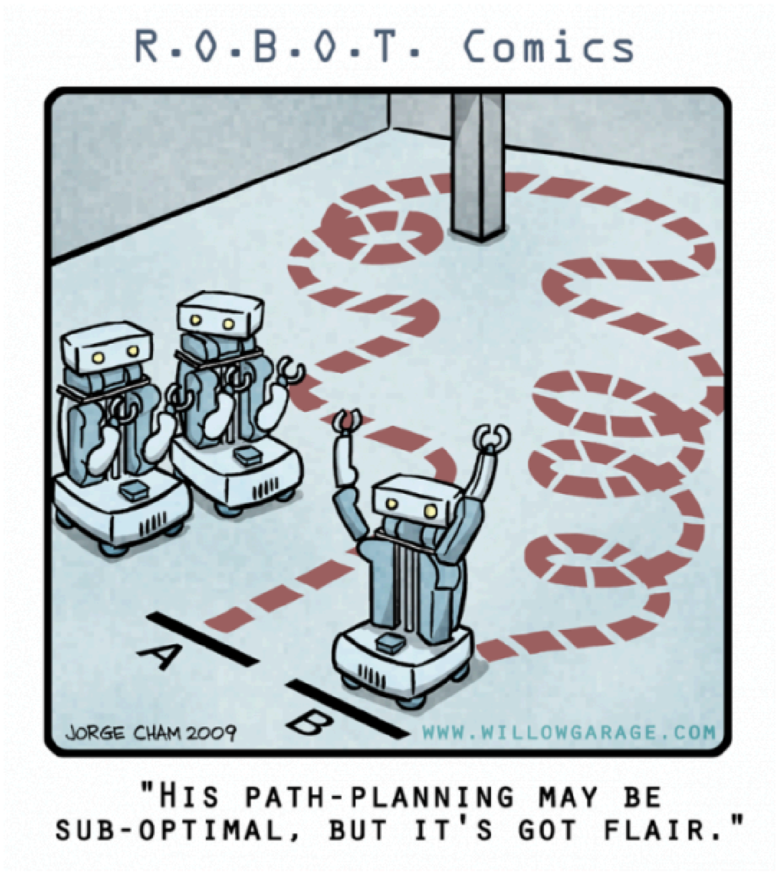
COMS4045A/COMS7049A-ROBOTICS

Project Report - SurveillanceBot



Tumi Jourdan ~ 2180153
Luca von Mayer ~ 2427051
Mohammad Zaid Moonsamy ~ 2433079
Shakeel Malagas ~ 2424161

17/05/2024

School of Computer Science and Applied Mathematics

INTRODUCTION

This project report outlines the development and implementation of a surveillance robot tasked with navigating an environment. Leveraging the capabilities of a TurtleBot, we utilized a combination of mapping, image processing, pathfinding, and control algorithms to navigate and scan the environment effectively. The core components of our approach included map generation using gmapping, image dilation for enhanced obstacle avoidance, the Probabilistic Roadmap (PRM) for path planning, and the A* algorithm for determining the shortest path. Additionally, we implemented a Proportional-Integral-Derivative (PID) controller to ensure precise and stable movement of the robot. This report details each of these components, their integration, and the overall performance of our surveillance system.

Map Generation and Image Processing

Map Generation

We used the gmapping module to generate the map that we used to feed into the AMCL ros module. To make the map we had to slowly step through the area, making sure to keep close to any distinguishable physical features (such as corners). Making the inside of the house was easy, but the outside area proved troublesome, especially in areas the robot moved with little to no visual information.

Image processing

To provide a greater margin for the obstacle avoidance, we implemented image dilation to expand the obstacles in the map generated from gmapping.

Probabilistic Roadmap (PRM)

The Probabilistic Roadmap (PRM) algorithm is a popular method used for pathfinding in complex environments. It is particularly useful for navigating through spaces with obstacles

Mapping

1. **Random Point Generation**: The algorithm begins by randomly generating a set of points within the map (or relevant section of the map for efficiency), and adds both the start and goal nodes. These points serve as potential nodes for the roadmap.
2. **Collision Checking**: For each generated point, the algorithm checks whether the point is within a free space (i.e. not colliding with any obstacles).
3. **KDTree for Nearest Neighbor Search**: This data structure is used for efficient nearest neighbor search. By organizing points in a k-dimensional space, the KDTree allows for rapid retrieval of the closest points, significantly speeding up the roadmap construction process. This efficiency ensures the algorithm remains scalable and performs well even with a large amount of random points..
4. **Connecting Points**: Once the points are generated, the algorithm connects each point to its nearest neighbors (found with the KDTree) to form a network or graph. The connections (edges) between

points are checked for collisions by checking if every point (discrete number of points based on robot radius) along the edge collides with an obstacle. Collision free edges are added.

5. **Data Structure**: The roadmap is represented as a graph where each point is a node, and each collision-free connection is an edge. This graph is stored in a dictionary where each key is a point (a tuple of coordinates), and the value is a list of adjacent points (neighbors).

A*

To find the shortest path between a start point and a goal point, the A* algorithm is used. The algorithm explores the roadmap by evaluating nodes based on their cost (distance traveled) and a heuristic (euclidean distance to the goal). It maintains two sets: an open set of nodes to be evaluated (frontier) and a closed set of nodes that have already been explored. Starting from the initial point, A* searches through the roadmap, selecting the path that minimizes the total cost until it reaches the goal.

PID

The PID control code is designed for the robot navigation system, leveraging proportional, integral, and derivative (PID) gains to compute both linear and angular velocities for reaching a target position. The PID class is initialized with specific gains and calculates errors to determine the control outputs. The code handles the linear distance to the target, as well as the angular rotation to adjust the robot's orientation. The robot's current state is extracted from the AMCL pose message, and the desired yaw is calculated based on the target position in relation to the goal. The code ensures the robot moves towards the target, correcting its path as necessary, and stops once the goal is reached within a specified threshold. The code separates the angular rotation from the linear movement of the robot to ensure that it is either rotating or moving but not performing both at the same time to reduce confusion.

CONCLUSION

In conclusion, our surveillance robot successfully demonstrated the ability to explore, map, and navigate an environment. By integrating advanced mapping techniques with robust pathfinding algorithms, the robot was able to efficiently avoid obstacles and reach designated locations. The implementation of image dilation further enhanced obstacle detection and avoidance, contributing to the overall effectiveness of the system. The use of the PRM and A* algorithms ensured reliable path planning and execution, while the PID controller provided stable and precise robot movement. The inclusion of image processing allowed for accurate detection of the target object. This project not only showcased the practical application of these algorithms but also highlighted the potential for further advancements in autonomous surveillance technology.