Advanced robotics (ARO)
Course Assignment 1
Total: 10 Marks

Steve Tonneau ©University of Edinburgh
29 September 2025 (homework handout)

This coursework counts for 10% of the final mark. Each point is equivalent to 1%.

Before starting, please consider the following points:

- This should not take more than a few hours. If you feel like a question requires a disproportionate amount of research / work, you probably misunderstood it.
- I am not expecting you to program anything. Paper and pen, or a LaTeX report if you want to practise, is all you need. Of course you are allowed to use code if that helps you, but I won't review it.
- The coursework does not require any calculations. I am more interested in assessing your understanding of the consequences of given design choices.
- You need to work alone on this homework. Collaboration is not allowed. You can ask for clarifications on EdStem, but any form of request for a solution will be sanctioned.

Warm-up

We consider a 3D manipulator robot, such as the one we used for lab 0. The configuration of the robot is described by $\mathbf{q} \in \mathbb{R}^n$. The obstacles of the environment are described by a set of n spheres $\mathcal{S} = \bigcup_{i=1}^{n} S_i$, each of center $\mathbf{c}_i$ and radius r_i.

question 1 (1 pt): Given a configuration $\mathbf{q} \in \mathbb{R}^n$, provide the pseudo-code for a predicate $sphereCollision(q, S_i)$ that determines whether the robot is in collision with a sphere S_i. You can assume that the rigid bodies of the robot are cylinders of a known radius w_i connecting the joints. Self-collisions are ignored for this question. You can use any function given by the pinocchio API, except for the distance and collision functions :).

Testing whether the robot is in collision then simply consists in calling *sphereCollision* for all S_i. Propose a simple means to determine which spheres do not require a collision test.

Motion planning

We consider the task of sequentially touching all the spheres of $\mathcal{S}$ with the end-effector, without entering in collision. We are not interested in any form of optimality (we do **not** consider shortest distance travelled etc) and we only consider the geometry of the problem (no dynamics).

question 2 (5 pt): In your own words, explain your approach to the problem. First, list what sub-problems need to be resolved. For each sub-problem, provide the inputs / outputs and describe your particular approach to resolving the sub-problem (with or without pseudo-code, as you prefer - any method from the pinocchio API can be used if you feel the need). I don't need details, but I need to understand exactly how you plan to solve the problem. What are the termination conditions for your framework? Provide a logical diagram (a simple sketch with boxes and arrows is sufficient) that illustrates the framework.

Marking criteria: 4 points for a complete framework and clear justification of each submodule. 1 point for the overall "quality" of the solution.

question 3 (3 pt): Comment on the following properties of your framework (no need for a formal proof, just a well-argued answer). Answer briefly (a few sentences for each question are enough).

- Are you guaranteed to find a solution if it exists? If there is no solution, can your approach maximise the number of spheres reached?
- How does the number of obstacles in the scene influence the efficiency of your framework?
- What are the limitations of your approach?

question 4 (1 pt): Propose a heuristic that could be used to estimate in which order the spheres should be touched to minimise the total distance travelled by the robot in the configuration space. Comment on the cases where it will be efficient, or not.