Rapidly Exploring Random Tree Planner: solving puzzle

Jakub Tomasek

Contents

I. Problem

The task was to find a path which would solve the puzzle using given code for RRT.

II. Rapidly Exploring Random Tree Planner

Rapidly Exploring Random Tree (RRT) is a randomized method method of planning to explore complex configuration space [1]. RRT was introduced in [2]. The basis for this method is to incrementally construct a search tree that rapidly and uniformly explore the state space.

Essentially, we bias the expansion of the tree towards unexplored configuration space. According to [1], the surrounding space pulls the tree towards it. In each step new point is generated. Function extend finds the closest existing vertex; the tree is expanded towards the generated point. This is how the tree is pulled towards the free space.

Specifically, in this solution we have 11 checkpoints on the path (array *FF_CS*). The planner must find path between these checkpoints. The random tree is saved into the structure *Tree_S*. RRT generates 200 random points uniformly distributed in between initial and goal location (function *Random_CS*); it is allowed to return a bit in each dimension. Function *expand* is same as in [1]; the function finds the nearest vertex and if possible (there is no obstacle) it expands towards that direction a bit (1/150 of the distance between these two).

The RRT planner must include a collision detector to determine whether given configuration satisfies the constrains; here it is solved by Minkowski sum.

III. Solutions

A. Random number generation

We were supposed to determine what random numbers are generated in the random number generation. I used uniform distribution from 0 to 1. This is example:

```
C_CS(i,:)=(FF_CS(i,ind)+SGN*15)+((FF_CS(i
  ,ind-1)-SGN*15)-(FF_CS(i,ind)+SGN*15))*
  rand(1,R_N);
```

B. Expand function

I rewrote the expand function so it didn't contain any for loops which are not optimal in Matlab:

```
function [Tree_S]=Extend(Q_rand,Tree_S)
    DS=sqrt(sum(abs(Tree_S.V(:,1:Tree_S.
      T_S)-Q_rand*ones(1,Tree_S.T_S))
      .^2,1));
    [NDS,IofT]=min(DS(1:Tree_S.T_S));
    Q_near=Tree_S.V(:,IofT);

    Q_new=Q_near+ (Q_rand-Q_near)/150;
    if(CollisionCheck(Q_new,ind)==0)
        Tree_S.T_S=Tree_S.T_S+1;
        Tree_S.V(:,Tree_S.T_S)=Q_new;
        Tree_S.E(:,Tree_S.T_S)=IofT;
    end
end
```

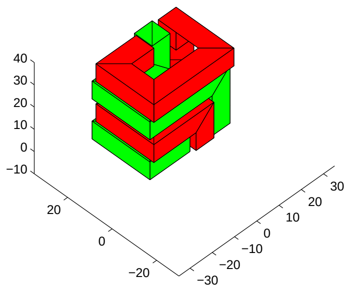
C. *Path*

I ran the code several times. There is most likely a bug in the code. When we switch to segment 7, the piece of the puzzle takes a shortcut right outside. It happened to me and to Mikkel independently. This is not likely not related to any edits we made in the code. Since I suppose it was originally working I assume the problem is related to the version of Matlab: we both use Matlab 2013b, 64bit.

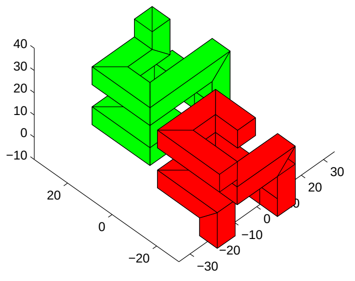
I suspect that the problem might be in the piece of code, which adds new tree point into the tree, when the coordinate system is changed. Particularly, function *find* probably doesn't work in the way the author intended.

Therefore, I had to run the path planning in two segments. When the error occurred, I ran the code again deleting last nodes and edges from the tree. I myself set the variables to values which are supposed to be set when it transforms from segment 6 to 7. At the end, I picked variable *RS* manually from the values of the tree.

In total, it took around 15 hours. The size of the whole tree was 13601 nodes. Please, see the video attached with the report: http://youtu.be/gXg34cUEC28. Figure III.1 shows initial and final stage of the puzzle.



(a) Initial stage



(b) Final stage

Figure III.1: Initial and final stage of the puzzle

References

[1] S. M. LaValle, J. J. Kuffner, and Jr, *Rapidly-Exploring Random Trees: Progress and Prospects*, 2000. 1
[2] S. M. LaValle, "Rapidly-exploring random trees a ĐŠew tool for path planning," 1998. [Online]. Available: http://coitweb.uncc.edu/~xiao/itcs6151-8151/RRT.pdf 1