Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Technical Aspects of Multimodal Systems
Department of Informatics
L. Einig

Robot Practical Course
Assignment #3

This assigment targets the creation and optimization of complex trajectories with direct and inverse kinematics. Remember to update the repository files with `git pull`.

Task 3.1 Setting parameters: In order to be able to compare results, all joint limits and joint velocities should be identical. Adjust your URDF file from Assignment 2 with the following limits.

Joint	1	2	3	4	5	6
Lower limit	-6.2832	-6.2832	-3.1416	-6.2832	-6.2832	-6.2832
Upper limit	6.2832	6.2832	3.1416	6.2832	6.2832	6.2832
Velocity	3.15	3.15	3.15	3.2	3.2	3.2

Task 3.2 Creating Waypoints: After updating the repository, launch the new task setup with `roslaunch itr_rpc task_3.launch robot:=<myrobot>.urdf`, where `robot:=` requires an absolute path. You will see a map with the outline of the `house of santa clause`. Write a script which requests an inverse kinematics solution for all five edge points of the house (waypoints) and make the robot move to these edge points using the forward kinematics configuration returned by the inverse kinematics solver (ROS Service). Be sure to respect the control rate, and give the robot sufficient time to execute the motion between the waypoints. The IK solver accepts position and orientation targets, the orientation is (yet) ignored. **Present your solution to a supervisor.**

Things you will need: `srv/IK_Solve.srv`

Task 3.3 Interpolating trajectories: After finding the required waypoints, you are now supposed to interpolate your trajectories. **Discuss the results for each subtask with a supervisor.**

3.3.1: Interpolate the trajectory for one of the paths in cartesian space (interpolate the requests to the IK solver). The path error must stay below 1cm in cartesian space. The orientation target should be $(0, 0, 0)$. It can be a valuable idea to first attempt the interpolation with velocity limits turned off. Ensure to leave sufficient time to reach the start configuration.

Things you might need: `numpy.arange()`, `numpy.linspace()`

3.3.2: Interpolate the full trajectory for the house without drawing any line twice. Keep the path error below 1 cm. The orientation target should stay $(0, 0, 0)$.

3.3.3: Reduce the orientation error with the dynamic reconfigure gui to 0.01. Discuss the execution of the trajectory with a supervisor and implement a solution to the problems arising.

3.3.4 *Bonus*: Optimize the trajectory using your knowledge on trajectories from the lecture.

Bonus Task 3.4 Interpolating in joint space:

3.4.1 *Bonus*: Interpolate the trajectory for one of the paths in joint space (interpolate the commands to controller). The path error must stay below 1 cm in cartesian space.

3.4.2 *Bonus*: Interpolate the full trajectory for the house without drawing any line twice. Keep the path error below 1 cm.