Maze Solving Robot (MAZE)

Project Lead: Christian Flewelling
Members: Daniel Guzman, David Boosi, Aman Manadath, Nishyanth Arimanda, Ananya Bandi

TEXAS A&M UNIVERSITY Engineering

TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Problem Definition

The maze is not constrained to that of a typical coloring-sheet maze, but rather the maze is composed of a closed boundary with a visually demarcated endpoint. Furthermore we do not require that the robot start along a boundary.

Software

To solve the subproblem of traveling between two waypoints in a known environment or section of the maze, the following algorithm is used.

1. A path consisting of joined line-segments is generated through the “Fast Matching Trees” algorithm (see *Figure 3*)
2. For each pair of line segments, a bezier curve is generated to interpolate between the endpoints of the pair (see *Figure 1*)
3. The bezier curve control points are adjusted to avoid any obstacles (see *Figure 4*)
4. A velocity profile is generated to accompany the path (see *Figure 2*)

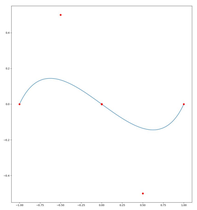

Figure 1: Bézier curve

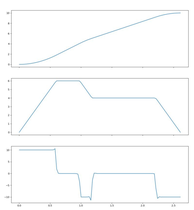

Figure 2: Velocity profile

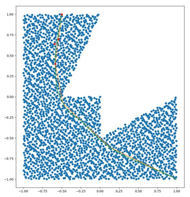

Figure 3: Fast Matching Algorithm Path

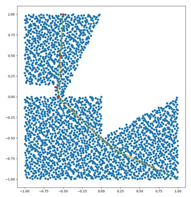

Figure 4: Bézier Curve Path

Software cont.

- Odometry ROS node
  - Wheel odometry node to inform pose based off wheel encoder data
- Refactoring of the sea-current library
  - The codebase was split into multiple modules
  - Several functions had their return types changed for better error reporting
- Gazebo SDF world generation for maze algorithm testing
  - To bridge the gap from the path planning algorithm to usage on hardware we need simulation
  - For validation that the algorithm works we require semi-arbitrary world generation that still meets the initial problem constraints
  - Differential drive robot and lidar also specified in the SDF file

Dependencies

1. Eigen: matrix and vector operations https://github.com/PX4/eigen
2. toppra: path parametrization https://github.com/hungpham2511/toppra
3. nlohmann/json: C++ JSON serialization integration https://github.com/nlohmann/json
4. matplotlib: plotting used for testing https://github.com/lava/matplotlib-cpp/blob/master

References

1. sea-current/reference/chap4.pdf at dev · turtle-robotics/sea-current
2. sea-current/reference/lau09iros.pdf at dev · turtle-robotics/sea-current
3. sea-current/reference/vanderlicio_rev2.pdf at dev · turtle-robotics/sea-current

Hardware

A new prototype CAD design was created with the following design considerations:

1. An enclosed design.
2. A spherical design.
3. Relatively small
4. Space for each necessary part and wire space to connect parts. These parts include: battery, encoder (2), lidar sensor, motor, motor support, Odrive, wheels (2), jetson nano, brake resistor (2), stepdown, caster wheels (2), and gear box.

During the design, a planetary gearbox system was decided and modeled in the CAD design shown below. with an internal view.

Figure 5: Planetary Gearbox System

This CAD design occupies a maximum diameter of 8.6 inches from spherical shell to shell and occupies a height of 10.15 inches. This prototype CAD design was 3D printed in two parts split in half.

Future Work

There are several remaining tasks for next semester:

Software:

- Implementation of the path planning algorithm
- Testing of the path planning algorithm in a virtual environment
- Testing the “Fast Marching Trees” method of solving a maze in a virtual environment

Hardware:

- Once gearbox arrives, 3D print a piece to attach the gearbox to the motor
- After have gearbox and motor are combined, update the CAD model to account for its size
- 3D print the updated CAD model, which will become the new body of the robot
- Add electrical components to new body, then test in maze built from totem poles

Documentation