parts of similar size, if done on a machine like a CNC router or mill, would require hours to finish, while the plasma cutter can cut them in minutes. This makes prototyping far more efficient. The only downside is that the parts need to be bent in the post-process on non-NC equipment, reducing the accuracy and precision of the parts. However, for a proof of concept, rapid prototyping is essential.

2.2 Collector

The collector comprises two main components: The arms and the trash manipulator. The arms serve a couple of purposes, most notably providing passive shock absorption for the manipulator as it moves over uneven terrain. This compliance reduces the likelihood of the manipulator becoming obstructed by minor surface irregularities such as ridges or small inclines, thereby enhancing the system's overall mobility and robustness. As illustrated in Figure 2.7, the arms are mechanically coupled to the robot's main chassis via a gearbox, with the litter manipulator located at the end of the arms.

The trash manipulator itself is inspired by conventional street cleaning mechanisms and incorporates two large counter-rotating roller brushes. This design facilitates effective litter collection by directing litter toward the center between the two brushes, allowing the object to be pushed upwards into the collector storage. The dual-brush configuration is intended to accommodate a wide spectrum of litter types, including varying sizes, shapes, and materials, ensuring reliable operation across diverse outdoor environments.

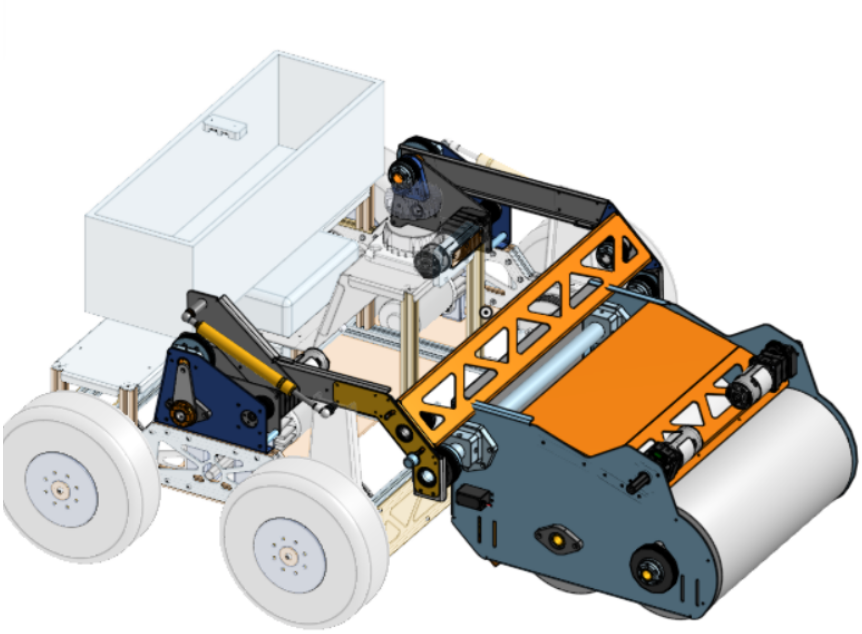


Figure 2.7: Trash Manipulator and Arms Model

2.2.1 Design

The design of the collector comprises two parts: the arm and the actual collector. The collector is what collects the litter, consisting of the two brushes, and the arm is used to hold and move the collector.

Arms

The arms serve two functions: The first being to lift the collector over the storage bin when the collector is full, and the second being to act as a shock neutralizer so the collector can conform to the contours of the ground.

The arms are powered by two 775 pro brushed motors with a 200:1 reduction. The two 775 motors drive a multi-stage planetary gearbox. This planetary is made up of three stages as shown in Figure 2.8. The first and second stages are both 7:1, followed by a 3:1. The output of this gearbox is then transferred to the arm via a belt and pulley system with a 22-tooth pulley directly on the output shaft and a driven 40-tooth pulley that is fixed to the arm.

A significant design constraint arose from the thermal limitations of the 775 Pro motors. These brushed DC motors rely primarily on integrated fans for active cooling during operation. When stalled, however, the fans cease to rotate, and due to the limited surface area and minimal passive cooling features, heat rapidly accumulates, posing a risk of thermal damage. This poses a challenge for the arm mechanism, which must support the collector at arbitrary angles for extended periods without continuous power draw.

To address this issue, a passive counterbalance system utilizing gas pistons was implemented. Gas pistons are linear actuators that apply a constant unidirectional force, functioning similarly to mechanical springs. Internally, they contain a pressurized gas, typically nitrogen, which exerts a consistent force on the output shaft across its stroke. These actuators are available in a range of sizes and force ratings.

In this application, the gas pistons are configured to counterbalance the gravitational torque exerted by the collector. In order to do so, a gas piston pair is chosen based on the weight of the collector at the end of the arm and the length of the arm. Knowing this, it is possible to make an

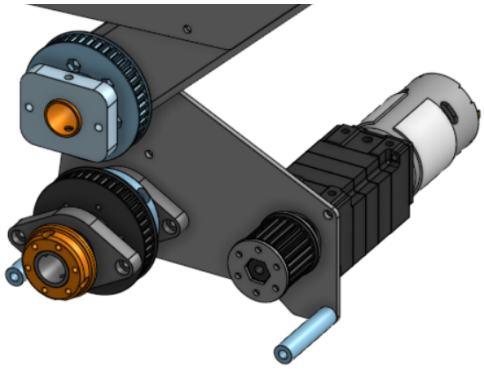


Figure 2.8: One of two Arm Gearboxes

estimate for the force required to counterbalance the weight of the collector at the end of the arm. The result of these equations can be seen below.

Gas Piston Force Calculation

The list below shows our variables used for calculating the required force contributions from the gas pistons:

- Mass: $m = 8$ kg
- Arm length: $d = 0.4$ m
- Gravity: $g = 9.81$ m/s^2
- Piston distance from pivot: $d_{\text{piston}} = 0.2$ m
- Number of pistons: $n_{pistons} = 2$

Using these values, the required force from each of the gas pistons (F_{piston}) can be calculated using Equation 2.1, where τ is the torque around the pivot; calculated as shown in equation 2.3.

$$n_{pistons} \cdot F_{piston} \cdot d_{piston} = \tau \tag{2.1}$$

To find τ, the weight force must first be calculated:

$$F_{\text{weight}} = m \cdot g = 8 \cdot 9.81 = 78.48 \text{ N} \tag{2.2}$$

Using the weight force, the torque around the pivot can be found:

$$\tau = F_{\text{weight}} \cdot d = 78.48 \cdot 0.4 = 31.39 \text{ Nm} \tag{2.3}$$

This gives the force required from each piston:

$$2 \cdot F_{\text{piston}} \cdot 0.2 = 31.39 \implies F_{\text{piston}} = \frac{31.39}{0.4} = 78.48 \text{ N} \tag{2.4}$$

As seen above, two gas pistons with a constant force of ~ 78.48 newtons are needed to counterbalance the weight of the collector. The force, however, will not be applied in a linear motion. The point of the pistons is to counteract the force of gravity. As the arm rotates over 90 degrees of rotation the force of gravity will now start to act on the other side of the arm or in the other direction due to the weight shift. The solution to this is to clock the arm and gas piston such that when the arm is at 90 degrees or perpendicular to the ground, it is parallel to the lever it is attached to. This means that when the arm is pointing straight up, gravity is having the least effect on the motors, making the gas piston neither helping nor hurting due to their positioning, as seen in Figure

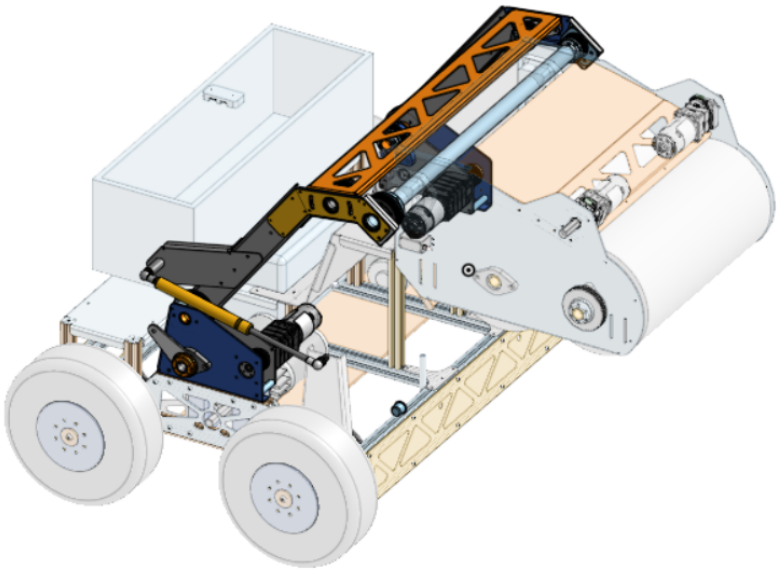


Figure 2.9: Collector Arms Supported by Gas piston

2.9. As the arm moves past this zero point, the gas pistons begin to help resist the arm from falling faster.

Lastly, the reduction needs to be chosen to make control of the arm accurate and efficient. Figure 2.10 shows the speed and position curves for the arm driven by two 775 pros at a 300:1 reduction to go from a starting angle of zero, which is parallel to the ground, to 150 degrees. These calculations were made using a tool called **ReCalc**. Figure 2.10 also shows the current draw throughout the motion of the arms arc or path of motion. Both the 775s are on 20 amp current limits and assume that the motors are spinning at max rpm. At this configuration, it is calculated that the arm can reach its position in 0.5 seconds. This assumes that there is no limit on the speed of the motor.

Another attribute of the arms is their ability to keep the collector level throughout the range of motion of the arms. This is due to a four-bar linkage that starts at the base of the arm. Using a stationary pulley at the base of the arm and a pulley that is fixed to the collector. When these two are connected via a series of belts, they will maintain the same rotation in relation to each other. This linkage can be seen in figure 2.11 where the non transparent parts are the linked pulleys. This overall helps with maintaining the collector parallel to the ground as well as not spilling trash out of the collector when lifting it to avoid obstacles or to empty it.

Collector

The collector is what is used to pick up the trash. It utilizes two large brushed rollers with bristles. These rollers spin in opposing directions and slightly overlap each other. This design is meant to mimic traditional street cleaning trucks with the hope that the large rollers with longer bristles can accommodate litter of various shapes and sizes.

The collector itself is powered by two 775 pros. The same ones that are used for the gearboxes. These motors are mounted directly to a three-stage planetary gearbox that results in a two-hundred-to-one reduction on each side. The output of these gearboxes is tied directly to the rollers via a belt and pulley system. As seen in Figure 2.12, the front roller is slightly raised above the back roller to allow for trash to go under and then be kicked up by the back roller in the collector.

The brushes themselves are made up of two half-horse scratch brushes that are designed to be placed on trees for livestock to scratch against. These two half brushes are mounted to an aluminum

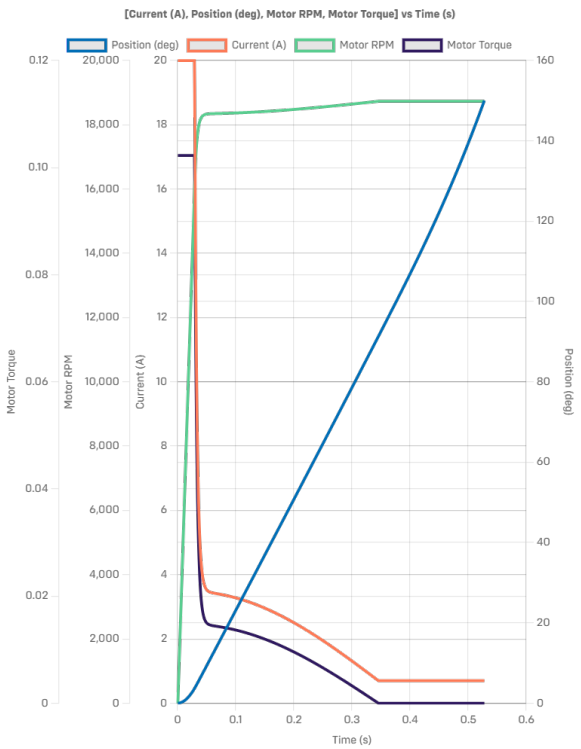


Figure 2.10: Speed and Position Curve for Arm Configuration

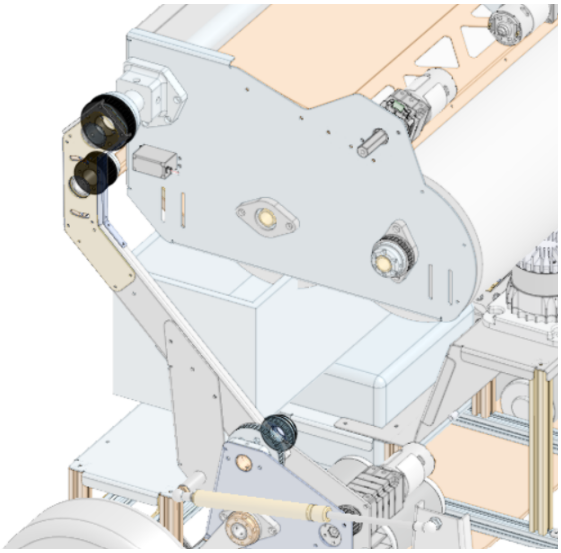


Figure 2.11: Arm Pulley Four Bar Linkage

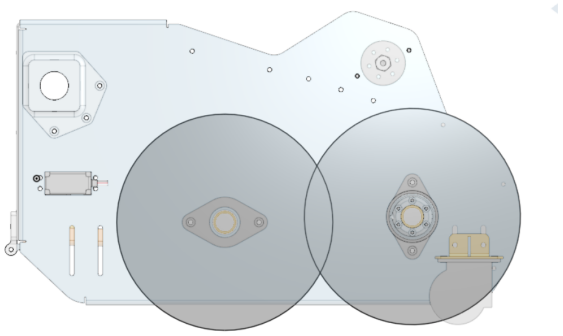


Figure 2.12: Collector Rollers

axle via 3D printed hubs and fitted into bearing housings on either side of the collector.

Once trash is collected by the two rollers, it is then scooped into a collection bin on the back of the collector. It stays in there until the collector is full, at which time it is then lifted above the trash bin on the robot and emptied. This configuration is shown in Figure 2.13.

The trash is retained in the collector by a simple trapdoor mechanism that is controlled by a lever arm on a servo. When closed, the lever is inline with the servo arm, preventing the weight of the trash from forcing the door open.

2.3 Electronics

The electrical system is a critical component of a mobile robot, responsible for distributing power, enabling sensor integration, and facilitating communication between various subsystems. This section details the custom and off-the-shelf circuits used to operate the robot effectively. It covers the power distribution network, motor controllers, sensors, and communication interfaces, along with the considerations made for key components. Together, these circuits form the backbone of the robot's ability to perceive its environment and respond with accurate, real-time motion.

2.3.1 Components

Multiple electronic components go into making a robot. Besides resistors, fuses, and wires, big electronic components like a battery, motors, motor controllers, a camera, and a LiDAR (Light Detection and Ranging) have gone into making WALL-E functional.

Battery

To choose the right battery for the robot, it is necessary to calculate how much power the other electronic components in the robot need, and for how long the robot should be able to run on one charge. Ideally, the robot can run for at least eight hours on a single charge, but as the cost of the battery also needs to be considered, this will not be possible. As a minimum to be able to run WALL-E with enough time for testing the planned functionality, the robot needs to be able to run for at least one hour on a single charge, but preferably more.

The amount of current the battery can sustain is given through its ampere hours [Ah]. The amount of ampere hours needed to sustain the current for the desired time is given by $Ah = Amps \cdot hours$.

The motors' datasheet holds information about how much current the individual motors use. This can be read through the motor curves as seen in Figure 2.14. Here, it can be seen that the motors can pull a maximum of 140 Amps, but the peak performance is delivered around 70 Amps.

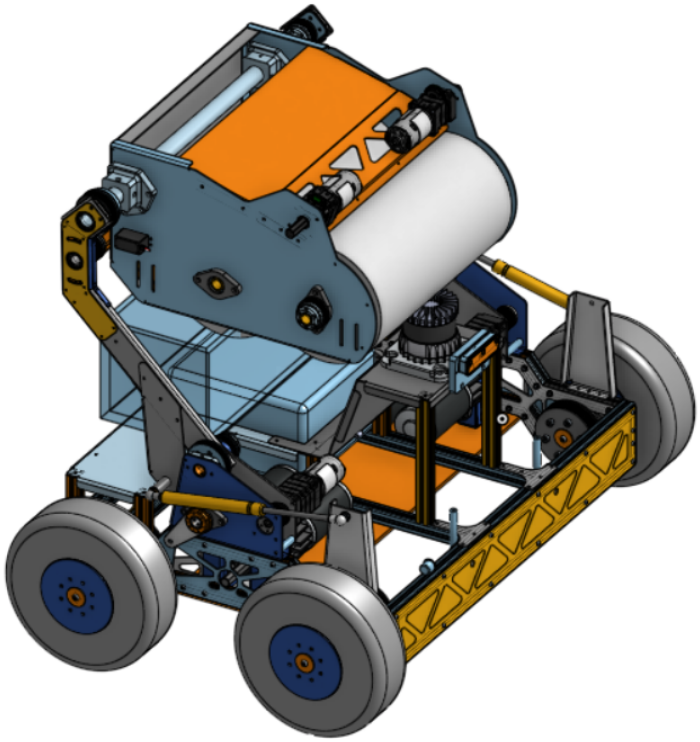


Figure 2.13: Trash Dump Configuration

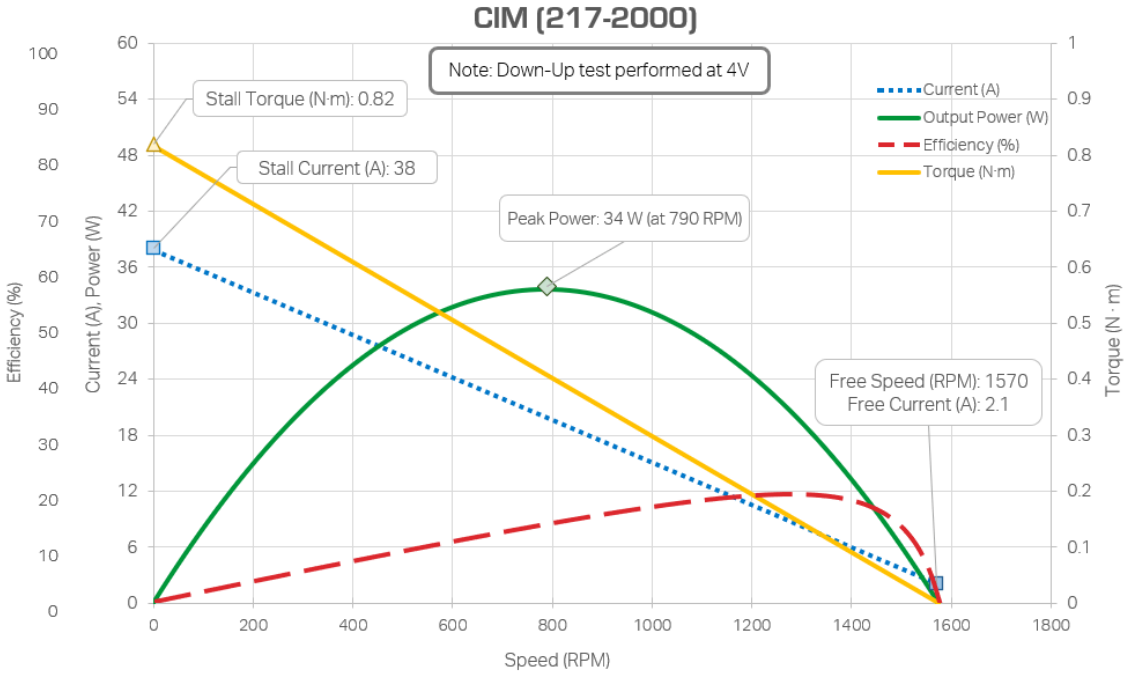


Figure 2.14: CIM Motor Curves from Vex[6]

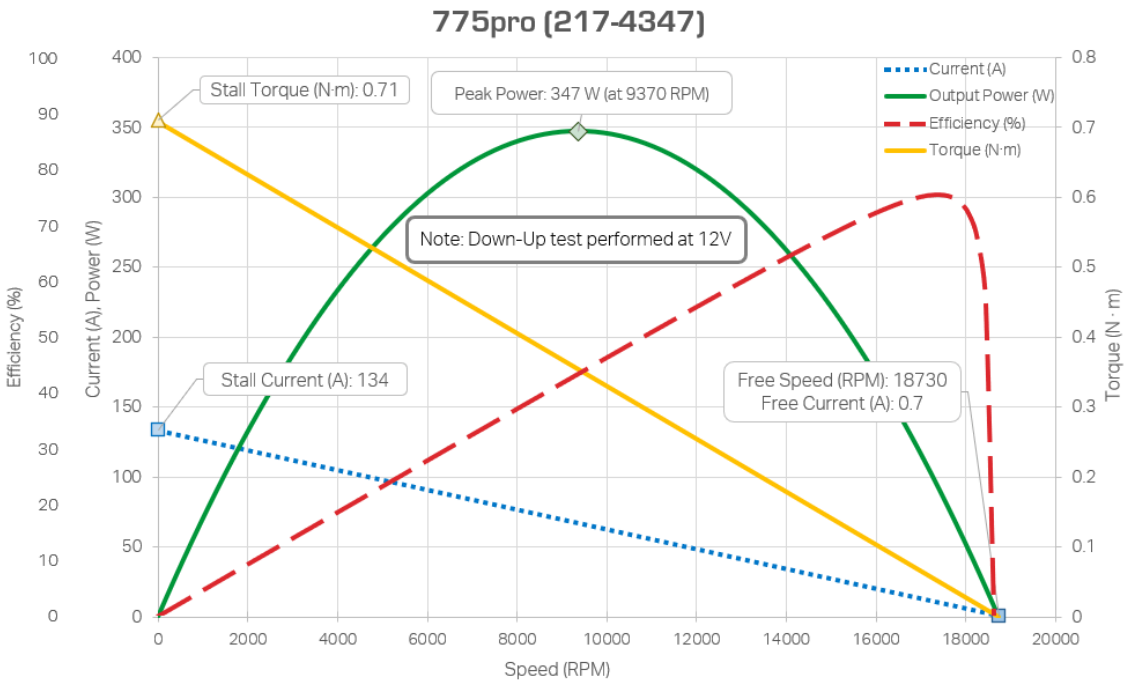


Figure 2.15: 775 Pro Motor Curve from Vex[7]

Motors

The entire robot is actuated using 12-volt DC motors of various sizes, RPMs, and torque outputs. The drivetrain takes advantage of four CIM motors. While the collector and arms utilize four 775 pro motors. The specifications for each motor can be seen in Table 2.1. The CIM motor curves for current use can be seen in Figure 2.14, and the 775 pro motor curves for current use can be seen in Figure 2.15,

Specification	CIM Motor	775pro Motor
Free Speed (RPM)	5310	18730
Weight (kg)	1.27	0.39
Stall Torque (Nm)	2.41	0.71
Free Current (A)	2.7	0.7
Stall Current (A)	131	134
Operating Voltage (V)	12	12
Max Power Output (W)	337	347

Table 2.1: Comparison of CIM and 775pro motor specifications

Controllers

This system uses VexPro Victor motor controllers, shown in 2.18, as the primary interface for driving the DC motors, both the 775 pro and the CIM models mentioned earlier. These controllers are designed for 12V operation and are commonly used in robotics due to their durability and responsiveness. They can receive control signals via either CAN (Controller Area Network) or PWM (Pulse Width Modulation).

For this project, CAN was selected over PWM primarily due to its compatibility with pre-existing libraries provided by *Cross The Road Electronics* (CTRE). These libraries simplify integration with

Figure 2.16: 775 Pro Motor

Figure 2.17: CIM Motor

Figure 2.18: Victor SPX Motor Controller

Raspberry Pi systems and offer robust tools for motor control, diagnostics, and network communication. While the advanced motion features themselves are not inherent to the CAN protocol, using CAN makes it possible to leverage CTRE's software stack, which supports high-level capabilities such as velocity and position control, current limiting, and fault monitoring.

Camera

The onboard camera used for detecting the litter is a Luxonis OAK-D Lite camera. It's a fixed-focus 13 MP central RGB camera with stereo depth using a USB-C for data and power transfer. The camera uses five Watts for optimal operations and has on-device AI and computer vision capabilities [8]. The five Watts are at the max capabilities of what a Raspberry Pi 5 can deliver through a USB port. This means that the Pi can only be connected to the camera if also applying it with power. To lower the power pull from the Raspberry Pi, the power and data are separated using a Y-adapter, making the camera get power directly from the battery, using a buck converter, which is a step-down voltage regulator.

LiDAR

To maintain accurate odometry and navigate effectively, this robot employs a LiDAR-based SLAM (Simultaneous Localization and Mapping) system. SLAM enables a robot to build a map of an unknown environment while simultaneously tracking its own position within that map.

While there are several sensor modalities that can be used for SLAM, LiDAR is often preferred for its speed, accuracy, and relatively low computational overhead. A LiDAR sensor works by emitting rapid pulses of laser light and measuring the time it takes for each pulse to reflect off surfaces and return to the sensor. This time-of-flight data allows the system to calculate precise distances to objects in the environment, generating a high-resolution 2D or 3D point cloud of the surroundings in real time.

In contrast, Visual SLAM (VSLAM) relies on camera data — often from stereo or monocular RGB or RGB-D cameras — to extract visual features (like corners, edges, or textures) and track their movement across frames. While VSLAM can be useful in GPS-denied or indoor environments and can capture rich scene information, it is often more sensitive to lighting changes, motion blur, and textureless environments. Additionally, it typically requires more complex image processing and can be more computationally demanding, particularly in real-time applications.

In the end, a LiDAR-based SLAM system was implemented using the Ouster 3D LiDAR sensor, as shown in Figure 2.19. This sensor was selected for its suitability in outdoor environments, where traditional 2D LiDAR systems may struggle due to the absence of flat surfaces or vertical structures required for consistent point cloud generation. The Ouster LiDAR's high-resolution 3D scanning capabilities allow for robust environmental perception across diverse and unstructured terrains. Furthermore, the sensor offers native integration with ROS 2, significantly simplifying

Figure 2.19: Ouster Lidar

its incorporation into WALL-E's existing ROS2-based robotic architecture, accelerating the LiDAR implementation into the autonomous navigation.

2.3.2 Circuit

The majority of devices within the robot's electrical system operate at a voltage of 12 volts. This standardized voltage level simplified power distribution, as it matched the output of the primary battery source, reducing the need for intermediate conversions across most of the system. However, while a common voltage improves efficiency and reduces complexity, careful design considerations were necessary to protect sensitive components. Over-current protection and safeguards against electrostatic discharge (ESD) were implemented to ensure the longevity and reliability of each device in the system.

Not all components operated at 12 volts, however. For instance, the Raspberry Pi single-board computers used for control and sensor processing required a stable 5-volt supply. To accommodate this, step-down voltage regulators, specifically buck converters, were employed. Buck converters are efficient DC-DC converters that reduce voltage without significant power loss, making them ideal for powering lower-voltage subsystems from a higher-voltage source. Their compact size and high efficiency made them a practical choice for ensuring safe and reliable operation of all 5V components throughout the robot.

In order to protect components in the system, fuses are placed before each device. The size of the fuse was determined based on the current limit of the component itself. In the case of voltage being converted, a simple calculation was done based on the current rating of the component after the converter. These equations were done based on the power requirement of the component as seen in Equation 2.5.

To know which fuse was appropriate, the component power requirement was used to adjust the input power based on the converter efficiency (Equation 2.6), then use that to determine the input current (Equation 2.7), which was needed for calculate the correct fuse rating (see Equation 2.8).

1. Calculate the Power Requirement on the Output Side

$$P_{\text{out}} = V_{\text{out}} \cdot I_{\text{out}} \tag{2.5}$$

where:

- P_{out} is the power required by the load (Watts),
- V_{out} is the output voltage of the converter (Volts),
- I_{out} is the current required by the load (Amps).

2. Adjust for Converter Efficiency

$$P_{\text{in}} = \frac{P_{\text{out}}}{\eta} \tag{2.6}$$

where:

- P_{in} is the input power (Watts),
- η is the efficiency of the converter (typically 0.85 - 0.95).

3. Determine the Input Current

$$I_{\text{in}} = \frac{P_{\text{in}}}{V_{\text{in}}} \tag{2.7}$$

where:

- I_{in} is the current draw on the input side (Amps),
- V_{in} is the input voltage to the converter (Volts).

4. Select the Fuse Rating Finally, the fuse rating should account for normal operational overhead. A safety margin of 125% to 150% is typically applied:

$$I_{\text{fuse}} = I_{\text{in}} \cdot 1.25 \text{ to } 1.5 \tag{2.8}$$

where:

- I_{fuse} is the selected fuse rating (Amps).

2.4 Experiments

To thoroughly test the hardware of WALL-E, both the drivetrain and the collector needed to be tested. This was tested by manually controlling WALL-E with an Xbox controller. The drivetrain was tested first to ensure that WALL-E was operable and could operate in different environments. When this was ensured, the collector mechanism was tested to test the variety of trash WALL-E could collect and to test the litter collection performance on different terrains. The software used to control the drivetrain and collector is described in Chapter 3.

2.4.1 Drivetrain

The drivetrain has been tested to ensure it can operate on different surfaces and inclines. To get a good view of the effect of the different environments on WALL-E's driving, the tests have been measured on speed, where a max speed has been set programmatically, and each test setting had a ramp up distance of five meters, allowing it to reach max speeds before the measuring started (if it can reach the max speed on the surface/incline setting). The drivetrain has been tested on different terrains: Grass, pavement, and asphalt, and with a single incline test on grass with an incline of $\angle 20.94°$.

The drivetrain tests were timed using a stopwatch over a two-meter distance. The two-meter distance was indicated using a ruler as a marker. The time was started when the start of WALL-E passed the start of the measured distance, as seen in Figure 2.20, and the time was stopped when the start of WALL-E passed the end of the measured distance, as seen in Figure 2.21.

The point of the drivetrain tests was not to see if the speed gets affected, as the speed is controlled with a PID controller feeding from wheel encoders. This means that if WALL-E meets a higher resistance, the current to the motors gets increased to reach the desired speed. The tests are meant to work as a binary test to see if WALL-E can operate in the different test scenarios. The speed measurements ensure that WALL-E reaches the full speeds, introducing more vibrations, and

Figure 2.20: Start position for time start

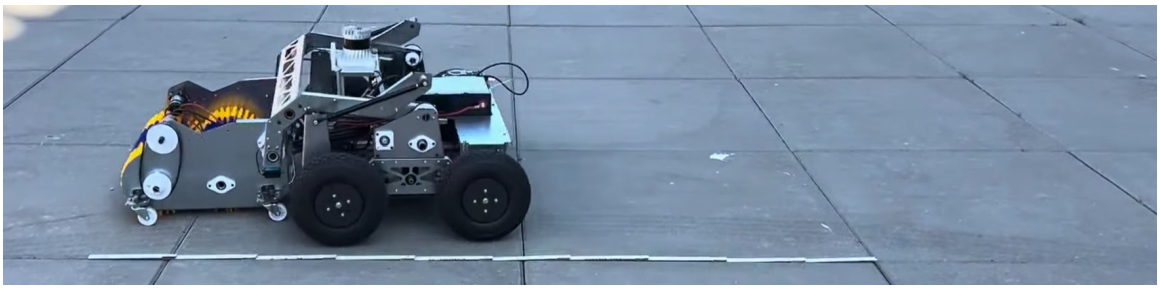


Figure 2.21: End position for time end

thereby testing the mechanical system's stability deeper than going at the planned collector speed. Giving a better image of our design's robustness.

The reason for testing on both asphalt and pavement is because pavement introduces small crevices that the robot needs to be able to navigate through. One concern here is the small caster wheels that may limit the size of crevice that WALL-E can navigate through. This is also a concern with navigating on grass surfaces, as they usually present uneven surface grounds underneath the grass.

2.4.2 Collector

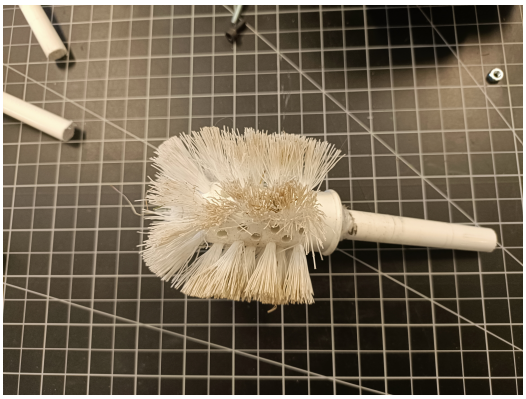
The collector tests were run in two different ways. The initial tests were run with a prototype of the collector, whereas the later tests were run using the full-scale collector. The prototype tests were mainly to ensure that trash could be picked up, and to find the best configuration of the brushes for the collector, i.e. the distance between the brushes and the best brush pattern.

Collector Prototype

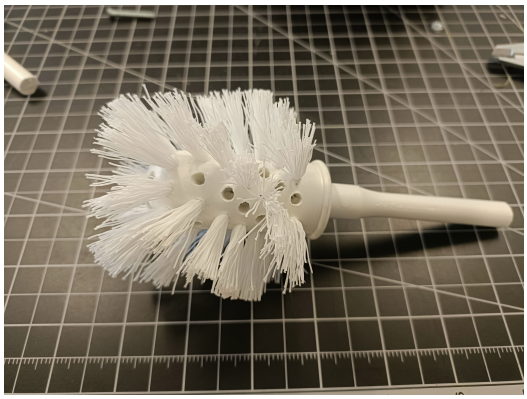
Several experiments have been executed to test (i) optimal distance between the bristles, and (ii) best bristle pattern (straight lines, spirals, triangle lines, and no lines, AKA. full (see Figure 2.22)). These experiments were tested on a scaled-down prototype of our collector. This was done to enable rapid prototyping and testing, and to keep the costs of tests low, before deciding on the best design for the final product.

The low-cost collector prototype was built using laser-cut plywood for the collecting box and toilet brushes to simulate the brushed bristles. A picture of the prototype can be seen in Figure 2.23.

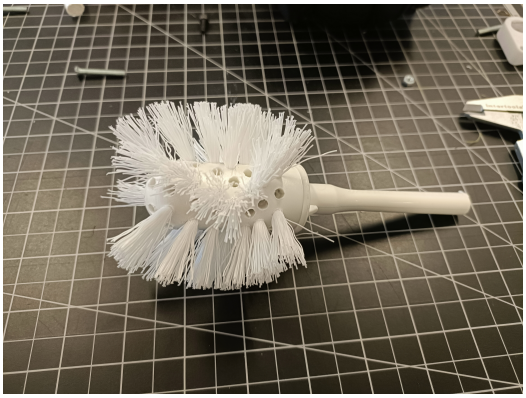
The tests were run on pavement and grass, where it was tested if the collector prototype could pick up scaled down versions of (i) paper, (ii) cardboard, (iii) soft plastic, (iv) hard plastic, (v) compressed metal, (vi) metal cylinders, (vii) glass shards, (viii) cigarette butts, and (ix) nicotine packs. A picture of the scaled-down trash that was used for the tests can be seen in Figure 2.24.



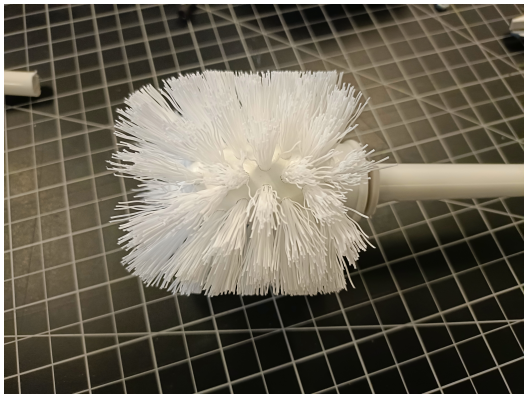
(a) Straight line bristles



(b) Spiral bristles



(c) Triangle line bristles



(d) No lines bristles (full)

Figure 2.22: Different bristle pattern designs

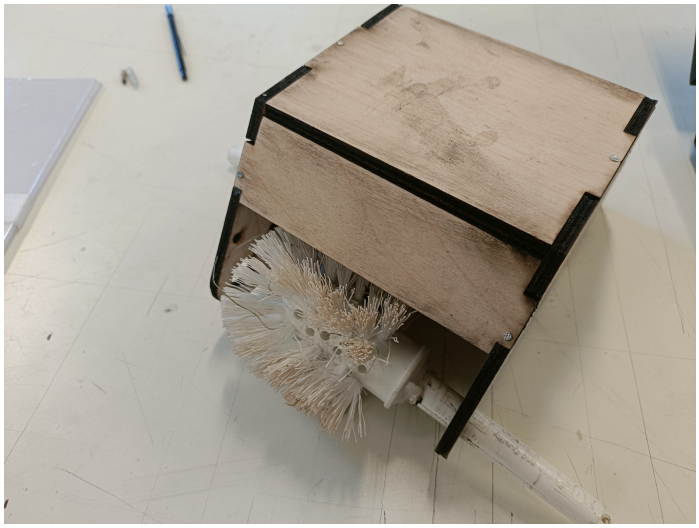


Figure 2.23: Collector prototype for testing

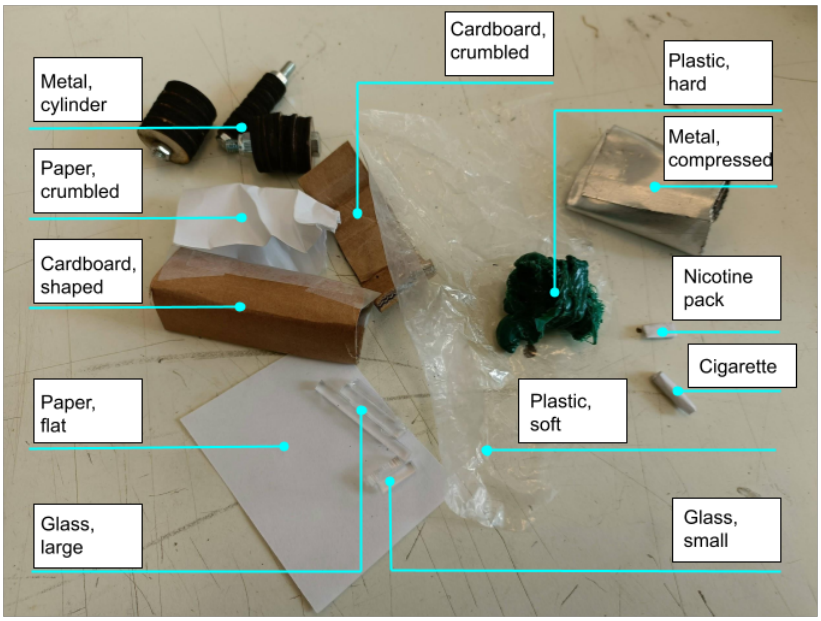


Figure 2.24: Down-scaled trash for testing

The toilet brushes were rotated using handheld drills, and plywood plates were used to hold the collector box at specific testing distances above the ground. This distance was set to 0 cm between the collector box and the ground when testing on pavement (25 cm from brush center to the ground), and 10 cm between the collector box and the ground when testing on grass (35 cm from brush center to the ground). The brushes needed to be raised when testing the trash collection on grass, as the brushes otherwise destroyed the grass.

Full-scale Collector

The full-scale collector was tested on pavement, asphalt, and grass. A diverse collection of trash had been scattered over an area, where the robot was driven over to test if it could pick up the trash. The trash that was used for testing included (i, ii) metal cans (one compressed and one not compressed), a (iii) metal lid, a (iv) glass bottle (cylinder shaped), (v) glass shards, a (vi, vii) cardboard container (one shaped and one compressed), a (viii) plastic bag, a (ix) plastic wrapper (soft), a (x) plastic carton (hard), a (xi) newspaper, (xii) laminated paper from commercials, a (xiii) crumbled and wet A4 paper, a (xiv) milk carton, and a (xv) thread. An image of the trash used can be seen in Figure 2.25.

With these 15 different types of trash materials, shapes, and sizes, we hope to get a good view of WALL-E's capabilities in picking up a diverse variety of trash. To ensure stability of the pickups, each trash was attempted to be collected ten times for the three different terrains (asphalt, pavement, and grass). The placement of the trash was random at each test iteration, simulating the real-world litter scenario, where litter appears in all sorts of placements and orientations. The trash was scattered over the testing area, as seen in Figure 2.26. WALL-E was driven over the trash in order to attempt to pick up the items.

Besides the different grounds, the full-scale collector was also tested with two different brush patterns (full, and straight lines), and at two different brush heights (fully up, and fully down), except on grass where only the "fully up" setting for the brushes was tested, to avoid harming the grass.

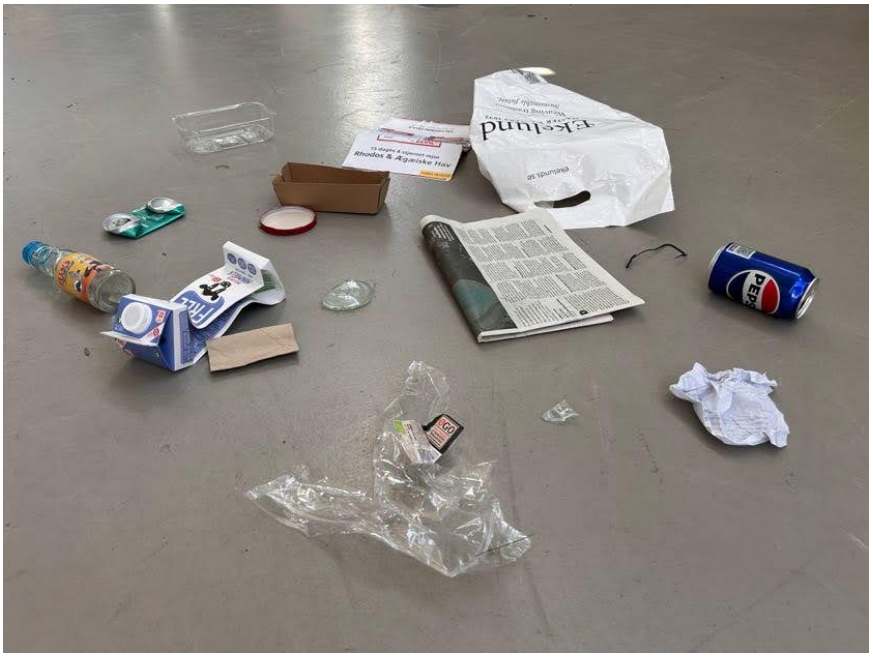


Figure 2.25: Full-scale test trash

Figure 2.26: Trash picking setup

2.5 Results

The tests were executed on the grounds of the University of Southern Denmark. This section describes the results from these tests of both the drivetrain and the collector.

2.5.1 Drivetrain

The experiment results from the ground test of the drivetrain can be seen in Table 2.2. It was measured the time it took WALL-E (x [s]) to move two meters (2 [m]) at full speed. This was used to calculate the meters per second speed through $\frac{2[m]}{x[s]}$.

Terrain	**m/s**	**m/s**	**m/s**	**m/s**	**m/s**	**m/s**	**m/s**	**m/s**	**m/s**	**m/s**
Pavement	1.67	1.53	1.68	1.77	1.75	1.57	1.89	1.94	1.68	1.65
Asphalt	1.83	1.72	1.74	1.71	1.77	1.74	1.75	1.79	1.79	1.74
Grass	1.59	1.75	0.94	1.77	1.77	1.75	1.87	1.82	1.85	1.74
Grass ∠20.94°	1.45	1.60	1.39	1.20	1.33	1.43	1.42	1.18	1.03	1.28

Table 2.2: Drivetrain ground test

To investigate how the robot's maximum driving speed varied across different terrain types, four terrain conditions were tested: asphalt, pavement, grass, and inclined grass. Each condition was tested with 10 independent drives.

Levene's test was conducted to evaluate whether the assumption of equal variances held across the four terrain groups. This should show if a certain terrain would increase the instability of the system. The test results had a p-value of 0.322, showing no statistically significant evidence to reject the null hypothesis that the speed variances are not significantly different from each other over the different terrains. Thus, the variances across terrain types can be considered homogeneous overall.

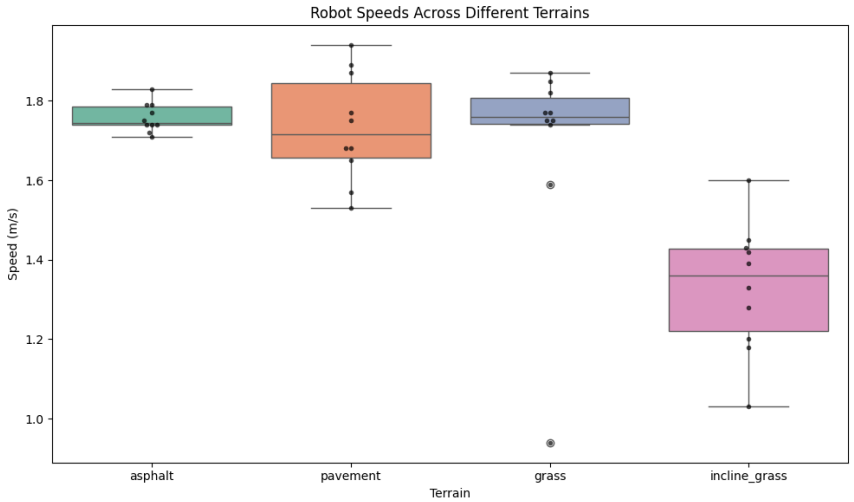


Figure 2.27: Box plot over drivetrain speed results

However, while the overall Levene's test suggests equal variance, the test results are also limited by the low sample size of 40 observations. In the box plot in Figure 2.27, it can be seen that the incline test looks to have a much bigger variance than the grass test, however still insignificant. The grass test does, however, also look to have some outliers. Therefore, Welch's ANOVA was chosen for robustness against differences in variance and unequal sample sizes (although not relevant in our case). The Welch's ANOVA was used to test for differences in mean speeds between the terrain types.

Welch's ANOVA showed that there is a highly significant difference in the mean speeds across terrain types ($p < 0.001$), with a large effect size of $\eta_p^2 = 0.523$.

To determine which specific terrain pairs differed, a Games-Howell post-hoc test was conducted, making pairwise comparisons. The key results can be seen in Table 2.3, where the significant results are highlighted.

Comparison	p-value	Significant?	Hedges' g	Interpretation
Asphalt vs Grass	0.835	No	0.36	No significant difference
Asphalt vs Inclined Grass	<0.001	**Yes**	**3.45**	Very large difference
Asphalt vs Pavement	0.942	No	0.24	No significant difference
Grass vs Inclined Grass	0.015	**Yes**	**1.51**	Large difference
Grass vs Pavement	0.958	No	-0.21	No significant difference
Inclined Grass vs Pavement	<0.001	**Yes**	**-2.55**	Very large difference

Table 2.3: Games-Howell post-hoc pairwise comparisons test results

It can be seen through the Games-Howell post-hoc test that WALL-E drives significantly slower on inclined grass compared to all other terrain types, as shown by the Hedges' g. This is, however, most likely because the ramp-up distance was not on an incline, due to the short incline availability, not allowing the PID controller to adjust for the incline.

Although the grass tests (without the incline) did not show to have a significant difference, during the test, the mechanical part of WALL-E showed to have some problems. For example, the bristles got stuck on the grass if the collector was too low, making the arm push up in an undesirable way. An example of this can be seen in Figure 2.28, where the blue line indicates the straight path where all the left-side wheels should be if they're touching the ground, and the red circle indicates the wheel lifting from the line. To avoid this, when testing the drivetrain, the front caster wheels on the arm were lowered, raising the collector to fully up, allowing WALL-E to drive on the grass.

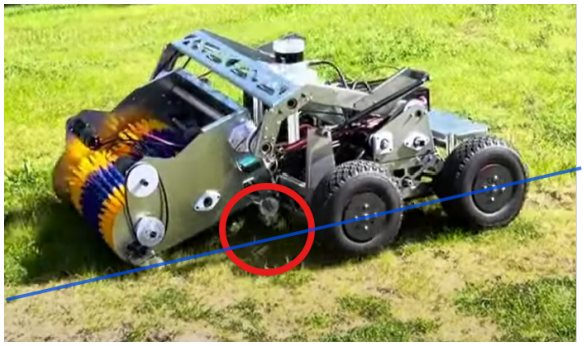


Figure 2.28: Bristles stuck on the grass, lifting the back wheel of the arm

The robot circuit furthermore needed to be re-turned on, as it frequently turned off when driving on grass, due to the vibrations and bad wiring. This happened 35% of the drives on grass over the 20 grass trials over the flat and inclined grass. This was not an issue on asphalt, but also happened twice on the pavement.

2.5.2 Collector

The collector was tested in two iterations; one for the collector prototype for testing the best bristle design and brush distances, and one for the full-scale collector to find the best testing configuration and ensure we can pick up the desired litter at a good success rate.

Collector Prototype

The results from the collector prototype tests can be seen in Table 2.4 for the tests on grass, and in Table 2.5 for the tests on pavement. The cells marked with red and 0 indicate that the item could not be picked up, whereas the cells marked with green and x indicate that the item could be picked up at the first attempt. The orange cells marked with o and p indicate that the items could not be picked up at the first attempt, and that the collection had to approach the item from a different direction if marked with o or that the item needed to be in a different position if marked with p, in order to pick it up. The position (p) requirement was mainly due to the item being so small that it slipped under the grass, preventing the collector from collecting the item without damaging the grass. In these cases, the collection only succeeded if the item was lying on top of the grass. Each item was attempted to be picked five times. If it couldn't succeed in those five attempts, the item was marked as non-collectable.

B. shape	**Dist.**	**Cardb.**		**Paper**		**Plastic**		**Metal**		**Glass**		**Cigs**	**Nicot.**
		Cr	Sh	Cr	Fl	Hd	St	Cr	Cyl	S	L		
Full	5.5	x	0	x	x	x	x	x	x	0	x	0	0
	6.5	x	x	x	x	x	x	x	x	0	0	0	0
	7.5	x	o	x	x	x	x	x	x	0	0	0	0
	8.5	x	x	x	x	x	x	x	0	0	0	0	0
Spiral	5.5	x	0	x	x	x	x	x	0	0	x	0	0
	6.5	x	0	x	x	x	x	x	0	0	x	0	0
	7.5	x	x	x	x	x	x	x	x	0	0	0	0
	8.5	x	x	x	x	x	x	x	0	p	p	0	0
Triangle	5.5	x	o	x	x	x	x	x	x	x	x	0	0
	6.5	x	x	x	x	x	x	x	x	0	x	0	0
	7.5	x	x	x	x	x	x	x	x	0	0	0	0
	8.5	x	x	x	x	x	x	x	x	0	0	0	0
Straight	5.5	x	o	x	x	x	x	o	o	0	x	p	x
	6.5	x	o	x	x	x	x	x	x	0	x	0	0
	7.5	x	x	x	x	x	x	x	o	0	0	0	0
	8.5	x	x	x	x	x	x	x	x	0	0	0	0

Table 2.4: Prototype trash collection experiments on grass

The meaning of the abbreviations in Table 2.4 and Table 2.5 is described in the list below.

- Cr → Crumbled
- Sh → Shaped
- Fl → Flat
- Hd → Hard
- St → Soft
- Cyl → Cylindrical
- S → Small
- L → Large
- Cigs → Cigarette butt
- Nicot → Nicotine bag

B. shape	Dist.	Cardb.		Paper		Plastic		Metal		Glass		Cigs	Nicot.
		Cr	Sh	Cr	Fl	Hd	St	Cr	Cyl	S	L		
Full	5.5	x	x	x	x	x	x	x	x	x	x	x	x
	6.5	x	o	x	x	x	x	x	x	x	x	x	x
	7.5	x	o	x	x	x	x	x	x	x	x	x	x
	8.5	x	o	x	x	x	x	x	x	0	0	0	0
Spiral	5.5	x	0	x	x	x	x	x	x	x	x	x	x
	6.5	x	0	x	x	x	x	x	x	x	x	x	x
	7.5	x	0	x	x	x	x	x	x	x	x	x	x
	8.5	x	x	x	x	x	x	x	x	x	x	x	x
Triangle	5.5	x	0	x	x	x	x	x	x	x	x	x	x
	6.5	x	o	x	x	x	x	x	x	x	x	x	x
	7.5	x	x	x	x	x	x	x	x	x	x	x	x
	8.5	x	o	x	x	x	x	x	x	0	0	0	0
Straight	5.5	x	x	x	x	x	x	x	x	x	x	x	x
	6.5	x	x	x	x	x	x	x	x	x	x	x	x
	7.5	x	x	x	x	x	x	x	x	x	x	x	x
	8.5	x	x	x	x	x	x	x	x	0	0	0	0

Table 2.5: Prototype trash collection experiments on pavement

From the prototype collector results in Table 2.4 and Table 2.5, it can be seen that the tests perform significantly better on pavement than on grass. This is due to the brushes needing to be lifted higher from the ground on grass, as the grass would otherwise be damaged, and due to the small objects falling underneath the grass.

To explore the significance of the test results, a logistic regression model was applied, testing which brush design and which distance between the brushes provides the best setting for good collection results. The dependent variable was binary, indicating whether a collection attempt was successful (success = 1) or not (success = 0). The model included the distance between the brushes (a continuous variable) and the categorical variable brush design, with "Full" as the reference category. The results of the analysis can be seen in Table 2.6.

Predictor	Coef	Std. Err	z-score	p-value	95% CI	Odds Ratio
Distance	-0.269	0.119	-2.26	0.024	[-0.503, -0.035]	0.76
Straight design	0.349	0.375	0.93	0.353	[-0.387, 1.085]	1.42
Spiral design	0.065	0.360	0.18	0.857	[-0.640, 0.769]	1.07
Triangle design	0.201	0.3367	0.55	0.584	[-0.518, 0.920]	1.22

Table 2.6: Logistic regression analysis results of prototype collector experiment

The model converged successfully after six iterations and demonstrated a log-likelihood of -179.23 compared to the null model (-182.33), resulting in a likelihood ratio test p-value of 0.184. This indicates that the model did not significantly improve over the null model. The pseudo R-squared was 0.017, suggesting that only about 1.7% of the variation in success can be explained by the independent variables of the model.

The coefficient for Distance was statistically significant (p = .024) and negative (β = -0.269), indicating that an increased distance between the brushes is associated with a decrease in the odds of a successful collection. The odds ratio of 0.76 implies that for every additional centimeter of distance, the odds of success decrease by approximately 24%.

However, none of the Design variants (Straight, Spiral, Triangle) showed a statistically significant effect compared to the reference design (Full). Although their odds ratios were all greater than 1 (suggesting a possible trend toward increased success), the confidence intervals were wide and all included 1, indicating a lack of statistical significance. This is likely due to the low number of data

points, with only 352 observations.

Full-scale Collector

The results from the full-scale collector tests can be seen in Table 2.7. The results are given in percentages of how many times the trash could be picked up in the specific setting, out of the ten attempts. Below is a list of the abbreviations used in the table with their meaning.

- Sh. → Shaped
- Cr. → Crushed
- Fl. → Flat

Ground	**Asphalt**				**Pavement**				**Grass**	
B. Pattern	**Full**		**Straight**		**Full**		**Straight**		**Full**	**Straight**
B. Height	Down	Up	Down	Up	Down	Up	Down	Up	Up	Up
Plastic Wrapper	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Sh. Plastic	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%
Sh. Cardboard	100%	80%	100%	100%	100%	40%	100%	100%	100%	100%
Milk Carton	80%	70%	100%	100%	100%	100%	100%	100%	100%	100%
Glass Bottle	50%	0%	100%	80%	70%	0%	100%	90%	100%	100%
Cr. Can	100%	0%	100%	0%	100%	0%	100%	0%	100%	100%
Glass Shards	0%	0%	100%	0%	0%	0%	100%	0%	0%	60%
Metal Lid	100%	0%	100%	0%	100%	0%	100%	0%	20%	100%
Plastic Bag	100%	80%	100%	100%	100%	80%	100%	100%	100%	100%
Wet, Cr. Paper	100%	0%	100%	100%	100%	10%	100%	100%	100%	100%
Newspaper	100%	90%	100%	100%	100%	100%	100%	100%	100%	100%
Soda Can	0%	0%	100%	100%	0%	0%	100%	100%	100%	100%
Fl. Cardboard	100%	0%	100%	0%	100%	0%	100%	0%	70%	100%
Commercials	100%	40%	100%	70%	100%	100%	100%	100%	100%	100%
Zip Tie	60%	0%	100%	0%	50%	0%	100%	0%	40%	70%
Total	**79.3%**	**37.3%**	**100%**	**63.3%**	**81.3%**	**42%**	**100%**	**66%**	**82%**	**95.3%**

Table 2.7: Results from full-scale collector test

To investigate the effect of the collector height and the brush design on the probability of successful litter collection, a logistic regression analysis was conducted again, with the full-scale data. The binary outcome variable was Success (1 = successful collection, 0 = failed collection). The primary predictors were on the collector height (up vs. down) and the brush design (full vs. straight). Additional control variables were included to account for variation due to litter shape (round or not, and flat or not), litter rigidity (soft or not), ground type (asphalt, pavement, or grass), and size (small, medium, or large). The baseline categories were with the collector up, the brush design "full", on asphalt, with large size, and non-round and rigid litter.

The final model included 1500 observations and converged successfully to the logistic regression after eight iterations. The model fit was strong, with a Pseudo R^2 of 0.5538, indicating that approximately 55% of the variance in collection success was explained by the model (i.e. the collector height and brush design). The likelihood ratio test was highly significant (LLR p <0.001), confirming that the model provides a significantly better fit than the null model.

The statistical results of the logistic regression analysis can be seen in Table 2.8.

We can here see that when the collector is placed in a lower position (down), the odds of a successful collection increase by a factor of 80.47 (p <0.001) as compared to the collector in a higher position (up). This represents a very strong and significant effect of collector height on success, and shows to be the most significant predictor of a successful collection. The litter collection on grass

Predictor	Coef	Std. Err	z-score	p-value	95% CI	Odds Ratio
Collector down	4.388	0.307	14.30	<0.001	[3.786, 4.989]	80.47
Straight design	2.562	0.224	11.41	<0.001	[2.122, 3.002]	12.96
Round shape	-2.053	0.236	-8.72	<0.001	[-2.515, -1.591]	0.13
Flat shape	-1.894	0.235	-8.06	<0.001	[-2.355, -1.434]	0.15
Soft rigidity	2.358	0.275	8.58	<0.001	[1.819, 2.897]	10.57
Grass ground	4.139	0.355	11.65	<0.001	[3.443, 4.836]	62.75
Pavement ground	0.079	0.198	0.40	0.692	[-0.310, 0.468]	1.08
Medium size	-0.682	0.279	-2.44	0.015	[-1.229, -0.135]	0.51
Small size	-2.084	0.302	-6.91	<0.001	[-2.675, -1.492]	0.12

Table 2.8: Logistic regression analysis results of full-collector experiment

also proved to significantly boost the odds of collection success with a factor of 62.75 ($p < 0.001$) compared to the other grounds. These two results may, though, be linked, as the grass raises the trash higher, having a similar effect to the collector being lowered. The effects are not equal as the distance from the grass to the collector brushes is still larger than a lowered collector, but also much less than a higher collector on a non-grass ground. This is contrary to the prototype collector experiments, likely due to the trash's larger size and a fuller grass lawn.

It can also be seen that the pavement has no significant difference from asphalt, with a p-value of >0.05. This means that the crevasses created by the pavement did not significantly affect the collection success.

The significance of the collector distance between the ground and the litter also explains why the smaller the trash is, the harder it becomes to pick up, as the distance grows larger. However, this is only relevant with the collector in a higher position, as the down position of the collector makes the brushes touch the ground over a distance of three centimeters, creating a zero distance between the litter of any size and the collector brushes. However, in this case, the smaller the litter object is, the higher the risk of the object staying in the brush deadzone (between the brushes where they don't touch each other) becomes.

For this same reason, the shapes flat and round significantly decreased the success of the collection ($p < 0.001$). On the other hand, soft litter increased the collection chances by more than 10 times. This can be explained by the fact that if an item is soft, it is easier to grab with a brush, as it can deform the trash to help pick it up.

The straight design of the brushes also significantly ($p < 0.001$) increased the success rate, making the collection a factor close to 13 times more successful. This was not significant in the prototype collection tests due to the low observation number, but it indicated that it would be true, and is now confirmed by the full-scale experiment.

These results highlight that both the collector's height and the brush design play a crucial role in the collection success, even when controlling for shape, ground surface, and object size. Notably, placing the collector down and using a straight brush design substantially increases the probability of success. Additionally, the control variables confirm that collection success is also strongly influenced by the physical properties of the objects and their environment.

Across all terrains, the straight design with the collector down on pavement and asphalt and up on grass has a success rate of 98.4%. The setting for raising the collector is currently manual, but can in the future be altered dynamically, enabling WALL-E to switch its area of operation from grass to pavement and back, autonomously.

2.6 Discussion

The prototype experiments proved that the distance between the brushes mattered and that a shorter distance between them was better. However, with the prototype test being driven by power drills and the brushes being smaller, the friction between them is smaller than the full-scale collector

Figure 2.29: Detached bristle

brushes that need to be driven by motors. The distance chosen for our collector brushes had this in consideration, and the distance was chosen to be the smallest it could, while a 200:1 ratio of the gearing for the motors could still drive the brushes, allowing for extra friction for when collecting bigger items.

The prototype experiments also showed a lack of significance for the brush designs, likely due to the low number of data points. Here the full-scale collector showed a significant improvement of the collection success by using a straight brush design over the full design.

With the proof of concept, our robot design, being a first iteration, still has a lot of issues. As shown during the drivetrain experiments, the small caster wheels would easily get shortly stuck when driving on the uneven grass. Here, bigger caster wheels could prove to significantly enhance the drivability of WALL-E.

Additionally, although picking up bigger objects was more likely to be successful, it also put a bigger strain on the collector design. The sides of our collector holding the brushes appeared to be too flexible, allowing the front brush axle to fall out of its bearing when picking up big items, as seen in Figure 2.29. This only happened sometimes when picking up the milk carton and the glass bottle. When the brush detached, the pickup attempt was recorded as a fail. However, the experiments proved that WALL-E is able to pick up bigger items, but that a reinforcement of the collector walls is necessary to make the collection more stable and reliable.

Another issue that occurred during the litter collection was that the brush bristles would tear up the newspaper, as the paper is very thin and brittle. As was seen in the results from Table 2.7, WALL-E was not able to pick up small flat items. This was due to them getting stuck in a deadzone between the two brushes, with the brushes unable to lift the item, just pushing it back and forth between themselves. This also meant that when the newspaper got ripped up, the majority was picked up by WALL-E, but the smallest pieces were left, leaving a bit of a mess behind, as seen in Figure 2.30. However, this specific issue is minor, as newspapers are biodegradable and will biodegrade fast when cut into small pieces.

It was, however, not all small items that the robot wasn't able to pick up. With the thread, the robot could pick it up if it wasn't lying completely flattened out. However, specifically with the thread, it would get stuck in the bristles, as seen in Figure 2.31. This meant that the thread would get whirled around with the brush, giving it a 50/50 chance of getting thrown into the collector storage or being thrown out into nature again with the full design. To prevent this, a brush cleaner

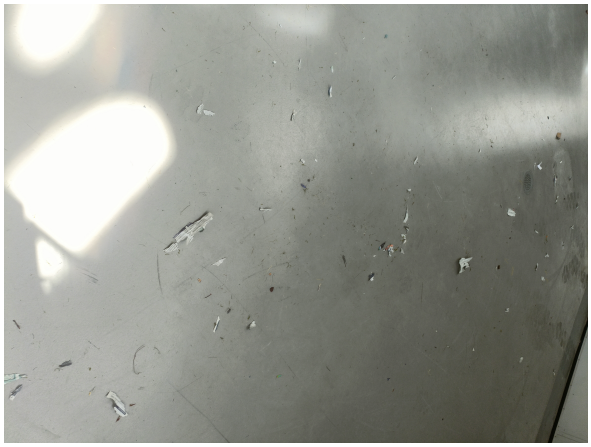


Figure 2.30: Aftermath of picking up a newspaper

Figure 2.31: Thread stuck in Brush

could be attached to the bristles to pull out things getting stuck in them, making sure the items would end up inside the collector. This would also help with the newspaper problem, as a lot of the paper pieces got stuck in the bristles before being thrown out.

Another thing that was witnessed during the experiments where the fact that it was difficult to control the drivetrain in a slow way, making it burst with power, breaking the inner pulleys by the motors. To combat this, we reinforced the new pulleys with six bolts. However, for a future iteration, more time spent on tuning the velocity control should help alleviate this. Other methods of motion profiling, like trapezoidal motion, can also be used to eliminate the jerky motion.

This system leveraged components that were fairly voltage and current tolerant, as well as converters that were efficient and stable. This allowed the safety circuits to be fairly minimal. However, the system still managed to burn one Raspberry Pi. Due to a lack of diagnostic data, the best conclusion is that static build-up when operating outdoors in the dry air was the culprit for this loss. The circuits in this system are built to handle overcurrent but not extreme voltage spikes. In the future, better use of common grounds and zener diodes should be used to limit damages like this, as well as better wiring to stabilize the circuit under vibrations.

It was also found that the battery ended up taking a lot of real estate in the drivetrain and takes far too long to charge. For the situation of creating a high fidelity prototype due to its ease of use and price, it was a perfect choice, but in the future, to further improve the system, a more efficient option in terms of size, power output, and lifespan is needed.

With a second iteration of the robot design, these things can be altered, further increasing the

collection success and improving the stability and robustness of the overall robot solution.

2.7 Conclusion

By starting with the development of a CAD model, to manufacturing the parts, assembling the robot, and testing both the drivability and the collection mechanism of WALL-E, we have successfully created a first iteration of a trash-picking mobile robot. The robot proved to be able to pick up all the test trash pieces with a collection success rate of 100% on pavement and asphalt with a straight brush design. Because of the limited height of the collector possible on grass, without damaging the surroundings, the success rate with the straight design came in at a lower rate of 95.3%, which is still within our desired collection success rate. More accurate alteration of the height can further increase the collection success rate. The best configuration over the three different terrains yields a success rate of 98.4%, passing our criteria of a 95% collection success rate.

We furthermore saw the drivetrain's ability to successfully traverse through different terrains, although with some limitations in terms of driving over deep crevasses, because of the small caster wheels on the collector. It was also seen that while WALL-E can successfully drive on a 20.94°incline, it does lose some speed, although not important for the litter collection use case, and still passes our binary test of being able to successfully traverse the three different terrains of grass, asphalt, pavement, and a 20°incline.

Although not a finished product, the current WALL-E works as a proof of concept that a mobile robot can successfully collect litter from the streets and grass. Further iterations include reinforcing the collector, adding a comb for the brushes to prevent trash from being thrown out of the collector after being collected, and increasing the size of the caster wheels for more stability when traversing uneven terrains, as well as a more stable circuit wiring.

Chapter 3

Software

With the mechanics done and the electronics wired, the hardware of the robot is now finished. The hardware of WALL-E can be remote controlled using an Xbox controller. Each input from the sensors and Xbox controller must be processed by the onboard computer, which translates these inputs (the camera feed, joystick commands, LiDaR, and encoder data) into motor outputs for the arm, latch, brushes, and drivetrain. The system is built using ROS2 and communicates between inputs and outputs through ROS nodes. The idea is that WALL-E will have modes: Manual and autonomous. The Autonomous mode will rely on the LiDaR and encoder data from the wheels to explore the area. In the autonomous mode, the camera is used to detect litter, and when detected, it overwrites the exploration to move the robot over to the litter position, where the camera activates the brushed rollers, making WALL-E pick up the litter. When the camera detects that the litter has been picked up, data is added to a database, which activates the arm motors when the collected litter amount exceeds the limit value. The arm then rises and activates the latch when the arm is in the correct position to dump the collected litter into the onboard trash can. When the latch closes, the data of the collected litter in the database is reset. The joystick can overwrite the autonomous mode and move the robot into manual mode. Here, the user can control the robot using the Xbox controller, from which the user can also activate the rollers. The joystick commands are described in Section 3.1. The overall communication between the different ROS nodes and systems can be seen in Figure 3.1.

As mentioned in 2.3, this system utilizes two Raspberry Pis for navigation, vision, and motor control. These two devices send and receive commands from a stationary external controller. In this case, a laptop. This laptop is responsible for controlling the robot's behavior. When operating, it collects data from the two Raspberry Pis to make decisions on whether to search for a new area to collect trash or to approach and collect trash that is visible to the robot. To allow for easy transfer of data between these different devices, one of the Raspberry Pis creates a wireless access point for the external controller to connect to. This allows for the ROS nodes running on the external controller to access the topics and data that are being collected by the two Pis on the robot. This access point also acts as a bridge for the Ouster lidar. A Docker container [9] created by Ouster that exposes the data collected by the sensor as ROS topics. This container runs on the external controller and is bridged to the lidar via the access point. The same is true for the node running on the navigation pi that communicates with the motor controllers via a CAN bus.

Additionally, the robot can swap between autonomous and manual modes dynamically using a tool called Twist Mux. Twist Mux is a message multiplexer for twist messages. There are a few sources from which the robot can receive twist messages that contain a linear and angular velocity. The message used at any given time is dictated by the multiplexer that is configured with a ranking for each message. The highest-ranking messages come from the Xbox controller. Meaning that at any time, the user can take over control of the robot. The next rank is the camera, providing the location of trash, and lastly, the default behavior of navigation or exploration. Meaning, unless the robot sees trash or receives an input from the controller, it will explore its environment.

In addition to the inputs described in Figure 3.1, the robot also has a GPS module. This module

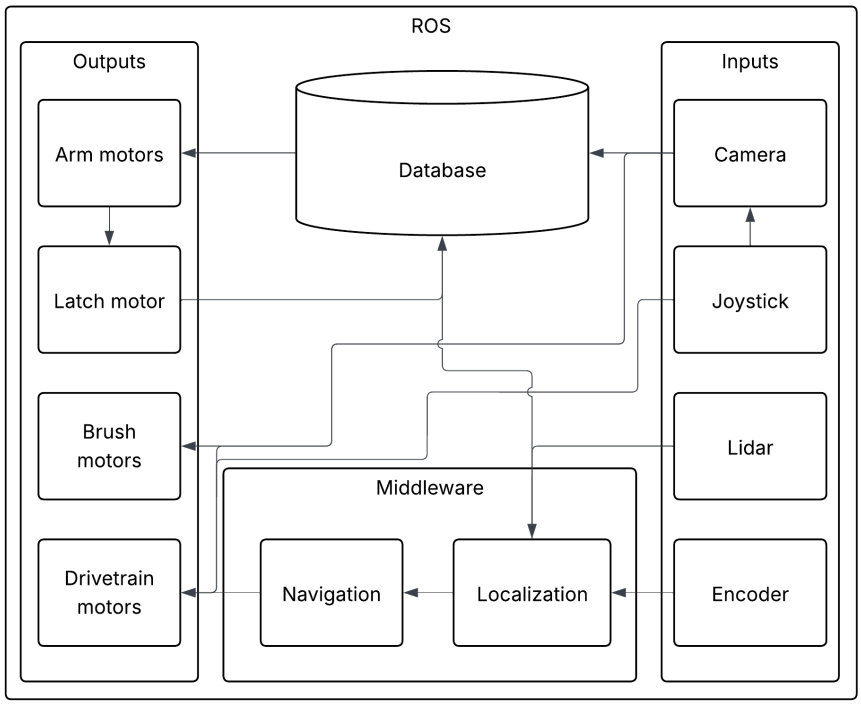


Figure 3.1: ROS communication

doesn't post using a ROS node, but is directly read in the exploration algorithm. The user can set up the robot using the GPS module through a website hosted by WALL-E. Here, the user can draw boundaries on a map, which is translated into the area WALL-E can explore and clean. The website is not further described in this thesis, as it was developed and used for another course. The boundaries are, however, further described in the exploration algorithm in Section 3.3.3.

3.1 Manual Control

The manual control is, as mentioned, controlled through an Xbox controller. The controller buttons are being read and sent through ROS nodes and translated into robot functions.

The manual control is activated through an enable button, which is the LB button on the controller. This needs to be held down in order to control the robot manually. The drivetrain can be driven using the left stick controller, where the percentage of direction the user pulls the stick will translate to a percentage of speed in the given direction. The RT is a trigger button in the Xbox controller, which also registers how hard the button is pressed. This is also converted into a percentage output, this time for the brushed rollers. With this, the user will need to interact with three buttons to manually pick up litter with WALL-E: the enable button, the drivetrain stick, and the roller trigger.

A diagram visually showcasing an overview of the used controller buttons and their functionality can be seen in Figure 3.2.

Figure 3.2: Mapped controller buttons.

3.2 Related Works

3.2.1 Litter Detection

The task of litter detection has gained increasing attention due to its environmental impact and the need for efficient waste management strategies. Deep learning-based object detection algorithms have played a crucial role in advancing the field, particularly in real-world outdoor settings where litter varies in size, shape, and background complexity. This section reviews relevant litter detection models, including models designed for general object detection with a focus on lightweight architectures suitable for edge computing applications.

General Detection Models

Among the general-purpose detection models, YOLOv9 [10] and YOLOv11 [11] represent the latest advancements within the YOLO family. YOLOv9 introduces a new lightweight architecture: Generalized Efficient Layer Aggregation Network (GELAN), based on gradient path planning that optimizes feature fusion and reduces computational complexity, leading to enhanced accuracy and real-time performance. Building on this, YOLOv11 further refines the previous YOLO architectures with novel attention mechanisms and structural optimizations, improving detection precision, especially for small objects. These improvements make the models promising candidates for litter detection, where objects can be irregularly shaped and blend into complex backgrounds.

Another notable model, YOLOR [12], extends the capabilities of YOLO-based detectors by integrating implicit knowledge representation into object detection. Unlike traditional YOLO models, YOLOR leverages both explicit and implicit learning, improving generalization and robustness in various detection scenarios. This approach has been particularly beneficial for detecting objects in cluttered environments while showing better model generalization, making it highly relevant for street-level litter detection.

For applications requiring efficiency on low-power devices, EfficientDet [13] has emerged as a competitive alternative to the YOLO series. EfficientDet utilizes a weighted bidirectional feature

pyramid network (BiFPN) and a compound scaling method to balance accuracy and computational cost. This model has demonstrated high performance in small object detection while maintaining lower inference times, making it suitable for deployment on edge devices

In addition to these models, classical object detection architectures have also been widely used and serve as a foundation for modern approaches. RetinaNet [14] introduced the focal loss function, which addresses the issue of class imbalance by down-weighting easy examples and focusing more on difficult detections. This approach has proven effective for detecting small and scattered objects, making it relevant for litter detection applications. Faster R-CNN [15], another widely adopted model, utilizes a region proposal network (RPN) to improve efficiency in detecting objects within an image. While computationally heavier than single-shot detectors like YOLO, Faster R-CNN remains a strong benchmark for accuracy. Similarly, Mask R-CNN [16] extends Faster R-CNN by incorporating an additional segmentation branch, enabling instance segmentation in addition to object detection. This added capability could be useful for precise litter localization, distinguishing between overlapping waste items in complex environments, however, with the downside of poor real-time performance due to the larger model sizes.

Litter Detection Models and Datasets

TACO: Trash Annotations in Context for Litter Detection [17] provides a dataset for litter detection. They tested their dataset using Mask R-CNN but only achieved the maximum mAP of 26.2% over one-class vs. ten-class segmentation, which the 26.2% mAP score being for the one-class segmentation.

Another litter detection dataset, PlastOPol was presented in Litter Detection with Deep Learning: A Comparative Study [18]. The study compares CNN architectures (Faster RCNN, Mask-RCNN, EfficientDet, RetinaNet, and YOLO-v5 [19]) for litter detection, measuring their performance on metrics such as detection accuracy, processing time, and memory footprint. The study found that models within the YOLO family has superior performance within these metrics.

pLitterStreet: Street-Level Plastic Litter Detection and Mapping [20] provides another extensive dataset collected from street-level imagery, pLitterStreet, which they use to test various object detection algorithms (Faster R-CNN, RetinaNet, YOLOv3 [21], and YOLOv5) for litter detection. They tested both for best model performance and for model generalization based on different outdoor environments from three different datasets (TACO, PlastOPol, pLitterStreet). The work also confirmed that models in the YOLO family showed the best performance for litter detection, achieving a mAP of 69.0%. The work additionally highlighted that domain specific datasets are crucial for good model performance, showing low model environment generalization based on the three different datasets.

Beyond ground-based approaches, aerial litter detection Onboard Vision-Based Trash and Litter Detection in Low Altitude Aerial Images Collected by an Unmanned Aerial Vehicle [22] explores the use of deep learning for detecting litter from aerial imagery, comprised into the UAVVaste dataset. The study explored different CNN-based models on different edge computing devices. They presented eight different metrics on which EfficientDet achieved the overall best performances, with the YOLOv4 model [23] having the overall highest accuracy within a single metric. The study additionally showed even the small models struggled with real-time inference on edge devices, with the fastest model having an average time per inference (ATPI) of 0.8 seconds, but not being any of the best performing models. YOLOv4 had a 14.4 ATPI [s], and EfficientDet had 3.6 ATPI [s] when tested on a Raspberry Pi 4B CPU, showing the struggle with real-time performance models on edge devices.

Overall, the combination of dataset-driven approaches like pLitterStreet and state-of-the-art detection models such as YOLOv9, YOLOv11, and EfficientDet presents a promising direction for improving litter detection accuracy and efficiency. The trade-off between model performance and computational feasibility remains a critical consideration, particularly for real-time deployment in edge computing environments.

3.2.2 Navigation

Many research efforts have been made towards the development of SLAM and autonomous navigation systems for mobile robotics. An early example done by Durrant-Whyte and Bailey[24] implemented extended kalmen filters for real time map generation and localization, starting the basics for probabilistic SLAM. Since this initial research on SLAM many other variants have emerged. Including graph based SLAM, and particle filter based SLAM. Each of these helping with the usability in different environments.

Cartographer based SLAM by Google[25] and Gmapping[26] are both examples of 2D SLAM algorithms used a lot in the ROS ecosystem. For 3D environments where 3D lidars return point clouds, systems like LOAM[27] and LeGO-LOAM[28] have proven to have high accuracy in point cloud based SLAM. Especially when high resolution sensors like the Ouster lidars are used. These systems rely on the processing of dense 3D data clouds.

This systems is built on the back of the developments from the mentioned tools and researchers. It works by leveraging the ROS 2 ecosystem and the packages and tools it provides, making development and testing far smoother. The integration of the Ouster OS0 LiDAR with an IMU-based odometry system provides robust environmental awareness and localization, while frontier exploration enables autonomous discovery of unmapped areas in both indoor and outdoor environments.

3.3 Methodology

The first step to autonomous driving, is being able to detect where the trash is located, allowing the robot to autonomously pick it up. This Section describes the litter detection of WALL-E, together with its navigation.

The navigation utilizes SLAM (Simultaneous Localization and Mapping). SLAM allows the robot to traverse and explore an unknown or known environment while mapping the area and avoiding obstacles. A method called frontier searching is used to explore unknown areas.

In autonomous robot navigation, exploration of unknown or partially known areas is a crucial attribute. SLAM provides provides the foundation for the robot to create maps of its environment and actively positioning itself within that map. This duel processing allows for effective path planning and obstacle avoidance.

In terms of simulation this system is simulated using the ROS 2 Navigation Stack (Nav2). The primary goal being to create a basic but functional representation of the system. This representation can then be used to test the autonomous features of the Nav2 stack within a simulated environment. This allows for testing and development of tasks such as SLAM, path planning and other skills the Nav2 stack provides. The use of the simulated environment also allows for development of the autonomous systems in parallel to the development of the physical system.

Utilizing the nav2 stack within a simulated environment allows for testing of functionality to be done without the risk of damaging physical hardware making development far cheaper and faster.

3.3.1 Litter Detection

As reviewed in the related works, it was found that litter detection is domain specific, meaning that the dataset that the machine learning model is trained on is poorly generalized for both unseen litter and for domain specific backgrounds. Having looked through the datasets used for the different models, it showed that their datasets were lacking in diversity in environments. One thing that makes trash and litter detection models hard to train, is the difficulty in gaining a dataset that can accurately represent the diversity of trash in both shapes, sizes, materials and conditions. With most detection models, artificial data can help diversify a dataset, but with litter being affected by weather conditions and object compression, generating artificial images is challenging due to the lack of realistic morphing of 3D objects. With the time needed to invest in creating a pipeline for generating realistic artificial litter images, it would be faster to take real images. However, even this would take a long time to get enough images for training and testing a deep learning model. To avoid this, we will limit domain specific images to only be used for evaluating pre-trained object detection

models, and for fine-tuning pre-trained models. As detection models are not trained for detecting trash and litter, the existing model needs to be trained using transfer learning. For this, the four datasets from UAVaste [22], TACO [17], PlastOPol [18], and pLitterStreet [20] will be used in a modified form as not all pictures in the datasets fit our domain, and images in water and indoor will not be included. All the datasets are of images from other countries, and does therefore not contain images close to the validation dataset. Different models have been tested to evaluate the best model for WALL-E's use case. The evaluation will be done both based on the model performance, but also the real-time capabilities of the models.

The object detection model used in this study is based on the YOLO11n architecture, a lightweight neural network designed for real-time inference on edge devices. YOLO11n ("nano") represents the smallest variant in the YOLOv11 family, optimized for low-latency applications and constrained environments, such as embedded systems on mobile robots [11]. Its architecture uses:

- CSPDarkNet-inspired backbone with fewer parameters.
- PANet-like neck for feature aggregation.
- Decoupled head for simultaneous bounding box regression and classification.

The model supports distribution focal loss (DFL) for precise bounding box localization and anchors-free detection to reduce hyperparameter sensitivity.

The overall training strategy can be seen as a two-stage domain adaptation pipeline:

1. **Stage 1:** Learn generalized feature representations from a broad dataset with high inter-class and inter-background variance.
2. **Stage 2:** Specialize via fine-tuning on data that approximates the test-time distribution.

This process enables robust deployment with both general and specific capabilities.

Pretraining Dataset

The first stage of training was performed on a curated dataset composed of selected images from four publicly available datasets:

- Plastopol[18]
- Plitterstreet[20]
- TACO [17]
- UAVVaste [22]

To construct a generalizable model for trash detection in natural outdoor environments, images were filtered based on the following criteria:

- Exclusion of indoor environments.
- Exclusion of marine or underwater scenes.
- Inclusion of trash items clearly visible in terrestrial natural settings (e.g., forests, fields, parks, streets).

The resulting dataset contained images with high diversity in lighting, weather, terrain, occlusion levels, and object poses, thereby promoting robustness to domain shift and a higher level of model generalization. Annotations followed the COCO-style format and included bounding boxes for plastic bottles, bags, cans, and general waste, which had been pre-processed to work as a binary classification for the detection model (trash vs. not trash).

Pretraining Procedure

The YOLO11n model was trained for 252 epochs using stochastic gradient descent with momentum. The loss function minimized was a weighted sum of:

$$\mathcal{L}(\theta) = \lambda_{box} \cdot \mathcal{L}_{box} + \lambda_{cls} \cdot \mathcal{L}_{cls} + \lambda_{dfl} \cdot \mathcal{L}_{dfl} \tag{3.1}$$

where:

- $\mathcal{L}_{box}$: Bounding box regression loss (CIoU).
- $\mathcal{L}_{cls}$: Classification loss (Binary Cross-Entropy).
- $\mathcal{L}_{dfl}$: Distribution Focal Loss for enhanced localization.
- λ_*: Task-specific balancing coefficients (default: 1.0).

Validation was conducted using a hold-out validation set of 532 images with 990 annotated objects. Early stopping was enabled, and training was halted at epoch 152.

Fine-Tuning Procedure

To improve the model's deployment performance, the pretrained model was fine-tuned on a second dataset consisting of images captured in the robot's target deployment environment. These environments were matched to the real-world operational contexts, including similar terrain, vegetation, and lighting conditions.

The dataset used for fine-tuning exhibits minimal domain shift relative to the final validation set:

$$\mathcal{D}train^{(fine)} \approx \mathcal{D}val^{(deploy)} \tag{3.2}$$

Fine-tuning was initialized from the pretrained weights θ_0, and continued using the same composite loss function. Training converged after 235 epochs.

3.3.2 Simulation

The robot was modeled using a Unified Robot Description Format(URDF) to define its physical and kinematic properties. In order to keep the model of the robot simple, blocks and cylinders were created then attached with offsets between their respective frames as shown in figure 3.3. The body and collector are treated as fixed frames, and each wheel had its own revolute frame. These frames are all related to each other through a transform tree that shows each frame and its parent that its attached to. Using this it is possible to get transforms that provide information on the position and orientation of each frame with respect to its parent. In this case the parent of all the frames being the base frame. Its this frame that is used to represent the overall position and orientation of the robot in its environment. The overall model is made to represent a differential drive system with four wheels where the left and right sides are linked together.

The main aspects of the code are the robot description, bring up files or launch file and configuration files. The robot description is where the model or representation of the robot is made. The launch files are responsible for loading the model and launching the desired nodes for the simulated environment. Lastly the configuration files define how these nodes are launched. These allow for some customization of the nodes or packages being used to better suit the system being designed.

Nav2 allows for the use of simulated sensor and state data. In the robot description a Lidar or IMU is able to be attached relative to the base frame. From here laser data and IMU data is generated based on the perceived simulated environment. This data is is accessed by the different ROS nodes such as Nav2 and SLAM toolbox via ROS topic.

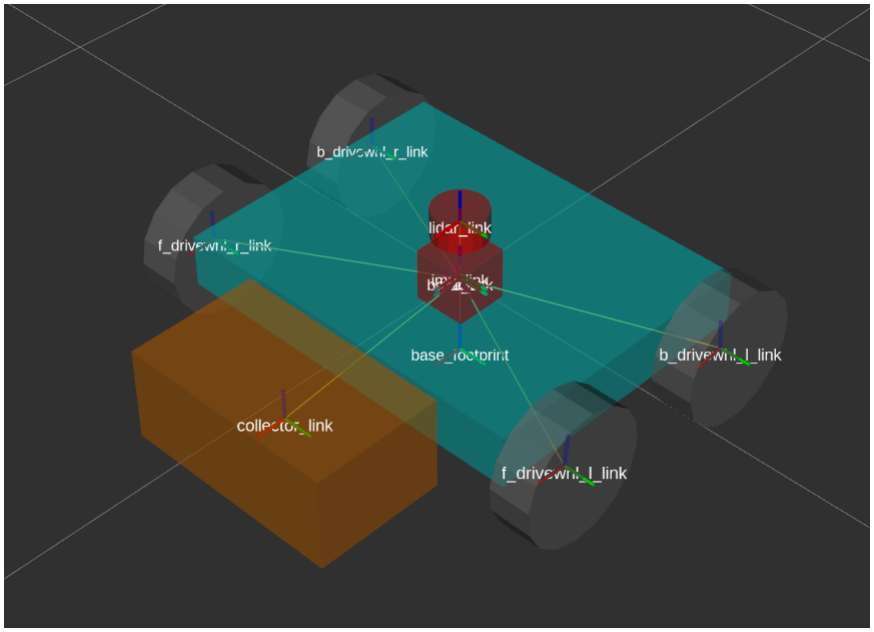


Figure 3.3: Simulated Robot Model

3.3.3 Navigation

System Architecture

The robot utilizes an Ouster OS0 LiDAR sensor, as mentioned in 2.3. The OS0 is a high-resolution 3D LiDAR capable of generating a dense point cloud that represents the surrounding environment in three-dimensional space. Each point in the cloud contains spatial coordinates (x, y, z) and intensity values, which reflect the strength of the laser return, providing additional information about the surface material and distance. The Ouster OS0 operates with a 360-degree horizontal field of view and a configurable vertical field of view, enabling it to capture comprehensive environmental data with each scan. This raw point cloud data is then processed and published as a PointCloud2 ROS message, which is natively supported in ROS and can be directly utilized by ROS Navigation 2 for tasks like obstacle detection, mapping, and path planning. By integrating the LiDAR-generated point cloud into the SLAM and path planning processes, the robot is able to perceive and respond to its environment with high precision and reliability.

Mapping and Localization

SLAM works by taking multiple laser scans and estimating the robot's movement between each scan. In doing so, the robot can create a dynamic map that is updated continuously as it moves through the environment. This map reflects changes such as newly discovered obstacles or the removal of previously detected ones.

The estimation of movement (also called "odometry estimation") can be achieved through various methods, commonly involving sensor fusion. External sensors like wheel encoders, Inertial Measurement Units (IMUs), and even visual odometry (cameras) are integrated with LIDAR or depth sensors to improve accuracy. This combination allows the SLAM algorithm to correct drift and improve localization, ensuring that the map remains consistent and reliable.

WALL-E uses ROS Navigation 2 (Nav2) packages, which include the SLAM Toolbox for Simultaneous Localization and Mapping (SLAM). SLAM Toolbox is a versatile tool that provides several methods for online asynchronous SLAM, enabling real-time map generation and localization during navigation as seen in figure 3.4. It supports both synchronous (blocking) and asynchronous (non-blocking) modes, allowing the robot to continue moving while mapping its environment. This

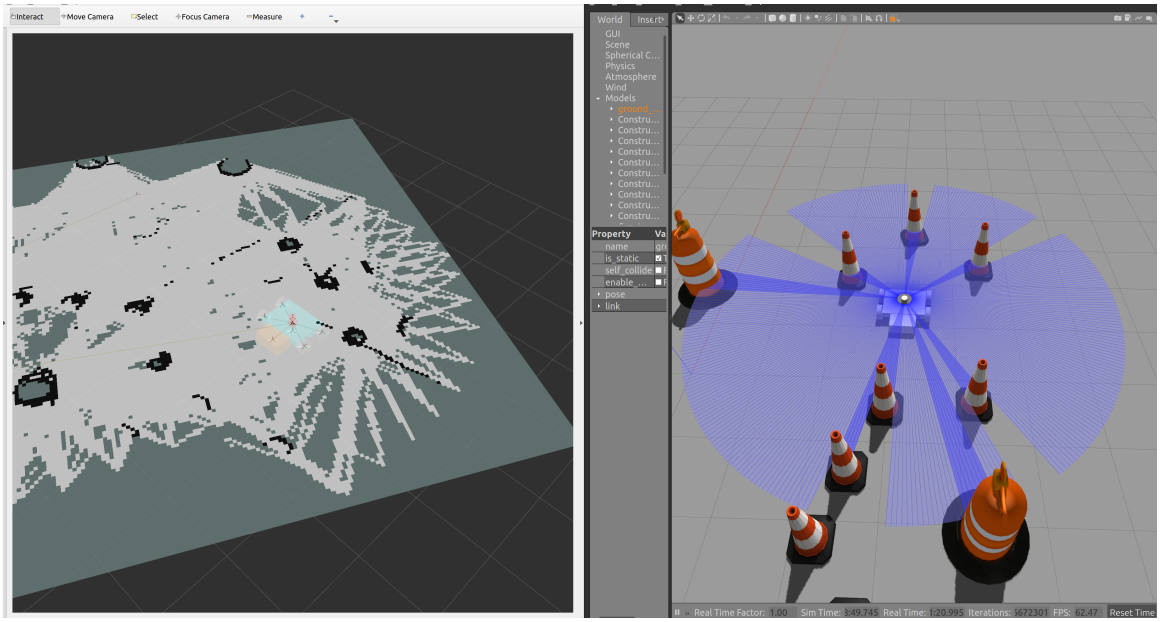


Figure 3.4: Asynchronous SLAM using SLAM Toolbox

is particularly advantageous for autonomous exploration, as it reduces idle time and improves efficiency. SLAM Toolbox also incorporates advanced features like loop closure for correcting drift errors and multi-session mapping for extending and updating maps over multiple exploration sessions. These capabilities enhance the robot's ability to build accurate maps of dynamic or large-scale environments while maintaining reliable localization.

For the odometry estimation a ICM-20948 9 degree of freedom IMU is used. This IMU is a combination of gyroscope, accelerometer and magnetometer. This system utilizes the accelerometer for the translation of the robot in the IMU frame by integrating the linear acceleration over time, and the gyroscope is used for orientation. Equations 3.3 through 3.16

$$\theta(t) = \theta(t-1) + g_x \cdot \Delta t \tag{3.3}$$

$$\phi(t) = \phi(t-1) + g_y \cdot \Delta t \tag{3.4}$$

$$\psi(t) = \psi(t-1) + g_z \cdot \Delta t \tag{3.5}$$

$$R = R_z(\psi) R_y(\phi) R_x(\theta) \tag{3.6}$$

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{3.7}$$

$$R_y(\phi) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \tag{3.8}$$

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

$$\mathbf{a}_{\text{world}} = R \cdot \mathbf{a}_{\text{imu}} \tag{3.10}$$

$$v_x(t) = v_x(t-1) + a_x \cdot \Delta t \tag{3.11}$$

$$v_y(t) = v_y(t-1) + a_y \cdot \Delta t \tag{3.12}$$

$$v_z(t) = v_z(t-1) + a_z \cdot \Delta t \tag{3.13}$$

$$x(t) = x(t-1) + v_x \cdot \Delta t \tag{3.14}$$

$$y(t) = y(t-1) + v_y \cdot \Delta t \tag{3.15}$$

$$z(t) = z(t-1) + v_z \cdot \Delta t \tag{3.16}$$

These equations are the basis for estimating odometry. However the sensor readings typically have a lot of noise. This noise leads to drift in the estimations. This drift can be significantly reduced through the use of filters. One of the most common filters for reducing noise in IMU readings is a complementary filter. An example of this can be seen in equation 3.17. The complementary filter combines the short-term stability of the gyroscope (high-pass filtered) with the long-term stability of the accelerometer (low-pass filtered).

$$\theta_{\text{est}}(t) = \alpha \cdot [\theta_{\text{est}}(t-1) + \omega_{\text{gyro}} \cdot \Delta t] + (1-\alpha) \cdot \theta_{\text{accel}} \tag{3.17}$$

Where:

- $\theta_{\text{est}}(t)$ is the estimated angle at time t.
- $\theta_{\text{est}}(t-1)$ is the estimated angle from the previous time step.
- ω_{gyro} is the angular velocity measured by the gyroscope.
- Δt is the time step duration.
- θ_{accel} is the angle estimated from the accelerometer.
- α is the filter coefficient (typically around 0.98 for high-pass filtering on the gyroscope and 0.02 for low-pass filtering on the accelerometer).

Frontier Exploration

As mentioned earlier online SLAM is a method that can be used for unknown and partially known maps or areas. One method by which a map can be further explored is frontier exploration. This method relies on approaching unknown areas or frontiers and allowing SLAM to further map that area. A frontier is defined by an area that is approachable and bordering an unmapped area. Once frontiers are established the system must choose which frontier to approach. This can be done in a few ways, randomly, distance to the frontier, or the size of the frontier. Typically some cost is applied to each of these factors and the frontier with the best cost is chosen. From here the robot will approach the center of the frontier, stop then reevaluate the map for more frontiers. This is done until a map is deemed to be completed, or in other words there are no more frontiers to explore.

Drivetrain Control

Once Twist Mux decides which twist message to send to the robot the linear and angular velocities need to be converted to left and right wheel velocities. For simulation a gazebo differential drive package is used. This just takes in the wheel frames and their distances in relation to each other to determine the movements of the robot in simulation. For the physical robot an adapter was made to continuously subscribe to the Twist Mux topic and wait for velocity commands to come in from one of the three sources. once a message is received it converts the linear X and angular Z components using the equations 3.18 and 3.19.

$$v_{\text{left}} = \frac{1}{7.75} \cdot \frac{v - \omega \cdot \frac{L}{2}}{r} \tag{3.18}$$

$$v_{\text{right}} = \frac{1}{7.75} \cdot \frac{v + \omega \cdot \frac{L}{2}}{r} \tag{3.19}$$

Where:

- v is the linear velocity command (`linear_x`).
- ω is the angular velocity command (`angular_z`).
- L is the distance between the left and right wheels (wheel separation), in meters.
- r is the radius of the wheels, in meters.
- The factor $1/7.75$ accounts for the gear ratio between the wheel and where the encoder is placed in the gearbox.

The output from these equations was then published to the hardware interface for the motor controllers as velocity targets for the left and right sides respectively. This hardware interface was an adaptation of the ros phoenix adapter [29]. This library made it possible to create ros nodes and messages for each Phoenix CTRE motor controller 2.18. It also made it easy to tune and run velocity controllers on each device. Once the PID parameters were tuned for the subsystem, a set command was simply published to the ros topic created for the desired motor controller.

Path Planning and Nav2

Using the Nav2 navigation packages modular and robust controllers out of the box making it possible to implement with a custom system. These controllers support real time path planning and obstacle avoidance in both simulation and real world environments.

Nav2 streamlines implementation by providing features that streamline navigation and path planning like global and local planners, and obstacle avoidance strategies that use cost maps like the ones shown in figure 3.5.The main components of the nav2 stack are seen below.

Global Planner: This component generates an optimal path from the robot's current position to the goal location using map data. The default planner used was `NavFn`, which implements Dijkstra's algorithm to compute the shortest path over a grid-based costmap. This ensures global optimality while accounting for known static and dynamic obstacles.

Local Planner (Controller): For real-time motion control, the `DWB (Dynamic Window Approach) Controller` was used. This planner evaluates possible velocity commands in a short time horizon and selects the one that best aligns with the global path while avoiding obstacles. It ensures smooth, safe trajectory following by taking into account the robot's dynamics and nearby hazards.

Costmaps: Nav2 maintains both global and local costmaps, which represent obstacle information at different scopes. The global costmap is used by the global planner to compute a full path to the goal, while the local costmap is continuously updated with sensor data from the lidar for dynamic obstacle avoidance.

Recovery Behaviors: Built-in recovery behaviors such as rotate recovery and backtracking were employed to help the robot escape from stuck or uncertain states.

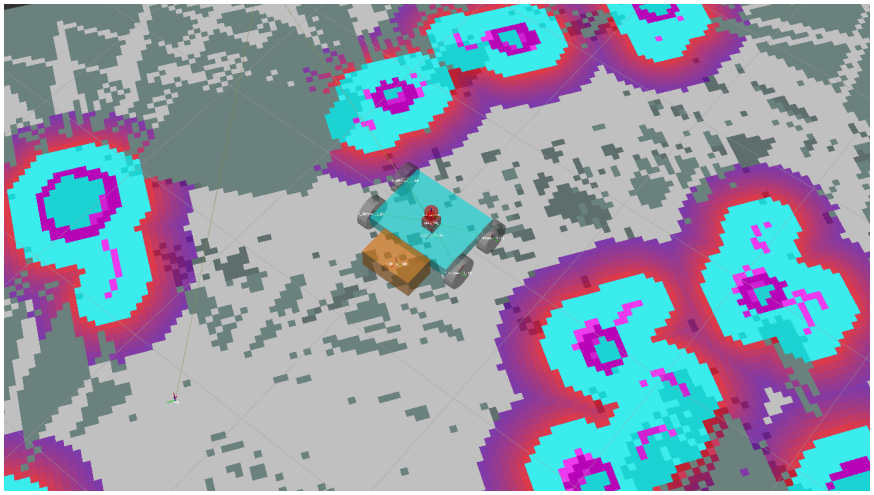


Figure 3.5: Nav2 Cost Map and Simulated Robot

SLAM Integration: If operating in an unknown environment, SLAM (Simultaneous Localization and Mapping) can be used to build a map in real-time. For simulation, a pre-existing map was used to demonstrate planning capabilities.

Using these tools allows for quick integration of autonomous navigation capabilities into both simulated and real-world robots, reducing development time while maintaining flexibility, modularity, and robustness in planning, control, and recovery behaviors.

3.4 Results

This sections describe the results from the litter detection model. Since navigation has not been implemented, there are no results from that part.

The results from both the pretrained (YOLOv11n-pre) and fine-tuned model (YOLOv11n-ft) can be seen in Table 3.1, where it is seen that the general model has a decent performance, which is increased further in the fine-tuned model.

Model	Dataset	Precision	Recall	mAP@0.5	mAP@[0.5:0.9]
YOLOv11n-pre	General (Outdoor Mix)	0.838	0.722	0.816	0.573
YOLOv11n-ft	Fine-Tuned (Local Env)	0.925	0.885	0.937	0.719

Table 3.1: Litter Detection Results

This performance increase in the fine-tuned model can be attributed to the high alignment between the fine-tuning and validation datasets' environments. In the detections for one of the validation batches for the general model, it could be seen that the model wrongly identified background features as litter. This can be seen in Figure 3.6, where it can be seen that there line between the tiles, bird poo, and the sewer cap is sometimes wrongly detected. This is because the general model's dataset environment has a high level of discrepancy from the validation dataset.

Looking at the training and validation graphs for the YOLOv11n-pre model in Figure 3.7, it can be seen that the loss curves for both training decreased smoothly, whereas the box loss for the validation stops decreasing and starts increasing slightly after $\sim$ 80 epochs, with extreme fluctuations. This points to the model slightly over-fitting on the bounding box sizes of the training dataset, and not improving beyond that point. The classification loss, on the other hand, steadily decreases in both the training and validation sets, without significant fluctuations. This points to the model

Ground Truth Predicted

Figure 3.6: Validation Batch Ground Truth vs. Predicted for YOLOv11n-pre

classification not over-fitting. The same is seen in the metrics, where the training and validation metrics steadily follow each other.

When looking at the fine-tuning training and validation curves in Figure 3.8, it is seen that the model converged well on the new dataset. Training losses decreased steadily, while validation losses followed similar downward trends with expected fluctuations due to validation set variability. The model achieved a precision of up to 0.98 and a recall ranging from 0.75 to 0.88, indicating robust detection capability with minor room for improvement in recall. The mean average precision reached approximately 0.94 at IoU=0.5 and 0.72 at IoU=0.5:0.95, demonstrating strong localization and classification performance across a range of detection thresholds. Although the fluctuations in the metrics seem extreme, this is due to the y-axis' granularity. It can be seen that the fluctuations only fluctuate around 0.08 points.

Given that the training and validation loss curves showed similar trajectories, it is indicated that overfitting was effectively mitigated. In particular:

$$|\mathcal{L}train - \mathcal{L}val| < 0.05 \quad \text{for all epochs after 50} \tag{3.20}$$

The mAP@0.5:0.95 metric improved from 0.57 to 0.72, highlighting the model's increasing localization accuracy across varying IoU thresholds. These results confirm the model's effectiveness for robust and precise object detection on the validation dataset.

The model speed was measured to be 0.3ms for preprocess, 2.4ms for inference, and 0.6ms for postprocess per image. This results in a speed from the time after an image is taken to when the detections have been identified of $\sim$ 3.3ms. This shows that the YOLO 11 nano model is indeed capable of a solid detection performance while running in a real-time application.

3.5 Discussion

The results of the fine-tuning procedure clearly demonstrate the benefits of domain-specific adaptation for object detection in real-world applications. The two-stage training approach follows the principles of transfer learning and domain adaptation [30], where a base model is pretrained on a diverse dataset and subsequently adapted to a target domain. This strategy leverages both the generalization capability of the pretrained weights and the specificity gained through fine-tuning.

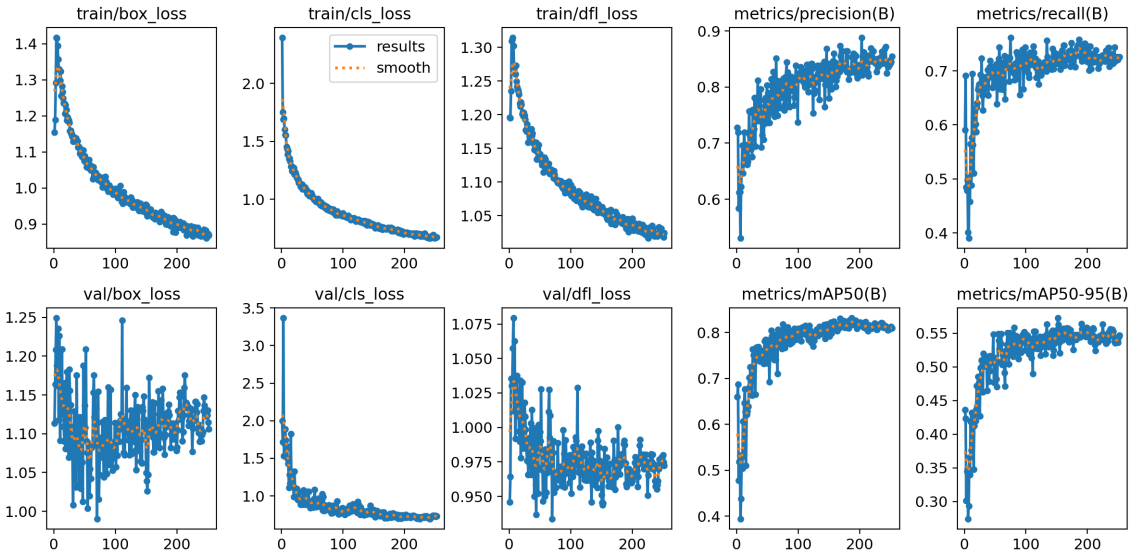


Figure 3.7: Training and Validation graphs for YOLOv11n-pre

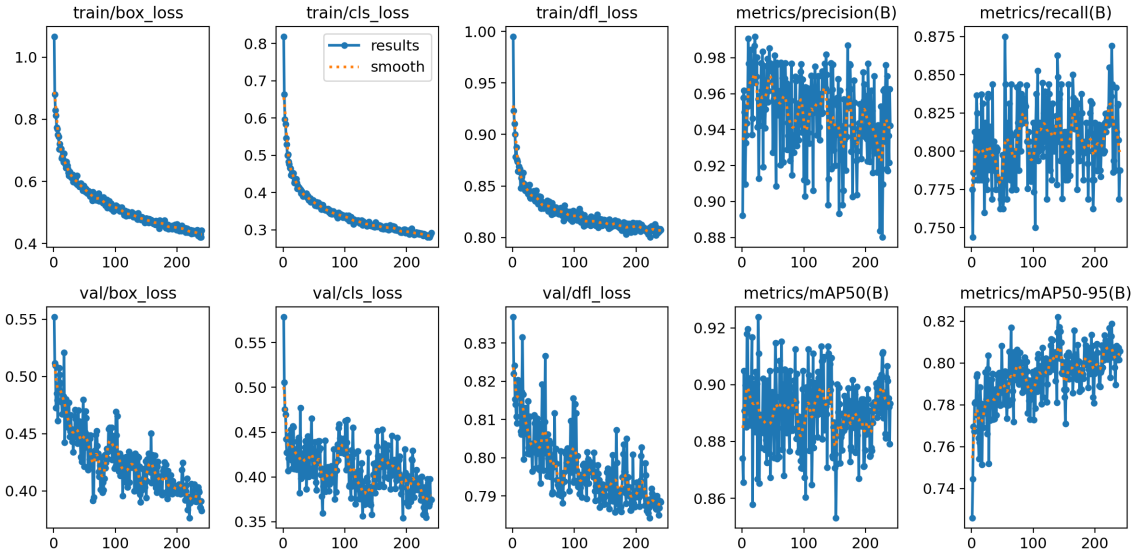


Figure 3.8: Training and Validation graphs for YOLOv11n-ft

The performance gains observed during fine-tuning (e.g., mAP@0.5:0.95 >0.82) can be attributed to the close alignment between the training and validation distributions. This suggests that the model benefits from the reduced domain gap and is able to learn task-relevant features with greater fidelity. The increasing validation mAP over time and absence of overfitting (as indicated by aligned loss curves) further support this claim.

From a theoretical perspective, the method of a generalized model to fine-tuning it to the environment minimizes the empirical discrepancy between the model and target domain [31].

Overall, these findings highlight the critical importance of environment-specific fine-tuning when deploying deep learning models in structured, real-world robotic applications.

Navigation for this system was initiated in simulation utilizing tools like RVIZ and Gazebo. This allowed for testing of different packages to be done with simulated sensing. For instance the SLAM toolbox was first testing with a simulated environment containing a representation of the robot, and a laser scanner as seen in figure 3.4. This allowed for initial testing to be done completely in simulation before moving to the real robot.

Due to the nature of how messages are handled in ROS and by the SLAM toolbox, it was pretty simple to switch from using a simulate sensor to using the physical one. Ouster provides its own ROS hardware interface. By building and running their pre made Docker containers it becomes very easy to expose the sensor readings as topics. Once they are exposed SLAM toolbox is launched with the configuration pointing at the newly exposed scan topics. The only problem to be solved here is that the sensor readings and the sim states need to have similar timestamps. A simple adapter was made to handle this to make sure sim and real time readings from the sensor matched.

Initial testing showed that the physical sensor published massive amounts of data and SLAM toolbox was having a hard time keeping up. Often the message ques for the scan topic would become full. For the time being the goal is to have the system be as real-time as possible. To accommodate these things an asynchronous version of SLAM was used to make sure only the most recent sensor reading is used when mapping and localizing.

3.6 Conclusion

This study demonstrates the effectiveness of fine-tuning a lightweight YOLO11n model for robust trash detection in natural outdoor environments. By employing a two-stage training pipeline, i.e. general pretraining followed by domain-specific fine-tuning, the model achieves both high accuracy and operational efficiency. The marked improvement in detection metrics (e.g., mAP@0.5:0.95 >0.82, precision >0.95) validates the importance of aligning training data with deployment conditions. These findings underscore the value of domain adaptation in practical robotic applications, especially when computational resources are limited and environmental variation is significant. The fine-tuned model complied with the desired detection accuracy of >90% litter detected.

Chapter 4

Discussion

The development of WALL-E as a litter-collecting robot works as a proof of concept for automated environmental cleanup. However, while the system meets many of the initial objectives, in hardware design, perception, and trash collection, it remains an early-stage prototype. This chapter explores the technical and practical implications of the design choices, experimental results, and current limitations, while proposing concrete directions for improvement.

4.1 Mechanical Performance and Design

The drivetrain of WALL-E was designed to handle a variety of outdoor terrains, including grass, pavement, and asphalt, with successful traversal of inclines up to $\sim 20°$. The test results confirmed that the robot can navigate most common urban outdoor settings. However, the reliance on small caster wheels for the front support structure presented a significant bottleneck. These wheels struggled on uneven or soft surfaces like grass, where they caused the robot to bounce or get caught, lifting the collector arm and impacting trash pickup reliability. Replacing these small front caster wheels with larger pneumatic casters could drastically improve terrain adaptability, however, it would be a more expensive solution than the small caster wheels. A cheaper solution here would be using the arms through vision techniques to identify the terrain type, and dynamically lift the arm by five centimeters to account for the needed collector height difference. A hybrid solution that combines the passive leveling attributes of the mechanical four-bar and an active closed-loop software solution that can react to the system's environment would drastically improve the robot's ability to handle different environments and obstacles.

Another major issue was motion control granularity. The drivetrain motors sometimes responded too aggressively, especially at low speeds or when starting from rest. This "jerky" behavior introduced unwanted vibrations, which not only reduced system stability but occasionally caused electrical components to disconnect due to poor cable connections. These issues can be alleviated using a software solution called motion profiling (e.g., trapezoidal velocity profiles or S-curve acceleration). It would smooth velocity transitions, reducing mechanical strain and vibrations. Another, easier solution would be to configure the Victor SPX motor controllers to limit the ramp rate and startup current using their implementation of trapezoidal profiling called Motion Magic [32]. Similar issues could be seen with the arm. On larger motions, it was hard to limit the angular velocity of the arm as it moved to a position, resulting in belt skipping or overshooting as it approached a target. Motion magic would allow for target velocities to be tuned throughout the arm's path of motion.

The choice to use custom-printed pulleys for all torque allowed for fast implementation and testing of systems while proving to be cheap and robust enough for the prototype stage of this system. There were a few instances where a pulley was used without the steel reinforcement shown in Figure 2.5. The solution in these cases was to reprint the pulley with a higher infill and add the support plates on either side of the pulley. In the future, off-the-shelf aluminum pulleys should be used for a longer life span of the pulley.

On the electronic side, one Raspberry Pi failed, likely due to static discharge. Improvements in grounding, voltage protection, and overall circuit robustness should be ensured to make WALL-E commercial-ready. Battery life and charging time also present a bottleneck, limiting long-duration deployments. The overall weight of the system could also be greatly reduced by switching to a different battery type, like Li-Po.

4.2 Collector Design and Litter Handling

The trash collection system was arguably the strongest-performing subsystem. The twin roller brush design enabled the successful pickup of a wide variety of litter types: cardboard, plastic, metal, glass, and paper, with a great 98.4% success rate in the optimal configuration; straight brush pattern and collector down on pavement and asphalt, and collector up on grass.

This change in collector height on grass vs. pavement and asphalt does provide a need for active height adjustment of the collector. With this, WALL-E would be able to dynamically change the terrain. This, however, is not necessarily a needed improvement to be able to commercially deploy WALL-E, as the deployed areas most likely would be in a homogeneous terrain.

More critical challenges were observed with very small, lightweight items like torn newspaper pieces and threads. These tended to get stuck between brushes or be thrown out of the collector, suggesting the need for a brush-cleaning mechanism or a comb-like component to improve reliability. Adding a "combing" mechanism between and beneath the brushes could eliminate the dead zone that caused small, lightweight items to get stuck on the brushes, rotating with them, and at times escaping collection.

It was also seen that the collector walls occasionally flexed under load when collecting big, rigid trash pieces, allowing the axle to dislodge. Reinforcing the side panels with cross-beams would greatly enhance the collector's robustness.

4.3 Perception and Detection

In terms of perception, the fine-tuned YOLOv11n model achieved high precision and recall metrics, which translated into reliable real-time litter detection in outdoor environments. The success of the domain-specific fine-tuning emphasizes the importance of aligning training data with deployment environments.

Despite the high accuracy, certain organic objects (e.g. bird poo) occasionally triggered detections. A simple rule-based or secondary classification filter could reduce these.

4.4 Autonomy and Next Steps

While WALL-E is equipped with a powerful 3D LiDAR and implements SLAM using ROS 2, it does not yet support autonomous navigation or exploration. This is one of the most significant limitations of the current system. The robot relies on manual control for both driving and litter collection.

To reach full autonomy, several additional developments are needed: Firstly, WALL-E needs an exploration algorithm. Implementing a frontier-based or information-theoretic exploration strategy would allow WALL-E to autonomously map and traverse a defined area. This defined area would need to be defined in some manner, either through a user interface where the user can draw on a map, or by having the user drive WALL-E around the border of the area, while WALL-E measures the border using a GPS module. The exploration would need to be integrated with a global path planner (e.g. ROS's Nav2 node) using the SLAM map data. A local reactive planner for dynamic obstacle avoidance is also needed, such as DWA (Dynamic Window Approach) or TEB (Timed Elastic Band). With this, the system can drive and explore an area autonomously, and the vision trash detections can then be used to align WALL-E to the litter and collect it through visual servoing.

4.5 Overall Robot Performance

In terms of a solution as a whole, in order for an autonomous solution for litter collection to become more relevant, the system would need to expand into innovations outside of the robot itself. A method to empty the collected trash would allow the robot to run without human interference for even longer, and the ability to operate at night would limit the robot's interaction with by-passers in the environment. As mentioned previously, a method by which the robot can explore its environment needs to be implemented, but this could also evolve into external devices reporting the presence and location of litter in the robot's operating space.

Overall, this project successfully developed a modular, low-cost prototype capable of real-time litter detection and physical collection. While WALL-E does not yet feature autonomous navigation, it provides a solid mechanical and perceptual foundation to build on. This discussion has highlighted key areas: terrain adaptation, collector robustness, electronic reliability, and autonomous navigation, offering the most immediate impact for future development.

By addressing these areas, WALL-E can move from a minimal proof of concept to a deployable system capable of operating independently in real-world environments.

Chapter 5

Conclusion

This thesis explored the design and realization of an autonomous robot, WALL-E, capable of cleaning public outdoor spaces by detecting and collecting various types of litter. The overall objective was to create a system capable of navigating urban and natural terrains, detecting a wide range of litter types, and collecting them with high reliability. Although the final system does not yet achieve full autonomy, the robot successfully met the core functional objectives and lays the groundwork for further autonomous development.

The robot's drivetrain was designed to operate across challenging terrains; grass, pavement, asphalt, and inclines of up to 20 °. The robot maintained high mobility on all surfaces, with a minor reduction in speed on steep grass inclines. Although mechanical stability was occasionally compromised, particularly due to vibrations and caster wheel performance on uneven terrain, WALL-E met all locomotion criteria of being able to drive on the mentioned terrains. Improvements such as upgrading caster wheels and better vibration tolerance were identified as future upgrades.

The trash collection system was rigorously tested using 15 diverse trash types across the three terrain types. The final system achieved a 98.4% collection success rate in its optimal configuration (straight brush pattern and the collector up for grass, and down for the two other terrains), surpassing the 95% target set for reliable litter collection. The robot consistently handled soft, irregularly shaped, and deformable objects, although minor limitations were found in picking up big, rigid items, caused by the collector walls' flexibility.

The perception system, based on a fine-tuned YOLOv11n deep learning model, demonstrated a strong detection performance with a maximum F1 score of 0.88 and mAP@0.5 of 93.7%. The system reliably differentiated between litter and natural background features, keeping false positives low and enabling stable collector performance.

These results highlight WALL-E's potential to reduce the need for manual litter collection. However, it still lacks autonomous navigation and exploration. Although SLAM with the onboard LiDAR is functional, the robot cannot yet explore or move independently. Implementing an exploration algorithm, along with safety features, obstacle avoidance, and adaptive collector control, will be key steps toward full autonomy and commercial viability.

In conclusion, WALL-E met the majority of its technical objectives, particularly in driveability, litter detection, and collection success. While full autonomous functionality is still under development, the system presents a promising, scalable platform for robotic litter collection. With targeted improvements in motion control, electronic reliability, and software autonomy, WALL-E could evolve into a viable tool for real-world deployment in public parks, sidewalks, and open spaces, offering a sustainable, low-cost alternative to manual litter collection.

Chapter 6

Individual contributions

Due to the special case of the exam being taken individually, the individual contributions have been divided into two tables: one for the project contributions (Table 6.1) and one for the report contributions (Table 6.2). The cells marked with "responsible" mean that the person has been over all aspects of that area, whereas the cells marked with "minimal efforts" mean that the person has not contributed significantly to that area.

Area	**Isabella Mia Lin Nielsen**	**Nikolas Neathery**
Drivetrain design & testing	Co-developed design concept, and testing responsible	Lead on drivetrain mechanical design and assembly
Collector system	Testing, and final integration	Lead on collector design, mechanical integration, and arm control
Electronics & wiring	Minimal efforts	Responsible
Vision system	Responsible	Minimal effort
SLAM integration	Minimal efforts	Responsible
ROS 2 & software	Manual control and brush control	Communication nodes, Twist mux, motor controller, PID tuning, and ROS setup
Testing & validation	Responsible	Prototype collector test

Table 6.1: Project work contributions

Report Part	**Primary Author**
Introduction	Isabella
Mechanical design	Nikolas
Electronics	Nikolas
Experiments & Results	Isabella
Software	Nikolas (Navigation, ROS, Simulation), Isabella (Vision, Manual Control, ROS)
Discussion	Joint effort
Conclusion	Joint effort
Figures and diagrams	Nikolas (CAD renders & electronic diagrams), Isabella (photos & software diagrams)

Table 6.2: Thesis writing contributions

Bibliography

[1] Ridly Rubbish Removal, "Facts about littering," 2025. Accessed: 2025-02-14.

[2] J. Lee, M. Kim, S. Park, and J. Choi, "Oatcr: Outdoor autonomous trash-collecting robot design using yolov4-tiny," *Electronics*, vol. 10, no. 18, p. 2292, 2021.

[3] C. Kao, A. Pathapati, and J. Davis, "Ai for green spaces: Leveraging autonomous navigation and computer vision for park litter removal." `https://escholarship.org/uc/item/9rw6h86w`, 2025. Accessed: 2025-05-10.

[4] 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.

[5] Hackaday, "Tank tracks," Accessed 2025. Accessed: 2025-06-01.

[6] VEX Robotics, "VEXpro CIM Motor," Accessed 2025. Accessed: 2025-06-01.

[7] VEX Robotics, "VEXpro 775pro Motor," Accessed 2025. Accessed: 2025-06-01.

[8] Luxonis, "Oak-d lite," 2025. Accessed: 2025-02-17.

[9] I. Ouster, "ouster-ros: Drivers and tools for using ouster lidars with ros." `https://github.com/ouster-lidar/ouster-ros`, 2024. Accessed: 2025-05-29.

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

[11] R. Khanam and M. Hussain, "Yolov11: An overview of the key architectural enhancements," 2024.

[12] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "You only learn one representation: Unified network for multiple tasks," 2021.

[13] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," 2020.

[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2018.

[15] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2016.

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2018.

[17] P. F. Proença and P. Simões, "Taco: Trash annotations in context for litter detection," 2020.

[18] M. Córdova, A. Pinto, C. C. Hellevik, S. A.-A. Alaliyat, I. A. Hameed, H. Pedrini, and R. d. S. Torres, "Litter detection with deep learning: A comparative study," *Sensors*, vol. 22, no. 2, 2022.

[19] G. Jocher, A. Chaurasia, A. Qiu, J. Stoken, J. Borovec, C. Kwon, Y. TaoXie, D. Michael, L. Fang, *et al.*, "Yolov5: An improved version of yolo for object detection," 2020. YOLOv5 is a family of object detection models optimized for real-time applications, providing a balance between accuracy and computational efficiency.

[20] S. R. Mandhati, N. L. Deshapriya, C. L. Mendis, K. Gunasekara, F. Yrle, A. Chaksan, and S. Sanjeev, "plitterstreet: Street level plastic litter detection and mapping," 2024.

[21] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[22] M. Kraft, M. Piechocki, B. Ptak, and K. Walas, "Autonomous, onboard vision-based trash and litter detection in low altitude aerial images collected by an unmanned aerial vehicle," *Remote Sensing*, vol. 13, no. 5, 2021.

[23] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[24] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[25] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): Part ii," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[26] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE Int. Symp. Safety, Security, and Rescue Robotics (SSRR)*, pp. 155–160, IEEE, 2011.

[27] S. Macenski, F. Martín, R. White, and J. García, "Slam toolbox: Slam for the dynamic world," in *Proceedings of ROSCon*, 2020.

[28] S. Macenski, M. Shoemaker, and B. Cook, "The navigation2 project," in *Proceedings of ROSCon*, 2021.

[29] V. Robotics, "ros_phoenix: Ros 2 driver for ctre phoenix devices." `https://github.com/vanderbiltrobotics/ros_phoenix`, 2024. Accessed: 2025-05-26.

[30] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[31] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of representations for domain adaptation," in *Advances in Neural Information Processing Systems* (B. Schölkopf, J. Platt, and T. Hoffman, eds.), vol. 19, MIT Press, 2006.

[32] C. T. R. Electronics, "Chapter 16: Closed loop control," 2024. Accessed: 2025-05-29.