Cooperative Robotics

Authors: "names"
Email: "email"
Date: "date"

General notes

- Exercise 1 is done with the ROBUST matlab main and unity visualization tools. Exercises 2-3 are done with the Franka simulation tools.
- Comment and discuss the simulations, in a concise scientific manner. Further comments, other than the questions, can be added, although there is no need to write 10 pages for each exercise.
- Aid the discussion with screenshots of the simulated environment (compress them to maintain a small overall file size), and graphs of the relevant variables (i.e. activation functions of inequality tasks, task variables, and so on). Graphs should always report the units of measure in both axes, and legends whenever relevant.
- You can add vertical bars to plots to highlight periods of time (e.g. when one action is active), or vertical ones to highlight thresholds (e.g. the minimum altitude value).
- Report the thresholds whenever relevant.
- Report the mathematical formula employed to derive the task jacobians and the control laws when asked, including where they are projected.
- If needed, part of the code can be inserted as a discussion reference.
- Provide your answers in the subsection where the question is made.

Use the following template when you need to discuss the hierarchy of tasks of a given action or set of actions. Please report the tasks in the same order as in the unified hierarchy implemented in your code.

Table 1: Example of actions/hierarchy table: a number in a given cell represents the priority of the control task (row) in the hierarchy of the control action (column). The type column indicates whether the objective is an equality (E) or inequality (I) one.

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
Task name A	I	1		1
Task name B	I	2	1	
Task name C	E		2	2

1 Exercise 1: Implement a Complete Mission

Implement several actions to reach a point, land, and then perform the manipulation of a target object.

Initialize the vehicle in a position that allows you to test all the tasks. Then, use a "safe waypoint navigation action" to reach the following position:

$$\begin{bmatrix} 10.5 & 37.5 & -38 & 0 & -0.06 & 0.5 \end{bmatrix}^\top$$

Then land, aligning to the nodule. In particular, the x axis of the vehicle should align to the projection, on the inertial horizontal plane, of the unit vector joining the vehicle frame to the nodule frame. Once landed, implement a "fixed-based manipulation action" to reach the target nodule (mimicking the scanning of the nodule). During this manipulation phase, the vehicle should not move for any reason. Considering that the goal position for the vehicle could be not so precise with respect to the position of the nodule, implement a solution that guarantees that, once landed, the nodule is certainly within the manipulator's workspace. If needed, you can change also the vehicle's goal position if you need to stress the system and force certain tasks to activate (e.g., the distance to the nodule one).

1.1 Q1: Report the unified hierarchy of tasks used, the actions and their tasks, specifying their priorities in each action.

1.2 Q2: Comment the behaviour of the robot, supported by relevant plots. For example, show the altitude and the altitude activation to show that when the altitude becomes too small, the tasks gets activated and a minimum value is guaranteed.

1.3 Q3: What is the Jacobian relationship for the Alignment to Target control task? How was the task reference computed?

1.4 Q4: What have you done to guarantee that the nodule is within the manipulator's workspace?

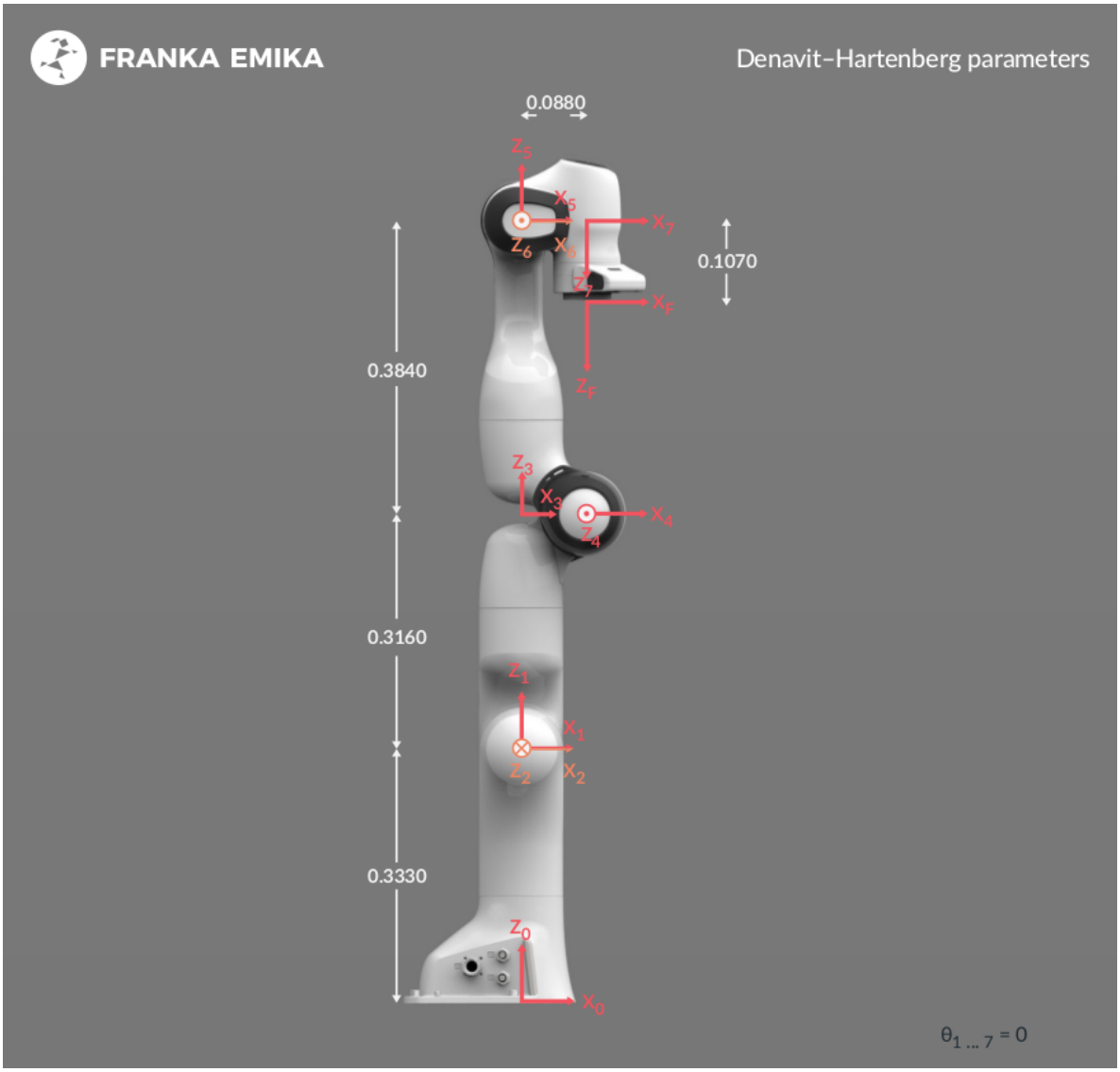

Figure 1: Robot Specifications

2 Exercise 2: Bimanual manipulation

In this exercise it is required to perform a bimanual manipulation by implementing the task priority algorithm considering two manipulators as a single robot. The manipulator adopted for this assignment is the Franka Panda by Emika (see Figure 1)). A simulation in python is provided, in order to visualize the two robots and test your Matlab implementation.

1. The transformations between the world and the base of each robot must be computed knowing that:
   (a) The left arm base coincides with the world frame.
   (b) The right arm base is rotated of π w.r.t. z-axis and is positioned 1.06 m along the x-axis and -0.01 m along the y-axis.
2. The transformation from each base to the respective end-effector is given in the code and is retrieved from the URDF model thanks to the *Robotic System Toolbox* of Matlab.
3. Then, define the tool frame for both manipulators; the tool frames must be placed at 21.24 cm along the z-axis of the end-effector frame and rotated with an angle of -44.9949 deg around the z-axis of the end-effector.

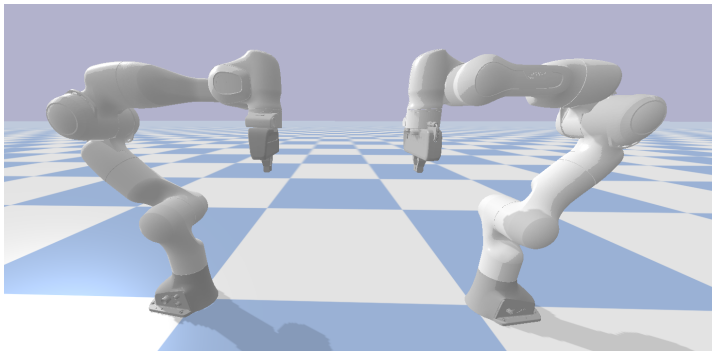

Figure 2: Initial position of the robots

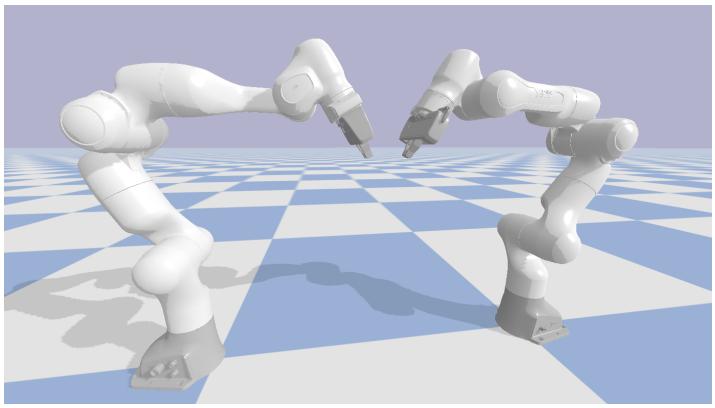

Figure 3: Relative orientation at the grasping position

4. Implement a "move to" action that includes the joint limits task.

5. The first phase foresees to move the tool frame of both manipulators to the grasping points, by implementing a "move to" action and its corresponding tasks. Define the goal frames for each manipulator such that their origin corresponds to the grasping points. **HINT**: the position of the grasping points can be computed by knowing the origin of the object frame ${}^{w}\mathrm{O}_{o}$ and the object length l_{obj}. The goal orientation of the tool frames is obtained by rotating the tool frames 30 deg around their y-axis; the manipulators should reach the configuration depicted in Figure 3.

$$ {}^{w}\mathrm{O}_{o} = [0.5, 0, 0.59]; \tag{1} $$

$$ l_{obj} = 10\,(\mathrm{cm}). \tag{2} $$

6. During this phase of the mission, the following safety tasks must be implemented: *end-effector minimum altitude* and *joint limits*. The joint limits specifications can be found in the datasheet of the robot. The minimum altitude from the table should be 15 cm.

Once the manipulators reach the grasping points, the second phase of the mission should start. Now, implement the Bimanual Rigid Grasping Constraint task to carry the object as a rigid body.

1. Define the object frame as a rigid body attached to the tool frame of each manipulator. **HINT:** Compute this quantity after reaching the grasping point.

2. Define the rigid grasp task.

3. Then, move the object to another position while both manipulators hold it firmly, e.g. ${}^{w}\mathrm{O}_{g} = [0.65, -0.35, 0.28]^{\top}(m)$

4. Note that the transition for the *Bimanual Rigid Constraint* should be a binary one, i.e., without smoothness. This is the nature of the constraint, i.e., either it exists or not. Modify the *ActionManager* class to take into account the different nature of this task (a constraint one).

Once the object has been moved to the required position you have to implement a final phase of the mission in which the joint velocities are set to zero, and every action is deactivated except for the minimum altitude task.

Repeat the simulation, using a goal position or by changing the grasping in such a way that one of the two manipulators activates multiple safety tasks, to stress the constraint task.

2.1 Q1: Report the unified hierarchy of tasks used and their priorities in each action.

2.2 Q2: What is the Jacobian relationship for the Joint Limits task? How was the task reference computed?

2.3 Q3: What is the Jacobian relationship for the Bimanual Rigid Grasping Constraint task? How was the task reference computed?

2.4 Q4: Comment the behaviour of the robots, using relevant plots. In particular, show the difference (if any) between the desired object velocity, and the velocities of the two end-effectors in the two cases. Add plots to show that the bimanual manipulation unfolds effectively (e.g., the distance between the tool frames is constant to show that the kinematic constraint is satisfied)

3 Exercise 3: Cooperative manipulation

In this exercise, it is required to perform cooperative manipulation by implementing the task priority algorithm considering the two Franka Panda manipulators as two distinct robots.

1. The first phase foresees to move the tool frames to the grasping points, by implementing the "move to" action for both manipulators. **Please note: each manipulator has his own Task Priority Inverse Kinematic Algorithm**. Define the goal frames for each manipulator such that their origin corresponds to the grasping points. **HINT**: the position of the grasping points can be computed by knowing the origin of the object frame ${}^{w}\mathrm{O}_{o}$ and the object length l_{obj}. The goal orientation of the tool frames is obtained by rotating the tool frames 20 deg around their y-axis.

$$ {}^{w}\mathrm{O}_{o} = [0.5, 0, 0.59]^{\top}(m); \tag{3} $$

$$ l_{obj} = 6\,(\mathrm{cm}). \tag{4} $$

   During this phase of the mission, the following safety tasks must be implemented: *end-effector minimum altitude* and *joint limits*. The joint limits specifications can be found in the datasheet of the robot. The minimum altitude from the table should be 15 cm.

2. Once the manipulators reach the grasping points the second phase of the mission should start. Implement the *Cooperative Rigid Constraint* task to carry the object as a rigid body.

   (a) Define the object frame as a rigid body attached to the tool frame of each manipulator.

   (b) Define the rigid grasp task.

   (c) You have to move the object to another position while both manipulators hold it firmly. The desired object goal position is

$$ {}^{w}\mathrm{O}_{g} = [0.60, 0.40, 0.48]^{\top}(m) \tag{5} $$

   (d) Compute the *non-cooperative* object frame velocities. **HINT:** Suppose manipulators communicate ideally and can exchange the respective end-effector velocities

   (e) Apply the coordination policy to the *non-cooperative* object frame velocities.

   (f) Compute the *cooperative* object frame velocities.

   Note that the transition for the *Cooperative Rigid Constraint* should be a binary one, i.e., without smoothness. This is the nature of the constraint, i.e., either it exists or not. Modify the *ActionManager* class to take into account the different nature of this task (a constraint one).

3. Once the object has been moved to the required position you have to implement a final phase of the mission in which the joint velocities are set to zero, and every action is deactivated except for the minimum altitude task.

Again, test it twice, once with the provided (reachable) goal, and then with a goal or with a different grasp configuration that triggers the activation of multiple joint limits or a joint limit and minimum altitude by one manipulator. The idea is that this second position should trigger the cooperation policy (i.e., the cooperative velocity of one manipulator should be different than the original desired object velocity, and that the cooperative velocity is very similar to its non cooperative one, showing that the other robot adapts).

3.1 Q1: Report the unified hierarchy of tasks used and their priorities in each action. Report cleary which action is used by TPIK1 and TPIK2 in the cooperative algorithm.

3.2 Q2: Comment the behaviour of the robots, using relevant plots. In particular, make sure there are differences between the desired object velocity and the non-cooperative Cartesian velocities at least in one simulation. Show also how the cooperative velocities of the two end-effectors behave. Add plots to show that the cooperation unfolds effectively (e.g., the distance between the tool frames is constant to show that the kinematic constraint is satisfied)