MSc Human and Biological Robotics

Author:
Edward McLaughlin

Lecturer:
Dr. Petar *Kormushev*

Robotics Tutorial 4: Motion Planning

Imperial College London

Department of Bioengineering

May 5, 2018

1 Question 1

Part A: Rapidly-exploring Random Tree (RRT) Motion Planning

Figure 1(a) shows the path made by the Rapidly-exploring Random Tree (RRT) motion planning algorithm with the parameters specified in the coursework handout. The nodes and path are shown in red, the obstacles in yellow. Each time the algorithm was run, the solution path which was found was different. This is due to the random nature of the motion planning, producing random movements into free space or towards the goal.

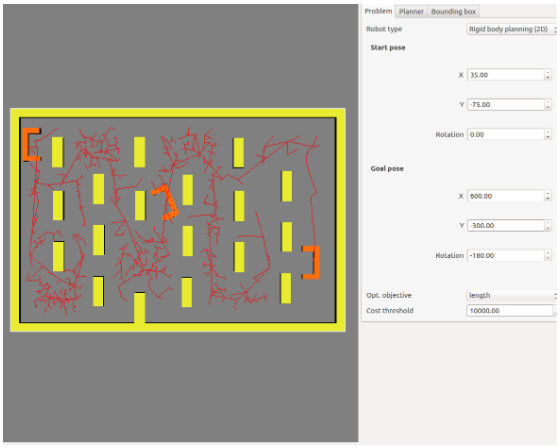
In this case, the solution is unidirectional, however, bi-directional RRT allows for the algorithm to find a solution quicker (and with less memory needed) as it must only explore half of the space due to the fact that, in 2D space, the area explored is proportional to the square of the distance between the start point and the end goal. For this problem, the dimensionality is of order 3 - position (x and y) and orientation (θ). Bi-directional RRT is especially useful in high dimensionality solutions, as when the dimensions increase the space explored is proportional to the distance between the start and the goal to the power of the number of dimensions. This effect is commonly known as the curse of dimensionality.

Figures 1(b) - 1(f) show the paths generated as a result of changing the motion planning parameters.

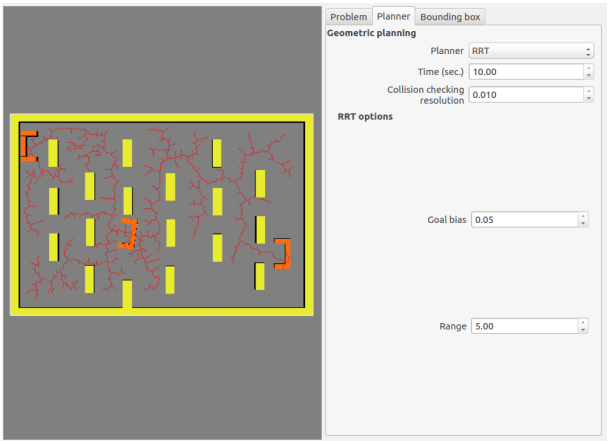
Figure 1(b) - 1(d) look at the effect of the 'range' parameter on the motion planning. In the primer document supplied in the tutorial, the range parameter is said to the maximum length added to the tree of motion [1]. This can be seen here in the fact that the path for a range of 50 contains longer straight movements planned in the tree compared with simulations with a range of 5. This being said, when the range was increased to a value of 150, the path lines do not increase significantly in length. This is due to the fact that as the lines get longer they are more likely to collide with the walls or objects in the map. Thus there is a limiting factor of the maximum range each random motion can take.

Figure 1(e) shows the motion planning when the goal bias was changed from 0.05 to 0.40. The goal bias is the propensity for the algorithm to move *towards* the goal, as oppose to a random direction into the open Voronoi space [1]. 'Towards' is highlighted here since if the algorithm made a movement all the way to the goal it could easy go through an obstacle, knowing only that the end goal is in free space. This can be seen in the figure by the fact that the Voronoi space is explored less than in figure 1(a), leading to a more direct and 'slim' path tree.

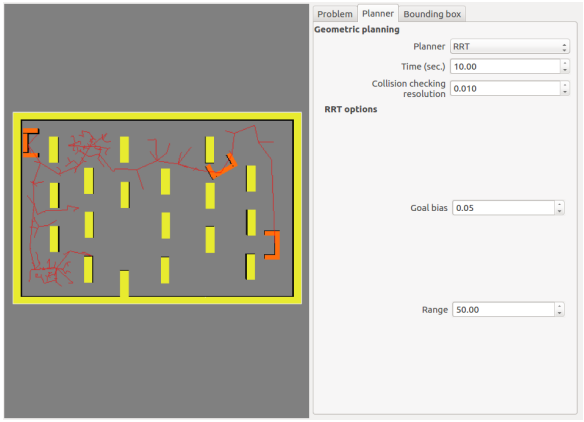
Figure 1(f) shows the motion planning for when collision check resolution is increased from 0.01 to 0.1. This results in the accuracy of the collision checking to be reduced - increasing the likelihood that the object collides with an obstacle as shown in the figure. Collision checking can be computationally expensive (even more so in the case of probabilistic models) so the correct level of collision checking is needed - if it's too resolute and the motion planning will take too long, if it's too relaxed the object will collide with obstacles.



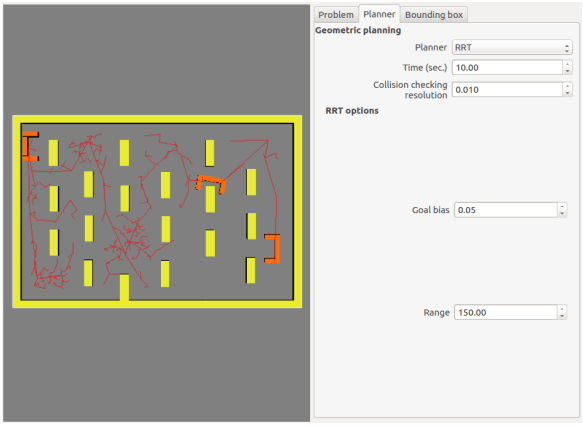
(a) Original Parameters



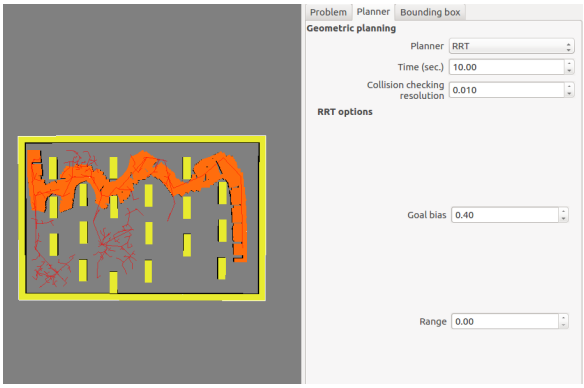
(b) Range = 5



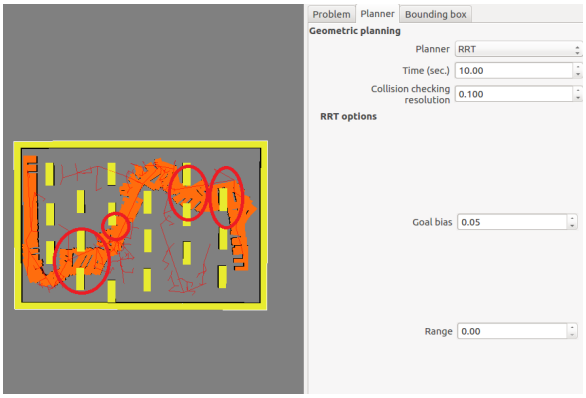
(c) Range = 50



(d) Range = 150



(e) Goal Bias = 0.40

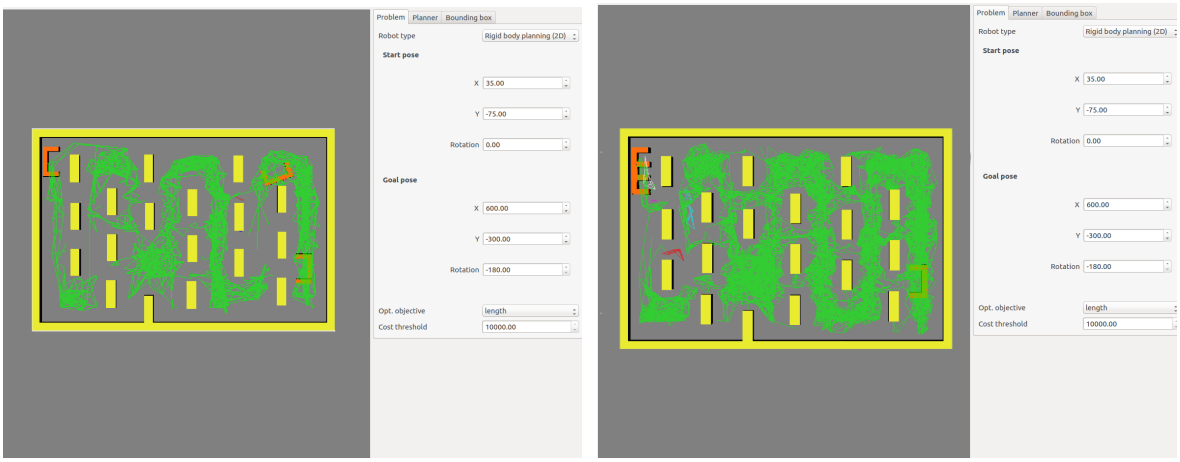


(f) Collision Checking Resolution = 0.1

Figure 1: Screenshots of RRT motion planning with various parameter configurations.

Part B: Probabilistic Roadmaps (PRM) Motion Planning

PRM motion planning finds a path in a probabilistic fashion. By sampling the entire space, points which are in free space are found - collision checking I. These nodes (or milestones) are then connected with the *nearest k* other milestones by vertices [1]. These vertices are subsequently check to see if they intercept any barriers (or any other areas outside of the free space) - collision checking II. This routine of double collision checking can be computationally expensive and thus the manner and resolution in which the space is sampled is important. These processes become even more complex if the object or robot is non-holonomic as the robots position and orientation must be considered for all collision checking. That being said, there is a benefit to this cost, once the milestones and vertices are found for a given map, they can be re-used for path finding on the same map. Another benefit of this algorithm is that, assuming the sampling is sufficiently extensive, if there is a solution, it will be found.
Figure 2(a) and figure 2(b) show the PRM motion planning path with the max. nearest neighbours parameter set to 8 and 30 respectively. It can be seen that by connecting each milestone with an increased number of its nearest neighbours results in a map more densely populated with possible paths. This is expected as the number of of vertices will increase significantly as the number of milestones increase.



(a) PRM with nearest neighbours = 8

(b) PRM with nearest neighbours = 30

Figure 2: PRM motion planning with varying nearest neighbour parameters.

Part C: Generic

As mentioned before, this is a 3-dimensional problem, considering x- and y- positions as well as orientation. Thus, each milestone or node must have all three of these parameters associated with it to ensure than the collision checking is accurate. Furthermore, if the path finding is being performed in C-space, the dimensions of the object relative to a given point on the object must be known. This given point on the object is crucial as the area the object occupies should always be defined relative to it. If the path finding is done in taskspace, the obstacles must be inflated appropriately to ensure the object doesn't collide with them.

The original RRT and PRM simulations were rigid body kinematic simulations i.e. considering position and space only. Since dynamic simulations consider not just physical properties of the system (space and orientation) but also dynamics (mass, acceleration), the algorithm can no longer assume infinite acceleration, and hence, any desired linear or angular velocity at a give time.

With regards to using either the RRT or PRM algorithm in a dynamic environment, the RRT is preferable. The PRM algorithm works by firstly by exploring the entire map (which in real life is not always known) and then computing a path to follow. This is not a very flexible solution and since obstacles can change position, it is not very adaptable in real time simulations. On the other hand, RRT

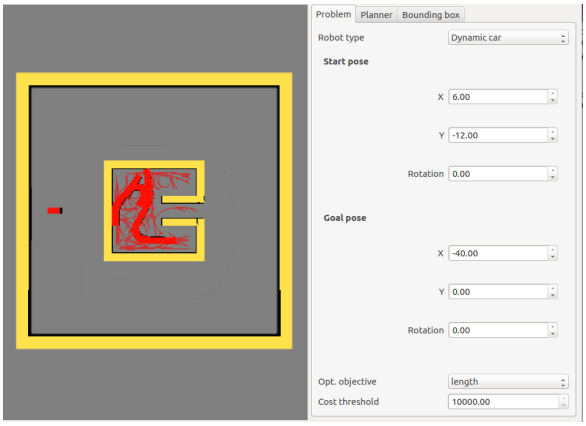
can continuously recalculate the path tree from its current position to the goal, regularly reassessing the obstacles in close vicinity. Finally, the RRT algorithm is better at dealing with non-holonomic problems as it can build in the limitations of the objects movements as constraints on the random movements.

2 Question 2: Kinodynamic Motion Planning

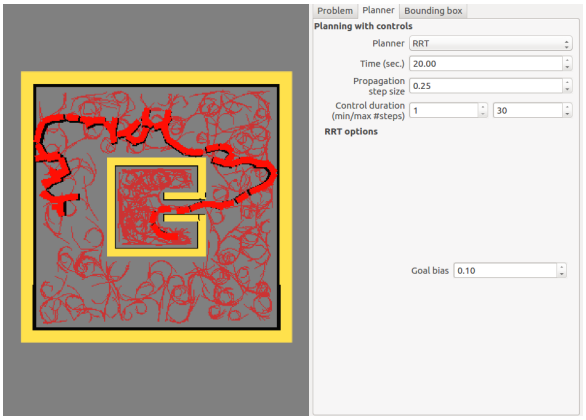
Part A: Tuning the parameters

Figure 3(a) shows the result for the kinodynamic problem with the parameters set to those given in the tutorial handout. With these parameters the car was unable to reach the goal.

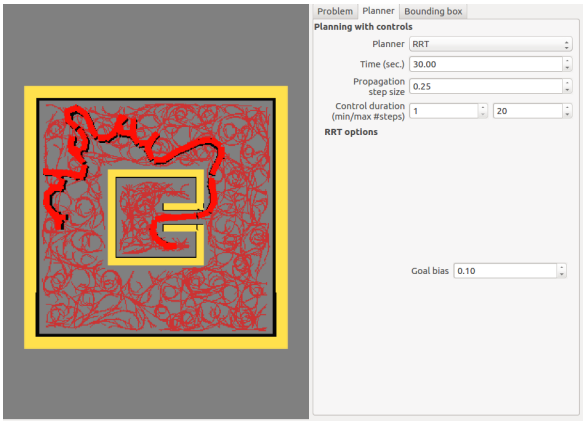
The parameters were changed in two ways to achieve successful motion planning to the goal (all other parameters were left unchanged from those in the tutorial handout): firstly, the number of steps allowed to take to reach the goal was increased to 30, secondly, the control duration was increased to 30 second allowing the car more time to find its way to the goal.



(a) Original parameters configuration



(b) Control duration: 1-30 steps



(c) Time = 30 seconds

Figure 3: Kinodynamic RRT initial parameter configuration and viable solutions.

Part B: Understanding

The planned paths for RRT in rigid body simulations compared to those in kinodynamic simulations in the fact that the 'branches' of the tree are much more jagged and discontinuous. In the kinodynamic simulation the path tree is smoother. This is due to the fact that the car must be able to perform the movements dictacted by the shape of the tree and so the limitations of the movement of the car must be considered (i.e. its turning circle). By considering a limitation on the car's turning circle, the algorithm is limiting the angular rotation to some value of rad/m. Thus, for a given linear velocity, the cars angular velocity, ω, (rad/s) is given by angular rotation, ϕ, (rad/m) $\times$ linear velocity, v, (m/s). The new angle of the car at time t is given by $\theta_{new} = \theta_{old} + v\phi t$. As previously mentioned, the kinodynamic simulation will also not allow the algorithm to assume instantaneous acceleration (angular or linear)

restricting the algorithms ability to give a desired velocity to the car at any given time. Hence, the car is unable to rotate on the spot and is unable to follow the paths given in the rigid body RRT.

References

[1] Open Motion Planning Library: A Primer. 2017.