TEXAS A&M UNIVERSITY
Engineering

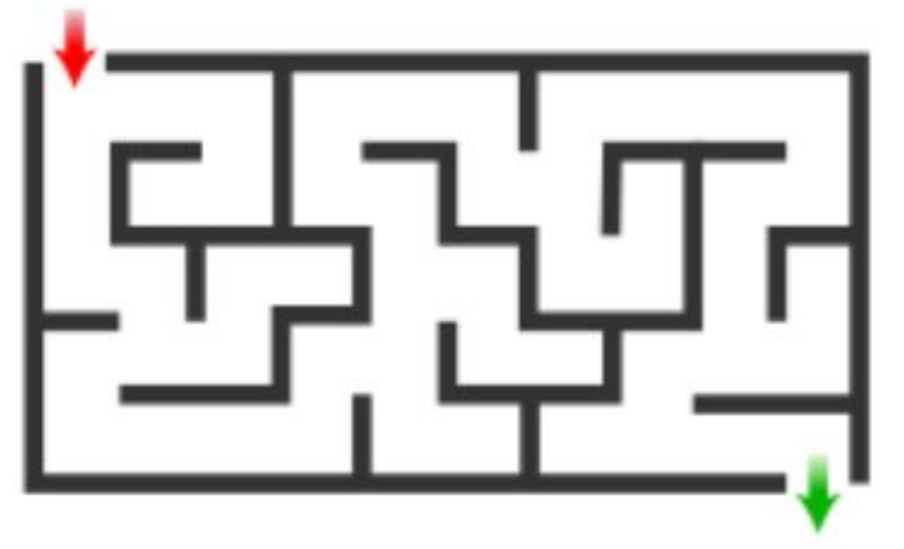
Maze Solving Robot (MAZE)

Project Lead: David Boosi
Members: Nishyanth Arimanda, Ananya Bandi, Daniel Guzman

TEXAS A&M UNIVERSITY
ROBOTICS TEAM & LEADERSHIP EXPERIENCE

Problem

Given an unknown maze, the robot should be capable of navigating from the entrance to the exit. The maze is not constrained to that of a typical coloring-sheet maze, but rather the maze is composed of closed physical boundaries with a visually demarcated endpoint. Furthermore we do not require that the robot start along a boundary.



Approach

The maze will be prototyped using cardboard sheets oriented vertically, and its shape will be randomly generated in software. Then, the physical maze will be constructed with demarcated entry and exit points and the robot will be responsible for locating the end and returning a map of the shortest path.

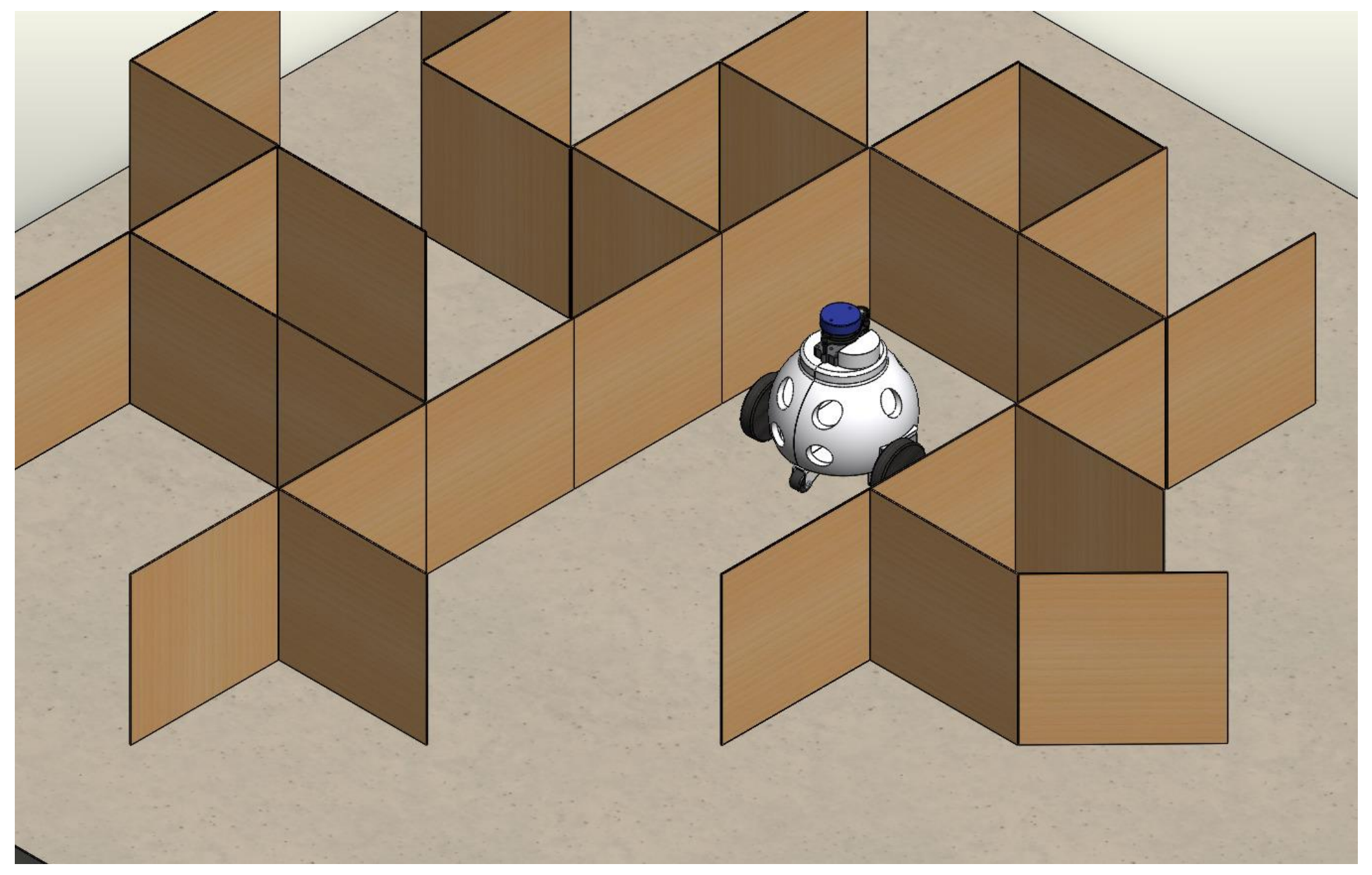


Figure 1: Robot in a Cardboard Maze Subsection

Software

The following software tools are used in this project.

- C++: Path Planning and Embedded Software
- ROS 2 Humble: Message Passing, Visualization
- Git: Version Control, Programmer Collaboration
- Jetson Linux 18.04: Jetson Operating System

Hardware

The following hardware components are used in this project

- NVIDIA Jetson Nano: Embedded Computer
- RPLidar A1: Boundary Detection, Maze Mapping
- Odrive 3.6: Motor Control Board
- DC Voltage Step Down: Converts battery input to 5V 3A
- Turnigy Nano-Tech Battery: Voltage Source

Algorithm

To solve the subproblem of traveling between two waypoints in a known environment or section of the maze, the following algorithm from last semester is still implemented

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*)

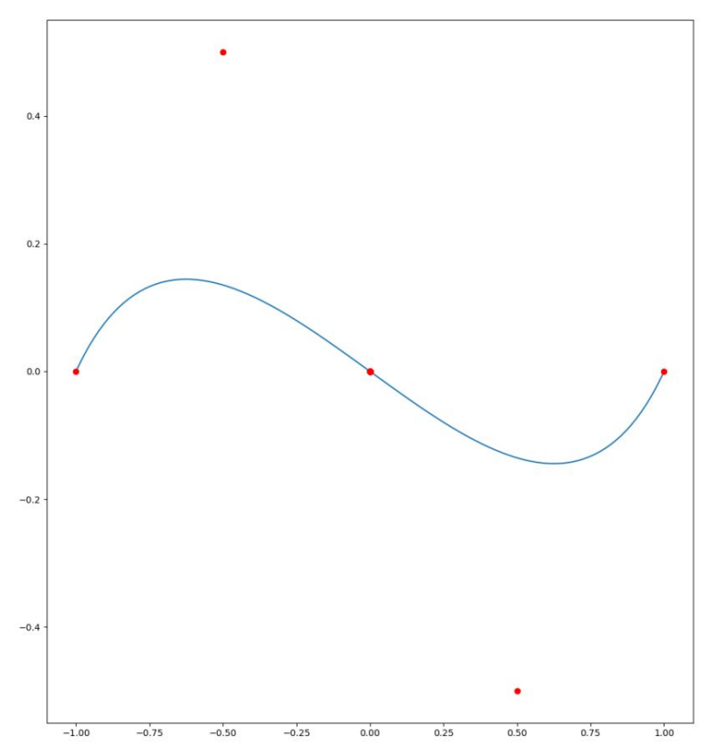


Figure 2: Bézier curve

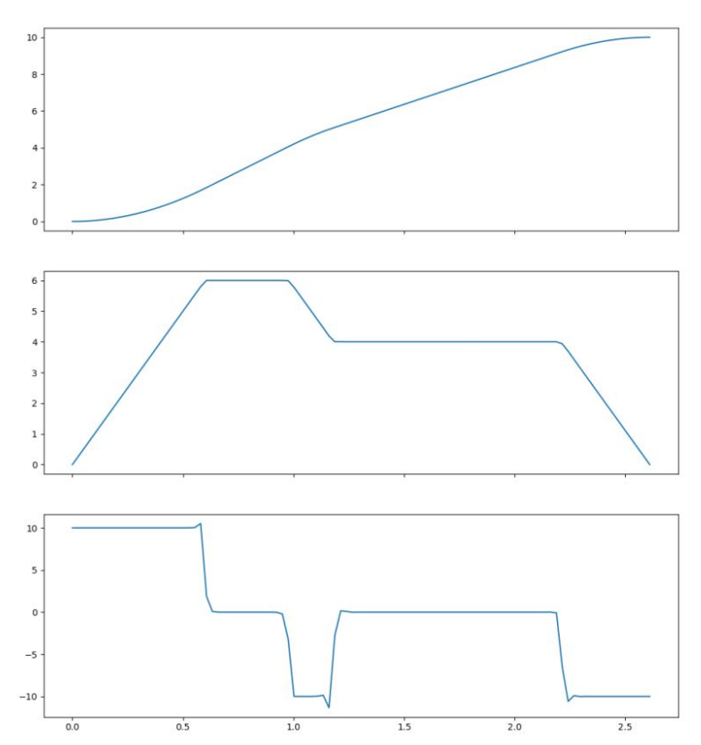


Figure 3: Velocity profile

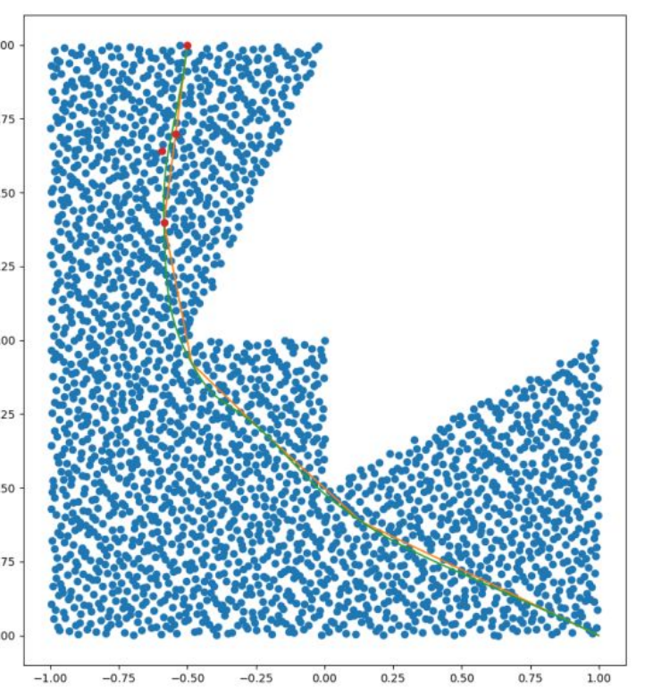


Figure 4: Fast Matching Algorithm Path

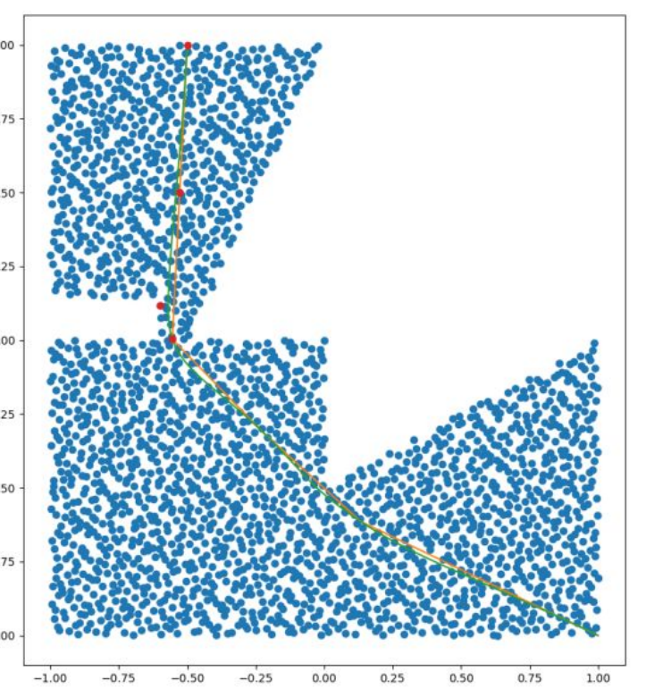
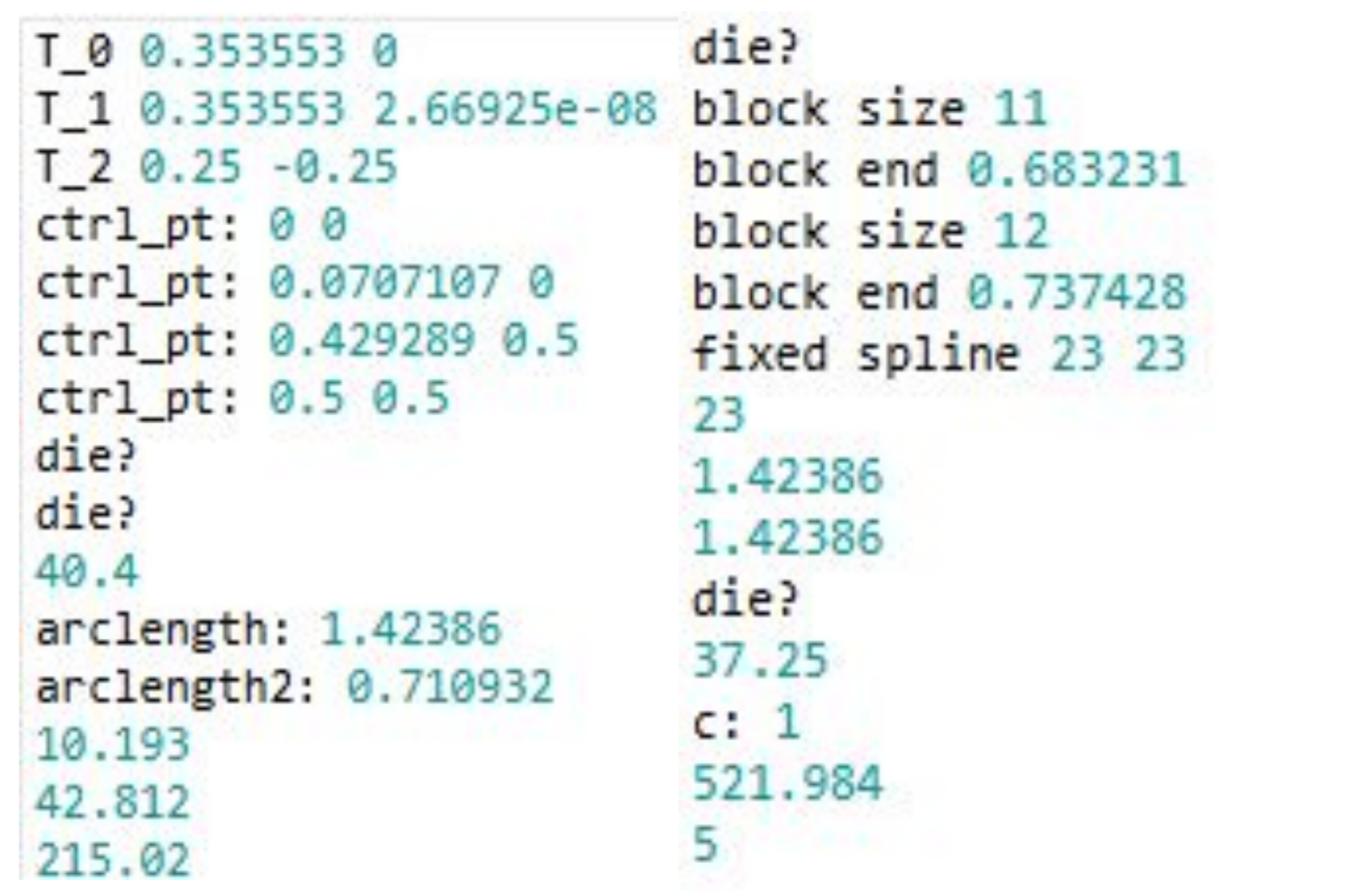


Figure 5: Bézier Curve Path

Dependencies

The following libraries were implemented in the codebase

- eigen: matrix and vector operations
- toppra: path parametrization
- nlohmann/json: C++ JSON serialization integration
- matplotlib: plotting used for testing



```
T_0 0.353553 0                die?
T_1 0.353553 2.66925e-08      block size 11
T_2 0.25 -0.25                block end 0.683231
ctrl_pt: 0 0                  block size 12
ctrl_pt: 0.0707107 0          block end 0.737428
ctrl_pt: 0.429289 0.5         fixed spline 23 23
ctrl_pt: 0.5 0.5              23
die?                          1.42386
die?                          1.42386
40.4                          die?
arclength: 1.42386            37.25
arclength2: 0.710932          c: 1
10.193                        521.984
42.812                        5
215.02
```

Figure 6: Path Planning Testing in a Virtual Environment

Design

The robot is designed in a spherical shape for ease of movement and navigation. Changes made to the CAD model this semester were made with assembly and maintenance in mind

- Exterior holes added for ease of assembly/disassembly and maintenance of internal parts.
- Wing-like extrusions generated to allow nut and bolt connections between both halves
- Thinner shell implemented to permit attachment of caster wheels

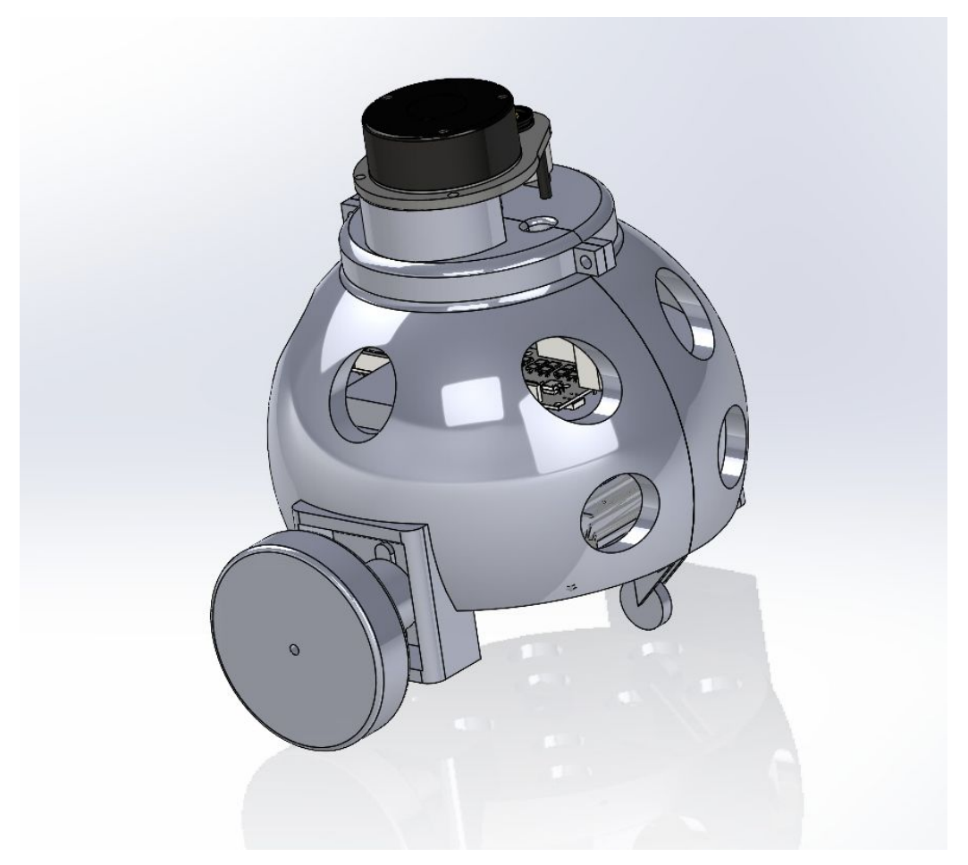


Figure 7: Robot Exterior

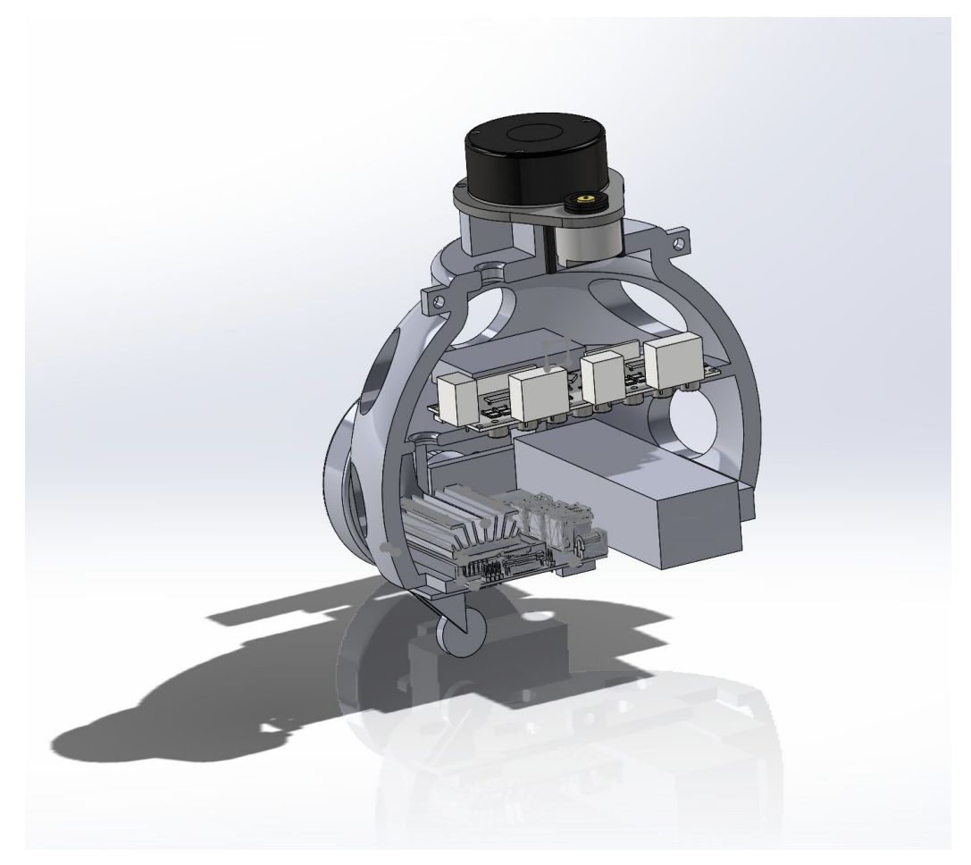


Figure 8: Robot Interior

References

The following references were implemented in this project

- *Numerical Methods: Design, Analysis. and Computer Implementation of Algorithms*
- *Kinodynamic Motion Planning for Mobile Robots Using Splines*
- *Efficient Computation of Bezier Curves from their Bernstein-Fourier Representation*

Future Work

There are several remaining tasks for next semester:

- **Software:**
  - Finish configuring Jetson Nano environment
  - Write software for RPLidar interfacing
  - Begin developing code for kinematics
- **Hardware:**
  - Assemble the robot body using the new shell
  - Finalize motors, wheels, and gearbox design
  - Explore methods of compacting current build

Documentation

The QR code below can be followed to reach the codebase

Texas A&M University Robotics Team and Leadership Experience (TURTLE) Lab