Lab 6 Report: Path Planning

Team 1

Vasu Kaker
Ben Evans
Emmanuel Anteneh
Johlesa Orm
Michelle Zbizika

6.4200/16.405 (RSS)

April 27, 2024

1 Introduction

Author of section:

The goal of this lab was to plan a path and execute controls to follow that path to its conclusion. Specifically, we use a breadth first search (BFS) algorithm in order to plan a trajectory. A pure pursuit algorithm with the localization program was implemented in order to follow that trajectory.

Overall, ...

2 Technical Approach

2.1 Path Planning

2.1.1 Image Dilation/Erosion

Author of section:

The obstacles were represented by an occupancy grid, which was a 2d array corresponding to x, y values in pixels, with value 0 if the space was free at 100 if the spot was occupied. In order to account for the real world size of the robot, we increase the size of the representation of the obstacles by the radius we desire for clearance around the robot.

2.1.2 Search Based

Author of section: Emmanuel Anteneh

Primary search based planning algorithms include BFS and A*. We implemented BFS to find an optimal path from our initial location to a given goal location. Our BFS implementation discretized the search domain into a grid space search space. Upon receiving a real world starting position and ending position, we translate them into the pixel representation of our grid space. We then initiate a standard BFS algorithm where a queue of potential paths to the goal state are built by repeatedly branching one step out from the end of the oldest generated path, discarding any paths containing coordinates that are occupied (have a value of 100 or -1 in the occupancy map). Once a path to the goal state is found, the coordinates are translated back into real world space and are used as the trajectory to be followed.

2.1.3 Sampling Based

Author of section: Michelle Zbizika

Primary sampling based planning algorithms include PRM (Probability Road Maps) and RRT (Rapidly-exploring Random Trees). An additional optimization of RRT is RRT*. We implemented RRT in addition to BFS. We used the collision checking function. The collision checker checks One small modification we included was at the start of the RRT search, it would check if there's already a valid, direct linear path to the goal. If there was, then output this path. If not, then the iteration starts. At every iteration, a random collision-free point on the map was sampled. Then, the vertex on the graph that is closest to this point was found. The RRT travels the function goes the specified maximum distance towards this point. In theory, sampling-based path planning is fast but can sometimes result in inefficient paths.

2.2 Trajectory Following

The predefined trajectory found by the path planner is followed by the robot by implementing pure pursuit and particle filter.

2.2.1 Localization

Author of section:

The localization nodes from lab 5 were used to estimate the location of our

robot in real time. This localization protocol was developed in part based off of text 'Probabilistic Robotics'(1)

2.2.2 Pure Pursuit

Author of section:

A pure pursuit algorithm was used to control our robot to follow a given trajectory using its estimated location. The pure pursuit algorithm received an array of points that represents the desired trajectory. The lookahead distance was set to 1 meter, the speed was set to 1 meter and the wheelbase length was set to 0.381.

2.3 Integration

Author of section:

The planned trajectory path was represented as piecewise points with x, y coordinates. After the path planning and pure pursuit algorithms were completed, these components were combined so that path planning and following were done in real time.

(2)

3 Experimental Evaluation

3.1 Simulation

3.1.1 Path Planning Performance

Author of section:

The main considerations for comparing the performance of the algorithm in simulation were: computation time, which includes initialization time and path computation time in a variety of scenarios, and the optimality of the trajectory, which we measure by the total length of the computed trajectory and the number of turns in the trajectory. The average time for initialization (out of 5 initializations) was 000 seconds for RRT and 000 seconds for BFS. The compute time trajectory distances for various path planning scenarios is summarized in the following graph. Initialization, straight line path, corner

Table 3.1.1 has a caption:

Average path planning computation time.

Scenario	RRT	BFS
Init time	cell5	cell6
Straight line path	cell8	cell9
corner 1	cell5	cell6
corner 2	cell5	cell6
Opposite corners	cell5	cell6
cell1	cell5	cell6

3.1.2 Trajectory Following Performance

Author of section:

3.2 Real World Testing

Author of section:

4 Conclusion

Section Author:

As a whole, we were successful in implementing robotic localization using particle filter in simulation. The main remaining work is to adapt our implementation to work on the physical race car, and conduct further experimental analysis in real world environments.

Implementing particle filter (aka MCL) required implementing motion model and sensor model. Given our estimations of the robot's current pose and orientation, motion model uses odometry data to update our estimations to reflect their likely future states. Sensor model then predicts the probabilities of each estimation being the true position and orientation. The models are then both utilized by particle filter which continuously estimates potential future poses/orientations for the robot, and weights the estimations by the likelihood that they reflect the true state of the robot, while averaging the current set of estimations to get future odometry.

Evaluating our particle filter in simulation yielded good results, as error stays low in most environments and noise values. While our implementation was

successful in simulation, running MCL on the physical racecar introduced challenges that we still need to address, such as resolving issues with running out of storage when running the algorithm. Moreover, once these problems are resolved, we will need to evaluate and further tune our particle filter to achieve ideal performance.

5 Lessons Learned

Person 1: something something

Michelle Zbizika: I learned that it's important to have multiple people work on a component so that we can make sure we're implementing something correctly and to be more creative in making the most efficient system.

Emmanuel Anteneh: I found this lab rewarding and frustrating, as it was great to see the final result after working on each component, but was a bit discouraged that we weren't able to get as good of a performance as I was hoping. I think a large part of this was poor time management, and in the future it would be good for our team to allocate time more effectively. Overall though, it was very cool to see our robot follow paths on its own, and I came away from this lab with a deeper appreciation for path planning and pure pursuit.

References

[1] S. Thrun, W. Burgard, and D. Fox, *Probabalistic Robotics*, 1st ed. Cambridge MA: MIT Press, 2005.

[2] RSS_Course_Staff, "Lab 6: Path planning," Available at https://github.com/mit-rss/path$_p$*lanning*(2024/04/24).