CENG 786

Efficient Spark PRM

Mert Kaan Yılmaz - 2099539
May 6, 2018

1. PROJECT OVERVIEW

Deterministic methods are known to be not very useful in high dimensional configuration spaces due to the lack of representation of obstacles in these spaces. Probabilistic Roadmaps(PRM) and Rapidly-exploring random trees(RRT) have proven themselves to be quite useful in these cases as they do not have such constraints. Even though all the advantages PRM has, it still cannot be able to handle narrow passages between obstacles very well in a reasonable time. There are many other approaches such as Obstacle-based PRM(OBPRM), etc. Spark PRM (Shi, Denny & Amato, 2014) combines ideas by using both PRM in general navigation and RRT in narrow passages which ensures safe passing. I wanted to extend this idea with an efficient closest neighbor algorithm such as stated in (Yershova & Lavalle, 2007). After combining these two, I should be able to make a 6 DOF serial manipulator accomplish decent tasks in a static 3D environment.

2. MOTIVATION AND GOALS

This project may not be considered as a great extension to Spark PRM, but uses it as a tool to understand the nature of PRM for the narrow passages. I see Spark PRM as an intelligent idea. Yet, it is unable to clear many things while presenting itself. The main motivation will be to get a better grip of detecting narrow passages, even in higher dimensions. Then, of course, there will be issues on the creating a connection between RRT and PRM. I believe these issues themselves are great topics to investigate. In the paper, authors only tried comparing Spark PRM with different PRM methods, such as OBPRM. I will try to extend this by an efficient neighbors algorithm and then, unlike the authors, I will concentrate on the unclear parts of Spark PRM.

1. Compare the different effects of different parameters cause in Spark PRM, such as how narrow passages determined, how many nodes are needed to start checking narrow passages, etc.
2. Face the implementation issues, state them clearly in a detailed fashion
3. Compare the difference of a nearest neighbor algorithm creates in the Spark PRM

Also, I will not be interested in the path lengths. Instead I will always be observing and comparing runtime values.

3. SPECIFICATIONS

- Simulation will take place in MATLAB. Even tough MATLAB is not the greatest environment to implement graph based algorithms, its easy and powerful visualization tools makes it great to use in this project.
- Environment is planned to be both 2D and 3D. 2D environment will be chosen so that configuration space can be visualized to get a sense of sample creation process.
- Robot knows the exact representation of the obstacles and itself. This will be done to decrease the time spent on implementation, but project can be configured for initially unknown environments. Though, it would be looking for different aspects of the algorithm.
- Robot will have a 2D circular shape in 2D environment. Robot will not have differential constraints in this simulation. It can be considered as an omni wheel robot.
- Robot joints will also have 3D polygonal shape in 3D environment. This is another specification to decrease the computational power required to run the simulation.

4. PROCEDURES

I plan to first implement a visualization function for the project. This is the highest priority, because visualization will play an important role throughout the research process. Not only it helps debugging, it also helps keeping track of the process management. And also, I will benefit from being able to start writing report from the beginning.

First a simple PRM, without any efficient closest neighbor algorithm, will be simulated in a 2D environment. As stated before, here, workspace will be same with the configuration space. Samples and their connections will visualized.

Then, an RRT for this 2D environment will be implemented. This part should be trivial. However, two issues will show itself here: when to use the RRT and when to stop it. All these questions are left unclear in the article I believe. Looking at the number of connected components should give me some idea about how isolated the sample is. But looking at this value too early, would result in many RRT instances which would greatly increase the run time. Then, connecting issue shows itself. What if there is really not a passage to the PRM samples? Algorithm has to give up after some time maybe. Maybe this is almost a zero possibility case. I will create scenarios and see the results in the simulation.

An efficient neighbor algorithm will be implemented for 2D case here. I expect a decrease the construction time and I will try to observe this and compare these two. An nearest neighbor algorithm might have negative drawbacks as some nodes can stay isolated for longer, and they can be considered as narrow passages by the algorithm. This could increase time. Test cases will help us observer whether this happens or not.

After that I plan to implement a simple PRM for the serial manipulator in a 3D obstacle-free environment. This part should be relatively trivial after all that done.

Then RRT will be added and tested in simple scenarios where obstacles exists in the workspace. By this time, Spark PRM should be the same as in the article. Similar questions to above will be asked again in 3D case. But here, some issue are even more problematic. For example, what is considered as a narrow passage in this 3D environment for a serial manipulator or will the number of samples rise exponentially with the dimensions to detect narrow passages correctly? I believe they will not be the same. These and many more that will come up during the implementation process and they will be tested in scenarios.

Then after I will improve the PRM with a KD-Tree implementation for an efficient closest neighbor. Again, same questions in 2D environment awaits in the 3D environment.

The report should be continued during this whole process.

5. CONCLUSION

Spark PRM seems to be promising algorithm for many cases. But unclear aspects of algorithm creates a distrust in the readers. I will try to clear these parts, and try to get a better picture of all these algorithms while doing so.

REFERENCES

Shi, K., Denny, J., & Amato, N. M. (2014). Spark PRM: Using RRTs within PRMs to efficiently explore narrow passages. 2014 IEEE International Conference on Robotics and Automation (ICRA). doi:10.1109/icra.2014.6907540

Yershova, A., & Lavalle, S. M. (2007). Improving Motion-Planning Algorithms by Efficient Nearest-Neighbor Searching. IEEE Transactions on Robotics, 23(1), 151-157. doi:10.1109/tro.2006.886840