Project 2

Thunder Sharks

Ryan McNulty

Adam Park

Greg Seaman

Navigation Overview

- CameraPose handles processing of squares
- Movement structured through calls to moveTo()
- TurnTo() handles turning
- StrafeTo() centers robot in hallway

CameraPose

- Camera images are filtered by cvInRangeS
  - Two calls used for separate pink values, or-ed together
- findSquares called on both filtered images
- Squares on top of each other removed with removeOverlap()
- matchSquares() creates pairs of squares
- getCenterError() numerically represents error from the camera image

MatchSquares

- Test each square with each subsequent square in the list
  - If y values are close, x values are far, and centers in top 3/5 of screen
  - Add both squares to squarePair structure
- After all squares matched, sort list
  - Largest squares most useful for calculating error
- getCenterError() called on these pairs
  - If no pairs, return 0, tell robot to use other sensors
  - Else return midpoint of largest pair minus center of image

MoveTo

- Updates and filters WE and NS as in project 1
- Camera image data processed every three updates
  - Strafe to center of hallway with strafeTo()
- turnTo() called on goal theta, same as project 1
- Error distance x and y calculated with respect to the robot
  - If y-error large move forward according to PID
  - If x-error large call strafeTo()
  - Else at cell center

StrafeTo

- Error calculated using getCenterError()
- If error greater than threshold, strafe left or right according to error sign
- Call turnTo() to ensure robot theta still correct

- StrafeTo() called on every camera update and when the error in the robots x direction is large enough