ENSC488: Introduction to Robotics
Simon Fraser University, Fall 2016
Demo 2

Group #
Member 1:
Member 2:
Member3:

Lab Project Demo 2 Instructions

Materials to be submitted:

- This page with the name and student no. of your group members written at the top.
- A succinct description of your trajectory planner specifying if it is a joint or Cartesian space planner, interpolation scheme used, type of continuity it ensures at the start, goal, and via points. This should not take more than a page.

Demonstrations:

Your program should demonstrate the followings:

- **Planning:** Plan a trajectory for the robotic arm from current Tool frame to a given goal Tool frame (and possibly up to three intermediate frames) provided by the user. Please note that the tool frame will be given as the position and orientation of the tool, i.e., (x, y, z) and φ with respect to Station frame.
- **Plotting:** Plot and show the planned trajectory (for all joints) sampled at a reasonable time resolution. For this you may use MATLAB or Excel. Please note that sampling resolution for plotting should not be too high (or you can potentially run into memory issues). It would be nice to have it interactive, i.e., send joint values to MATLAB as you move the robot, so that the plot is seen as the robot moves. But it is also ok, to simply do it offline, i.e. dump the values to a file and then use MATLAB to plot it after the robot motion is finished. Do make sure that it does not take you more than a few tens of seconds to massage/edit the data to plot. We have limited time for demos.
- **Grasp/Ungrasp:** Have the ability to execute a Grasp/Ungrasp command at the Goal Frame. We will use this to show a pick&place operation.
- **Emulator/Robot:** Execute the planned trajectory on the emulator and on the robot.

The planner should:

- select (for joint space trajectory planning) appropriate inverse kinematic solutions for each via point;
- detect and inform the user if joint limits, velocity limits and acceleration limits are exceeded (via say a message to console) at any instant in the trajectory.
- handle modulo 360 degree arithmetic for all angles where needed.

Please note that modules from Demo 1 (forward and inverse kinematics) must also be working (within the same program) since we may need to query your code for current joint and tool values.