COMP3702 Assignment 1 – Report

Name: Sean Manson
SID: 42846413
Tutorial: I02 Tuesday 10am

Problem Formulation

The task was to program a pathfinding AI robot consisting of a base and straight arms connected by joints. The robot had to travel through square workspace between two given configurations while avoiding rectangular obstacles. To solve this task, I made several simplifications and assumptions that reduced the problem of designing an agent to one which could be easily solved.

One of these simplifications was the splitting of robot movements into discrete steps, where the robot's base could only move up to 0.001 units, and joint angles 0.1° per step. This meant that the **action space** (the set of moves the robot can perform) could be easily defined as all robot configurations within a single step of the current configuration. Another assumption made was that the robot can see the entire workspace and obstacles at all time, removing the need to define a **percept space** and **percept function**.

The only dynamic aspect of this problem is the robot. Hence, because each individual robot configuration can be discretely defined, the **state space** for our agent simply consists of all valid configurations of the robot, and our **world dynamics** are simply moving from one of these configurations to another as our action space permits. We can think of each configuration as being an N-dimensional point in **configuration space**, where N is the number of arm joints + 2 for the x and y axes.

Ultimately, the problem for this assignment is to find a series of valid points in C-space that lead sequentially from the start configuration to the end configuration.

Method Description

Initially, I approached this task by using **Rapidly-exploring Random Trees** (RRTs), which are a method that starts at a particular node in space and quickly generates a tree expanding from this point in all dimensions, working around obstacles and through tunnels as needed. By using two of these – one for the start and one for the goal – I planned to run until they overlapped, and then create a path using this overlapping point from the root of one tree to the root of the next. I soon found, however, that this method becomes unacceptably slow for higher dimensions of C-space, as the degrees of freedom for expansion increase exponentially. As I wanted to have a robot which supported 12 arms or more, this method had to be scrapped.

The approach I decided on instead was a **probabilistic roadmap (PRM)**. This method works by taking random points within C-space (called 'samples'), repurposing them as nodes in a graph (the 'roadmap'). Each sample is tested to see whether they can reach other samples already in the roadmap, and if they can, an edge is added between their two nodes. After a certain amount of samples taken, we then test to see whether a path exists from the start node to the end node in the roadmap, and if so, we have our overall path for the robot.

In order to implement this, it was necessary to define how to tell if two nodes on the roadmap could connect. The definition I decided upon was whether the robot could travel from one to the other 'linearly', moving its arms and position constantly from one point to the next over a series of discrete steps. If it could do this without colliding with itself or an obstacle then the two nodes were considered to be connected. Collisions were defined as robot points or lines intersecting or being within obstacle boundaries.

The **A*** algorithm was used to find a path in the roadmap. This algorithm works by favouring nodes which bring us closer to the goal by assigning a heuristic weight to each node, in addition to a weight given by its distance from the start node. In practice, however, I found that the roadmap pathfinding process completes far quicker than finding samples, and so the difference between algorithms was negligible.

In order to improve this method, significant changes were made to the sample-finding process, which is the slowest part of the algorithm – the core change being the strategy used when finding samples. Instead of choosing entirely random configurations to use as samples, a path of 0.01x0.01 grid squares is found corresponding to the solution for a point robot. Samples are taken as random configurations contained within one of these squares. This path is pre-calculated using A* before the general method begins, and in practice does not add much overhead.

Several further improvements were also made. One of these prevents nodes outside a certain distance from being tested for connections, which sped up sample finding significantly. Another was the ability to use both a random selection strategy as well as the path strategy, with the former being selected every now and then with a small probability (20% or so).

Conceptual Arguments

For an entirely visible workspace, it can be seen that a probabilistic roadmap is probabilistically complete – meaning that as we take more samples, the probability that no path will be found despite one existing approaches zero (Kavraki, 1996). In other words, the main challenge with this method is not whether we can find a path, but whether we can do within a limited period of time.

The reason this probabilistic roadmap method is successful is because it is able to handle higher-dimension C-spaces without much increase in complexity, instead of taking exponentially longer to calculate. This proves invaluable for this particular problem, where the robot can have up to 14 degrees of freedom. It does this by reducing a complex workspace problem to the simple process of finding a path on a set of graph nodes.

The core limiting factor on this process, especially in higher dimensions, is the creation and connection of new sample nodes. This is due to the need to detect collisions with obstacles in the workspace, which requires a costly iterative process with hundreds or thousands of steps. Thus, to be effective, improvements were made to the method to decrease the amount of collision detection required. For example, when connecting nodes, only neighbouring nodes within a certain distance are considered, which greatly reduces the number of wasted collision detections to nodes which likely could not be reached anyway.

Another approach undertaken was to improve the sampling method, which reduces wasted collision detection by needing less samples to find a path. The path-based sampling method is able to generate paths through tunnels and tight areas far more efficiently, and so less computation time is necessary. To make this even more effective, a weighting system was used, where squares on the path that collide more often are chosen more, as they often represent tunnels or tight areas, which need more samples to be passed effectively.

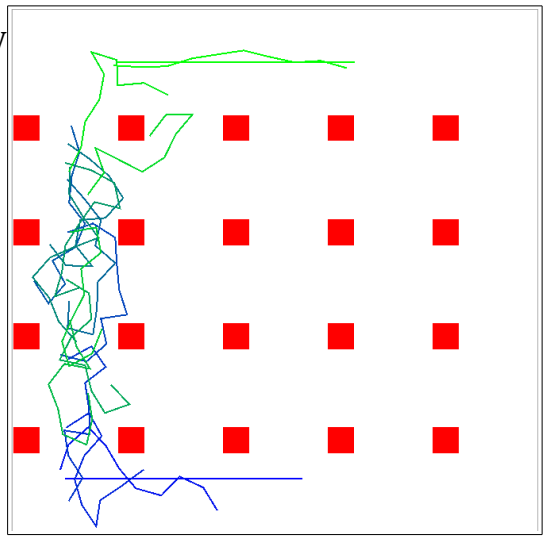
There may be cases, however, where purely path-based sampling may not be the best strategy. For example, there might be a path leading through a very tight tunnel which the robot arms are unlikely to fit through, with an alternate route around the obstacle existing. In order to accommodate for these cases, I combined both random and path-based sampling, choosing the former 20% of the time and the latter 80%. This ratio was chosen after originally implementing an epsilon-greedy choosing method, where it was found that path-based strategy practically always came out on top. Rather than waste overhead on calculating weights, I decided to hardcode the random percentages instead.

Though this AI should be able to solve most simple 12-arm problems within 60 seconds, there are some cases I doubt it would be able to solve effectively. Some of these are due to limitations of the probabilistic roadmap method – for example, it would be quite difficult to solve a problem where the robot has to move its arms around in a very tight space, as PRM tends to result in rather erratic arm angles and random positions. Also, cases where a grid path cannot be found for path-based sampling (i.e. cases with very tiny gaps between obstacles) would be practically impossible, as the chances that random sampling could find a path on its own through this gap would be tremendously low.

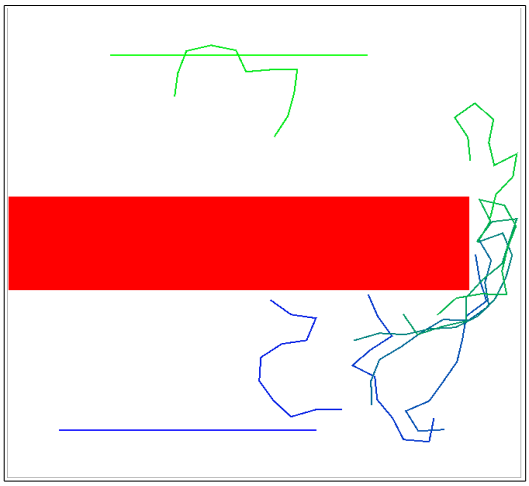
In general, I believe my robot AI is moderately successful at solving high-level problems, though I admit some improvements could be made for especially difficult cases.

Experimental Arguments

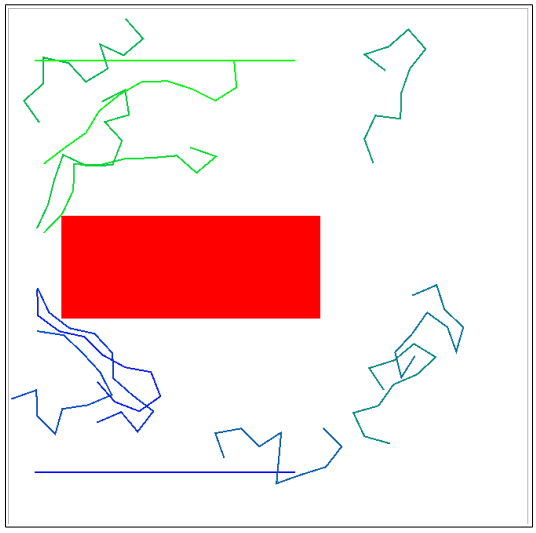
Several test cases were chosen to illustrate some of the points made above.



The first screenshot on the right shows the path taken by a robot with 9 arms as it tries to pass through a series of small obstacles. This took 20 seconds to calculate. It dexterously manoeuvres around each box to the goal position, demonstrating its ability to deform, fold and right itself as it moves. This demonstrates the power of the probabilistic roadmap method – a more general movement method would not be able to handle the complex arm movements required to solve this problem.



The second screenshot justifies the use of a path-based algorithm over a random one. The robot is able to squeeze through the tiny gap because the path algorithm chosen prefers squares on the path to useless configurations in the middle of nowhere. The weighting system used meant that more configurations were chosen at the most difficult square in this path as well. This 10-arm robot was able to solve this problem in about 30 seconds.



The third screenshot demonstrates how leaving random sampling as an optional strategy works to help the robot in some cases. In this problem, the path chosen leads through a very cramped area, which the robot is unlikely to fit through. Thanks to the extra random samples however, the robot is able to realise a better path around the obstacle, which leads to a simpler solution. This takes this 10-arm robot only 5 seconds to calculate.

Through the testing I've done, I've found that my AI is able to readily solve up to 9-arm problems within a minute or so (barring the absurd), while 10 arms or more can depend on several factors. Hence, I'd say that my AI is moderately successful at fulfilling the problem statement.

References

Kavraki, L. E. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on, 12*(4), 566 – 580. doi:10.1109/70.508439