Autonomous Mobile Robots

CDA 4621

Lab Report 3

Motion Planning - Motion to Goal and Bug Zero

Claude Watson

Objective

The primary goal of this lab was to implement a motion-planning strategy for navigating a robot towards a specified goal (represented by a yellow cylinder) while avoiding obstacles. This lab involved developing the 'Motion to Goal' and 'Bug Zero' algorithms. These approaches utilized the robot's RGBD camera and LIDAR sensors to detect obstacles, maintain an appropriate distance, and adjust its path to avoid obstacles until it could resume its trajectory toward the goal.

Methodology

Task 1 - Motion to Goal

The robot needed to identify and navigate toward a yellow cylinder using the camera and LIDAR sensors. The main steps involved:

1. Detecting the Goal: Using the camera's object detection capabilities, the robot detected the goal object, noted its position, and calculated the distance and orientation from the goal.
2. Centering the Goal in the Frame: A PID controller adjusted the robot's rotational speed, aiming to center the goal within the camera's frame.
3. Moving Toward the Goal: Once the goal was centered, the robot moved forward and stopped within 0.5 meters from it.

Key Functions

1. Distance and Orientation Calculation

   - distanceToGoal(goal_x, goal_y): This function calculates the Euclidean distance from the robot to the goal using GPS coordinates.

   - orientation_to_goal(goal_x, goal_y): This function calculates the robot's orientation to the goal using the compass readings and position of the goal.

   2. motionToGoal()

   - This function combines goal detection and movement, rotating the robot to center the goal in the camera frame and then moving forward until the target distance is reached.

- Input: Camera recognition objects and LIDAR data.
- Output: Motor velocities to move toward the goal.

3. PID Control

- centerPID: Centers the goal within the camera's frame by adjusting the rotational speed based on the position of the goal in the frame.
- forwardPID: Controls the forward speed based on distance to the goal, aiming to stop the robot within 0.5 meters.

Task 2 - Bug Zero Algorithm

The Bug Zero algorithm was used for obstacle avoidance. When an obstacle was detected:

1. Obstacle Detection: The robot switched from the motion-to-goal strategy to obstacle avoidance using LIDAR to measure the distances from nearby objects.
2. Wall Following: The robot followed the obstacle boundary until it could resume a direct path toward the goal. This was achieved using PID control for angular velocity adjustments.
3. Path Resumption: The robot continued checking for a clear path to the goal while following the obstacle boundary, resuming the direct path once possible.

Key Functions

1. wallPID() and wallFollow()

- wallPID: Adjusts the robot's angular velocity based on the distance error to the obstacle's edge.
- wallFollow: Controls the linear and angular velocities to maintain a fixed distance from the wall and avoid collisions with the obstacle.

2. rotate()

- Allows the robot to perform a 90-degree turn when an obstacle is directly in its path.

3. Main Loop Logic

- Detects when the goal is visible and controls the robot using motionToGoal.

- When the goal is not visible and an obstacle is detected, the robot engages rotate to avoid the obstacle and wallFollow to trace the obstacle boundary.

Formulas Used

1. Distance Calculation

$$Distance = \sqrt{(goal\_x - gpsX)^2 + (goal\_y - gpsY)^2})$$

2. Orientation Error Calculation

$$Orientation\ Error = ((goal\_angle - current\_orientation + \pi)\ \%\ 2\pi) - \pi$$

3. PID Control

- Proportional Control: P = Kp * error
- Integral Control: I = Ki * sum(error) * dt
- Derivative Control: D = Kd * d(error)/dt
- Total Output: Output = P + I + D

Results

The robot successfully completed both tasks:

- In Motion to Goal, the robot was able to center the goal in its camera frame, approach it, and stop within the 0.5-meter target distance.
- The Bug Zero Algorithm allowed the robot to navigate around obstacles smoothly, maintaining the correct distance to avoid collisions and resume its direct path to the goal when possible.
- When right wall following in Bug 0 algorithm, the robot successfully reached the goal within the 3-minute mark. In contrast, when left wall following, the robot eventually reached a point in the maze where it could not exit.

Conclusion

I was successfully able to program the robot to complete both tasks. However, some problems were encountered during each task.

Task 1

The motion to goal task was fairly simple. Only two problems were encountered: setting the velocities too high and what to do when the goal was not in the lidar sensor range. Setting the velocities too high caused the robot to jerk violently and sometimes flipping. To combat this, the velocities were cut in half. To address the other issue where the goal was out of the lidar sensor's range, I had the rotate until the goal was in the center of the camera's frame, then slowly move forward until it was within the lidar's range. Once it was in range, the velocities were controlled by the forwardPID() function.

Task 2

The main issue when completing task 2 was figuring out the logic for the main loop. All the functions in this task were already made in previous projects. With the help of generative AI, I was able to properly implement a main loop that fulfilled the needs of the task.

Videos

Task 1: LAB3TASK1.mov

Task 2: LAB3TASK2.mov

Author Statement

I developed all the code and written report with the following exceptions:

Task1: Lines 109 - 143

Task2: Lines 278-324

[Claude Watson: C.W., 11/11/2024]