Cooperative Robotics

Authors:	Email:
Dikshant Thakur	s5938924@studenti.unige.it
Ouassim Milous	s5943225@studenti.unige.it
Girum Molla Desalegn	s6020433@studenti.unige.it

Date: 02/02/2025

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 at the position:

$$\begin{bmatrix} 8.5 & 38.5 & -36 & 0 & -0.06 & 0.5 \end{bmatrix}^\top$$

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.

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

$\mathcal{A}_1$ - Safe Waypoint Navigation Action.
$\mathcal{A}_2$ - Nodule Alignment Action
$\mathcal{A}_3$ - Safe Landing Action
$\mathcal{A}_4$ - Fixed Base Manipulatin Action

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$
Minimum Altitude	I	1	1	1	1
Horizontal Alignment	I	2			
Nodue Alignment	E			2	2
Landing	E		2		3
Vehicle Stop	E				4
Tool Manipulatin	E	3			
Position Control	E				5

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 Defintions
Minimum Altitude - This task maintains the altitude between vehicle and sea floor.
Horizontal Alignment - This task responsible to maintain the horizontal position with respect to sea floor.
Rock Alignment - This task is responsible to align the vehicle head with x axis of nodule.
Landing - This task is responsible to land the uvms system on the sea floor.
Vehicle Stop - To make sure that vehicle will not move during the manipulation task.
Tool Manipulatin - This task move the manipulator arm to the goal position.
Position Control - It controls the vehicle to achieve its desired location.

1.2 Q2: Comment the behaviour of the robots, supported by relevant plots.

On the basis of the actions , which are mentioned above, we define our phases and here are our observation.

Comment 1 - In the action of Safe Waypoint Navigation , vehicle starts from its intial position and goes linearly to its desired position. Keeping the vehicle horizontal stable and the manipulator is still not active.
PLOT OF THE GRAPH
Comment 2 - After reaches the desired position, whole system align itself with the x axis of nodule. Keeping the whole system horizontal stable.

TOP VIEW
Comment 3 - In this phase, keeping the vehicle aligns with respect nodule x axis and sea floor, whole system goes to the sea floor vertically. PLOT THE GRAPH FOR ALTITUDE.
Comment 4 - Once the vehicle lands the manipulator arm will reach to the noduel, which is its desired position.
PHOTO and PLOT

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

To align the vehicle with the nodule, we calculated the angle difference between vehicle and nodule. Here is the calculation step: First we calculated the the distance between the current posiition and the nodule with respect to world frame.

$$^{w}v_t = {}^{w}v_{goal} - {}^{w}v_{current}$$

With that distance vector, we find the angle between the current position of the vehicle and the target position of the vehicle with respect to world frame.

$$^{w}v_t = {}^{w}v_{goal} - {}^{w}v_{current}$$

Once we have the angle difference then we constraint the yaw angular vehicle velocity in like this:

$$\dot{x}_{yaw} = \psi_{taget} - \psi_{current}$$

The Jacobian we are using is the same we used in the vehicle posiition control task. Because we constrained our velocities only for yaw angular vehicle axis, so only solution we will get is for the yaw angular vehicle velocity.

$$^{w}J_v = \begin{bmatrix} \mathbf{0}_{3\times7} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times7} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix}$$

1.4 Q4: Try changing the gain of the alignment task. Try three different values: 0.001, 0.1, 1.0. What is the observed behaviour? Implement a solution that is gain-independent guaranteeing that the landing is accomplished aligned to the target.

We observed that with gain 1.0, the time taken by vehicle to reduce the angular error between its body and nodule around z-axis with respect to world frame is faster than if we assign the gain 0.1 and 0.001. To be precise the time take in the first case, when gain is 1, is 10.5 seconds. In the second and third case, which are 0.1 and 0.001 respectively, it takes 52 seconds and in the latter case it moves infinitinemaly slow.

1.5 Q5: After the landing is accomplished, what happens if you try to move the end-effector? Is the distance to the nodule sufficient to reach it with the end-effector? Comment the observed behaviour.

After landing task, the manipulator easily achieved its task without vehicle moving.

1.6 Q6: Implement a solution that guarantees that, once landed, the nodule is certainly within the manipulator's workspace.

To make sure that nodule is always in the work space of the manipulator arm. We need to control the vehicle in such a way it always come closer to the nodule. For that first we need to calculate the x and y component of linear error, because we always activate this task after we have done the horizontal alignment, between the rock and tool.
After getting the x and y component of linear error, we constrain our vehicle linear x and linear y velocities in such a manner:

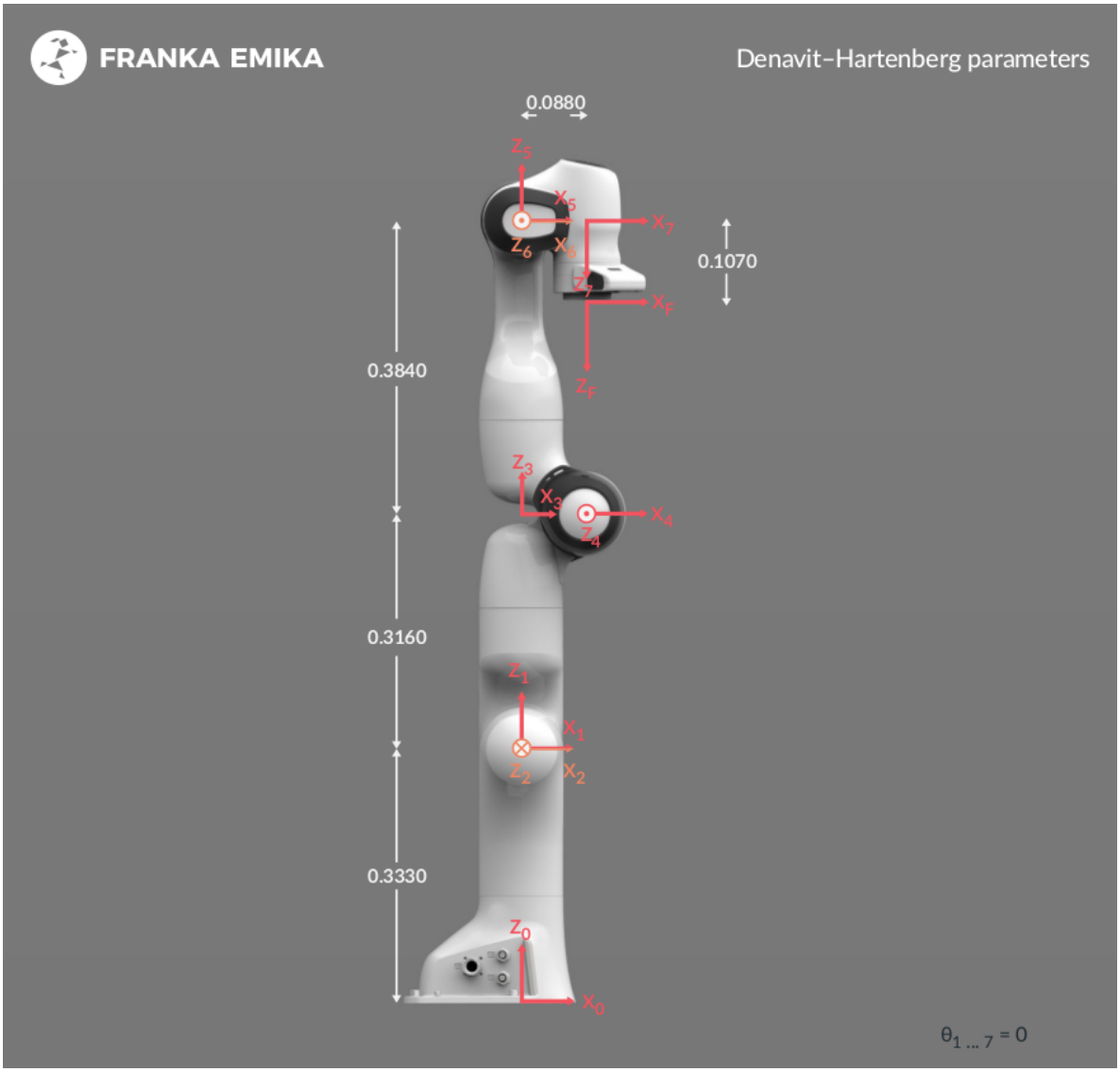

Figure 1: Robot Specifications

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 21.04 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.

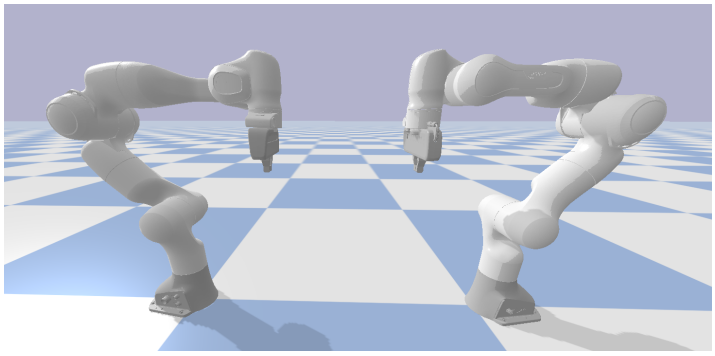

Figure 2: Initial position of the robots

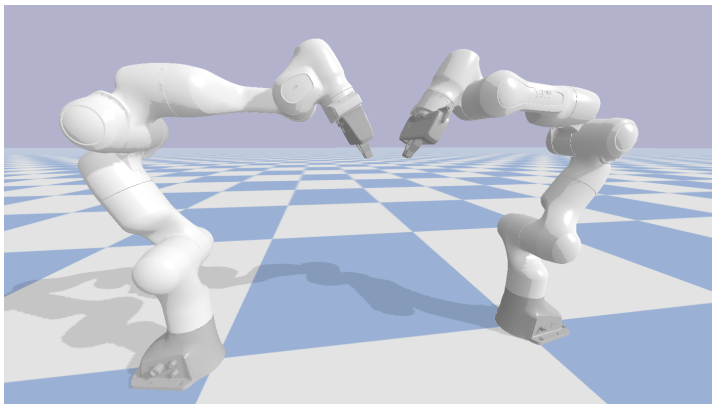

Figure 3: Relative orientation at the grasping position

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 *Action-Transition* script seen during the 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.

Actions :

$\mathcal{A}_1$ - Move-to.
$\mathcal{A}_2$ - Co-operative move-to
$\mathcal{A}_3$ - Halt manipulator

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
Joint limits	I	1	1	
Minimum Altitude	I	2	2	
Bimanumial rigid grasping constraint	E		3	
Move-To	E	3	4	

Table 2: 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 Defintions
Joint limits - This is safety task to constraint the each joints of the manipulator , which is responsible to maintain the joint angle in its range.
Minimum Altitude - This is the safety task where the tool of the manipulator keeps the minimum distance from the ground.
Bimanual rigid grasping constrain - In this task we constraint our robot to act together.
Move to - this task the manipulator move to the desired position

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.

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 *ActionTransition* script seen during the 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 should be different than the original desired object velocity).

3.1 Q1: Report the unified hierarchy of tasks used and their priorities in each action, and report clearly the actions used in the two phases of the cooperation 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.