6.881 Project Update PSet 5

Chris Mauck, Christian Moroney, Kevin Lyons

{cmauck10, cmoroney, kalyons}@mit.edu

---

1 Project Update

Our team has been able to advance further than anticipated based off of our original proposal PSet 5 deliverable. We began by working on implementing a 3D RRT algorithm from scratch in a GitHub repository. With our algorithm developed, we began looking into some way of visualization for the path returned from our algorithm. We arrived at using plotly for a friendly way of viewing and toying around with RRT algorithms. On top of this, we have been implementing these algorithms in a way that has our future analysis easily available. We have also begun looking into more advanced variations of the RRT algorithm. Essentially, we want to familiarize ourselves with multiple variations of the RRT algorithm in a friendly environment, perform some initial analysis of these algorithms, and then transport the algorithms to our environment with the Kuka Iiwa in MeshCat. We advocate for this two-pronged algorithm analysis approach to be able to provide a theoretical analysis of various MP algorithms, as well as more applied analysis of performance of the robot in a physical environment. Also, it provides a simple way to isolate future algorithm development from implemented analysis. We decided to begin looking into variations from RRT due to some interesting papers/resources encountered as we began project work, also due to the PSet giving us an opportunity to actually implement RRT for the Kuka Iiwa. We also looked at some open source data from the OMPL benchmarking dataset to get a first glimpse at how different sampling-based motion planning algorithms differ across various performance metrics.

2 Future Work

In the coming weeks leading up to the PSet 6 update, we anticipate to have more concrete implementation of RRT-variations not only in our theoretical testing environment but also in an applied testing environment with the robot. Furthermore, we plan on having RRT* implemented, as well as RRT*-Smart (designed to accelerate runtime of RRT* using path optimization and intelligent-space-sampling). Currently, we are not performing any collision checking, but we plan on pulling this in (sourced from OMPL or public GitHub repo's suggested from TA's). With that being said, we will also begin construction of environments for applied testing and analysis of algorithms.

---

3 Appendix

- Our Project Repository: https://github.com/cmronz/6.881_mp_2020.git
- Quick Look at OMPL Benchmarking Dataset: clickable link
- Paper Introducing RRT*/RRT*-Smart: clickable link
- RRT*-Quick: clickable link
- GitHub Library w/ Various Motion Planning Algorithms: clickable link

Visualizations we've made using Plotly to view paths from RRT and Directed RRT