Imperial College London

DE3-ROB1 ROBOTICS 1

Tutorial 4: Motion Planning using OMPL

Marcin Laskowski

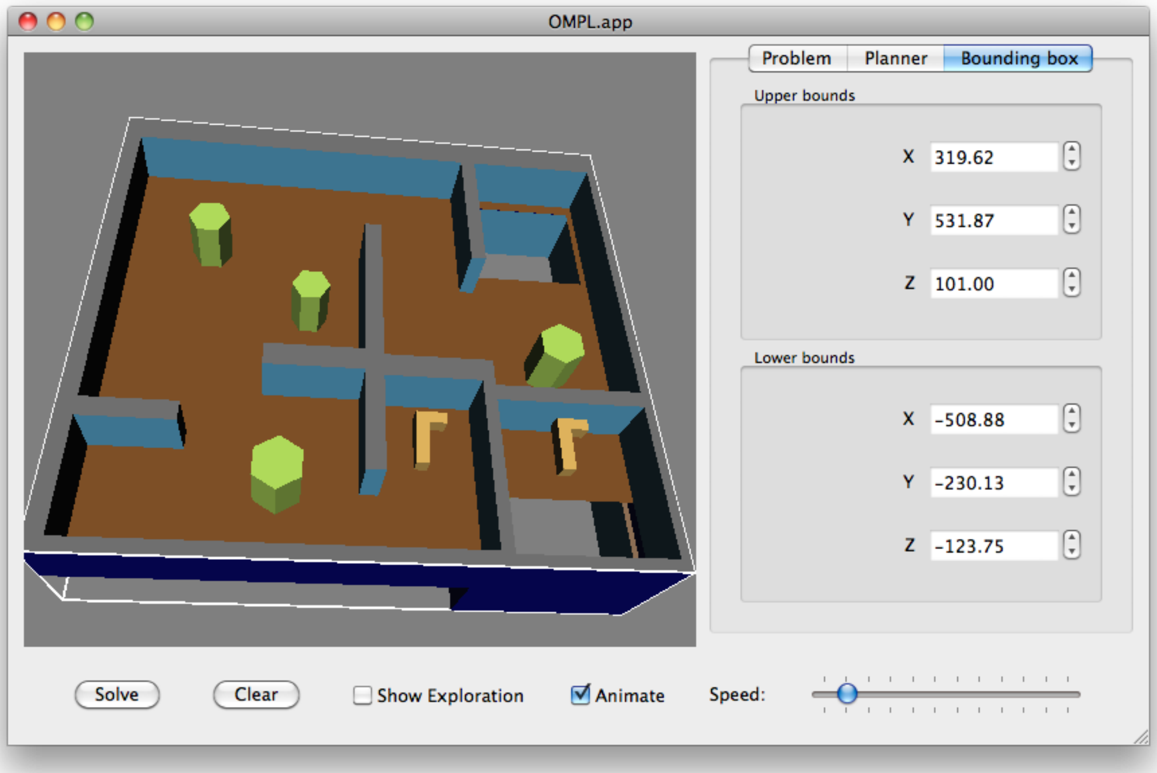
supervised by
Dr Petar Kormushev

December 8, 2017

1 Introduction

The goal of the following tutorial was to familiarize with the two sampling-based motion planning algorithms:
a) Rapidly-exploring Random Trees (RRT)
b) Probabilistic Roadmaps (PRM)

To perform the following task special software package for motion planning in robotics called Open Motion Planning Library (OMPL) was used.



Upper bounds

X 319.62

Y 531.87

Z 101.00

Lower bounds

X -508.88

Y -230.13

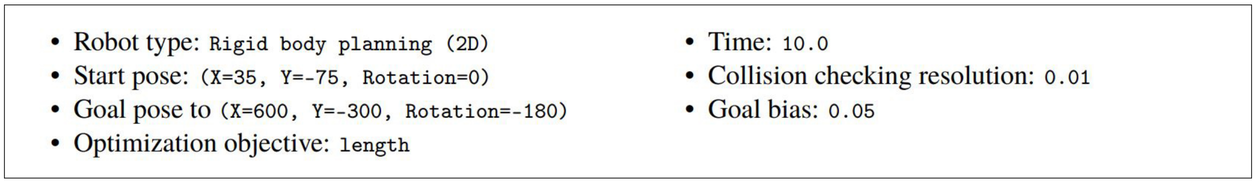
Z -123.75

☐ Show Exploration ☑ Animate

Figure 1: Screenshot presenting OMPL library (`http://ompl.kavrakilab.org/images/gui_bbox.png`)

2 QUESTION 1

In the first task it was given initial parameters (Fig.2) which needed to be apply for the `Barriers.cfg` file



- Robot type: `Rigid body planning (2D)`
- Start pose: `(X=35, Y=-75, Rotation=0)`
- Goal pose to `(X=600, Y=-300, Rotation=-180)`
- Optimization objective: `length`
- Time: `10.0`
- Collision checking resolution: `0.01`
- Goal bias: `0.05`

Figure 2: OMPL parameteres for the first task

2.1 Planning with RRT

In this task Rapidly-exploring Random Trees (RRT) has been investigated. General idea of the algorithm is as follows. The trees generated by these methods are analogous to the probabilistic roadmap. These method starts with rooting a tree at the initial configuration of the robot. First node of the tree intact and after that occurs random sampling of the free space. The planner employs an heuristic expansion from which the sample is connected to the tree along a collision free path.

2.1.1 Screenshots of the resulting exploration tree

Below in the Fig.3 it is presented the roadmap created using RRT alogrithm.

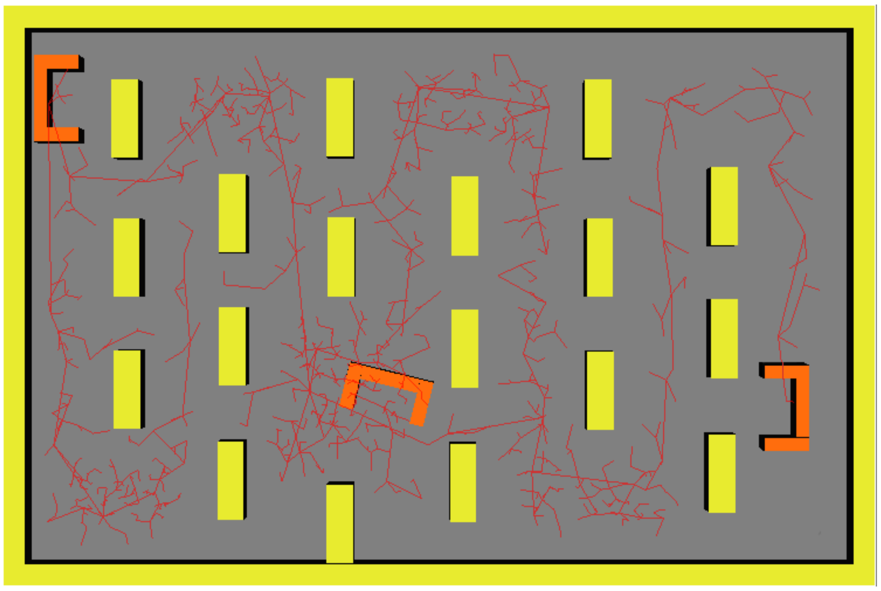


Figure 3: Screenshot of the RRT with above initial parameters

2.1.2 Several solution for the same configuration of the parameters

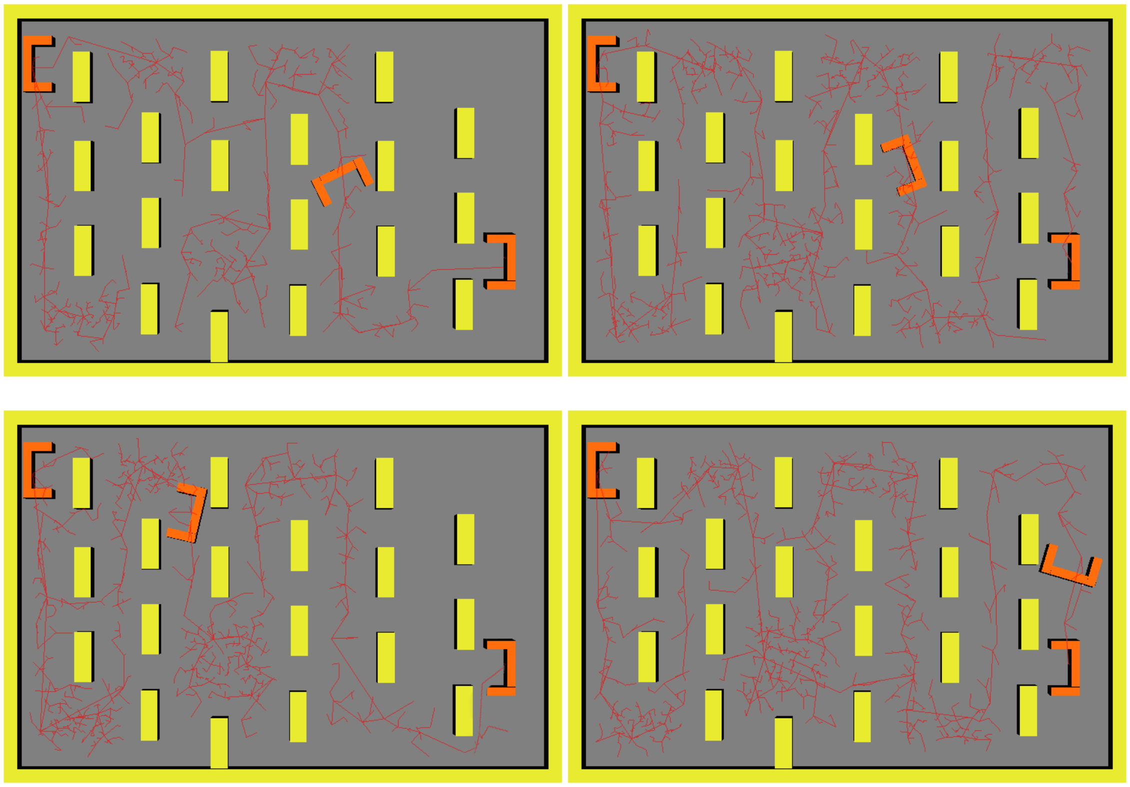


Figure 4: Screenshots of the RRT performed sever times

It can be noticed in the Fig.4 that configuration of the RRT on the every picture is different. That is due to the fact that RRT is a method that uses probabilistic roadmap. Each running of the code produce different (random) configuration of the tree.

2.1.3 Several solutions for different values of the parameters

For the following task several parameters have been investigated. First one is `Goal bias`.

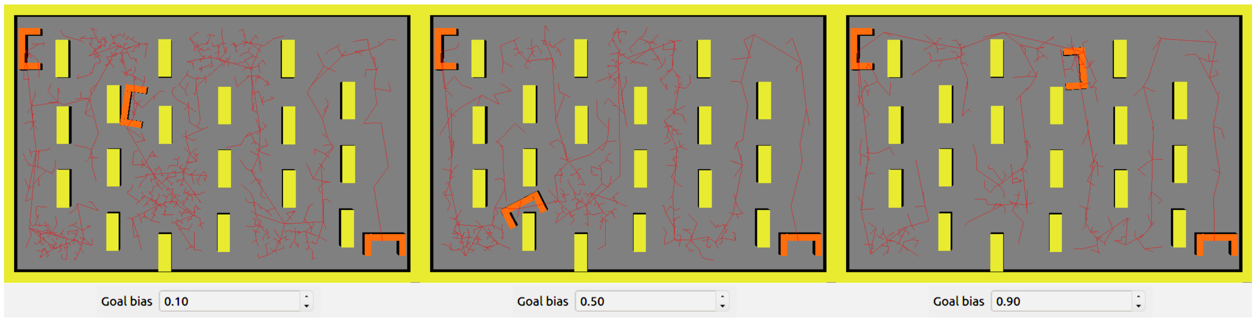
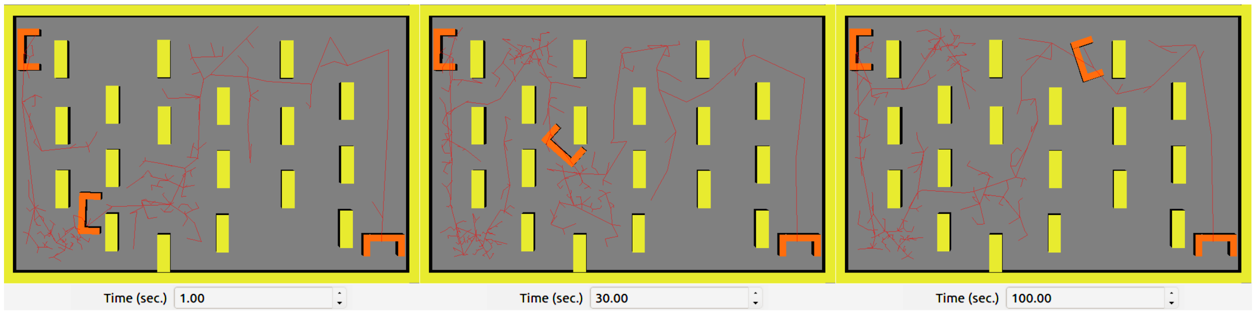


Figure 5: Plot of the PRM with different `Goal bias` values

As it can be noticed in the Fig.5 the smaller the goal bias the more complex the structure of the tree. In the process of randomly selecting states in the state space to attempt to go towards, the algorithm may in fact choose the actual goal state with some probability. This probability is a real number between 0.0 and 1.0; it should not be too large, and usually lies around 0.05.

Next invastigated parameter was `time`.


Time (sec.) 1.00

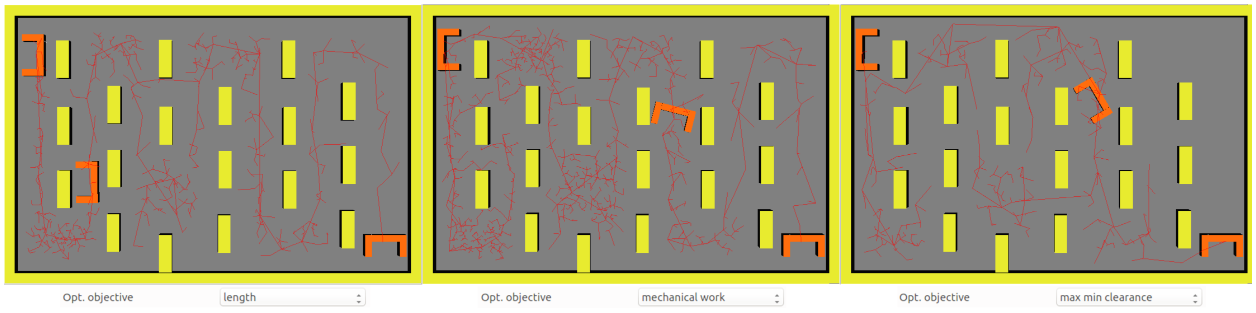
Time (sec.) 30.00

Time (sec.) 100.00

Figure 6: Plot of the PRM with different `time` values

In Fig.6 can be noticed the influence of the time on the structure of the tree. Structures of the tree are quite similar.

Last parameter that was observed was `Opt.objective` which is Optimization Objective.


Opt. objective length

Opt. objective mechanical work

Opt. objective max min clearance

Figure 7: Plot of the PRM with different `Opt.objective` values

Certain motion planners can meet optimization objectives as well. These objectives can implement different cost functions. In this case there was compared 'length', 'mechanical work' and 'max min clearance'. Second produced the most complex tree while the last one creates very simple structure.

2.2 Planning with PRM

A probabilistic roadmap (PRM) is a network graph of possible paths in a given map which is based on free and occupied spaces. General idea of this approach it to create the connection between the nodes which were randomly generated. That connections are based on the locations of the obstacles in the environment. PRM algorithm uses network of connected nodes to find and obstacle free path from a start to an end location.

In the following two tasks there is presented PRM algorithm in which maximum nearest neighbour parameter is investigated.

2.2.1 Max nearest neighbors = 8

Below in the Fig.8 is presented picture of the PRM motion planning with the Maximum nearest neighbour parameters equal to 8.

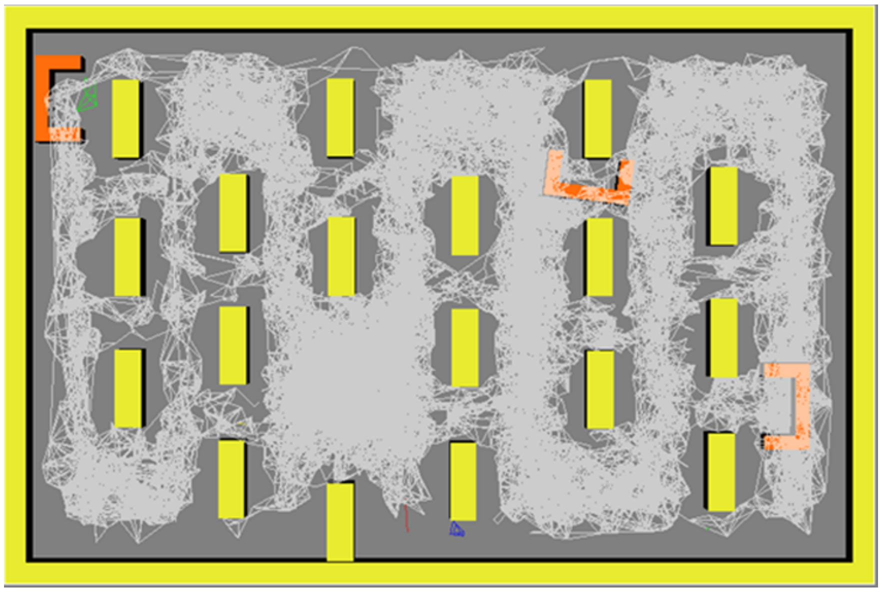


Figure 8: Plot of the PRM with parameter `Max nearest neighbors = 8`

2.2.2 Max nearest neighbors = 30

Fig.9 presents picture of the PRM motion planning with the Maximum nearest neighbour parameters equal to 30.

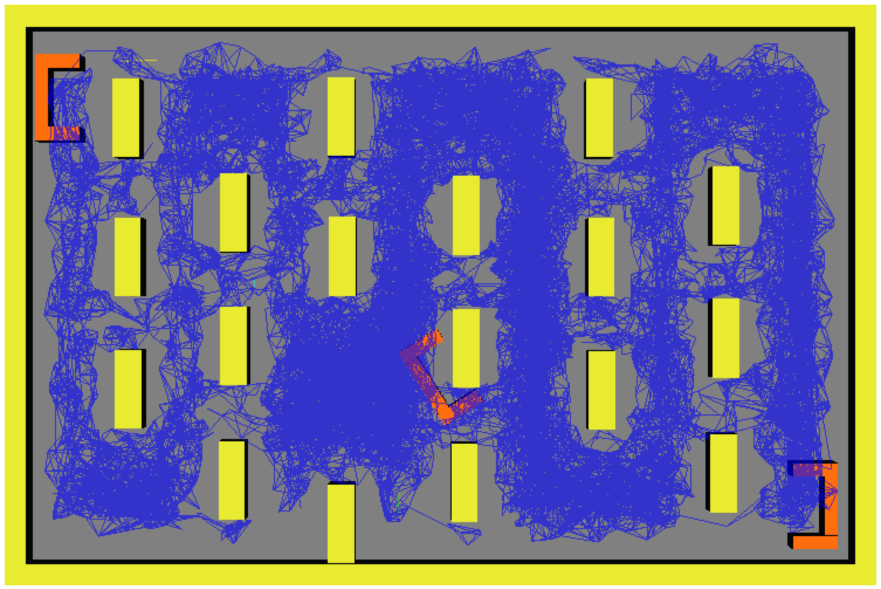


Figure 9: Plot of the PRM with parameter `Max nearest neighbors = 30`

It can be noticed that first picture (Fig.8) contains slightly less condensed structure of the connections. Increasing number of nodes can increase the efficiency of the path by giving more feasible paths. However increasing number of nodes means also greater complexity which leads to greater computation time needed to calculate the path. For the good coverage of the map there is need for large amount of nodes. Due to random placement of the nodes some areas of the map are not be connected.

2.3 Generic

2.3.1 Generated state

Each generated state (such as node or milestone) for the motion plan to result feasible collision-free trajectory in both above methods should contain the current position and orientation of the robot in space. Additionally in PRM there should also be information of the position of the node and its nearest neighbours. What is also important is the are of the obstacle (The reason for this is that during computation algorithm check if the node is in the obstacle, if yes than there is no connection, if no nodes are connected. After that there is also checked connection whether the connection itself is in the obstacle or touch obstacle. If yes, than such a connection is removed. Finally there are only connections which do not goes through the obstacles.)

2.3.2 Above algorithms in dynamic environment

Original PRM algorithm is not applicable to dynamic environment which require real-time motion planning because the algorithm of that method works in a way that scans the environment once (randomly distribute points in the map) and than on that bases connects the points which are not in the obstacle and looks for the most optimum path. Every change of the position of the object in the environment could provide a collision.
When it comes to RRT it is important to notice that this algorithm is viewed as a technique to generate open-loop trajectories for nonlinear systems with state constraints. This means that this particular algorithm could not work in the dynamic environment. For the dynamic environment that algorithm need to be modified. First and foremost the length of the branch need to be short because long branches are insufficient. The algorithm also need to have a maximum number of nearest neighbors, or neighborhood size that would be configurable to the algorithm. Example of improved RRT algorithm is RRT^* or RRT^X. It allows the tree branches to go with the agent without discarding previously sampled paths due to good performance in the real time planning of the path in the dynamic environment.

3 QUESTION 2

In the second problem environment to `BugTrap_dcar.cfg` have been changed and the following tasks as been performed using parameters presented below.

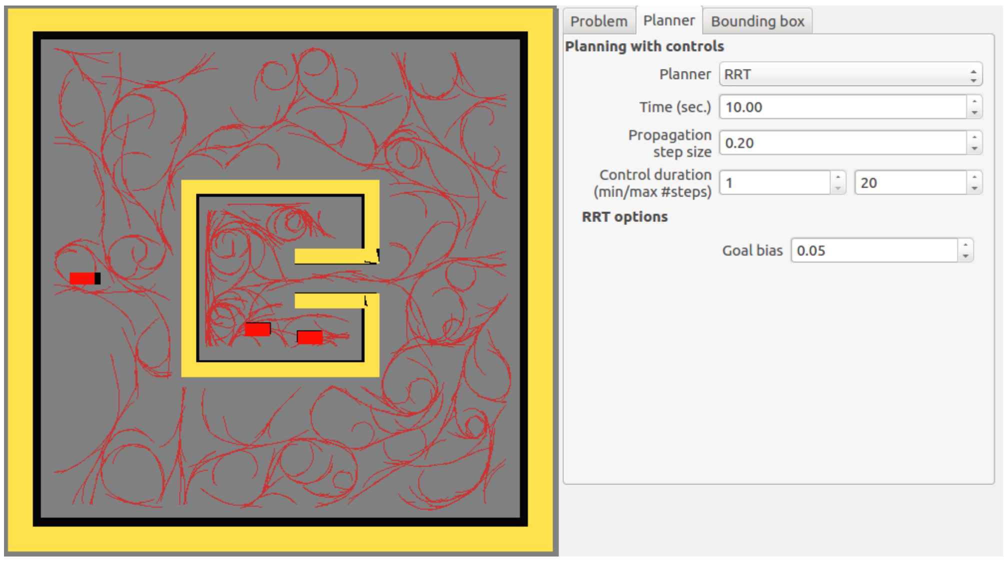
- Robot type: `Dynamic car`
- Start pose: `(X=6, Y=-12, Rotation=0)`
- Goal pose to `(X=-40, Y=0, Rotation=0)`
- Optimization objective: `length`
- Time: `10.0`
- Propagation step size: `0.20`
- Control duration: `0-20`
- Goal bias: `0.05`

Figure 10: OMPL parameteres for the second task

In the following tasks it was investigated The kinodynamic motion planning. The general idea of this kind of motion planning is to solve a robot motion problem subject to kinematic and dynamics constraints calculated in the real time. The goal of that algorithm is to find a minimal-time trajectory that goes from a start position and veloclty to the final (target) position and velocity. During motion robot avoid obstacles by a safety margin and respecting constraints on velocity and acceleration. Such a solution is for example used in autonomous cars where vehicle represents non-holonomic system.

3.1 Tuning the parameters

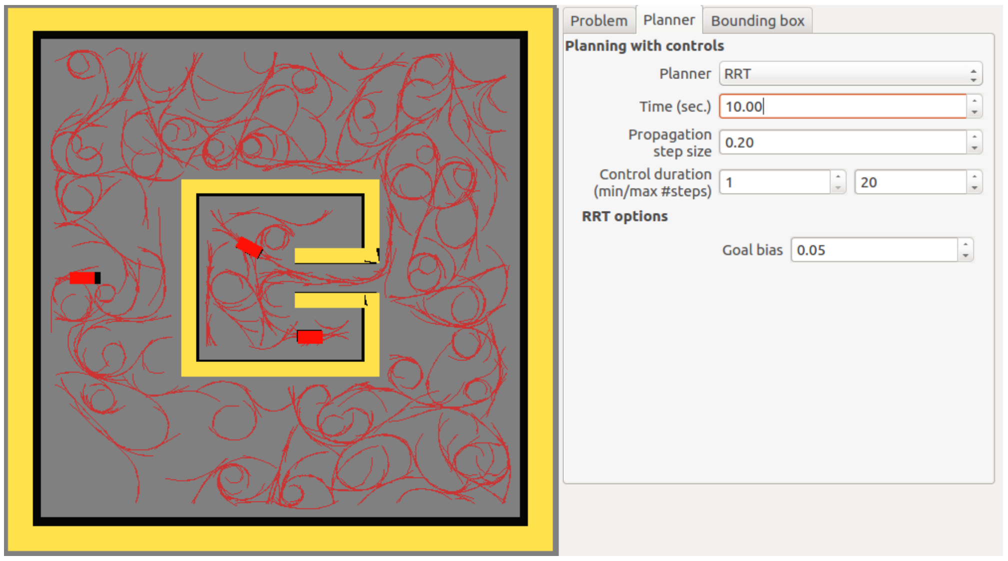
Below in the Fig.12 it can be noticed the screenshot from the successful plan with provided parameters.



Planning with controls

Planner: RRT
Time (sec.): 10.00
Propagation step size: 0.20
Control duration (min/max #steps): 1 20
RRT options
Goal bias: 0.05

Figure 11: Screenshot of the successful plan with parameters



Planning with controls

Planner RRT

Time (sec.) 10.00

Propagation step size 0.20

Control duration (min/max #steps) 1 20

RRT options

Goal bias 0.05

Figure 12: Screenshot of the successful plan with parameters

3.2 Understanding

In the first question robot was treated as the point mass, while dimension of the robot was taken into the consideration in the transformed environment which is called C-space. That is why RRT was able to spread in all directions equally to find the goal position.
However in the second question there was taken into the consideration kinematics and dynamics of the robot which meant that the robot was not treated anymore as a point mass but in this case as a object with mass and dimension. Transformation of the environment to c-space in this case was not needed.

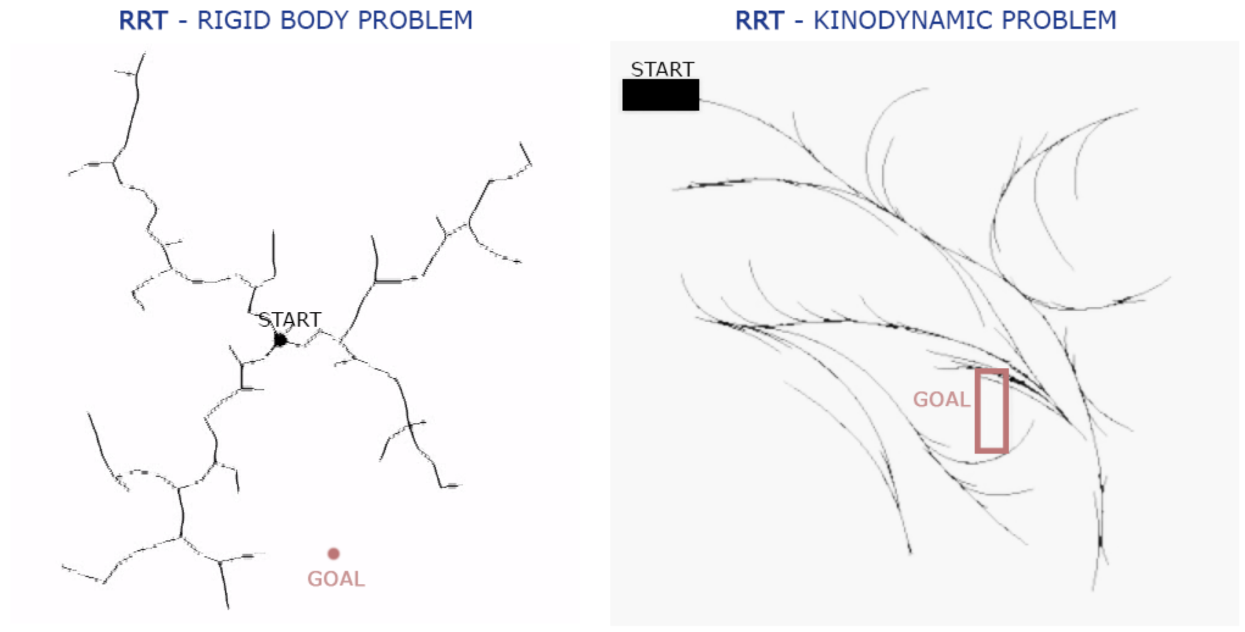


Figure 13: Comparison of the two cases of RRT

Looking at the Fig.13 it can be easily noticed that the path in the second case (Kinodynamic problem) is more smooth, there is no sharp edges. Roots are not spread in all direction but only in the direction of motion which is in some case obvious (e.g. car is a non-holonomic system, it cannot move sideways.)