Project 3

Thunder Sharks

Ryan McNulty

Adam Park

Greg Seaman

Strategy Overview

- Decide on which cells to move to using depth-first search with iterative deepening
- Move to each cell using North Star and wheel encoders
- Once within center threshold, use camera to center
- Camera centering takes into account current location in maze

MazeStrategy

- Search map for highest value paths starting at length 2
- If no paths of value, increase path length up to 35 (depth-first iterative deepening)
- Value determined by points on path
  - Bonus given for straight paths
  - Penalty given for going in the top row where North Star data is bad and bottom row where lighting is poor
- Call moveToCell() on next path location
- If path blocked, calculate a new path

moveToCell()

- Used to navigate to next cell based on path produced by MazeStrategy
- Takes the index of the next cell to move to
- Turns to cell location
- Set goal (x,y) to current Kalman plus cell distance in new direction
- Call moveTo() as in project 1 to get within 10cm of cell center
- Once within threshold call centerInCell() to center bot

centerInCell()

- Gets list of squares like in project 2
- If the robot sees no pairs, turn to look for squares
  - Array holds direction for each cell where the bot can see a pair of squares
  - Turn to direction plus a small random theta in [-15°, 15°]
- If the robot sees pairs, use visual error to center
  - Strafe based on the center of the largest pair of squares
  - Move forward or back based on height largest pair
- Center until error threshold reached or too many iterations

Improvements

- Added state machine to centering algorithm for robot to correct in all four directions
- Reset Kalman with each cell to avoid error accumulation
- Changed Kalman uncertainty appropriately
- Additionally used height of squares for centering