Lab 5 matge373

5.1

1. What is the difference between Configuration Space and Work Space?

Workspace is the physical space in which we work. Can be either 2D or 3D depending on the problem. The configuration space includes the degrees of freedom of an object. A car for instance have 3 physical degrees of freedom, and the configuration space of it would be:

- location in the plane (x/y)
- Angle (Θ)

Using these values, motion planning takes place in configuration space.

In essence the main differences seem to be that Work Space describes the environment in which we work in and the Configuration Space describes the possible configurations of the robot.

2. Which one is used when searching for a plan? Why is it used?

The Configuration Space is used when searching for a plan. This is because it provides a more concise representation of the system's states. Since the work space is abstracted away the search algorithm can focus on the relevant attributes and constraints of the system, exploring the space of possible configurations to find a valid plan. Effectively, searching in this space is simply more efficient.

3. What are the formal requirements for a robot to be considered holonomic?

A robot is considered holonomic if the controllable degree of freedom is equal to total degrees of freedom. This basically means that the robot can move to any location without relocating itself before moving (assuming there are no obstacles blocking its way).

5.2

The planners used in this lab are RRTstar, PRMstar and SPARS.

How does PRMs work?

(for example, what happens during preprocessing / during actual planning)

PRMs randomly generates a configuration q in free config space. If this q was previously unreachable, it is added to M, which is an empty roadmap. There is also the case where q already was reachable, but now connect two different configs (that were not connected). Then q is added together with associated edges to M. Repeat these two steps until sufficient coverage has been reached.

This basically means that in the construction phase (preprocessing) PRMs expand a sort of "explore area", with the goal of covering as much space as necessary. There is an example of when sufficient coverage is met, and if n is 1000 it is very likely that 99.9% of the free config space is covered.

After this, node placement occurs, which is random but not always uniform. These nodes can then be connected in the query phase (adding edges basically), check if valid (not going

through obstacles). Then one can run A* search through this node network to find a path from start to finish.

How does RRTstar work?

RRTs are “Rapidly-exploring Random Trees”. They iteratively grow a tree structure by randomly sampling different configurations and connecting them to the nearest existing node. Doing this “tree expansion” it eventually reaches the goal (if possible) and produces a path from the root to the goal configuration.

How does SPARS work?

SPARS provinces asymptotic near-optimality and includes a meaningful stopping criterion. Its goal is to construct a sparse roadmap while preserving important connectivity properties. It removes redundant edges from the iterative sampling using geometric and topological criteria. This results in a reduced graph which keeps key connectivity information for efficient planning.

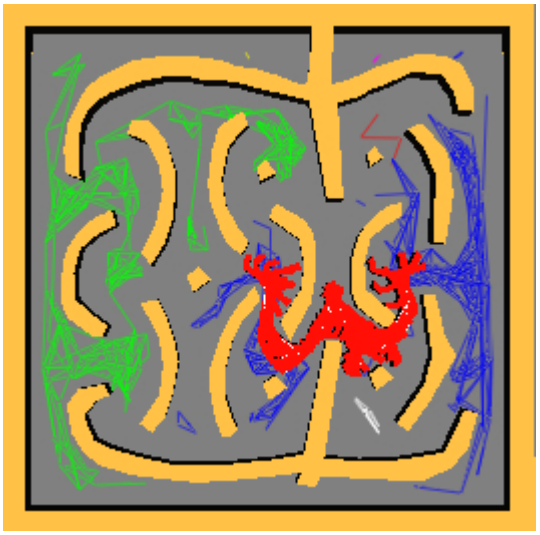
Map1 - Maze_planar_env

All of the planners below were given the same starting point and goal. These were the same as in the lab description (not .03, just 0 though), start = (0, 0) and goal = (26.00, 0). The planning time was set to the default 10 seconds for the RRT and PRM planners, but SPARS didn’t find a solution within 20 seconds the first time, so it was set to 30 seconds.

- RRTstar

Solution found in **1.061152 seconds**
Interpolating solution path to **55 states**
Distance: **82.31514424459365**
Total rotation (radians): **10.304462388781916**

- PRMstar



Solution found in **0.907516 seconds**
Interpolating solution path to **50 states**
Distance: **72.81645181429847**
Total rotation (radians): **14.278145628955265**

- SPARS

Solution found in **24.172126 seconds**
Interpolating solution path to **107 states**
Distance: **161.9295933362306**
Total rotation (radians): **15.281689605898915**

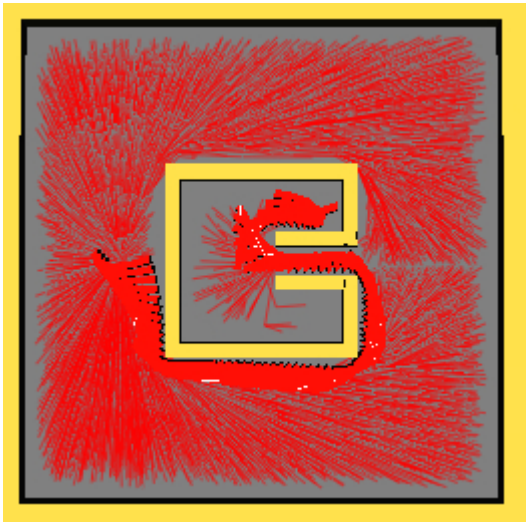
Discussion Map1:
For the first map, it seems like the PRMstar performed the best out of the three. It was the fastest to come to a solution as well as use the least amount of states. We can clearly see the different config spaces from the preprocess (different colors in search paths) and that it expanded quite far but still managed to find a solution very quickly. It looks like the RRT expanded a bit too much in the wrong direction for it to be able to reach a solution as quickly, but it still managed to do it in 1 second. The SPARS planner was very inefficient for this

problem, probably because it has so many different paths to choose from which does not seem to be very good for this planner. It started searching in many different directions but eventually came to a solution. This solution, however, reached 107 states. So in comparison, this planner performed poorly on this problem instance.

Map2 - BugTrap_planar_env

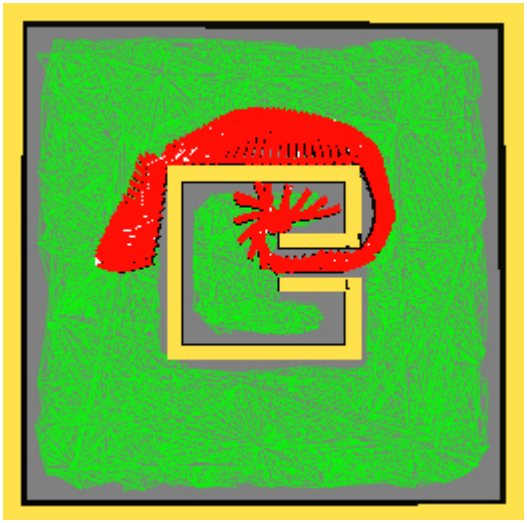
For the 2nd map I chose BugTrap because I thought it would be interesting to see how the search looks and if this, rather simple problem (for us humans), is simple or complicated for the planners. For this map, I had to change the time to 20 seconds for the SPARSE planner or else it would not manage to find a solution. The other two still run at 10 seconds. All start and goal values are the same, start = (-30, 0) and goal = (9, 11).

- RRTstar



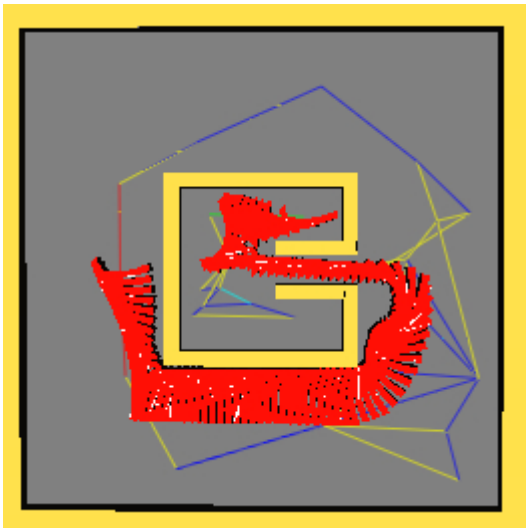
Solution found in **10.006717 seconds**
Interpolating solution **83 states**
Distance: **124.75106880901482**
Total rotation (radians): **9.815301374677189**

- PRMstar



Solution found in **8.918065 seconds**
Interpolating solution path to **86 states**
Distance: **129.19617451501284**
Total rotation (radians): **11.956758209936503**

- SPARS



Solution found in **20.196460 seconds**
Interpolating solution path to **101 states**
Distance: **154.70039048915885**
Total rotation (radians): **10.527872155948998**

Discussion Map2:
For this map, the result was much closer than previously. This is to be expected though as the map itself is less complex than the previous one. Once again the PRM “won” with the shortest time, but it did not beat the RRT when it came to minimizing states. Worth noting here is also that I had to rerun PRM two times as it did not find a solution within 10 seconds. I extended its time but then it might have gotten lucky and reached this solution within 9 seconds. The RRT and PRM both search through the entire space possible basically, while

the SPARS one illustrates how unnecessary edges are removed and provides a more “clean” search path. For this problem, the SPARS planner was still the worst one, but it was not as bad as for the first problem. The search space being smaller might indicate that it is less resource heavy, so that could be an advantage for larger problems, but if this is not true, it seems like RRT and PRM are simply better!