MTRX3760 - Project 1 Turtle Bots

Report of Project 1

Created By:

SID	Date
460386784	20/09/2019
470318966	20/09/2019
470023233	20/09/2019
470416309	20/09/2019

Introduction/Problem Statement

The problem or exercise requires the Turtlebot to successfully complete or solve a maze. This is to be both in a simulation as well as a physical model. The physical model of the maze is to be created using cardboard with the Turtlebot moving through the cardboard environment (where most of the material is utilized represents the maze walls). The simulation is modelled with a combination of programs including:

- RViz
- Gazebo
- ROS Kinetic & Melodic
- Ubuntu 18.04 & 16.04

The main method of completing the maze would involve the use of wall following. The method uses only detection of a wall to guide the robot through the maze. The robot will continue this motion until it reaches the maze exit. The simulation will involve further modification of a turtle bot drive file with the addition of a tested algorithm of turtle bot left-wall follower.

The physical simulation that requires autonomous solving of the turtlebot employs the use of a remote PC program to monitor the progress of completion of the robot in the maze. All decisions, movement and overall solution is to occur autonomously on the robot. There are three levels of independence that are an objective:

1. *Remote Run:*
    - Remote computer runs the program.
    - Starts immediately on request.
        - Roscore and decision-making node run on turtlebot.
    - Program runs autonomously without need of remote PC.
2. *Remote Prime:*
    - Remote connection to run the program.
    - Main node sits idle until pushbutton is pressed on the robot.
        - Sits idle for two seconds then runs (if button is pressed).
    - Runs independently from the remote PC.
3. *No Remote PC:*
    - PC runs automatically on boot.
        - Edit **.bashrc** on the robot to make your program launch automatically.
    - Waits for a pushbutton to be pressed.
        - Sits idle for two seconds and runs.

Requirements & Conditions

The following section highlights a variety of different requirements from this project and to allow for the operation of the turtle bot maze solver in context of a specification:

1. Must be at least 0.3 metres from the wall of the maze (specifically the left-side).
2. Must be completed as fast as possible.
3. Minimum hallway height is 0.38 metres.

4. May be placed in a scenario with:
    a. Walls all around the robot or…
    b. With walls only on two or less sides, etc.
    c. No wall will be placed in front of the turtle bot (on starting position).
    d. Wall will always (at starting position) be on the left of the robot.
5. Total time for the solving of the robot must last less than or equal to eight minutes in total.
    a. Total of two time trials with the best time used as a final result.
    b. No handling of the robot within the time trials.

System Design

Using the drive file of the turtle bot, the direction of movement can be determined using different functions. Furthermore, the utilization of a ROS node for the initialization of the turtle bot to allow for the indepence cases to be adequately implemented.

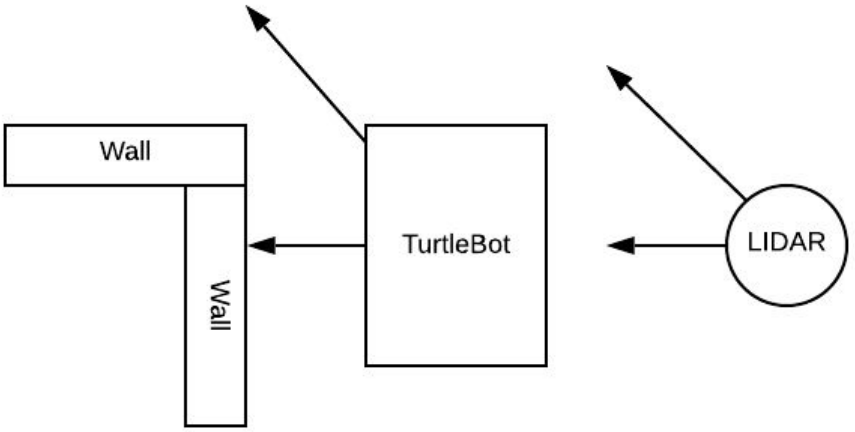
The system design of the left wall follower utilizes distances as constants in centimetres to define how far the turtle bot should travel from the left wall as well as how far it should travel before turning a corner. Specifically, the front and front left distances are utilized initially with the right side being implemented after (plan). There is planned to be a minimum and maximum distance in the LIDAR detection to allow for the bot to determine what it is looking at and to program the bot to follow close objects (such as the wall), ignore interference from distant objects and detect openings. The scan will allow for the turtle bot to detect a corner and stay close enough (as specified) to the corner using the angular geometry and detection of the laser system.

A control loop within the system would allow for the wall follow program to have a configurable speed and would be required to fix the distance and detection rate of the walls. This would allow for smoother running of the simulation and in physical implementation. The speed would also allow for detection in openings of the maze (or modelled maze boxes with openings). Furthermore, the cornering would be smoother and no collisions would occur. The speed configuration would also allow for time efficiency.

- Use switches to decide when to turn the turtle bot and keep the turtle bot straight.
- Use switches to control velocity and therefore, define a turn action of the robot.
    - The velocity also defines the stopping of the robot with wall ending detection and therefore, defining rotation.
    - The velocity also defines wall following and therefore, linear movement and speed.

The wall follower program must utilize the button pressing boolean statement therefore working with a flag.

Figure 1: Displaying Angular Distances



Testing

The testing of the turtle bot within the maze solving objective was accomplished using two different categories, the **simulation testing** and the **physical performance testing**.

Acceptance Testing

Acceptance testing within the project is accomplished with the data being correctly coordinated in the program in conjunction with integration. Each function must operate as expected with each function for example, the wall follower and the button flags.
The final acceptance test also refers to the project being properly satisfied (as per the customer's request).
The significant component of this testing included the reading of a value of a component from the program command line and ensuring that the expected results are outputted.

Simulation Testing

The simulation testing utilizes the experimentation of the turtle bot drive code as well as the incorporation of the wall detection algorithm. The testing itself involved the following methods:

1. Using Gazebo and RViz modeling with simulated objects and obstacles.
2. Running code alterations with instantaneous results to compare with changes.
3. Revising code with multiple members of the group
    a. Achieved through **git** as well as on the same terminal in person.
4. Comparing with initial setout specifications.
5. Increasing the obstacles and testing cases outside of the initial requirements for a perspective on the flexibility of the code operation.

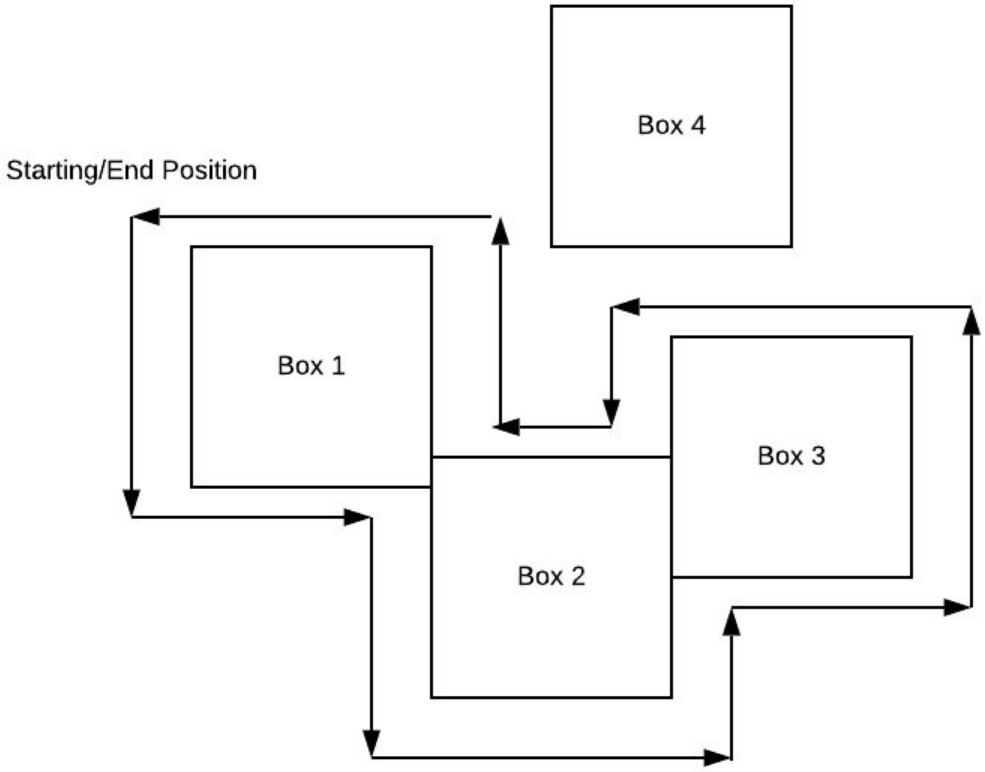


Figure 1: Simulated Tests of Turtle Bot

The above figure displays the test of the simulated RViz & Gazebo model maze solver denoting that the turtlebot can pass through right and left side walls, stick to the left wall as a follower system, asymmetric setups and also pass through tight corridor/tunnel setups.

Physical Testing

The physical testing in the project was accomplished by specifically the following illustrations of arrangements of the model maze. This allowed for the full testing of the turtle bot in a range of possible maze scenarios.

The first singular box test and the dual box test setup tests the interior tracking and exterior tracking of the box with the LIDAR on the turtle bot. Therefore, allowing for a tight turns and close quarters to be tested.

Note: The arrows denote the pathing of the turtle bot in a counterclockwise direction in all figures.

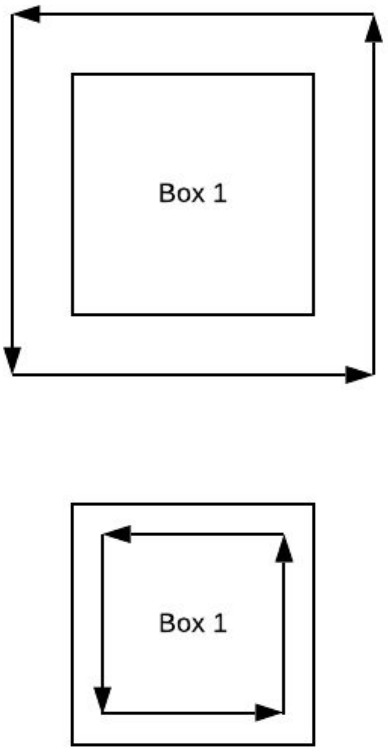
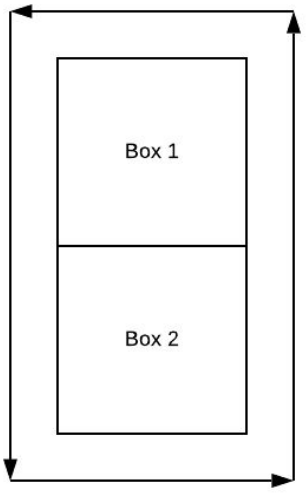
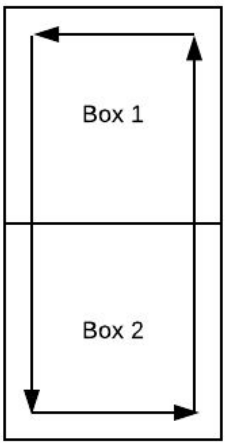


Figure 1: Single Box Test





Note: The next figure denotes a test of the turtle bot passing through openings of the interior and exterior of each box allowing for the height restrictions and cornering specifications to be tested. The dotted boxes are displaying these openings that the bot can pass through.

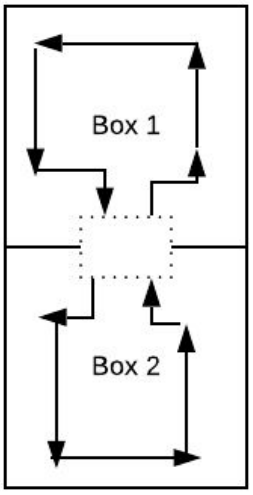


Figure 2: Dual Box Setup

Note: The following setup was tested with multiple different paths and starting points and the left wall was always successfully followed. Therefore, the arrows denoting a singular path is not displayed.

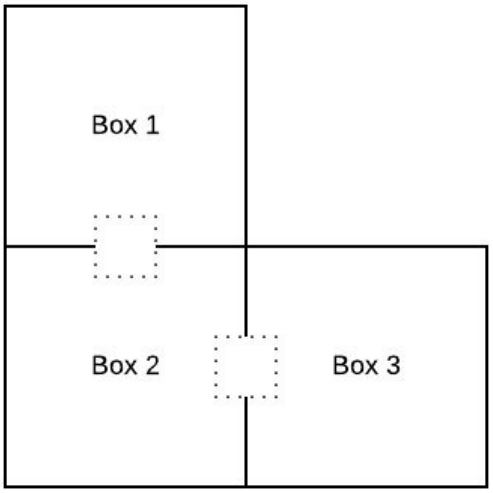


Figure 3: Full Test/Right & Left Wall

Testing Independence Cases

The primary integration and unit testing was accomplished through the wall follower file with the main launch node file with different functions being tested to output correct information and carryout correct commands with the button flags and functions.

Stage 1:

- The most common independence case was tested by running the program with a ROS node in combination with a switch to allow for the checking if the follower function was enabled. This was implemented using a boolean statement.
- Testing was accomplished physically without simulation utilized.

Stage 2:

- Configured remote connection to test the running of the program. Most of the components was tested using integration (along with most of the other stages) allowing for the interfaces between the components to be upto specification in the project.

Stage 3:

- Configured through editing of **.bashrc**. The testing was accomplished using development and constant testing in conjunction. Tests include unit testing that allowed for the verification of the functionality of the program with reference to the specification.

Teamwork & Project Management

The team was coordinated by having a total for four meetings with the following minutes detailed below. The team was broken into two main sections of focus, two individuals out of the total four members would concentrate on the design of the system, one member would complete the report and a final member would complete a test and evaluation of all changes in the system.

Meeting Minutes

Meeting 1:

Date: 09/09/2019
Time: 11am-2pm
Attendees:

- Benson McClelland
- Manan Vora
- Tranks Naing
- Vishant Prasad

Notes & Completed Objectives:

- Initial connection to hardware and setup.
- Starting the configuration on PC's and Ubuntu systems.
- Planning algorithms and overall system design (basic).
- Plan of project management:
    - Communication
    - Meetings
- Setup of git and collaborative documents.

Meeting 2:

Date: 11/09/2019
Time: 3-6pm
Attendees:

- Benson McClelland
- Manan Vora
- Tranks Naing
- Vishant Prasad

Notes & Completed Objectives:

- System design and continuation of programming.
- Editing of drive and LIDAR.

Meeting 3:

Date: 16/09/2019
Time: 11am-2pm
Attendees:

- Benson McClelland
- Manan Vora
- Tranks Naing
- Vishant Prasad

Notes & Completed Objectives:

- Simulated program testing.
- System design and further code/programming.
- Starting of report.

Meeting 4:

Date: 20/09/2019
Time: 4-7pm **(approximation)**
Attendees:

- Benson McClelland
- Tranks Naing
- Vishant Prasad

Notes & Completed Objectives:

- Completing of final testing.
    - Both simulated & physical
- Satisfying independence stages (all three).
- Adjustments of distance of to follow.

Completion:

At this stage the final creation of the report and discussion of performance, final evaluation and comparison to the specifications of the project as well as demonstration practice took place.

Final Roles:

- The code & primary system design was completed by Benson.M with assistance of Tranks.N.
    - Turtlebot hardware & setup.
    - Autonomy from host PC was completed by Benson.M.
    - Feedback control.
- The report was completed by Vishant.P with assistance in testing and code clarity.
- The refinement of time efficiency and partial system design was completed by Benson.M and Manan.V.
- The refactoring was completed by Manan.V.

References

ROBOTIS e-Manual. (2019). Retrieved 22 September 2019, from http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

Robots/TurtleBot - ROS Wiki. (2019). Retrieved 22 September 2019, from https://wiki.ros.org/Robots/TurtleBot