Maze solver robot

Team – 18

Bhuvan(49), Varun(63), Salman(64), Asok(72)

Problem statement

- Statement was to make an autonomous robot that will solve the maze using three different algorithms.
- The three algorithms employed include A-star, wall follower algorithm and tremaux algorithm. The maze can be either a wall maze or a white line maze.
- We can either make it to trace back to start node or place it at start and go to end in optimal path.

Requirements/Task specifications

- The only hardware part we required is the robot.
- We also thought of implementing the algorithms on the player-stage simulator. So, the software part required by the project was the player-stage simulator.
- Coding the algorithms in the robot was another important task.

Project plan

- First we tried to make a perfect maze design so that we can easily compare and contrast between the different algorithms.
- Then we tried to get familiar with the robot, and code the algorithms for the robot.
- Then we tried to implement the algorithms using the player-stage simulator.

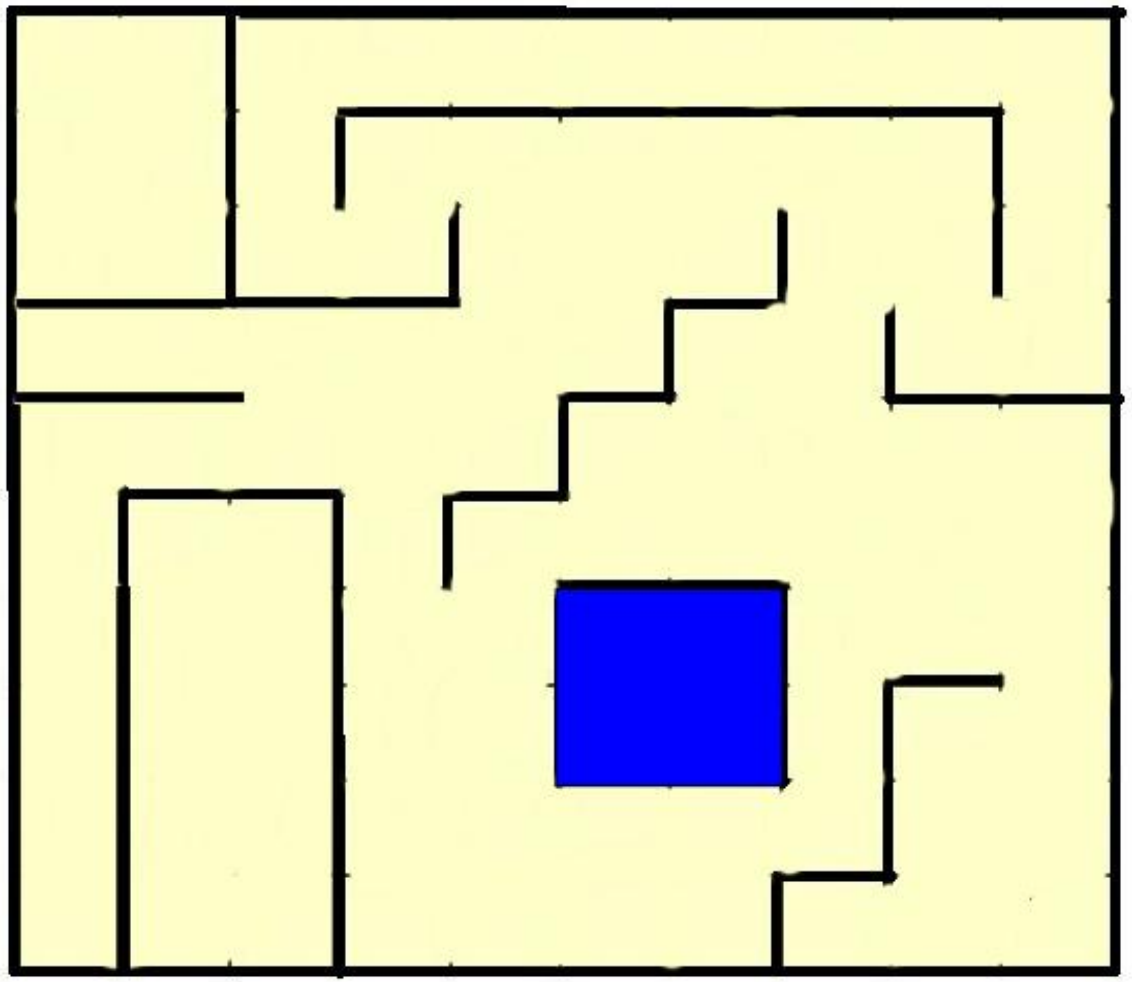
Innovation and challenges

- The task of getting the robot to follow the maze as directed was very hard. The sensors on the robot that we chose(spark V) were not suitable for the maze solving problem, and so we had to fight it for long and then change the robot.
- For every algorithm, we had to write its three forms, namely pseudo code, code for robot and code for player-stage.

Task completed

- Coding the algorithms: Main problems faced were during storage of the maze data. We wrote the code for robot in C, and as there are no data structures in C, we had to store the whole maze data in Arrays.
- Player-stage: We implemented the three algorithms, and the key challenge in this is the time it takes to converge.
- Robot: We couldn’t test the code on the robot, as none of the robots we selected were working properly(ten almost).

Test maze



Review and tests

- Our algorithms were working properly on player-stage, as we checked it. The test criteria we used was, we changed the destination of path, and made sure that the algorithm works for any destination.
- We also used different test rigs for different algorithms to compare between different algorithms.

Performance metrics

- Performance metrics for maze solving include time taken for the robot to explore the maze, and time taken by the robot to solve the maze.
- When running code in the robot, the accuracy of the sensors, the relative rotations of motors, and the brightness in that location effect the metrics of maze solving.

Performance metrics

- When running the code on the player-stage simulator, the factors that affect the working of the project are, the speed of the computer on which the simulator is running, the speed given to the robot, both traversal and rotational.
- Speed must be optimal so that robot should not take much time, and in the same time, should not crash into walls.

Re-usability features

- Like we said above, the code is written in three ways for each algorithm. One is pseudo code for human understanding, one is code to run on robot, and one is code to run on player-stage simulator.
- The code is clearly divided into hardware dependent and independent parts, so that changing the platforms won’t be a big issue.

Future enhancements

- Any other maze solving algorithm can also be easily incorporated into our code, as the code has all the general functions required for the robot to traverse through the maze.

Thank you