Wall Following Robot in Webots

Cormac Duffy, B-Number: B00806474

School of Computing, Engieering and Intelligent Systems,

Ulster University

*Abstract*— This report investigates the development and evaluation of a python-based controller for a wall following robot implementing the left-hand rule. Tested in the Cyberbotics Webots environment. The report uses the e-puck robot to navigate a maze by detecting and following walls. Through testing various environment scenarios, the report assesses the effectiveness of a rule-based or reactive control system and discusses the potential benefits of transitioning to proportional control system in the future.

Keywords— Robotics, Python, Wall Following, Left-Hand Rule, Webots, e-puck. (key words)

# Introduction

The aim of this report is to explore the development and implementation of a python-based wall-following robot controller algorithm using left hand rule.

The controller will be tested using the e-puck robot within the virtual environment of Cyberbotics Webots, which will navigate a maze by detecting and following walls using its sensors.

This report will discuss existing solutions or similar research to provide an understanding of the current robotic wall-following technologies.

## Toyota Human Support Robot

As we continue to rapidly develop and iterate robotics technology to automate jobs and mundane tasks that humans would rather not do we get products like the Toyota Human Support Robot. As the name suggests, it’s a robot that can help humans’ complete tasks around their homes such as picking up items and opening curtains. This robot uses a range of sensors to follow walls and avoid obstacles like its laser sensor, bumper sensor, or its cameras for 3D vision.

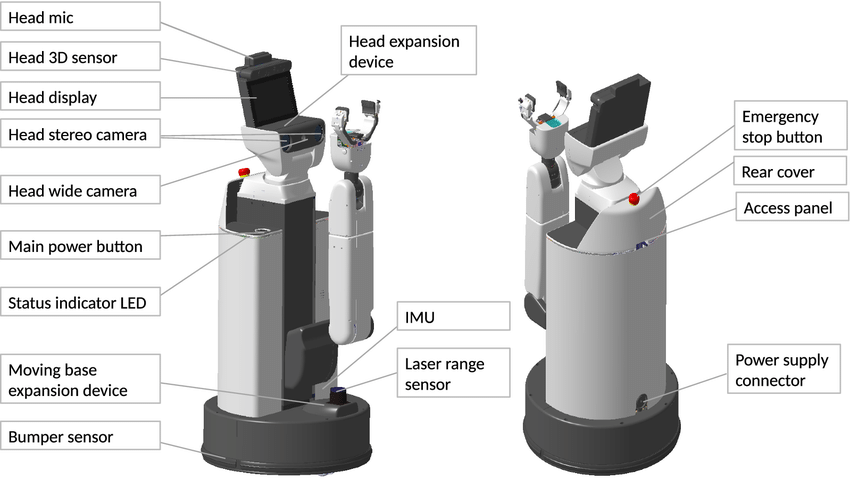


Figure 1 – HSR Sensors and Equipment [3]

A case study released during the COVID-19 pandemic implemented machine learning algorithms and made use of the Human Support Robots 3D vision sensor to detect, follow walls either left or right decided by the user, avoid obstacles and clean large wall surfaces instead of exposing humans to the potential health risk of COVID. [1] Although this robot and its system design is a lot more advanced than what we will be investigating with the e-puck robot and our Webots simulation software, it follows the same principle of using a robot’s sensors to detect walls and navigate successfully in its environment.

## Robotic Vacuums

Robotic Vacuums represent a more prevalent application of sensors-based navigation in household environments. These robots typically employ a combination of obstacle sensors including lasers, bump sensors and cameras to navigate around furniture or other obstacles. These sensors allow the robot to be effective at following walls to clean edges and use the cliff sensors to avoid falls. These robotic vacuums are a good practical example of autonomous navigation. [2]

# Methodology

This section outlines the methodology used to develop and evaluate our python-based wall-following robot controller. We will discuss all potential scenarios the robot could face in our environment and the algorithm we developed.

## Configuration Space

Configuration Space describes all the possible scenarios that the e-puck robot could encounter while navigating the environment. These scenarios are based on the robots’ interactions with the environment, particularly how it will detect and respond to walls.

A grid of black check marks

Description automatically generated

Figure 2 – Environment Scenarios Truth Table

The above chart represents the scenarios our left-hand rule robot will potentially face in its environment and the corresponding operations that should take place in those conditions.

## Controller Logic (Pseudo Code)

With our environment scenarios identified, the next step we must take is the initial development of our wall-following. This algorithm will make use of our robot’s lasers sensors to detect walls and make navigation decisions. The initial pseudocode for our algorithm is shown below.

A screenshot of a computer program

Description automatically generated

Figure 3 – Initial Algorithm Pseudocode

This simple loop continuously reads the robots sensors and decides which conditions they meet to decide movement based on left hand rule wall-proximity.

## Simulation in Webots

To test our robot controller algorithm a virtual maze environment was designed in Cyberbotics Webots that includes elements such as walls, corners, T-junctions, and dead ends. This will allow us to test all the scenarios that we refer to in the Configuration Space section and discover any shortcomings in our initial algorithm.

A maze on a chessboard

Description automatically generated

Figure 4 – Image of Simulated Environment

## Testing and Iteration of our algorithm

Testing of our algorithm involves running the e-puck robot through the virtual maze under different conditions and observing its performance in each scenario. These observations will be documented in terms of success in completing a lap of the maze without leaving the wall, crashing into the wall and the ability to maneuver around corners.

## Evaluation Criteria

The final part of our methodology is the evaluation of the robot’s performance based on the following criteria:

* Efficiency – How quickly the robot completes the maze starting from different scenarios.
* Accuracy – The robots’ ability to maintain ideal wall distance of 0.2m and follow the left-hand rule.

# Testing and Discussion

## Testing and Evaluation of Initial Algorithm

To test and evaluate the performance of our initial wall following algorithm, we will conduct a series of tests across a range of predetermined points in the simulation environment.

Each of these points are chosen to represent the different scenarios the robot may encounter like no-wall, right-wall and left-wall as outlined in the key below.

A screenshot of a game

Description automatically generated

Figure 5 - Testing Scenario Locations

The testing process is as follows:

* Placement and Observation – Place the robot at each starting position and activate the algorithm. During the runs we will observe the robot’s behavior
* Performance Metrics – We will record performance metrics such as time to complete maze.
* Behavioral Analysis – Observe robots’ behavior in each scenario to determine issues with the system design or algorithm. For instance, we will look at if it finds difficulty in consistent distance to the wall.

## Observation and Evaluation of Intial Algorithms

**Placement and Orientation**

**Left Wall Scenario**

* The robot was placed in 5 different positions where the left sensors were detecting walls at the beginning.
* The robot was riding the wall.

**No Wall Scenario**

* The robot was placed in 5 different positions where no walls were in proximity.
* The robot showed consistent behavior in seeking out the nearest wall.
* Run 4 involved a complete turnaround leading to a right-wall scenario that the robot then corrected by turning it into left-wall scenario.

**Right-Wall Scenario**

* The robot was placed in 5 different positions where the right sensors were detecting walls at the beginning.
* The robot followed the right-walls, until it met a front wall then it turned right and adapted left-wall following.
* In run 3 of using our right-wall scenario the robot came up to a turn causing the robot to miss part of the maze, leading to an additional lap to cover the missed area.

**Performance Metrics**

* Time – Across all scenarios and test, times to complete the maze were relatively consistent with most runs clocking over at just over 8 minutes. However, run 3 in the right-wall scenarios took twice as long due to the robot missing part of the maze and needing an additional lap.

A table with numbers and a few minutes

Description automatically generated

Figure 6 – Initial Algorithm Test Lap Times

**Behavior Analysis**

**Left Wall Scenario –** During these scenarios the robot had a large turning radius at corners and a tendency to hug the wall, indicating overadjustment when turning and not enough conditions in the algorithm to determine it is too close to the wall.

**No Wall Scenario –** The robot immediately turns left in search of a wall, as expected. This behavior is to ensure left hand rule is established quickly. However, the 180 degrees turn in run 4 creating a right-wall scenario may suggest excessive corrective behavior.

**Right Wall Scenario –** The robot successfully manages to correct from right wall to left wall scenario by following the wall until it reaches a corner indicating flexibility in navigation. However, the missed section in Run 4 indicates a fundamental flaw, causing an additional lap.

A computer screen shot of a program code

Description automatically generated

Figure 7 – Initial Algorithm

## Algorithm Improvements and Testing

Based on the testing and observation of my initial algorithm I have made the following improvements to my algorithm as shown below.

A computer screen shot of a program

Description automatically generated

Figure 8 – Improved Algorithm

Using additional sensors on the e-puck robot a condition was added that checks if the robot is too close to the wall and corrects the behavior by putting the right wheel in reverse. I also modified the turn left condition so that the left wheel turns slower around the corner to reduce the turning radius and the right wall condition so that it now turns in place, meaning it will correct too left-hand following quicker.

**Placement and Orientation**

**Left Wall Scenario**

* The robot was again tested in the same 5 locations as the old algorithm.
* It consistently maintained a safe distance from the wall and was no longer riding against it.
* Turning corner radius was reduced.

**No Wall Scenario**

* The robot was again tested in the same 5 locations as the old algorithm.
* The robot showed varied results, due to the reduced speed settings when searching for a wall, in one run it failed to find a wall in a reasonable amount of time.

**Right Wall Scenario**

* The robot was again tested in the same 5 locations as the old algorithm.
* The robot showed uniform and consistent response across all 5 tests, reorienting itself to transform the right wall into a left wall, adhering to left-hand rule.

**Performance Metrics**

* Time - The left-wall scenario runs were consistent, with completion times ranging from 6.47 to 6.49 minutes. The no-wall scenario presented more varied results, with times from 7.03 to 8 minutes, and included one test that failed after 19 minutes due to the robot's inability to locate a wall. The right-wall scenario showed consistency, with all runs clocking in between 7.02 and 7.07 minutes.

A close-up of a running schedule

Description automatically generated

Figure 9 – Improved Algorithm Test Lap Times

**Behavior Analysis**

**Left Wall Scenario –** There was minor jittering when the robot was adjusting its path due to being too close to the wall. It no longer hugs the wall and its corner turning radius is largely improved, helping the robot meet the desired behavior. Overall, the time to completion of a lap is reduced.

**No Wall Scenario –** The robot continues to turn left to find a wall, but due to the reduced speed on the turn left condition, it has a smaller radius for finding walls, this led to a failure in run 2.

**Right Wall Scenario –** The robot now corrects the right wall to left wall by turning 180 degrees in place, instead of following the wall until it reaches a corner. This led to faster completion time and removal of outlier results that existed in the original algorithm.

## Results

The results of the new algorithm show an overall improvement in lap times and a strong improvement in expected behavior for the robot including following the wall at a safe distance, within 0.2 meters, improved corner turning and never colliding with the wall. However, there are parameters that could be fine-tuned, specifically the No Wall scenario where the robot aimlessly wanders while never finding a wall like in run 2. Additionally, the jittering observed in the left wall scenario could be improved.

## Other Experiments

I explored other solutions and experiments to no wall scenario to improve its handling, but failed to implement anything that fixed the issues without breaking other conditions. One of the experiments I tried was creating an additional condition for the controller that detected that none of the sensors were detecting walls and to find one as shown below.

A screen shot of a computer program

Description automatically generated

Figure 10 – Attempt at Better No Wall Detection

This condition conflicts with the turn left condition as both are activated when no walls are detected. This essentially breaks the turning functionality of the robot. This is among one of the many solutions I tried.

## Future Considerations

**Controller Improvements**

The current implementation of our controller follows a reactive or rule-based paradigm. This is because of its reliance on predefined rules that translate sensor inputs into immediate action. This approach has the advantage of being relatively simple and easy to implement, but it limits the robot’s ability to navigate, especially in more dynamic environments.

**Transition to Proportional Control**

Transitioning from a rule-based or reactive controller to a proportional controller presents an opportunity for improvement. A proportional controller allows for more granular and gradual adjustments. It does this by adjusting the control signals to the motors in proportion to the distance from the wall. This would help reduce the oscillations we see when the robot corrects itself from being too close to the wall, maintaining a more consistent path, execute smother turns and adapt its behavior based on the continuous feedback it receives from its environment.

**No Wall Detection Improvement**

The No-Wall Scenario presented as a challenge, as the robot is programed to search for walls and initiate the left-hand rule. However, the absence of walls leads to inefficient search patterns and sometimes failure to find a wall.

**Combining Sensor Types**

Combining different types of sensors such as infrared and camera could provide more data for the robot, improving its ability to find walls at greater distances.

**Simultaneous Localization and Mapping (SLAM)**

Implementing SLAM algorithms could allow for the robot to create a map of its environment in real-time. This could improve its ability to find walls in no-wall scenarios.

# Conclusions

The development, implementation and testing of the wall-following controller based on the left-hand rule using the e-puck robot has provided insight into the capabilities and limitations of a reactive or rule-based system in a simulated environment. The robot successfully navigates common scenarios involving direct wall interactions like when it detects left or right walls, however, shows limitations in no-wall conditions.

The reactive or rule-based control shows some minor control issues such as jittering.

Analysis of existing robotic technologies has highlighted the relevance of sensor based autonomous navigation and has helped guide future considerations. The discussed potential transition to proportional control system would help address current shortcomings by enabling smoother and more precise adjustments due to continuous feedback from the environment.

The future considerations discussed will focus on improving the detection of walls in a no wall scenario by combining different types of sensors or implementing SLAM. These improvements would aim to improve the robot’s adaptability and efficiency, preparing it for more dynamic environments.

##### References

1. T. W. Teng, P. Veerajagadheswar, B. Ramalingam, J. Yin, R. Elara Mohan, and B. F. Gómez, “Vision Based Wall Following Framework: A Case Study With HSR Robot for Cleaning Application,” *Sensors*, vol. 20, no. 11, p. 3298, Jun. 2020, doi: https://doi.org/10.3390/s20113298.
2. R. Digital, “The Workings of Robotic Vacuum Cleaners | | Resource Centre by Reliance Digital,” *www.reliancedigital.in*. <https://www.reliancedigital.in/solutionbox/the-workings-of-robotic-vacuum-cleaners/#>
3. T. Yamamoto, K. Terada, A. Ochiai, F. Saito, Y. Asahara, and K. Murase, “Development of Human Support Robot as the research platform of a domestic mobile manipulator,” ROBOMECH Journal, vol. 6, no. 1, Apr. 2019, doi: https://doi.org/10.1186/s40648-019-0132-3