HiWI Position: Robot Motion Planning

November 22, 2024

The goal of this assignment is to test your understanding of some fundamental concepts in programming, source control, and the ability to navigate concepts pertaining to robotic manipulators. Please keep track of approximately how much you spend on this and note it when you submit your solution.

1 Instructions

- You will have **7 days** to complete this test.
- Your submission should contain a local git repository and a remote. The changes you make to the project should be tracked and committed in a organized fashion in order to demonstrate your workflow.
- If certain assumptions need to be made for a specific problem, please go ahead and report them later in a separate file.
- Feel free to choose either **Python**/**C++** as your programming language.

2 Required Software(s) & Problems

The following software and libraries are needed and should be installed in the system before starting:

- Ubuntu 22.04 LTS or higher
- Pinocchio Robotics library[1] or your favorite kinematics library.
- MuJoCo[2]

[1] Installation instructions: https://stack-of-tasks.github.io/pinocchio/download.html
[2] Installation instructions: https://mujoco.org/

Now that you have properly configured the required software, your tasks are:

1. Artifical potential fields (APF) have been used for 3 decades, in the context of real-time obstacle avoidance, since they were initially introduced in the seminal paper by Prof. Oussama Khatib[3]. Watch this video if you need a quick primer[4]. Design a trajectory in MuJoCo using APF for avoiding a spherical obstacle (choose your own radius and start, goal positions) and visualize the path using some marker;

2. Now that there's a reference trajectory, import a Franka Emika Robot model[5] and track the reference generated in the previous step. The following control law can be utilized[6]: Given the desired end-effector pose $\underline{\boldsymbol{x}}_d$ and the task-space error (remember that you need to consider translation and orientation) defined as $\underline{\boldsymbol{e}} = \underline{\boldsymbol{x}} - \underline{\boldsymbol{x}}_d$, the control law generates joint velocities $\dot{\boldsymbol{q}}$ at each step,

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{\dagger} k \underline{\boldsymbol{e}} + \boldsymbol{N} d \left(\boldsymbol{q}_c - \boldsymbol{q} \right), \tag{1}$$

   where $\boldsymbol{J}^{\dagger}$ is the Moore-Penrose pseudoinverse of the Jacobian matrix $\boldsymbol{J}$, the proportional gains are $k, d > 0$, the nullspace projector is $\boldsymbol{N} = \boldsymbol{I} - \boldsymbol{J}^{\dagger}\boldsymbol{J}$, in which $\boldsymbol{I}$ is the identity matrix, and the joint center $\boldsymbol{q}_c = 0.5 \left(\boldsymbol{q}_{\min} + \boldsymbol{q}_{\max} \right)$, with $\boldsymbol{q}_{\min}$ and $\boldsymbol{q}_{\max}$ designating, respectively, the minimum and maximum positions of the robot joints.

3. Create a video (you can just record your screen) that shows the generated path (trajectory), obstacle, and the robot end-effector moving from start to goal on the visualized path while avoiding the obstacle.

3 Submit your solutions

Zip or tar your project directory including all generated files and video(s). Attach your archive along with any comments/issues within **7 days**.

[3] https://journals.sagepub.com/doi/abs/10.1177/027836498600500106
[4] https://www.youtube.com/watch?v=3PYWezYama0
[5] https://mujoco.readthedocs.io/en/stable/models.html
[6] https://github.com/kevinzakka/mjctrl?tab=readme-ov-file