ODTÜ
METU

06.06.2018

CENG 786: Robot Motion Planning & Control Term Project

Term Project

Implementation of a Multi-Robot Path Planner

"Finding a needle in the haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning"

Volkan OKBAY

1741479

Ins. : Assoc. Prof. Uluç SARANLI

TABLE of CONTENTS

INTRODUCTION & OVERVIEW

In this report, realization and experiment details will be presented regarding term project for our course. The project was implemented on the article “Finding a needle in the haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning” [1]. The simulation assumptions, limitations, difficulties and bonus work are discussed below.

The method proposed by the authors is a new approach to multi-robot case of motion planning in both 2D and 3D environments. But due to time concern, only 2D cases are built in the project.

MATLAB platform is used to code and simulate results. The sample maps and hardware are discussed in the experiment section.

Multi-robot motion planning has been a popular subject in the recent years. As sometimes it is more profitable to perform a job with a number of simplier robots connecting each other, instead of a single complex one.

In the study, the objective is to find a path solution for a several robots moving from a starting point to a target point, each having their own. To do so, they focused on the “completeness” rather than “optimization”. Furthermore, this is a probabilistic algorithm involving randomness, we can talk about “probabilistic completeness”, which guarantees to find a solution in a finite time amount, if at least one exists.

It is claimed to solve a multi-robot problem like that at least 10 times faster than its predecessor algorithms of probabilistic base. This claim supports novelty of the method.

ALGORITHM

In this part, a short description of the methodology will be given. Firstly, introducing discrete RRT concept is a good start. A definition called "direction oracle" should be clear before understanding dRRT.

Direction Oracle: Consider a G = (V,E) graph consist of vertices v_i . Let there is a random point *u* in the graphical space and *v'* are neighbouring vertices of *v*. And *$L_v(x,y)$* denotes the smaller angle between v-x and v-y lines. So oracle is found as below:

Definition *Given a vertex $v \in V$, and a point $u \in [0,1]^d$ we define*

$$\mathcal{O}_D(v,u) := \underset{v'}{\operatorname{argmin}} \left\{ \angle_v(u,v') \mid (v,v') \in E \right\}$$

By definition, formula returns a neighboring *v'* vertex that gives the smallest angle between v-v' and v-u. As a result, *v'* is the adjacent vertex in the direction of random point *u*.

dRRT: Consider a single robot given, and a PRM graph calculated with a pre-determined number vertices. Implicit graph of PRM includes s and t vertices. The algorithm initiates a tree *T* with a single vertex (s: starting vertex). In the loop, tree tries to expand (see next paragraph). Then, a check is done if current can be locally connected to the t (target vertex). When this connection is established the path is retrieved and algorithm terminates; otherwise, continue from line 3 (algorithm 1)

While expanding (algorithm 2), for N times following is done. A random sample in the Eucledian space is generated. Then, following direction oracle explained above, a neighboring vertex is found in the direction of random point. If this calculated node is a new one, tree is expanded.

Algorithm 1 dRRT_PLANNER (s,t)

```
1: $\mathcal{T}$.init($s$)
2: loop
3:     EXPAND($\mathcal{T}$)
4:     $\Pi \leftarrow$ CONNECT_TO_TARGET($\mathcal{T}, t$)
5:     if not_empty($\Pi$) then
6:         return RETRIEVE_PATH($\mathcal{T}, \Pi$)
```

Algorithm 2 EXPAND $(\mathcal{T})$

```
1: for $i = 1 \rightarrow N$ do
2:     $q_{rand} \leftarrow$ RANDOM_SAMPLE()
3:     $q_{near} \leftarrow$ NEAREST_NEIGHBOR($\mathcal{T}, q_{rand}$)
4:     $q_{new} \leftarrow \mathcal{O}_D(q_{near}, q_{rand})$
5:     if $q_{new} \notin \mathcal{T}$ then
6:         $\mathcal{T}$.add_vertex($q_{new}$)
7:         $\mathcal{T}$.add_edge($q_{near}, q_{new}$)
```

Composite Roadmaps: This roadmaps are the representation of states (configuration spaces) for the multi-robot case. They are obtained by Cartesian/Tensor products of PRMs of individual robots.

In the project, like the authors, Tensor Product will be taken into account. Shortly, every configuration change is allowed as long as there is no collision during robots move from one configuration to another.

MRdRRT: Combination of the previous knowledge gives rise to MRdRRT concept. This is simply applying dRRT algorithm on the composite map, instead of a PRM of a single robot.

Consequently, pathfinding is done from starting position to target positions of each robot without collision.

IMPLEMENTATION

Grasping the idea behind and going through clues in the article, we started to code a simulation environment in MATLAB. Below given the system properties and software features:

- Intel Core i7 2.50GHz CPU
- 16 GB 2333 MHz RAM
- 8GB GPU (not critical)
- Windows 10 OS

Software: MATLAB 2016a Student Version with Robotic System Toolbox v1.2 installed. Parallel processing was not used.

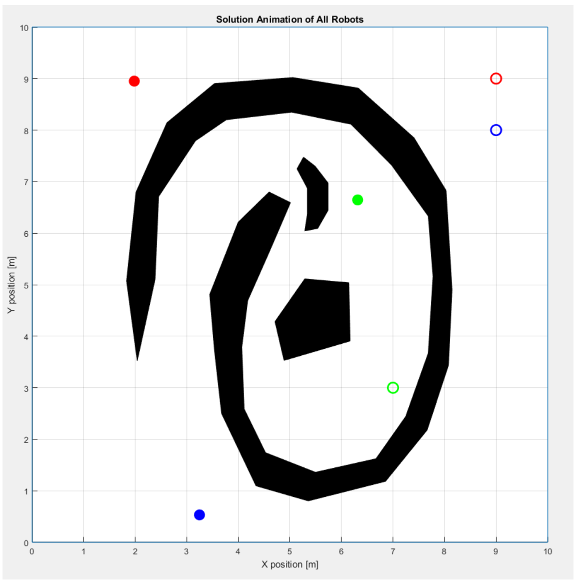
Above mentioned detail are given due to give an idea about timing discussions in the next paragraphs.

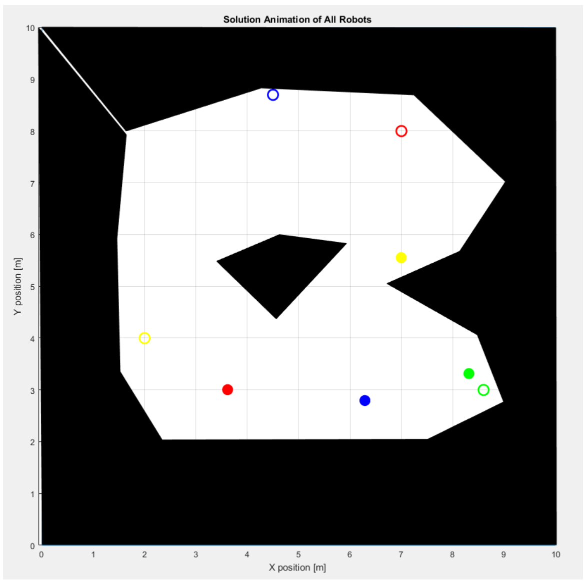
The simulation may be run in the MATLAB workspace by calling the main function with two parameters; namely, “case number” and “number of vertices”.

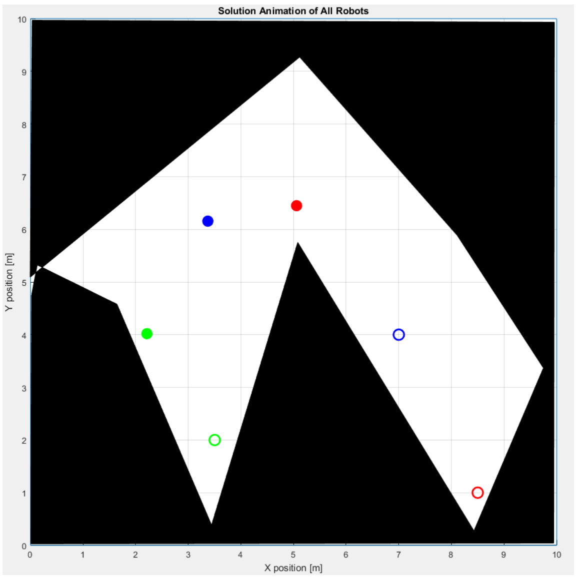
```
main_project_ceng786(case_number, number_of_vertices);
```

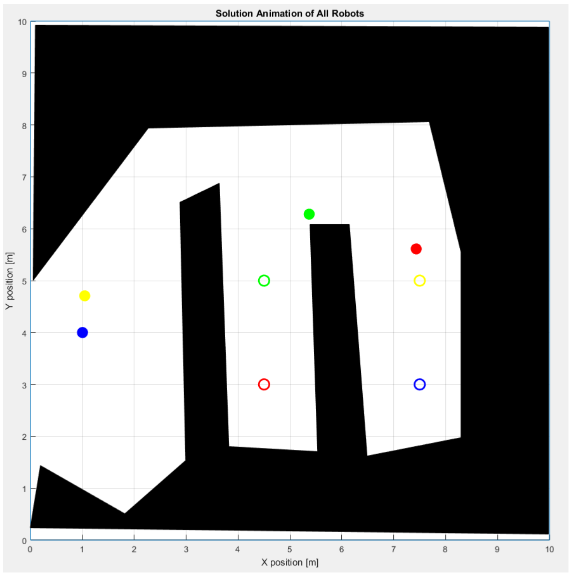
Case number determines the arena map, obstacle type and positions; as well as, number of robots and start/target points for each. There are 8 cases integrated in the coding, some of which are originated from the article (see Figures 1-8).

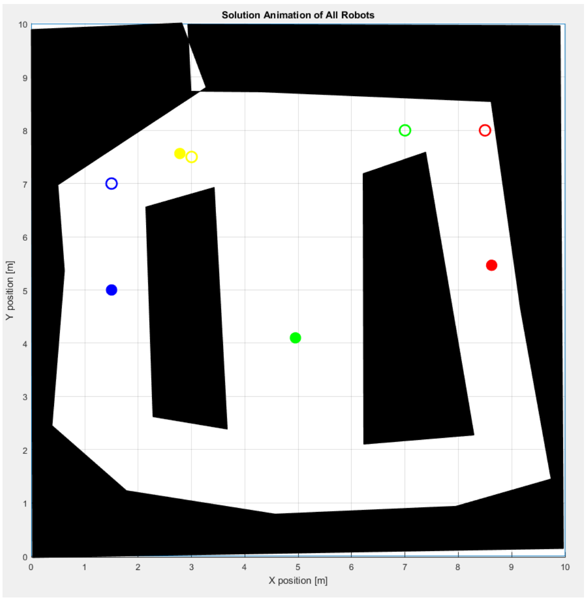
Second parameter, on the other hand, determines the number of vertices for individual PRM graphs. Authors have chosen 300 for 2D case, and some number in the order of thousands for 3D cases. Our implementation suggests around 5 to 10 vertices for this. It is because they have written in C++ which is much faster and less memory consuming and what we like to show is the main idea of the algorithm in a short time. The timings are given in the Table 1. It can be inferred that composite graph generation is the time-consuming, computationally complex part. It has the complexity of O(v^r), where v is number of vertices parameter and r is the robot number. As a result, it is a good idea to keep vertex amount parameter less than 10, especially highly-coupled 4 robot cases presented.

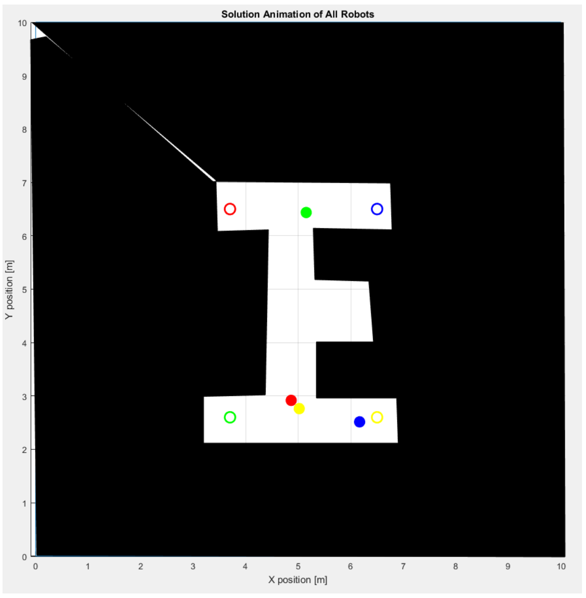


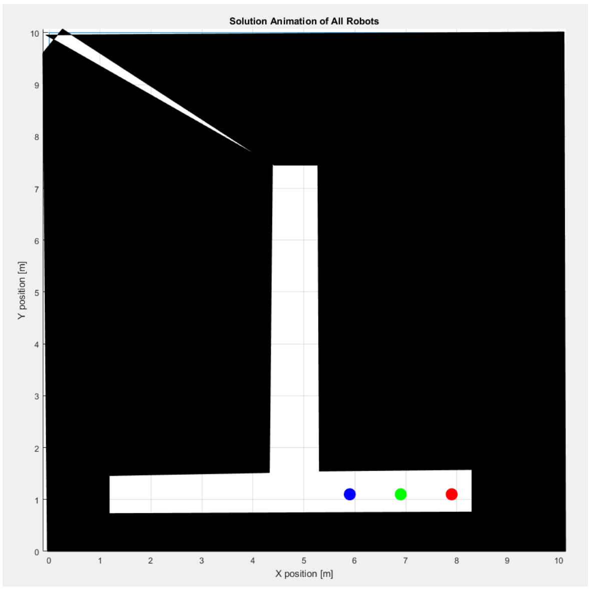


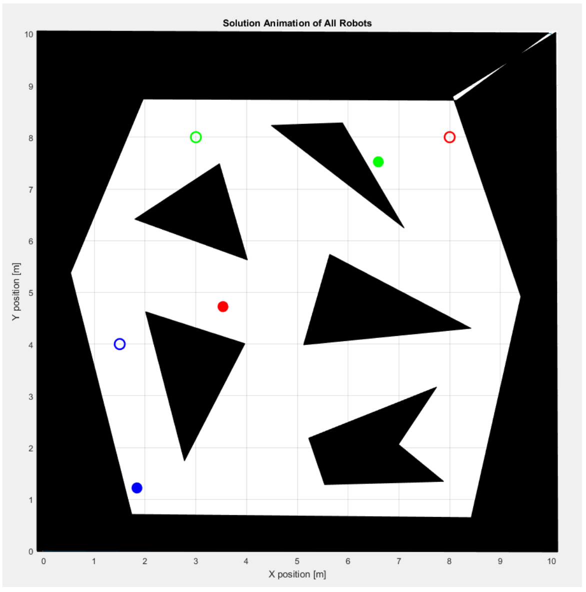












Figures 1-8: Test arenas with circular robots

In the figures above, all arena samples integrated in the code are given. Some are 3, others are 4 robot problems (yellow robot is sometimes hard to spot). Black area represents obstacles and walls, while hollow circles indicates target points. The images are taken from random points during resulting animation. Running main function will also plot individual PRMs for each robot seperately. User may also check timing for different process sections from the workspace.

Below given timings on average of 10 runs for each case shown above in the figures. Please, note that vertex parameter is 6 for all.

All in seconds	**PRM Generation**	**Composite Tree Generation**	**dRRT Path Finding**	**Total Time**
Case 1	1.05	1.87	0.12	3.04
Case 2	2.64	77.27	3.32	83.23
Case 3	2.10	2.77	0.02	4.89
Case 4	2.63	105.05	8.49	116.17
Case 5	2.32	78.18	3.22	83.72
Case 6	3.90	117.62	22.05	143.57
Case 7	2.91	2.93	0.01	5.85
Case 8	1.87	3.01	0.23	5.11

Table 1: Average timings for each case

An obvious conclusion is that composite tree generation is sometimes very expensive in time. It was even longer, then a recursion implementation is done which reduced time consumption around 1/5th as before. Another change is skipping candidates of new nodes of implicit tree during expansion, which has no neighbor (in other words, they are not expandable).

Some dRRT steps seems to remarkably short. That is probably because we lack of some robot-robot collision checks (discussed in the next section), and those cases become simplier. Those times may change for different systems, but proportions should give some idea.

The first step, namely Probabilistic RoadMap generation, was not clearly defined in the article. To be more precise, the exact method (visibility graphs, sweeping etc.) was not mentioned. So, the original PRM algorithm by Kavraki et. al. (1999) is used. Here, robotics toolbox was helpful. This part is deciced to be “out-of-scope” of the project, then it seemed to be a fair choice using custom methods of MATLAB. At this point, it should mentioned that original PRM class of the toolbox is alternated in small details. Some of the required properties were originally private.

It can be inferred that cases (such as 1,3 and 7) without an obstacle in the middle (only walls) consumes a little time for dRRT step. And other highly-coupled cases like 2,4 and 6 needs a lot more (app. 50 times) seconds to acquire a composite tree.

NOTE: A final note on simulation, PRM block tries to generate a roadmap and checks if it is valid (a path exist for individual robot from start to target). If no path is found, it continues to try for 255 times. Having still no path, an error is prompted. In this case, user may run main function again or increment the number of vertices parameter.

DISCUSSION & FUTURE WORK

All in all, it may be a successful replica of the project for 2D case. Although, 3D case is graphically different, it is not different in methodology, but only more computationally complex. Only important missing part is robot-robot collision checks. We assumed robots to be points regarding robot-robot collisions, but circles with 0.1 meter radii for robot-obstacle collisions. So, first future work should include below:

- Robot-robot collision check for implicit tree node, collision check of PRM nodes of distinct robots
- Robot-robot collision check while local connector trials (this only affect the last movement of the path)
- Robot-robot collision check while moving one configuration to another

In fact, the last one is the most important and general. Hence, there we have a simple check for robot paths. Assuming point robots, the sub paths at every step are examined if they intersect or not. To fulfill real world realization, that check should be a area intersect check instead of linear. In order to do so, a simple determinant operation is used. The reason that we omitted such simple mathematical operations (robot-robot collision in general) that they are run during composite tree generation. As mentioned before, that is the costly part of computation. So, to comfort that step collision checks are simplified. Consequently, that situation does not influences algorithmic performance.

A second development should be transferring codes into a lower level software language. This is obvious that it will speed up the process and facilitate simulation of the 3D cases, which may be another future work.

A clever idea is that applying a bi-directional growing RRT tree in the main dRRT flow. This is also mentioned by the authors (actually they were able to solve one of the 3D problem by only bi-directional developed tree). This, again, will speed up the process definitely and able to solve some extra problems.

In conclusion, the method is proved [1] to be probabilistically complete and seems to be a practical solution in the case PRMs and composition of all PRMs are ready. Conversely, considering total time consumption, it is better to use this algorithm in highly-coupled cases. As the authors agree with that their method is not so efficient in simple cases, usually common tree search algorithms outsource dRRT. When robots share paths (more intersections, shadows each others target points) and number of robots goes up dRRT become profitable. A delicate selection of node number and robot number must be considered, so that user will not encounter with out-of-memory or excessive time consumption situations. This novel approach of application of discrete RRT on an implicit graph of PRMs may advance in the next studies.

Please, see the original article highlighted in .zip-file, in-line comments in MATLAB coding and similar competitor studies [3] for further information.

REFERENCES

[1] Solovey K., Salzman O. and Halperin D. (2016). ”Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning”. The International Journal of Robotics Research (2016) Vol. 35(5) 501–513

[2] Kavraki, L.E., P. Svestka, J.-C. Latombe, and M.H. Overmars. "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," IEEE Transactions on Robotics and Automation. Vol. 12, No. 4, Aug 1996 pp. 566—580.

[3] Wagner G and Choset H (2015) Subdimensional expansion for multirobot path planning. Artificial Intelligence 219: 1–24.