Brandon Rishi
brishi

TP1: Mapping and Path Planning with Cozmo

Description:

This project will focus on mapping and path planning with the Cozmo robot. A GUI will be shown that will allow a user to select the action of the Cozmo. The actions consist of map, three different items to retrieve, and stop. During map, the Cozmo will use its cliff sensor to gauge size of the table it is on, and detect obstacles (specifically light cubes) with its camera. The locations of the bounds and other objects will be stored within the Cozmo. Later, when an object is selected for pickup, the Cozmo will find the most efficient pathing to the object, pick it up, and bring it back to the user.

Competitive Analysis:

Unlike most other path planning/maze mapping robots, this allows offers convenience to a user by having the Cozmo retrieve small objects and bring them back to the user. However, it will be similar in the sense that it will utilize A* and other path planning algorithms.

Structural Plan:

The two major methods, map and retrieve, will be called within one main file that also has access to flask and the HTML file with the submit buttons. From there, mapping and planning will each have their own files and classes. Within mapping, a separate class will be made for the nodes located within the map that correspond to each coordinate. The path planning method will also have its own file and class to keep track of the relevant information when pathing. It will also have another class that tracks the heuristics of the potential environments at each coordinate within the map.

Algorithmic Plan:

The trickiest part of this problem will be creating the most efficient path planning algorithm. There are many pathing algorithms that already exist, such as A* and Dijkstra's algorithm, however I will need to implement them within Python and the Cozmo sdk. This will be done with heavy use of the Numpy library, as well as a well maintained map. Because there is not a limit in terms of direction, simply moving from coordinate to coordinate, even with diagonal movement, will not be efficient enough.

Timeline Plan:

- Tech Demo: Have the Cozmo moving and able to be manipulated via flask
- TP1: Have a basic mapping algorithm
- Sunday 11/25: Have a basic path planning algorithm
- TP2: Have a more in depth path planning algorithm (A* or Dijkstra's algorithm)
- Sunday 12/2: Use OpenCV to better identify obstacles
- TP3: Use OpenCV to detect walls and allow for ground movement and mapping as opposed to being limited to a table

Brandon Rishi 15-112:D
brishi TP1

Version Control Plan:

I will be using GitHub to manage versions of my code and back it up.

https://github.com/brishi28/TermProject

Module List:

Pre-MVP
- Cozmo
- Flask
- Numpy

Post-MVP:
- OpenCV