Coursework 2: code. Instructions and marking criteria

The code submission accounts for 40% of your total mark. 32 marks are given for achieving the lab “as is”, while the remaining 8 marks will be given for excellency criteria.

Code submission:

It is your responsibility to submit a code that is expected to run “as is” on a DICE machine. If you include third-party libraries not initially planned in the lab, you should contact me to ask for authorisation well ahead of the deadline.

If you don’t apply strictly to the instructions mentioned in the following, **you will lose points.**

Zip file submission:

Your code should be submitted as a single zip file named as follows:

STUDENT1NAME_STUDENT2NAME.zip where STUDENT*NAME is to be replaced with the family name of each student.

Each zip should contain:

- The entire project (should run “as is”)
- A video of a successful run of your code. (Optionally, a video for the extra credits if it applies)
- A README file that describes in maximum 300 words (less is good):
    - What other software is needed to run the code, (assuming this has been validated with us).
    - For each part, which algorithm was used to solve the problem
    - Any short description of the work is accomplished to get extra points.

I expect that I should be able to unzip the files and obtain the following behaviours regardless of whether you have solved the task or not:

- I can run inverse_geometry.py. After execution, configurations q0 and qe should be defined, consisting in the initial and final configurations as described in the instructions. computeqgrasppose is defined with the same API as in the template.

- I can run path.py. This should display a path that satisfies the constraints. computepath is defined with the same API as in the template, with the possible definition of extra parameters for the cube and robot objects. It returns a list of configurations as a path

- I can run control.py. This opens a pybullet window and executes the grasping motion until the target is reached. I can modify q0, qe and the path and the changes should be reflected in the results (even if it fails)

Marking criteria:

You will get up to 32 points if each task is solved with the initial placements of the cube. You can get extra points if your framework is able to handle noise in the initial and target positions for the cube.

- 8 points for successfully solving the inverse geometry problem
- 12 points for solving the motion planning part
- 12 points for the control part. 4 Points if the control is achieved without managing to accurately grasp the cube (i.e. you only achieved optional task 0)

The remaining 8 points are distributed as follows (you can actually get more than 8 points, but this will be capped at 8 points)

- 2 points if you implement version 1 of the trajectory optimisation
- 2 points depending on the robustness of your code to different initial / goal positions for the cube (I ll test this using an unknown method)
- Up to 5 points for going beyond expectation and extending the scope of your framework (make sure to describe this in your report). **WARNING**: Please consider carefully the time available and the efforts that you can put into this task before committing to this. If you decide to abandon, make sure you describe your efforts in the report. Examples of extensions include but are not limited to:
    - Implementing an efficient search for the nearest neighbour in the RRT (eg with a kd-tree)
    - Changing the scene such that improvements in the RRT are necessary to find a solution (to be justified)
    - Using force feedback from the effector to implement a better control law
    - Throwing the cube into a basketball
    - Etc...