Exercise cooperative Robotics – Cooperative Manipulation

December 12, 2024

1 Introduction

In this exercise, it's required to perform a cooperative manipulation by implementing the task priority algorithm for two separate manipulators. Suppose manipulators communicate ideally and can exchange the respective end-effector velocities. The manipulator adopted for this assignment is the Franka by Emika, a versatile 7-DOF manipulator for researchers (see the datasheet attached and Figure 1). Moreover, a simulation in Python is provided, to visualize the two robots and test your Matlab implementation (see the *Franka Visualization Tools* folder in the Teams channel of the course).

The template of the code for this assignment can be found in the folder *Matlab Scripts* \ *cooperative_manipulation* in the Team channel of this course. The folder contains the file *main.m* and a list of functions that should be implemented to complete the exercise; while inside the *simulation_scripts* folder there are some useful functions already implemented at your disposal.

2 Assignment

1. To complete this exercise, we need to define the frames that describe the system depicted in Figure 2, the objective is to obtain two kinematic chains, one for each manipulator. First, define the transformation matrices between the word frame and the two end-effector frames. 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.

   (c) 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.

   Then, define the tool frame for both manipulators; the tool frames must be placed at 21.04 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.

   After defining the frames involved in the manipulation we can move on to the implementation of the various phases of the mission. The cooperative manipulation mission should be implemented with three different phases: *reach the object*, *move the object*, and *stop the motion*.

2. The first phase foresees to move the tool frames to the grasping points, by implementing the *Tool move-to* equality task for both manipulators. **N.B 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; the manipulators should reach the configuration depicted in Figure 3.

$$ {}^{w}\mathrm{O}_{o} = [0.50, 0, 0.30]^{\top}(m); \tag{1} $$

$$ l_{obj} = 6\,(\mathrm{cm}). \tag{2} $$

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 given with this assignment. The minimum altitude from the table should be 15 cm.

3. Once the manipulators reach the grasping points the second phase of the mission should start. Now you have to 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. Below is the object goal position.

   $$^{w}\mathrm{O}_{g} = [0.60, 0.40, 0.48]^{\top}(m) \tag{3}$$

   (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 *ActionTransition* script seen during the class to take into account the different nature of this task (a constraint one).

4. 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.

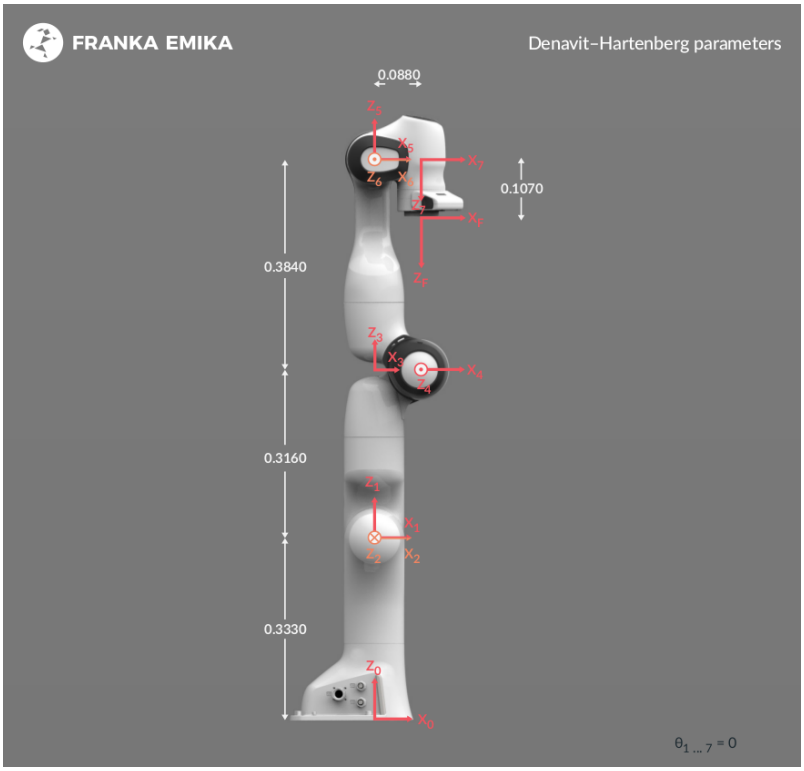

Figure 1: Robot Specifications

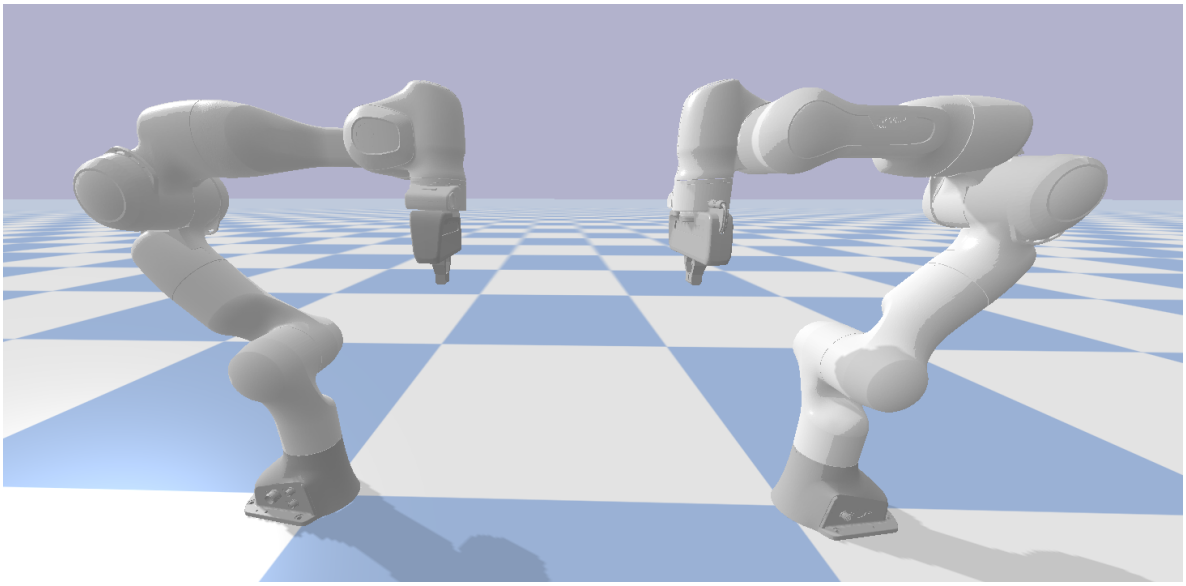

Figure 2: Left and Right Arm of the bimanual system

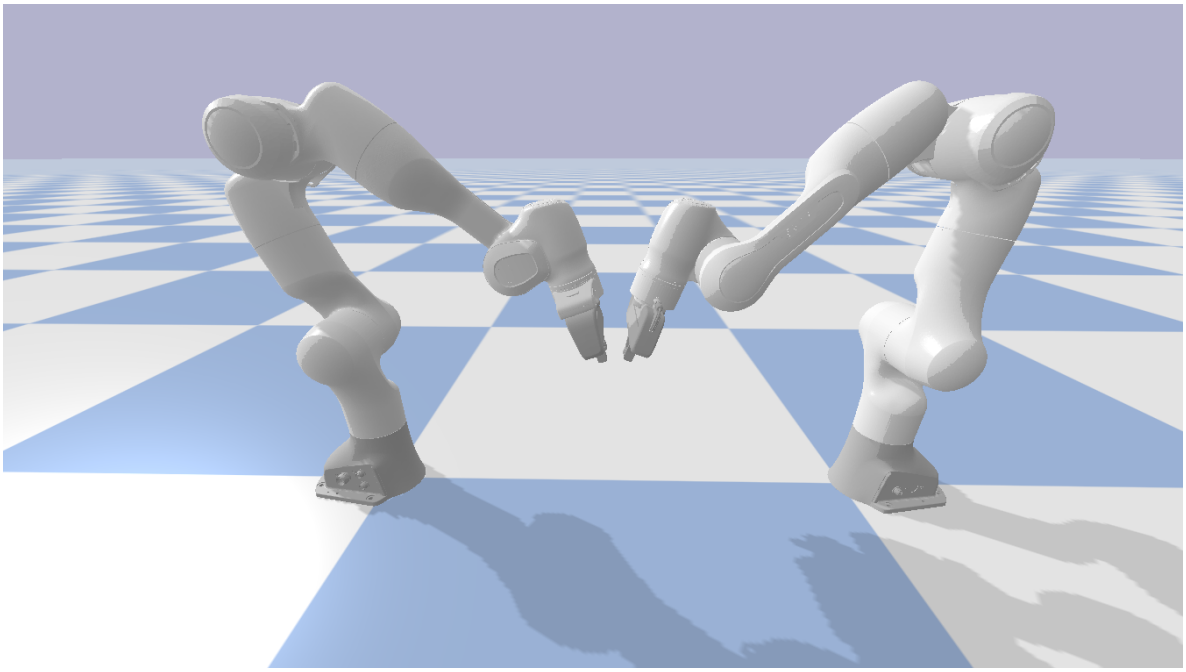

Figure 3: Robot configurations for grasping the object