WALT

Wheeled Autonomous Locomotion Traveler

Leon Greiner, Vineeth Parashivamurthy, Yichen Hu, Yitong Wu, Zichu Zhou

December 11, 2024

1 INTRODUCTION

1.1 Objective

The objective of this project is to design, develop, and implement a wheeled quadruped robot capable of performing autonomous navigation in a controlled environment. The robot will integrate both walking, climbing and driving capabilities, allowing it to:

1. Drive fully autonomously in a controlled environment from a start to an end point.
2. Detect and avoid obstacles autonomously while maintaining a smooth path.
3. Climb and overcome stairs.
4. Switch autonomously between driving and climbing modes based on environmental conditions, such as encountering an staircase.

This project aims to combine real-time sensor data processing (RGB camera) with intelligent decision-making algorithms to allow the robot to autonomously navigate. Moreover, the robot has to be designed, and a sophisticated selection of the hardware components (such as servos, microcontroller, sensors, etc.) is necessary, to ensure that all elements are combined effectively into a mobile system. Particular focus is on the development of a control system of the locomotion and gaits.

1.1.1 Target Route

The target route for the robot is designed to test its ability to navigate both driving and climbing modes in a structured environment. The route goes through a corridor, includes dynamic obstacle avoidance, and a staircase climbing task. Below is a detailed breakdown of the route and its key features:

1. **Start:** The robot begins at the designated Start Point, located in a corridor.
2. **Climbing an Obstacle:** To test and demonstrate the robot's climbing and walking capabilities, it will encounter a staircase that requires a transition from driving mode to climbing mode.
3. **Driving along the corridor:** The robot drives fully autonomously through the aisle and should be able to navigate past various obstacles.
4. **Goal:** After successfully completing the route, the robot reaches a predefined endpoint.

This route evaluates the robot's capabilities in obstacle detection and avoidance, control and mechanical design, seamless transitions between driving and climbing modes, and successfully overcoming challenges to reach the predefined endpoint.

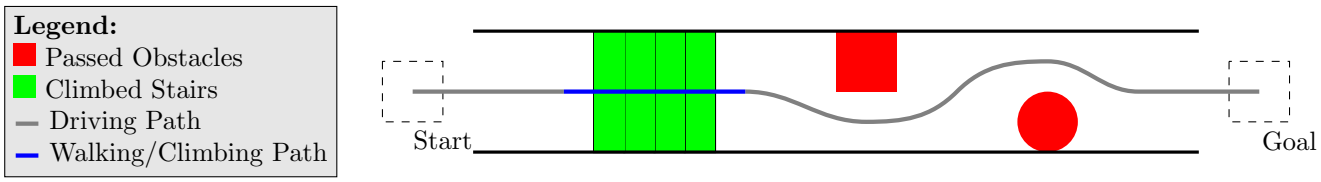


Figure 1: Target route of the project

1.1.2 Real-World Application of the Project

The quadruped robot is well-suited for a variety of real-world applications. It excels in autonomous delivery, particularly for last-mile logistics, where fast-paced driving on flat surfaces is required, but the ability to overcome obstacles, such as stairs or curbs, is essential. In industrial environments, such as factories and warehouses, the robot is ideal for navigating primarily flat surfaces at high speeds while still being capable of overcoming occasional obstacles, ensuring smooth and efficient operations. Its versatility makes it valuable in any environment where fast driving is the primary mode of transport, but the ability to switch to walking or climbing is mandatory to handle challenging obstacles.

2 SYSTEM ARCHITECTURE

The figure [2] illustrates the hierarchical architecture of the robot system, showing the flow of data and control signals through its main components:

1. **Perception & Path Planning**: This module processes RGB camera data on a laptop to detect drivable surfaces and obstacles such as stairs. Using this information, a path planning algorithm calculates optimal routes and sends high-level commands to the robot.
2. **Electronics & Microcontroller**: This component manages the execution of commands from the path-planning system, handling tasks such as motor control, sensor communication, and interfacing with the perception module.

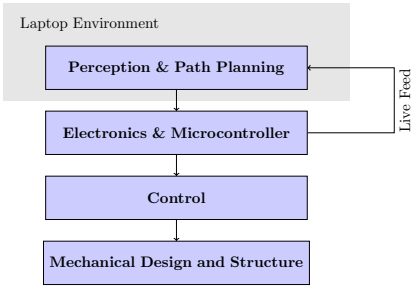


Figure 2: Systems Architecture

3. **Control**: The control system converts the high-level decisions and sensor inputs into precise commands for the actuators, ensuring stable and effective locomotion.
4. **Mechanical Design**: This subsystem physically carries out the movements based on control signals, enabling the robot to interact with and navigate its environment.

The system includes a live feedback loop that allows the robot to send real-time data back to the perception module. This ensures continuous monitoring and enables dynamic adjustments for efficient decision-making in changing environments.

3 MECHANICAL DESIGN

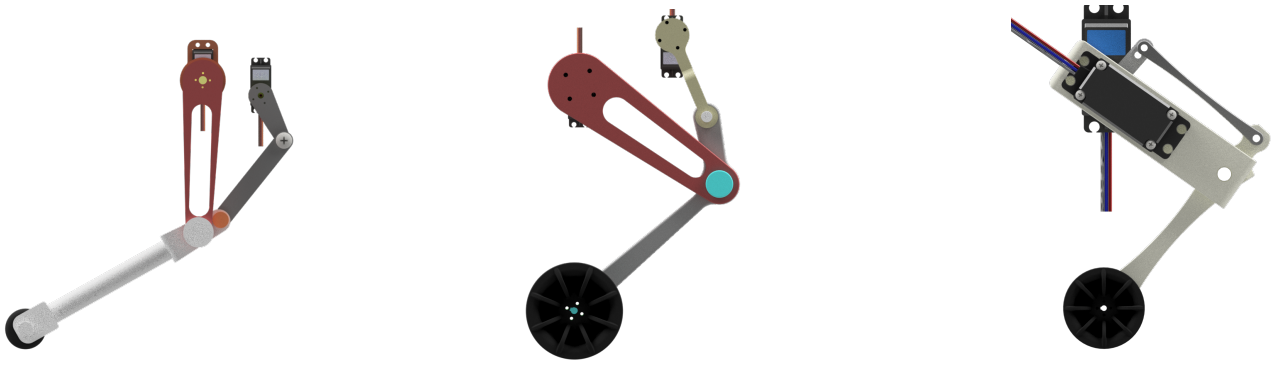
The mechanical design process initially focused on the leg's conceptualization and development, following an iterative approach. This focus was deliberate, as the leg is the most critical component of the robot, directly influencing its overall performance and capabilities. Factors such as the robot's ability to walk, traverse various terrains, and climb obstacles (and the limitations of these capabilities) are all heavily dependent on the leg design. As a result, this aspect became the primary focus during the initial development phase. Subsequently, the main body of the robot was designed to house all essential components, including the MCU, a smartphone for camera-based perception, the battery, and the mounting points for the four legs.

3.1 Leg Design

As mentioned, the leg design is the most crucial aspect of the robot's mechanical system. For this project, we opted to develop an 8-DOF robot, with 2 DOF per leg. Each leg is equipped with a hip pitch joint and a knee joint, but the hip roll joint (responsible for abduction and adduction) is absent. This decision was made to simplify the design and reduce resource requirements for the project. Additionally, omitting the hip roll joint eased control complexity, offering advantages in terms of simplicity and efficiency.
However, this simplification does come with trade-offs. The lack of a hip roll joint restricts the leg's workspace to a 2D plane, limiting the robot's mobility in uneven terrain or under external disturbances. For instance, a lateral force (e.g., a side impact) could destabilize the robot and potentially cause it to fall over. Nonetheless, given the objectives and constraints of this project, the absence of the hip roll joint was deemed acceptable, albeit requiring careful implementation of walking and climbing controls to ensure stability.

3.1.1 Iterative Design Process

Recognizing the critical importance of the leg design, the development process followed an iterative approach, with multiple prototypes tested in real-life scenarios. This iterative process, illustrated in Figure [3], progressed from the initial prototype (V1) to the final version (V3). All prototypes used linkages 3D-printed from PLA. A preliminary stress analysis of the V1 PLA linkages showed a safety factor of 14.7, indicating sufficient strength. Consequently, further stress analyses were deemed unnecessary for subsequent iterations.



(a) Leg Design V1 (b) Leg Design V2 (c) Leg Design V3

Figure 3: Evolution of the Leg Design

V1: The first prototype featured a basic design with a dummy wheel as the end effector. It employed a rod system for actuation, with two rods connecting the servo to the joint. However, this design introduced significant limitations: the hip and knee joints were not independently controllable, leading to complex control requirements. Additionally, the workspace was restricted, making walking gaits challenging. In some

cases, the entire leg would lock due to mechanical interference between the rods, particularly when the knee servo counteracted the hip servo.

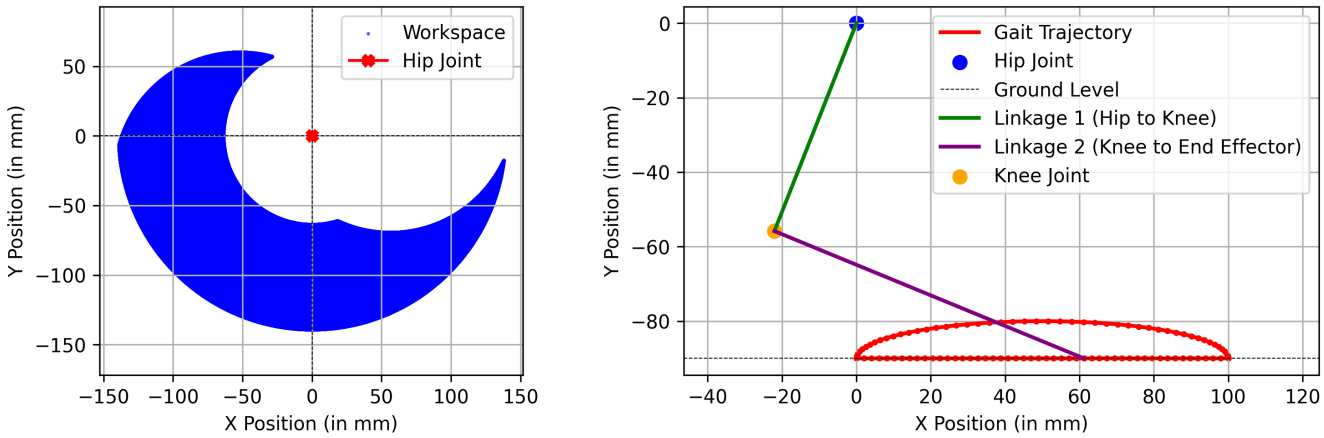
V2: To address the issues in V1, the second iteration reduced the rod system to a single rod, simplifying the design. While this improved the control slightly, the fundamental problem of limited independent joint control persisted, along with the workspace and locking issues.

V3: The final version introduced a significant improvement. While still using a rod system, the design relocated one of the servos. In V3, only the hip servo is fixed to the robot's body, while the knee servo is mounted on the moving thigh linkage, sharing the same rotational axis as the hip servo. This configuration allows full independent control of both joints, addressing the limitations of the previous designs. Additionally, the overall design was resized and made more compact, simplifying fabrication and improving control compatibility with the available servos.

3.1.2 Gait Cycle Simulation

First, we verified the capabilities of the leg design to meet the objectives of this project through a workspace analysis. The workspace of the final leg design (V3) is shown in Figure [4a], highlighting the range of motion of the end effector (wheel). The analysis considers a hip joint angle range of -150° to 20° and a knee joint angle range of -45° to -135°. From the workspace diagram, it is evident that the leg is well-suited for our purposes, with a sufficient range for walking and climbing, particularly due to the extended forward reach (visible on the left side of the diagram).

Building on this, we designed an appropriate gait trajectory within the leg's workspace to ensure an efficient and functional gait cycle. Using inverse kinematics, we calculated the joint angles required to follow this trajectory, and the resulting end-effector path is plotted in Figure [4b]. The trajectory exhibits the characteristic elliptical shape with a flat section on the ground, representing the ground contact phase. This trajectory aligns with expectations for a functional gait and supports stable locomotion. When this trajectory is synchronized for the right front and left back legs, as well as for the left front and right back legs, the robot achieves a trot gait.



(a) Workspace of the 2 DOF V3 Leg

(b) Gait Trajectory

Figure 4

3.2 Main Body

The design of the main body was developed after the completion of the leg design and was adapted to accommodate the servo mountings and dimensions of the legs. The size of the body was determined based on the leg dimensions, ensuring sufficient space to securely mount all four legs. Once the leg integration was finalized, additional mounting points were implemented to house components such as the microcontroller, battery, and phone. To reduce weight while maintaining structural integrity, the body features a perforated honeycomb pattern (see Figure [5a]). This pattern not only minimizes weight but also provides adequate airflow for cooling the components inside the housing. The phone holder (see Figure [5b]) was designed with a differential approach, allowing the phone to be securely mounted in the holder, which can then slide easily and securely into the main body. This ensures stability while enabling easy assembly and disassembly. Like the leg components, the main body and phone holder were fabricated using 3D printing with PLA material, ensuring a lightweight yet robust construction.

(a) Main Body

(b) Phone Holder

(c) Prototype assemble of WALT featuring leg design V2

Figure 5

4 ELECTRICAL DESIGN

The complete electrical diagram illustrates the structure of WALT's electrical system, detailing the connections of all eight servos used for walking and climbing, as well as the four DC motors responsible for driving, all integrated with the Arduino MEGA (for clarity in the diagram, we used in the diagram two Arduino UNOs).

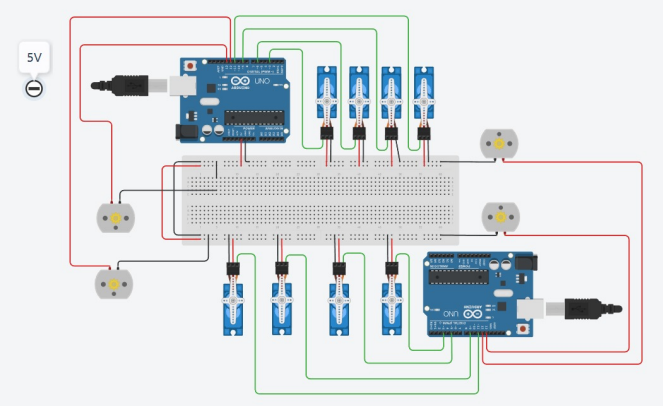
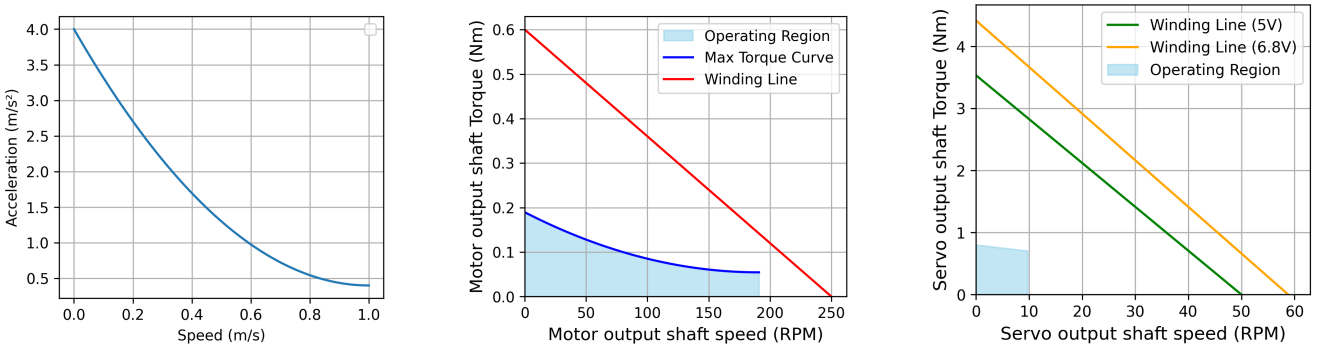


Figure 6: Electrical Diagram

4.1 Performance analysis

In our performance analysis, we examined the requirements for the DC motors used in the wheels and the servos controlling the joints. The requirements for the wheel motors were calculated by estimating the robot's mass, wheel dimensions, and setting a target maximum speed of 1 m/s. Additionally, we incorporated a decreasing acceleration profile relative to the speed (see Figure [7a]) along with an estimation of other contributing factors. For the joints, the torque and speed requirements over a gait cycle were determined using a Lagrangian-based analysis derived from gait cycle simulation (see Figure [4b]). Based on this evaluation, we selected a wheel motor ("DC TT Motor") and a servo ("Servo DS3240-270") that met these requirements. We then verified that the chosen motors not only fulfilled the calculated demands but also provided sufficient safety margins. Figures [7b] and [7c] illustrate the winding lines of the wheel motor and the servo, respectively. These were calculated using data obtained from the datasheets ([1], [2]) and aligned with the corresponding requirements for driving and walking functionality. In both cases, the motors meet the requirements. However, it is notable that the servos exceed these requirements with ease. This is because we opted for more robust servos, which were only marginally more expensive, offering greater reliability without significantly increasing costs.



(a) Desired Max. Acceleration vs. Speed when Driving

(b) Motor Performance Analysis

(c) Servo Performance Analysis

Figure 7

4.2 Quadruped Hybrid Movement Algorithm

The Algorithm coordinates the motion of a quadruped robot's legs and wheels for movement across various terrains. It defines movement sequences for different modes, such as forward walking, wheel-based rolling, stair climbing, and turning. By specifying servo angles for leg movements and velocities for wheels, the algorithm ensures precise execution of tasks while adapting to dynamic environments.

With:

- $\boldsymbol{\theta}^{\uparrow}$ = lift angle
- $\boldsymbol{\theta}^{\rightarrow}$ = forward swing angle
- $\boldsymbol{\theta}^{\downarrow}$ = lower angle
- $\boldsymbol{\theta}_h^{\uparrow}$ = high lift angle
- $\boldsymbol{\theta}_h^{\rightarrow}$ = high forward swing angle
- $\boldsymbol{\theta}_s^{\downarrow}$ = stair lower angle
- $v^{\rightarrow}$ = forward velocity
- $v^{\nearrow}$ = incline velocity

Algorithm 1 QuadrupedHybridMovement

function QUADRUPEDMOVE(mode)
  Let $L = \{L_1, L_2, L_3, L_4\}$ be legs
  Let $\boldsymbol{\theta}_s \in \mathbb{R}^3$ be servo angles per leg
  Let $v \in \mathbb{R}$ be wheel velocity

  Define sequences S(mode):
  Forward:

$$S_f = \big[(\{L_1, L_4\}, \boldsymbol{\theta}^{\uparrow}, 0.5s) \to (\{L_1, L_4\}, \boldsymbol{\theta}^{\rightarrow}, 0.5s) \to (\{L_1, L_4\}, \boldsymbol{\theta}^{\downarrow}, 0.5s) \\ \to (\{L_2, L_3\}, \boldsymbol{\theta}^{\uparrow}, 0.5s) \to (\{L_2, L_3\}, \boldsymbol{\theta}^{\rightarrow}, 0.5s) \to (\{L_2, L_3\}, \boldsymbol{\theta}^{\downarrow}, 0.5s)\big]$$

  Wheel:

$$S_w = \big[(v^{\rightarrow}, 1.0s)\big]$$

  Stairs:

$$S_s = \big[(L_1, \boldsymbol{\theta}_h^{\uparrow}, 0.6s) \to (L_1, \boldsymbol{\theta}_h^{\rightarrow}, 0.6s) \to (L_1, \boldsymbol{\theta}_s^{\downarrow}, 0.6s) \\ \to (L_2, \boldsymbol{\theta}_h^{\uparrow}, 0.6s) \to (L_2, \boldsymbol{\theta}_h^{\rightarrow}, 0.6s) \to (L_2, \boldsymbol{\theta}_s^{\downarrow}, 0.6s) \\ \to (v^{\nearrow}, 1.0s) \\ \to (L_3, \boldsymbol{\theta}_h^{\uparrow}, 0.6s) \to (L_3, \boldsymbol{\theta}_h^{\rightarrow}, 0.6s) \to (L_3, \boldsymbol{\theta}_s^{\downarrow}, 0.6s) \\ \to (L_4, \boldsymbol{\theta}_h^{\uparrow}, 0.6s) \to (L_4, \boldsymbol{\theta}_h^{\rightarrow}, 0.6s) \to (L_4, \boldsymbol{\theta}_s^{\downarrow}, 0.6s)\big]$$

  TurnLeft:

$$S_{tl} = \big[(v_L = v_{slow}, v_R = v_{fast}, 1.0s)\big]$$

  TurnRight:

$$S_{tr} = \big[(v_L = v_{fast}, v_R = v_{slow}, 1.0s)\big]$$

  Execute(n_{steps}):
  for step $\in [1..n_{steps}]$ **do**
    for all action $\in S$(mode) **do**
      if action.type = leg **then**
        for all $l \in$ action.legs **do**
          $\boldsymbol{\theta}_s[l] \leftarrow$ action.angle
          updateServo(l, $\boldsymbol{\theta}_s[l]$)
        end for
      else
        updateWheels(action.v)
      end if
      delay(action.t)
    end for
  end for
end function

5 PERCEPTION & PATH PLANNING

The entire processing pipeline structure of the perception and path planning is illustrated in Figure [8]. In the following sections, we will cover our approach and the solutions we implemented for each step.

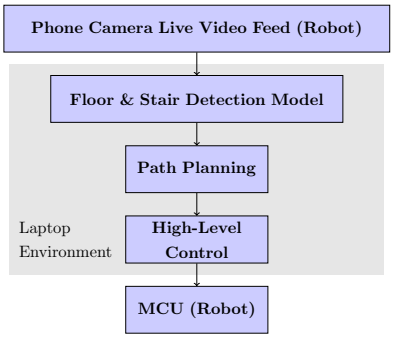
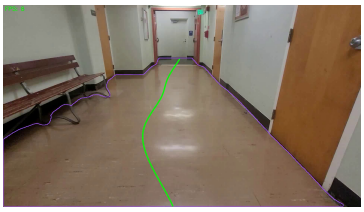


Figure 8: Processing pipeline of the Perception & Path Planning



(a) Only Floor

(b) Floor & Stairs

Figure 9: Cutouts of Real-time detection and computed path visualization.

5.1 Video Feed

As the camera, we are using a smartphone, which is securely attached to the robot using a custom-made 3D-printed mount. The camera's live feed is transmitted via Wi-Fi to a laptop using an already available app. This live feed operates at a resolution of 720p and maintains a smooth frame rate of 60fps.

5.2 Detection Model

5.2.1 Model Architecture

We used the YOLO (You Only Look Once) model for object detection in our project. In specific, we used the latest version, YOLO11, due to its superior performance in terms of both speed and precision, compared to older versions. Moreover, there are pretrained models tailored for segmentation (pixel-wise classification), making it particularly useful in our application. While a detailed analysis of YOLOv11's architecture is beyond the scope of this project paper, we briefly outline its structure here. The YOLOv11 architecture consists of three primary components: the backbone, the neck, and the head. The backbone is "responsible for extracting essential features from input images"[3]. The neck pools features, which make the model capable to detect objects of different sizes, while the head generates the final object detections.[4][3]
In our case, we are using this model architecture to detect the drivable floor and stairs. There are different model sizes available of YOLO11 for segmentation (see table [1]), but even the smallest model needs 65.9 ms for the inference per image (about 15 fps) on an Amazon EC2 P4 [4]. Given that we would likely get even lower fps on our laptop, we chose the smallest model available, with 2.9 million parameters.

Model	Size (pixels)	$mAP_{mask,\ 50\text{-}95}$	Speed CPU (ms)	Params (10^6)	FLOPs (10^9)
YOLO11n-seg	640	32.0	65.9 ± 1.1	2.9	10.4
YOLO11s-seg	640	37.8	117.6 ± 4.9	10.1	35.5
YOLO11m-seg	640	41.5	281.6 ± 1.2	22.4	123.3
YOLO11l-seg	640	42.9	344.2 ± 3.2	27.6	142.2
YOLO11x-seg	640	43.8	664.5 ± 3.2	62.1	319.0

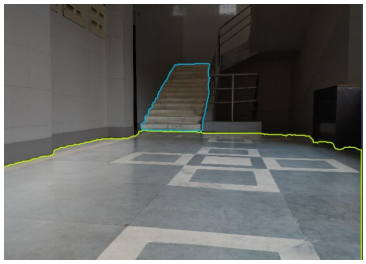
Table 1: Performance Metrics of YOLO11 Models [4]

5.2.2 Data, Training & Performance

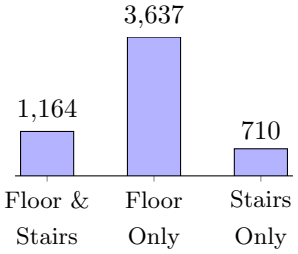
As is often the case in machine learning, the training data is one of the most crucial elements. The data needs to be high-quality, with correct masks, relevant to our application, diverse, and simply plentiful. In our project, we required images of corridors (since our target route is through a corridor) with floor masks, as well as images of masked stairs. To enhance diversity and gather enough data, we pulled images from various datasets available on the web. In total, we used labeled images from 11 different datasets containing images with labeled floors, labeled stairs, and sometimes both in a single image (see example 10a). This gave us a total of 5,511 labeled images (see figure 10b).

To further diversify the data and improve the model's robustness, we applied data augmentation techniques such as rotation, shear, adjusting saturation, adding noise, and changing exposure. This process expanded our dataset to 28,617 images. Additionally, to enhance the model's performance, we included preprocessing steps like auto-orientation, converting images to grayscale, and resizing. We executed the training on a remote service because it would have taken too long on our local computer.

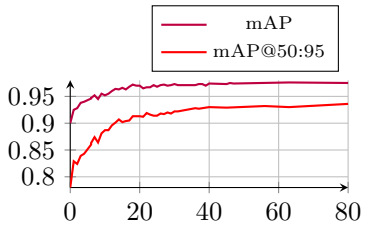
The trained model achieved a precision of 97.4% and a mAP@50:95 of 94% (see figure 10c).



(a) Example of a floor and stair masked image



(b) Number of samples with different mask classes



(c) mAP and mAP@50:95 Progression Curve over the Epochs

Figure 10

The real time inference performs on a local computer and achieves a consistent frame rate of over 10 FPS, which is for our application absolutely sufficient. During the real-world evaluation of the model, we found that it works quite well in the environment we trained it for (corridors). However, when we used it in other environments, such as outdoor or large rooms, the model had problems recognizing the floor, though it handled stairs relatively well. This is understandable because we trained it on a variety of different stairs, which generally have less variation than floor surfaces. Since our training focused only on corridors, tight spaces with walls on both sides, the model struggles outside this setting. Therefore, if we want to deploy the robot in environments beyond our target route, the model needs to be improved and more generalized, but that is beyond the scope of this project.

5.3 Path Planning

The path planning process in this application is relatively straightforward, as the primary objective is to follow the centerline of a corridor. To achieve this, we first applied denoising techniques to refine the detected floor and stair masks, then the path was calculated by identifying the midline between the left and right boundaries of the detected mask, followed by applying smoothing. This approach effectively keeps a sufficient distance to walls and obstacles, enabling it to navigate the corridor efficiently. The results of this detection and path planning process are illustrated in Figure [9].

5.4 High-Level Control

The high-level control of the perception pipeline manages the transmission of key parameters from the computer to the robot. These parameters, derived from the perception and path-planning stages, include the turning rate and stair detection parameter, that gets sent via Bluetooth from the laptop to the robot. The turning rate parameter, ranging from -100 to 100, instructs the robot on the direction and speed of its turns to follow the computed path. This control leverages an enhanced version of a lookahead algorithm. Specifically, the method calculates the weighted average distance of the first 15% of the path points from the center pixel (320th pixel in a 640-pixel-wide image). A carefully tuned PID controller then uses this deviation to determine the turning rate, ensuring alignment with the center of the camera image, which corresponds to the robot's central axis, which means a smooth path following. The stair detection parameter ranges from 0 to 100 and indicates the proximity of a detected stair. A value of 0 means no stair is detected, while 100 signals the robot to switch to climbing mode. This parameter is computed based on the pixel height of the lower edge of the detected stair mask. Since the camera's position and angle on the robot are fixed, this straightforward computation provides a reliable measure of stair proximity. However, the stair climbing itself requires a predefined control implemented directly on the robot. This control accounts for the known height and depth of the stairs, as these dimensions cannot be estimated accurately using a monocular camera alone, especially given the system's computational constraints. This means that, within the scope of this project, the robot is capable of autonomously climbing only predefined stairs using manually implemented control.

6 RESULTS AND FUTURE WORK

6.1 Results

This project successfully achieved the design and conceptualization of a leg mechanism capable of both walking and climbing stairs. Through iterative prototyping and careful analysis, we developed a final leg design that meets the functional requirements of mobility and stability. Additionally, we designed the main body, which securely houses all essential components, mounts the four legs, and provides a compact, lightweight structure.

A comprehensive analysis of the leg and its gait cycle was conducted, starting with a workspace evaluation to confirm its suitability for walking and climbing. From this analysis, we simulated a gait cycle and used it to compute the Lagrangian for the stance phase, which informed torque-speed requirements for a motor performance analysis. This enabled the selection of appropriate servos and motors for the robot.

We also implemented an electrical system and developed a perception pipeline from scratch. This included training a custom model using a dataset specifically curated for our use case. The perception pipeline was integrated with a path-planning module and high-level control system, which was successfully tested in simulation.

However, due to limitations in resources—both time and financial—we were unable to complete the physical assembly and real-world demonstration of the robot. Despite this, we are confident in the project's success, as we have developed a robust design and validated key components, paving the way for future construction and implementation.

6.2 Future Work

To build upon this foundation, several improvements and extensions can be made. With additional resources, the robot can be fully constructed and tested in real-world scenarios, allowing for a complete evaluation of its design and functionality. Enhanced perception systems could be developed by incorporating advanced sensors such as LiDAR, enabling the robot to autonomously detect stair dimensions and navigate more effectively in

unknown or uncontrolled environments. Furthermore, the robot's autonomy could be expanded by developing more robust algorithms for decision-making and adaptability, allowing it to operate dynamically in diverse and unpredictable settings. These advancements would significantly enhance the robot's capabilities, ensuring it can perform effectively in a variety of environments and fulfill its intended applications.

Data and Code Availability

The code, CAD files and all other data used in this project are publicly available at:
https://github.com/Loneli999/WALT.

References

1. Adafruit. DC "TT Motor" Datasheet. Accessed: 2023-11-14. URL: https://www.adafruit.com/product/3777.
2. microrobotics. Servo DS3240 Datasheet. Accessed: 2023-11-16. URL: https://github.com/microrobotics/DS3240.
3. Sulake NR. A Comprehensive Guide to YOLOv11 Object Detection. Accessed: 2024-11-24. 2024. URL: https://www.analyticsvidhya.com/blog/2024/10/yolov11-object-detection/.
4. Ultralytics. Ultralytics YOLO11. Accessed: 2024-11-24. 2024. URL: https://docs.ultralytics.com/models/yolo11/#supported-tasks-and-modes.