

# COOPERATIVE ROBOTICS

Authors:

Dikshant Thakur  
Ouassim Milous  
Girum Molla Desalegn

Email:

s5943225@studenti.unige.it  
s5938924@studenti.unige.it  
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:

$$[8.5 \quad 38.5 \quad -36 \quad 0 \quad -0.06 \quad 0.5]^\top$$

Use a "safe waypoint navigation action" to reach the following position:

$$[10.5 \quad 37.5 \quad -38 \quad 0 \quad -0.06 \quad 0.5]^\top$$

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.

Actions list:

- $\mathcal{A}_1$  - 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	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$	$\mathcal{A}_4$
Horizontal Alignment	[I, S, P]	2			
Minimum Altitude	[I, S, P]	1	1	1	1
Vehicle control	[E, AD]				4
Approach Nodule	[E, AD]				5
Landing	[E, AD]		2		3
Tool	[E, AD,]	3			

Table 1: The numbers in the table represent the priority of each action, with the first task having the highest priority and the last task having the lowest. For example, in Action 2, only the Minimum Altitude and Landing task are activated, so they have priority numbers. A number next to an action indicates that the task is activated for that particular action. The higher the number, the lower the priority for that action. The "type" column indicates whether the objective is an equality (E) or inequality (I).

### Tasks' Defintion

- **Minimum Altitude**—This task maintains the altitude between the vehicle and the sea floor.
- **Horizontal Alignment**—This task is responsible for maintaining the horizontal position with respect to the sea floor.
- **Rock Alignment** - This task is responsible for aligning the vehicle head with the axis of the nodule.
- **Landing** - This task is responsible for landing the UVMS system on the sea floor.
- **Vehicle Stop** - To make sure that the vehicle will not move during the manipulation task.
- **Tool Manipulation** - This task moves 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 initial 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 nodule, 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 yaw angle difference between the vehicle and the nodule. The calculation steps are as follows:

First, we calculated the distance between the current position and the nodule with respect to the world frame.

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

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

$$\psi_{target} = \tan^{-1} \left[ \frac{{}^w v_{target}(y_{component})}{{}^w v_{target}(x_{component})} \right]$$

Once we have the  $\psi_{target}$ , we then constrain the yaw angular velocity of the vehicle as follows:

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

$$\dot{x} = \begin{bmatrix} \mathbf{0}_{5 \times 1} \\ \dot{x}_{yaw} \end{bmatrix}$$

The Jacobian we are using is the same one we used in the vehicle position control task. Since we constrained our velocities only for the yaw angular velocity of the vehicle, the only solution we will get is for the yaw angular vehicle velocity.

$${}^w J_v = \begin{bmatrix} \mathbf{0}_{6 \times 7} & \mathbf{I}_{6 \times 6} \end{bmatrix}$$

## 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. lol my name is

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

To ensure that the nodule is always within the workspace of the manipulator arm, we need to control the vehicle in such a way that it always moves closer to the nodule. To do this, we need to control the linear x and linear y velocities so that the arm is always in the projection of the nodule. Here are the steps we followed: First we need to find the linear distance between the current position of vehicle and the nodule.

$$e_{linear} = {}^wT_g - {}^wT_t$$

where  $e_{linear}$  is the linear error, which is 3x1 matrix,  $< w >$  is world frame, and  $g$  and  $t$  represent the goal and the tool of the arm, respectively.

$$\dot{x} = \begin{bmatrix} e_{linear_x} \\ e_{linear_y} \\ 0 \end{bmatrix}$$

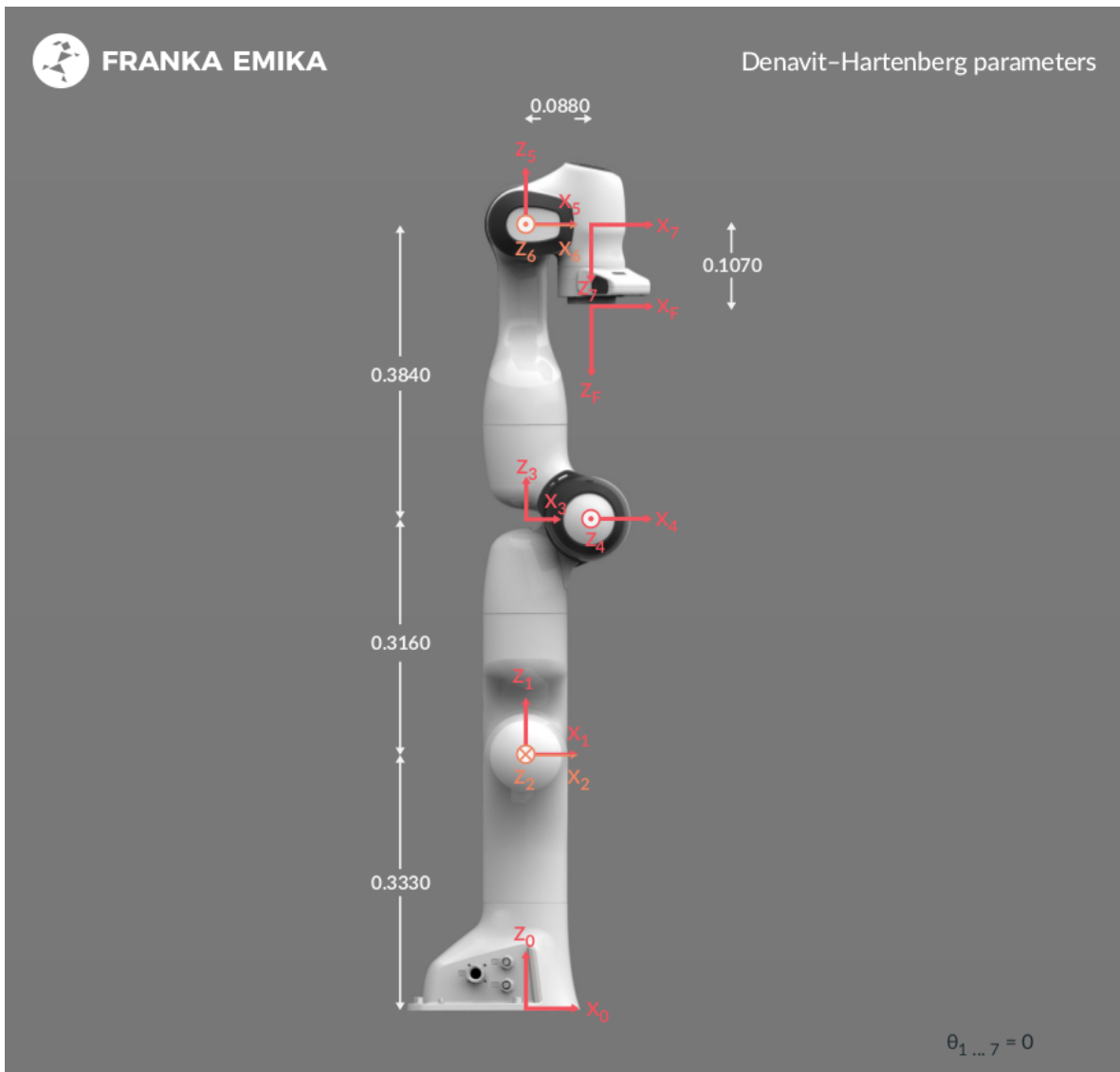


Figure 1: Robot Specifications

## 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  $\pi$  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.

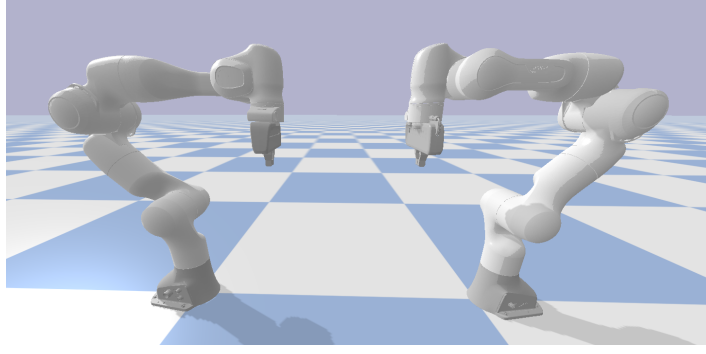


Figure 2: Initial position of the robots

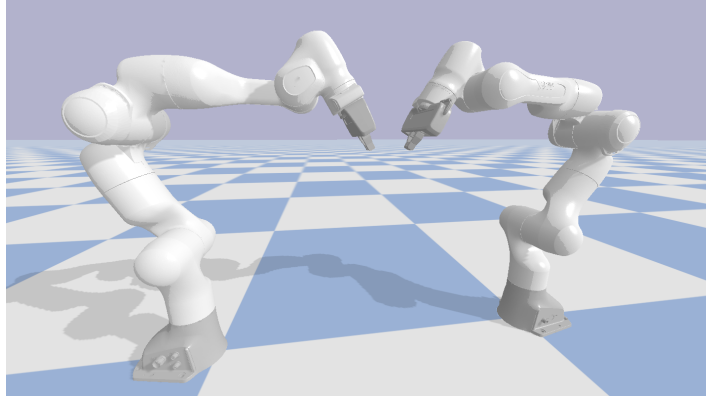


Figure 3: Relative orientation at the grasping position

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  ${}^wO_o$  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.

$${}^wO_o = [0.5, 0, 0.59]; \quad (1)$$

$$l_{obj} = 10 \text{ (cm)}. \quad (2)$$

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.  ${}^wO_g = [0.65, -0.35, 0.28]^T (m)$

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 list:**

- $\mathcal{A}_1$  - 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	
Bimanual rigid grasping constraint	E		3	
Move-To	E	3	4	

Table 2: 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.

### Tasks' Definition

- **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?

The Jacobian for the limits task is the identity matrix. For this task reference, we calculate the velocities in this way, where we constrain the joints to return within their limits. In the formula below,  $Jl_{\min_i}$  and  $Jl_{\max_i}$  represents the lower and upper joint limits of all joints  $i_{th}$  and  $q_i$  is the current of position of  $i_{th}$  joint.

$$\dot{q}_i = \left( \frac{Jl_{\min_i} + Jl_{\max_i}}{2} \right) - q_i$$

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

We took a reference point, denoted as  $O$  (which is the center point of the object in our case), and calculated the Jacobian for both of the arm tool, left arm tool and right arm tool, with respect to the  $< O >$ .

$$J_{\text{combined}} = [J_{l,o} - J_{r,o}]$$

Here,  $J_{l,o}$  refers to the Jacobian of the left arm tool with respect to  $< O >$  and  $J_{r,o}$  refers to the Jacobian of the right arm tool with respect to  $< O >$ . The negative sign accounts for the opposite direction of motion of the right arm relative to the left.

In the task reference, we aim to make the velocities of both arms equal, meaning that the difference in the end effector velocities should be zero in the context of the Bimanual Rigid Grasping Constraint task. This ensures the synchronization of both arms.

$$\begin{aligned} J_{o,a}\dot{q}_a &= J_{o,b}\dot{q}_b \\ [J_{o,a} - J_{o,b}]\dot{y} &= 0 \end{aligned}$$

The task reference should be zero in our case.

### 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  ${}^wO_o$  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.

$${}^wO_o = [0.5, 0, 0.59]^\top (m); \quad (3)$$

$$l_{obj} = 6 \text{ (cm)}. \quad (4)$$

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

$${}^wO_g = [0.60, 0.40, 0.48]^\top (m) \quad (5)$$

- (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.

We used the hierarchy given below for both arms, maintaining the same order.

**Actions list:**

- $\mathcal{A}_1$  - Move-to : This action is responsible for moving the robot without cooperation, meaning the robot will only follow joint limits and maintain a minimum altitude
- $\mathcal{A}_2$  - Co-operative move-to : In this action, we constrained both robots to move together while respecting each other's individual constraints
- $\mathcal{A}_3$  - Halt manipulator : In this action, we deactivate all tasks for both robots so that they reach the rest position and remain there

Task	Type	$\mathcal{A}_1$	$\mathcal{A}_2$	$\mathcal{A}_3$
Joint limits	I	1	1	
Minimum Altitude	I	2	2	
Move-To	E	3		
Cooperative Rigid Constraint move	E		3	

Table 3: 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.

**Tasks' definition**

- **Joint Limits :** This is a safety task in which we keep the joints within their range.
- **Minimum Altitude:** This is also a safety task in which we have restricted the robot to a specific distance. This means the robot's end effector will not go below that distance with respect to its base.
- **Cooperative Rigid Constraint Move:** In this task, we apply a coordination policy in which we provide velocities to both robots in such a way that they communicate with each other and move together. These velocities are calculated considering all the constraints.

### 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.