“Roomba” CSCI 3302 Final Project

Adam Chehadi
adch0507@colorado.edu

Olivia Golden
olgo0225@colorado.edu

Josephine Martin
joma1927@colorado.edu

Abstract

For our final project, we created an autonomous Roomba that navigates through the Webots Break Room to reach an endpoint designated with a traffic cone. The Roomba is an epuck robot. The Break Room contains 2 other “obstacle” TurtleBot3Burger robots that move to different waypoints assigned throughout the room. The autonomous Roomba robot reaches its endpoint (traffic cone) without colliding with stationary obstacles or the other moving robots in the Break Room.

Our team mapped the Break Room without the “obstacle” robots by manually driving the epuck. Once the map was defined, the epuck utilized an A* algorithm to create a path to the traffic cone. Then, we utilized a distance sensor to avoid collisions with the “obstacle” robots. If the epuck encountered a robot, it slowed down or sped up to avoided a collision and then continued on its path to the endpoint.

Equipment

- Webots Application
- Webots Break Room World
- Main Epuck
    - Sensors
        - Epuck LIDAR for radar for mapping of obstacles
        - GPS
        - Compass
        - Distance Sensor
- Simulated Obstacle Robots (Turtle3)
    - GPS
    - Camera

Deliverables and Implementation Plan

- Configure Webots Environment - Lead: Jo, Deadline: 11/12
    - Create repo on Github for the final project
    - Add Webots world to project
    - Configure Webots world for Break Room
    - Add robot controller file
- Obstacle Robots - Lead Olivia, Deadline: 11/22
    - Create robots
    - Find waypoints throughout the world for the robots to move to
    - Implement odometry to have the robots move to each waypoint to reach the final goal
- Epuck
    - Odometry - Lead Adam, Deadline: 12/9
        - Find odometry pose_x, pose_y, pose_theta equations to update odometry, this will be used in mapping and planning
    - Create Mapping - Lead Jo, Deadline: 12/10
        - Use LIDAR sensors, GPS, and camera to map the world, this will be used in the planning section
    - Planning - Lead Olivia, Deadline: 12/11
        - Use A* algorithm to find the optimal path between two waypoints
    - Obstacle Avoidance - Lead Olivia, Deadline: 12/12
        - Use Distance sensor to detect any obstacles. If the obstacle robot is detected in front, the epuck will stop till the obstacle is no longer there. If an object is detected to the sides, the epuck will speed up to get out of the way of the robot.
    - Autonomous mode Odometry - Adam and Jo, Deadline: 12/13

        - Find the alpha, rho and modify the x dot and theta dot values to get the epuck to navigate to the end state.

Demo

Our demo is a simulation of a successful run of our project. The simulation contains the Webots break room world, obstacle robots, and our epuck which navigates the course autonomously without colliding with any of the obstacles. The Webots display window is included to display a live map of the course, so there is a visual representation of how the main epuck perceives its surroundings.

Analysis

Our project is able to successfully create a map of the epuck's environment, generate an optimal path between a designated start and end point, and avoid obstacles while traversing the path. Our team encountered difficulty with the feedback controller code for the epuck to follow the path of the waypoints. Instead of making sharp turns and moving forward at a reasonable speed, the epuck takes wide turns, and moves at a snail's pace.

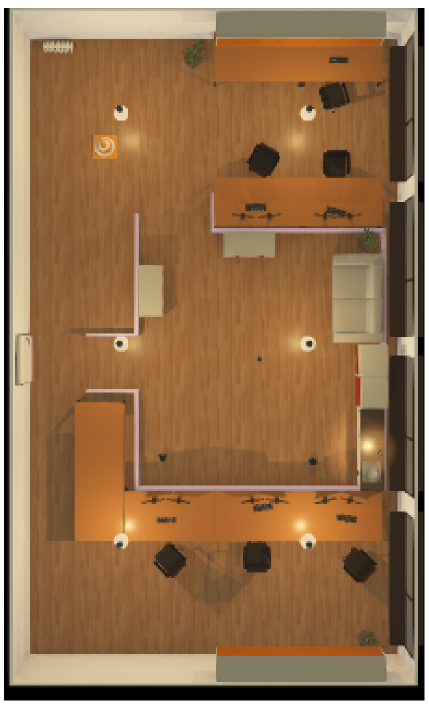
We are unsure of why this is occurring, but took several steps to debug our problems. We attempted to tune our x_dot and theta_dot values, but that didn't help much. We also analyzed the feedback controller algorithm from Lab 3, Lab 4, and Lab 5, and compared it to our epuck controller. After collectively reviewing our code, and comparing it to functioning lab code, we were unable to find a meaningful difference between our process and the functional process defined with Lab 3/4/5. Outside of the non-optimal feedback controller, we are pleased with what we accomplished with this project. This project builds on all of our previous labs and homeworks, and adds the additional functionality of dynamic obstacle avoidance and creating a world, nodes, and code from scratch with minimal instructor guidance.

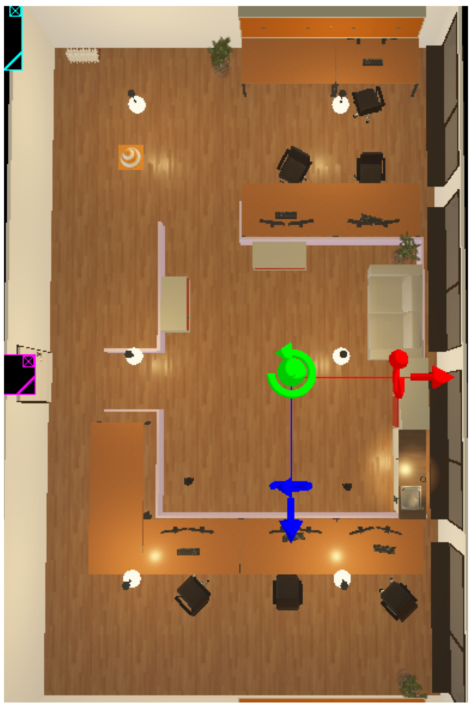
Mode Breakdown:

- In Manual mode the ebot can be driven around the world to map the surroundings using LIDAR. Use the arrow keys to navigate and the S key to save the world map.
- In Planning mode the code will create buffers around the obstacles. Then plot a path of waypoints and save them.
- In Autonomous mode the ebot drives to the waypoints and avoids the other robots and objects in the room till it gets to the traffic cone.

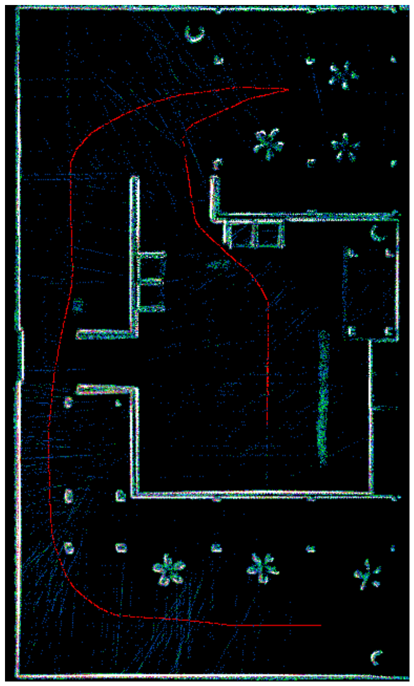
Images:

Starting position of the Epuck:

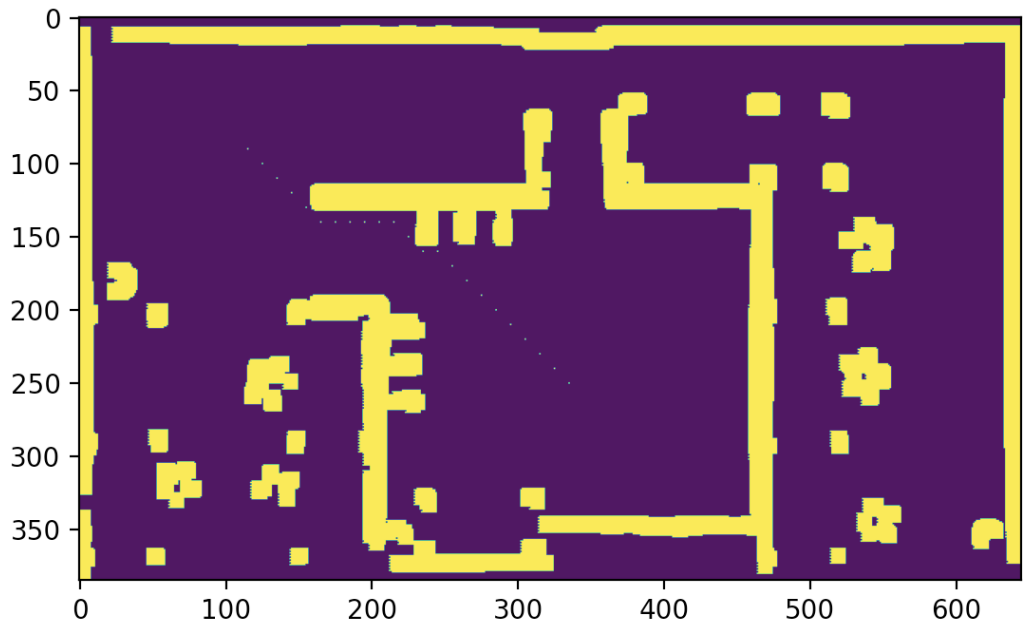




World Map:



Path planning



End Position of the Epuck