ME 570: Robot Motion Planning

Homework 4 Report

By Cameron Cipriano

12/02/2021

**Problem 1: Graph Search**

Question 1.1 \_code\_: Graph.heuristic

Implemented using numpy’s linalg.norm() and subtracting the two points’ physical locations in the graph

Question 1.2 \_code\_: Graph.get\_expand\_list

Implemented using the graph’s neighbor list

Question 1.3 \_code\_: Graph.expand\_element

Implemented by following the logic of lines 9-16 of the A\* pseudocode

Question 1.4 \_code\_: Graph.path

Implemented by basically treating the nodes as a linked list and iterating over the backpointers until I got to the beginning

Question 1.5 \_code\_: Graph.search

Implemented by following the logic of the A\* pseudocode

**Problem 2: Application of A\* to the Sphere World**

Question 2.1 \_code\_: SphereWorldGraph.\_\_init\_\_

Implemented by using my lambda functions from the previous homework and modifying it to translate values into 0s and 1s before the grid2graph function would vectorize that into booleans

Question 2.2 \_code\_: Graph.search\_start\_goal

Implemented by following the description’s indication to use the nearest\_neighbors function with number of neighors as 1 and calling search.

Question 2.1 \_report\_: nb\_cells discretizationDiagram, engineering drawing

Description automatically generatedA picture containing text, mask, wheel, vector graphics

Description automatically generatedA picture containing text, wheel, gear

Description automatically generated

Top Left: Discretized with 8 Nodes

Top Right: Discretized with 40 Nodes

Bottom Left: Discretized with 80 Nodes

Question 2.2 \_report\_: SphereWorldGraph run\_plot method

Diagram

Description automatically generatedDiagram

Description automatically generated

Left: Discretized with 8 nodes goal location 1

Right: Discretized with 8 nodes goal location 2

A diagram of a heart

Description automatically generated with low confidenceA diagram of a heart

Description automatically generated with low confidence

Left: Discretized with 40 nodes goal location 1

Right: Discretized with 40 nodes goal location 2

A picture containing chart

Description automatically generatedA picture containing chart

Description automatically generated

Left: Discretized with 80 nodes goal location 1

Right: Discretized with 80 nodes goal location 2

Question 2.3 \_report\_: A\* Behavior given choice of nb\_cells

Given the choice of nb\_cells, A\* has a few properties. The first would be the speed of execution, as the number of cells remains small, A\* will complete very quickly. If the number of cells is very large, then it will take much longer.

Another property based on the choice of nb\_cells is how close the path is to the true path that a robot would take to get to a goal. The higher nb\_cells, the higher the resolution the environment is discretized into, thus allowing for finer control of where the robot will go in the environment.

Something else that is noticeable is the relative size of the explored nodes for certain sizes of nb\_cells. When nb\_cells is still below a reasonable size, the A\* planner will tend to explore more nodes as the nodes have relatively similar distances to the goal. When nb\_cells is larger, this won’t be as apparent since each node will be slightly different from the others, allowing for a more direct path to the goal.

Question 2.4 \_report\_: A\* Behavior with respect to the potential planner

The potential planner from homework 3 has no effect on the behavior of A\*. This is because the graph simply takes the presence of some potential to be true and the absence of potential to be false, and thus an obstacle. As long as the potential function chosen spans the entire environment (which is the case for attractive potentials defined as a distance), then the only metric that matters is the resolution of the graph.

**Problem 3: Application of A\* to the Two-Link Manipulator**

Question 3.1(a) \_report\_: TwoLinkGraph.load\_free\_space\_graph

Question 3.1(b) \_report\_: TwoLinkGraph.plot

A picture containing diagram

Description automatically generated

Question 3.1(c) \_report\_: TwoLinkGraph.search\_start\_goal

Question 3.2 \_report\_: Plot the points obstacle\_points

Chart

Description automatically generatedChart, diagram

Description automatically generated

Left: obstacle\_points with straight end-effector positions

Right: obstacle\_points with end effector rotated by π/7

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

A picture containing diagram

Description automatically generatedA picture containing histogram

Description automatically generated

Left Side: Easy theta\_start to theta\_goal configuration and final path

Right Side: medium theta\_start to theta\_goal configuration and final path

Question 3.3 \_report\_: Comment on the *unwinding* phenomenon in the easy case

For the easy case, if we take a look at the path in the bottom left image, it is simple to see that despite the final and ending theta for the first link being identical, there is no straight black line connecting the two final theta’s. Because of this, the robot thinks that its range of motion is restricted in particular areas and must maneuver itself in a complicated way to avoid the “obstacles” that it thinks are present. Because we are not using the torus option, it doesn’t know that the path can wrap around to the other side, allowing the first theta to stay fixed and find the true optimal path. This unknown knowledge causes the sort of “back and forth motion” when traveling up the entire graph, unwinding itself.

Question 3.4 \_report\_: Comment on the obstacle closeness issue and practical solutions

Given the fact that A\* is optimal with an optimistic heuristic, it is always going to try and hug the obstacles because that will be the shortest path between any start and the goal. In order to potentially avoid the closeness to obstacles, we can arbitrarily expand the borders of all obstacles. This expansion of the obstacle’s borders will allow the A\* algorithm to still hug the “obstacle”, finding the true optimal path in the graph, while now avoiding the true obstacle’s dimensions very closely.