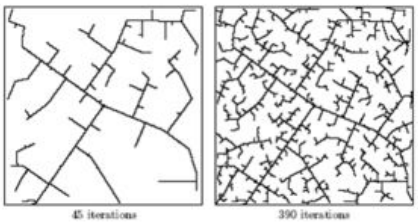

KUKA

Analysis of PRM for the KUKA iiwa

6.881 Project Proposal

Chris Mauck, Christian Moroney, Kevin Lyons

{cmauck10, cmoroney, kalyons}@mit.edu

---

1 Overview of Deliverable for Project

For our final project, we propose an analysis of probabilistic roadmap planning (PRM) and the particular implementation of the RRT algorithm as it relates to motion planning for the Kuka iiwa manipulator. Based on the discussion in lecture and our own expertise with designing robust software algorithms, we feel this is a good project area for us to explore.

We aim to implement PRM via RRT, and then potentially extend to other implementations as well. We would like to create various environments in Drake where the iiwa has to navigate certain obstacles in order to move from a start state to a goal state. Pick and place is one area where this behavior is especially important. After implementing this algorithm in the most basic functional version, we would like to analyze its behavior in various environments, as well as with various parameters (i.e. extension length, post-processing, etc.).

If time allows, we can also consider other PRM variants. These include the optimal planners (like RRT*) which are guaranteed to return the optimal path, rather than just some suboptimal but complete path. We could also potentially explore other OMPL variants and compare their performance.

2 Importance of Project

Motion planning is an interesting concept that relies on multiple topics we have covered throughout the semester. We believe pursuing a project deliverable implementing PRM via RRT with analysis of performance will be a great way to build off a conglomeration of what we have (and will continue to) covered in class. By not limiting ourselves to pure implementation, we enable our project to take on a real-world type experiment. We will be able to see various environments where different algorithmic approaches to Motion Planning perform well (similar to how we have seen different representations work well in various environments. Eg. Rotation Matrix vs. Quaternion).

3 Topics Covered in Class

- Kinematics
- Motion Planning
- Force-Control -> Actually Instructing the robot to move
- Pick and Place
- Object Detection/Segmentation (cool idea to explore where we can have the camera view these objects, and then determine location/adjust planned motion of the path for the arm to go through.)
- Geometric Pose Estimation

4 Related Prior Work

We will build off of previous collab notebooks we've seen in both lecture and on psets for combining various concepts covered throughout the course. (Pick/Place, Kinematics, Force Control, etc.) These will give us a framework for piecing together everything needed for our full project. Furthermore we will pull from these resources:

https://github.com/caelan/pddlstream/blob/stable/examples/drake/motion.py

- Motion planning, force calculations, joint movement calculations, collision detection

https://github.com/RobotLocomotion/drake/blob/master/bindings/pydrake/test/geometry_test.py

- Collision detection in Drake

OMPL

- Open Motion Planning Library documentation/code

RRT

- Article found online discussing RRT vs RRT*

5 Specific Goals for Key Dates

Problem Set 5

- Implement RRT algorithm in python (would like to include some way of visualizing)

Problem Set 6

- Incorporate RRT algorithm into Drake framework for Kuka iiwa motion planning
- Construct environments for robot to perform planned movements (Pick and Place)

Final Presentation/Report

- Analyze performance of PRM algorithm in these environments
- Enhance RRT algorithm (RRT ⇒ RRT*) (+ other enhancements)
- Incorporate other Motion Planning approaches for comparison (pulling from OMPL)

Potential to include more depending how project updates **5/6** show progress towards final deliverable

---

Appendix

For reference, we are including some feedback we got from the course staff regarding this project proposal.

Cool! Although a quick search of RRT and PRM in Drake documentation and codebase did not lead to any meaningful results, I believe there are many 3rd party implementation/integration of motion planning algorithms in Drake. A quick search leads me to

https://github.com/caelan/pddlstream/blob/stable/examples/drake/motion.py

I think it includes pretty much all of the components you will need, especially the collision checker.

You may find it helpful to take a look at

https://github.com/RobotLocomotion/drake/blob/master/bindings/pydrake/test/geometry_test.py

which contains example lines of using the collision checkers in Drake.

Let us know if you have any additional questions!
I agree sticking to PRM or RRT, then implementing an efficient version of it in Drake is an awesome idea.

As we saw in OMPL, there are tons of variations of it that could be interesting so it is easier to control the difficulty of the project as well!

If you're willing to build locally, then using ompl is a nice route to being able to explore a lot of the algorithms.
