vc68 mk232 Project5 Report

Vincent Chang, Mohamad Kreidieh
Nov. 16 2025

I. Introduction:

1. Problem Statement:
    - During the planning of manipulators, the state can be described by a vector q of joint angles (revolute joints only). We rely on motors to apply torques to exert acceleration on joints to move and achieve collision-free paths under robot physical constraints.
2. Motivation
    - Although it is possible to plan a geometric path (a sequence of waypoints {q0,q1,...,qn}), we aim to get a smoother motion such as continuous velocity $\dot{q}$ and acceleration $\ddot{q}$ after performing trajectory optimization within a limited time to boost the quality of the path.
3. Approach
    - In this research-oriented project, we adapt TrajOpt to optimize the paths generated by the RRT-Connect planner in VAMP and observe paths in terms of length and planning time. We first conducted experiments to test trajectories planning with Trajopt under environments with and without obstacles, using random initialization joint angles, and then comparing it with replacing the initialization paths with planning paths generated by RRT-Connect.

    - Since the optimization is sensitive to initialization, having multiple candidate initialization generated by VAMP increases the chance of getting paths with higher quality.

II. Environment

1. Robot

    Panda robot with 7 controllable joint angles, $(S^1)^7$
    Joint constraints: Velocity < 2.175 rad/s^2, Acceleration and Jerk < 1 rad/s^2

2. Obstacles

   Fourteen spheres with radius = 0.2, centered at (x, y, z)

$$\begin{bmatrix} 0.55 & 0 & 0.25 \\ 0.35 & 0.35 & 0.25 \\ 0 & 0.55 & 0.25 \\ -0.55 & 0 & 0.25 \\ -0.35 & -0.35 & 0.25 \\ 0 & -0.55 & 0.25 \\ 0.35 & -0.35 & 0.25 \\ 0.35 & 0.35 & 0.8 \\ 0 & 0.55 & 0.8 \\ -0.35 & 0.35 & 0.8 \\ -0.55 & 0 & 0.8 \\ -0.35 & -0.35 & 0.8 \\ 0 & -0.55 & 0.8 \\ 0.35 & -0.35 & 0.8 \end{bmatrix}$$

III. Experiments and Analysis

See all the 3D visualization in robotic_projectG_visualization

1. TrajOpt performance analysis

   In this section, we fixed two sets of start-mid-goal waypoints to compare the planning time and path length to observe the same path under the environment with and without obstacles.
   Note: Randomly choosing waypoints won't work most of the time for an environment with lots of obstacles. Since in this section, we only want to check whether the obstacles limit the planning, **we manually chose** the joint waypoints.

   Given four joint waypoints (manual choose) to do interpolation and ask TrajOpt to plan trajectory 20 times:
   start_joint = [2.8, 0.0, 0.0, 0.0, 2.2, 2.2, -0.5]
   mid_joint = [0.0, 0.0, 0.0, -0.5, 1.2, 1.0, 0.0]
   mid_joint = [0.0, 0.0, 0.25, -1.8, -1.0, 0.5, 0.5]
   goal_joint = [0.0, 0.0, -0.5, -2.2, -1.0, 2.0, 1.5]

- Result

Task1	Free Env1	Obs Env1
Average Path Length (m)	1.555	1.533
Average Planning Time (s)	0.236	1.021

- Observation
  We can see that the average path length is similar, but it takes four times more time to generate a path avoiding the obstacles.

b. Given one joint waypoint as start and three cartesian waypoints
   start_joint = [0.0, -0.785, 0.0, -2.356, 0.0, 1.571, 0.785]
   mid_joint = (0.3, 0.0, 0.55)
   mid_joint = (0.35, 0.0, 0.6)
   goal_joint = (0.2, 0.05, 0.65)

- Result

Task2	Free Env2	Obs Env2
Average Path Length (m)	0.362	0.496
Average Planning Time (s)	5.364	69.011

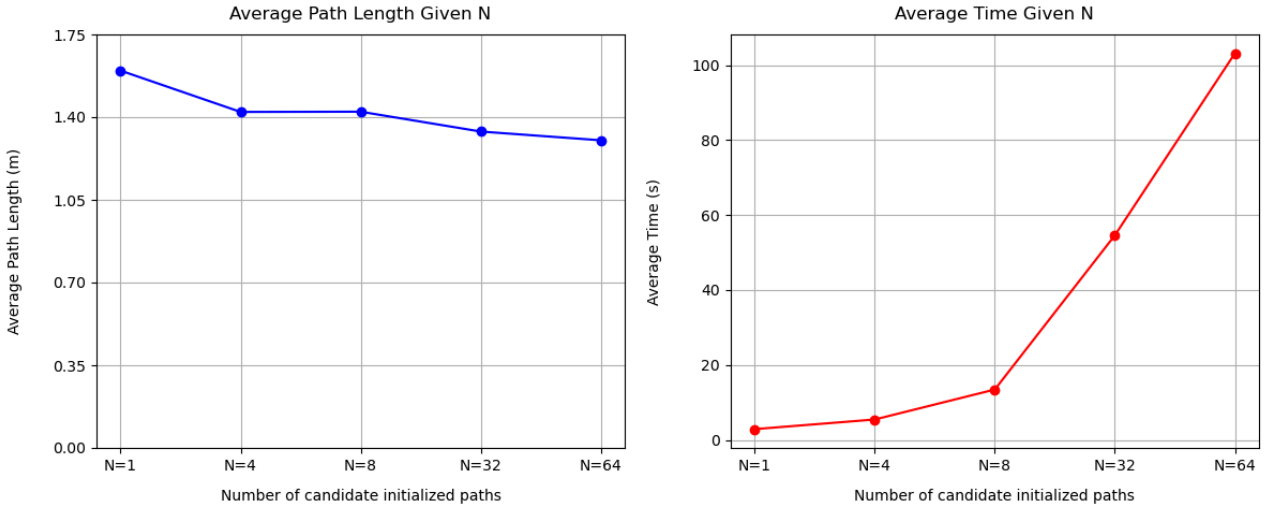
- Observation
  Since giving cartesian waypoints involves both IK and FK solvers, it takes a huge amount of time to perform collision checking in an environment with obstacles, and we also get a longer path to avoid obstacles in this case.

2. Planning with VAMP + TrajOpt optimization
   In this section, we use VAMP with its simplifier to generate the candidate initial paths and optimize with TrajOpt. We conducted experiments to examine the effect of VAMP , its simplifier, and multithreading for TrajOpt. We use the same start and end joint waypoints in Task 1 to conduct the following experiments.
   Note: We apply sampler.skip() with different values each time to simulate a new seed. The amount to skip is given by a random number generator taking the time at initialization as input to ensure different values for each trial. Note that for each set of results, the average is computed over 20 trials.

a. Using VAMP with its simplifier to initialize **N candidate paths**, applying interpolation on the result, optimizing them with TrajOpt, and selecting the one with the shortest states. (Thread = 4)

- Result

Task2	Part 1	N = 1	N = 4	N = 8	N = 32	N = 64
Average Path Length (m)	1.533	1.597	1.422	1.423	1.339	1.302
Average Planning Time (s)	1.021	2.884	5.453	13.453	54.601	103.16

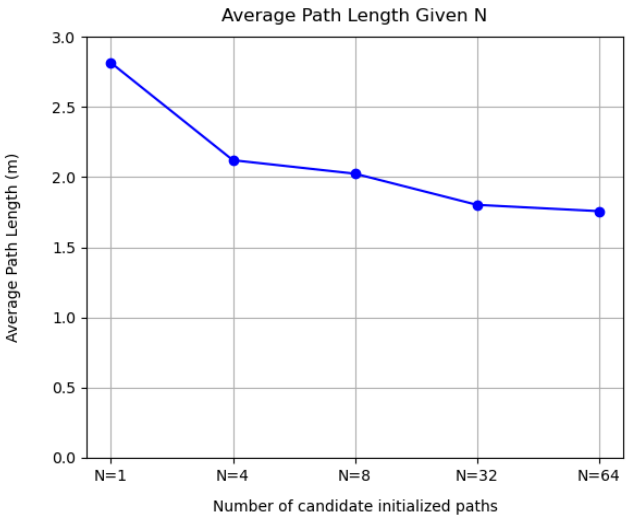


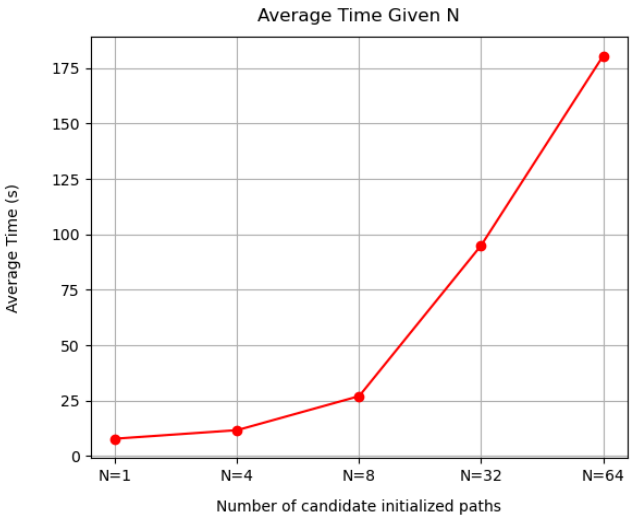
- Observation
  We notice that when using the simplifier, although we are adding interpolation, the overall shape of the path has already been simplified. Thus, the path is already close to ideal, and while an effect is still there, running VAMP and Trajopt multiple times has little effect on the overall length of the route, while significantly impacting the overall planning time.

b. Without simplifier in VAMP (Thread = 4)

- Result

Task2	Part 1	N = 1	N = 4	N = 8	N = 32	N = 64
Average Path Length (m)	1.533	2.814	2.120	2.024	1.802	1.757
Average Planning Time (s)	1.021	7.873	11.724	27.070	95.060	180.549





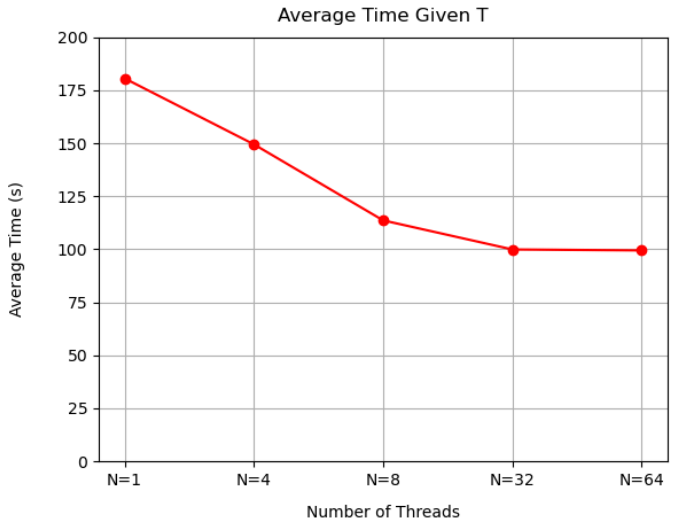
- Observation

  We ran TrajOpt with and without the simplifier. In both cases, we interpolated the initial path (i.e. the path generated by VAMP or the path generated by the simplifier. This gives us further insight into a 'lower-bound' for the potential paths (i.e. the path generated by TrajOpt can become as small as the path generated by the simplifier). However, we notice that in all cases, the path generated by TrajOpt alone is longer than that generated by TrajOpt after implementing the simplifier.

c. Effect of multithread in TrajOpt (N = 64, **T = number of thread**)

- Result

Task2	T = 4	T = 8	T = 16	T = 32	T = 64
Average Planning Time (s)	180.549	149.458	113.604	99.901	99.448



- Observation
  We can see that planning time gradually decreases and reaches the performance limit. Planning with 32 threads saves 45% of time compared to planning with only 4 threads when N = 64.

IV. Conclusion

According to the experiment, we can find that, although optimizing the path with TrajOpt over multiple trials does lead to shorter path length, it comes with a huge cost in planning time. Even with multithreading, the speed up is limited by Amdahl's law and the cost of context switch. Users have to strike a balance between them, with the number of candidate initializations depending on whether the task is urgent or quality-oriented. We also note that while increasing the number of generated paths (i.e. increasing N), the rate of change of the time over path length is exponential, and that after a certain number of trials, the overall performance gain may not be worth the additional computational resources required for the additional runs.

V. Appendix

- Difficulties:
  Understand the topic and build the environment: 7, 4 hours
  Part 1: 9, 20 hours
  Part 2: 8, 5 hours
  Experiments: 8, 5 hours
  Report and Visualization: 8, 5 hrs

- Member Contribution:
  vc68: part1 (75 %, Build TrajOpt Env, Finish Main Pipeline), part2 and bonus (25%, Vamp Connection), experiment (40%, Design Experiment), report (60%)
  mk232: part1 (25%, Pipeline Simplification), part2 and bonus (75%, Vamp Connection, Multithreading), experiment(60%, Conduct Experiment), report (40%)

V. Reference

1. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," The International Journal of Robotics Research, vol. 33, no. 9, pp. 1251–1270, 2014. https://rll.berkeley.edu/~sachin/papers/Schulman-IJRR2014.pdf
2. W. Thomason, Z. Kingston, and L. E. Kavraki, "Motions in microseconds via vectorized sampling-based planning," in IEEE International Conference on Robotics and Automation (ICRA), 2024, pp. 8749–8756. https://ieeexplore.ieee.org/document/10611190