Cooperative Robotics

Authors: Alberto Grillo, Filippo Gandolfi
EMAILs: — filippo.gandolfi95@gmail.com
Date:

General notes

- Exercises 1-4 are done with the ROBUST matlab main and unity visualization tools. Exercises 5-6 are done with the DexROV matlab main and unity visualization 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.
- 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.

1 Exercise 1: Implement a “Safe Waypoint Navigation” Action.

1.1 Adding a vehicle position control objective

Initialize the vehicle far away from the seafloor. An example position could be

$$\begin{bmatrix} 10.5 & 35.5 & -36 & 0 & 0 & \pi/2 \end{bmatrix}^\top$$

Give a target position that is also sufficiently away from the seafloor, e.g.,

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

Goal: Implement a vehicle position control task, and test that the vehicle reaches the required position and orientation.

1.1.1 Q1: Report the hierarchy of task used and their priorities. What is the Jacobian relationship for the Vehicle Position control task? How was the task reference computed?

The **hierarchy of the tasks**, in order of priority:

- Manipulability
- Horizontal Attitude
- Vehicle Position

We use the following notation to characterize each task:

- R/NR, reactive or non-reactive.
- I/E, inequality or equality.
- C/S/P/AD/O, constraint, safety, prerequisite, action-defining, optimization.

Manipulability [R, I, P], it’s used in order to prevent dangerous position of the robot arm.
Horizontal Attitude [R, I, S], it’s fundamental for being sure the robot it’s parallel with the seafloor.
Vehicle Position [R, I, AD], what the system really do. Action-defining task, so, lower hierarchy. This is the first task we implement, the goal is to perform velocity to enable the robot to reach the desired position.

The **Jacobian** for the vehicle position control task is the following one:

$$\boldsymbol{J}_v = \begin{bmatrix} \underset{6\times 7}{\mathbf{0}} & \begin{matrix} \underset{3\times 3}{\mathbf{0}} & \underset{3\times 3}{{}^w\boldsymbol{R}_v} \\ \underset{3\times 3}{{}^w\boldsymbol{R}_v} & \underset{3\times 3}{\mathbf{0}} \end{matrix} \end{bmatrix} \tag{1}$$

We compute the **task reference** applying the Cartesian error between the vehicle frame and the goal frame, both projected on the world frame.
The misalignment between the two is computed by the Unit Vector Lemma.
We multiply the final vector by a gain (k) factor.

$${}^w\dot{\boldsymbol{x}}_{v\_g} = k\,({}^w\boldsymbol{x}_g -{}^w\boldsymbol{x}_v) \tag{2}$$

Code for Jacobian definition and computation:
`ComputeJacobian -> Vehicle Position Task`
Code for computing distance and misalignment:
`simulation script -> CartError`
`simulation script -> UniteVersorLemma`

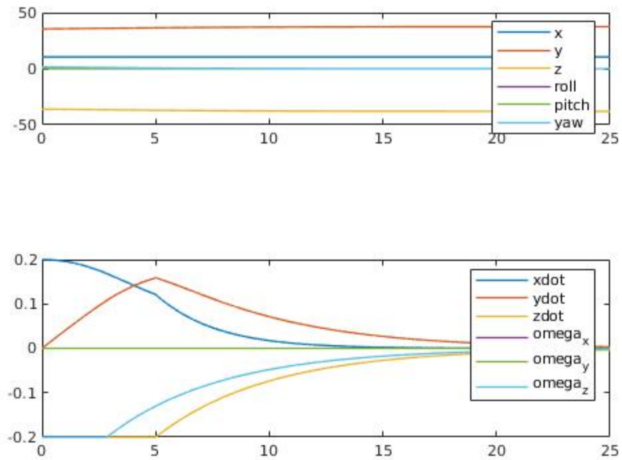

Figure 1: Resulted Graph for the first task

1.1.2 Q2: What is the behaviour if the Horizontal Attitude is enabled or not? Try changing the initial or target orientation in terms of roll and pitch angles. Discuss the behaviour.

By default the Horizontal Attitude is enable and has higher priority than the vehicle position task. So, if the target position has pitch and roll different from zero, the robot don't perform that "vertical" position (Figure 3).
When the horizontal attitude is not enabled, the UVMS trying to find the alignment between its frame and the goal one, without taking (of course) into account to stay in parallel with the seafloor. If the target position has pitch or roll, the robot will try to perform them.

1.1.3 Q3: Swap the priorities between Horizontal Attitude and the Vehicle Position control task. Discuss the behaviour.

The problem is the same as before. This time with the priorities switched, if the target, again, as roll or pitch, the robot will try to achieve this orientation. So, if we switched the position we will have the same performance as if we disable the Horizontal Attitude task.
In the end, it's not usefull to switch the position and it's wrong because the Horizontal Attitude is a Safety Task, so it's, every time, more important (so high priority)than a AD task.

1.1.4 Q4: What is the behaviour if the Tool Position control task is active and what if it is disabled? Which of the settings should be used for a Safe Waypoint Navigation action?

If the Tool Position Control task is enable, the vehicle could not reach the desidered position. If it's enable ad it as higher priority than the Vehicle Position Control task, the robot try to achieve the goal position for the arm, not the ones referred to the vehicle it self.
For this reason we split the goal position in two different, the first for the body of the vehicle and another one for the tool. For this particular exercise we didn't use a arm goal position.
However, we continue to use the Manipulability task, in order to maintain a safe position for the arm.

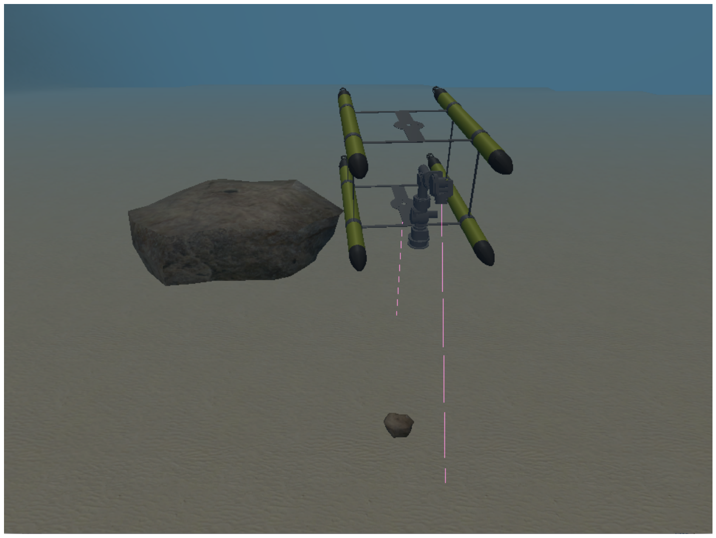

Figure 2: Image of the SIM, Horizontal attitude enable, Robot is parallel with the seafloor

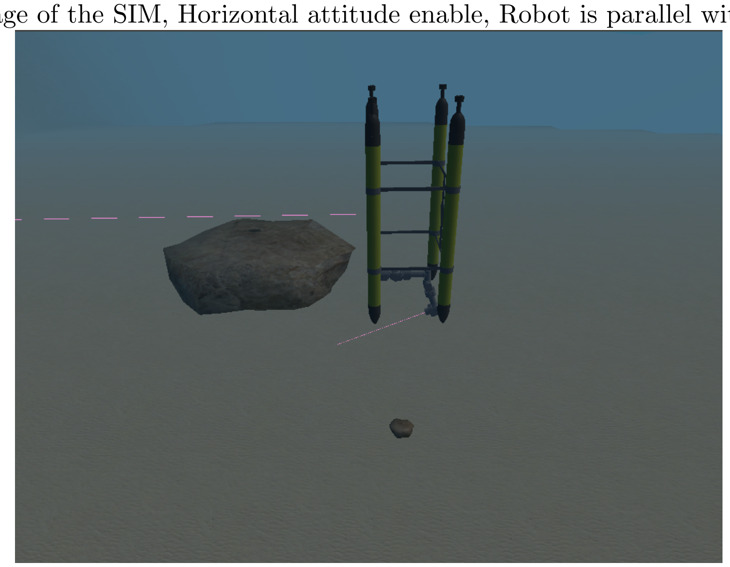

Figure 3: Image of the SIM, Horizontal Attitude not enable, Robot is no more parallel with the seafloor

1.2 Adding a safety minimum altitude control objective

Initialize the vehicle at the position:

$$\begin{bmatrix}48.5 & 11.5 & -33 & 0 & 0 & -\pi/2\end{bmatrix}^\top$$

Choose as target point for the vehicle position the following one:

$$\begin{bmatrix}50 & -12.5 & -33 & 0 & 0 & \pi/2\end{bmatrix}^\top$$

Goal: Implement a task to control the altitude from the seafloor. Check that at all times the minimum distance from the seafloor is guaranteed.

1.2.1 Q1: Report the hierarchy of task used and their priorities. Comment how you choose the priority level for the minimum altitude.

The new hierarchy of tasks is:

- Minimum altitude
- Manipulability
- Horizontal attitude
- Vehicle position

The news task implemented:
Minimum Altitude [R,I,S], it has high priority because it's a safety task, UVM must avoid crashes with the seafloor.

1.2.2 Q2: What is the Jacobian relationship for the Minimum Altitude control task? How was the task reference computed?

We need to control the movement along the z axis, so we use the same Jacobian of the vehicle position, but selecting only the component related to the z axis.

$$\boldsymbol{J}_{mav} = \begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1\end{bmatrix} \begin{bmatrix} \underset{6\times7}{\mathbf{0}} & \begin{matrix} \underset{3\times3}{\mathbf{0}} & \underset{3\times3}{{}^{w}\boldsymbol{R}_v} \\ \underset{3\times3}{{}^{w}\boldsymbol{R}_v} & \underset{3\times3}{\mathbf{0}} \end{matrix} \end{bmatrix} \tag{3}$$

Note that in order to select the desired component we used the vector [0 0 0 0 0 1] because the control variable uses a the convention [R P Y X Y Z].
Task reference is computed as the difference between the minimum distance given from the operator and the measured distance vector projected on the world frame.

$${}^{w}\dot{\boldsymbol{x}}_{mav} = k((d_{limit} + \Delta) -^{w} d_{sensor})) \tag{4}$$

- k is the control gain.
- d_{limit} is the desired minimum distance from the seafloor.
- Δ is the safety distance at which the activation of the task triggers.
- ${}^{w}d_{sensor}$ is the distance vector measured by the sensor and projected on the world frame.

Code for Exercise:
`Compute Task Reference -> MAV Control`

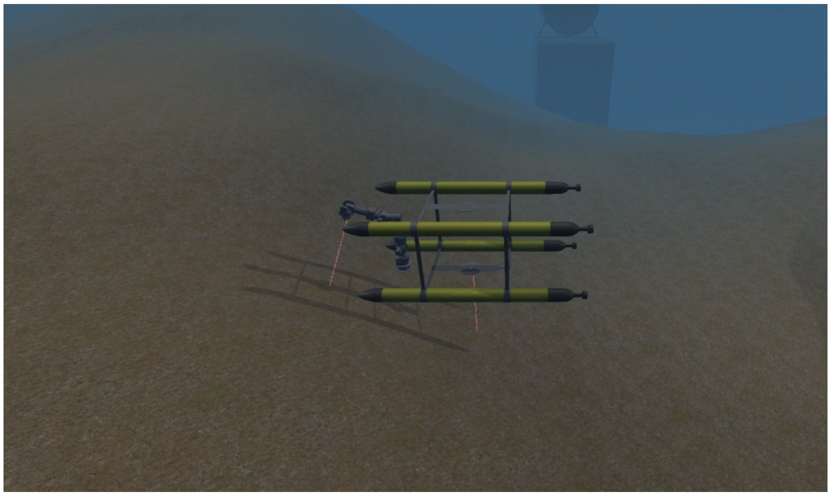

Figure 4: Image of the SIM, Minimum Altitude Vehicle set to 0.5m, Robot is following the seafloor

1.2.3 Q3: Try imposing a minimum altitude of 1, 5, 10 m respectively. What is the behaviour?

To achieve this task we perform the different minimum altitude, all with the same k gain (equal to 0.2), and the same Δ for the inequality nature of the problem (equal to $0.5m$).

- **10 m:** It seems no a reasonable value. The UVMS has initial position below this threshold, so it start floating to the surface very quickly (it is possible to notice this huge vertical velocity in the graph (Figure 5), $\dot{z}$). Of course, the task is not accomplished because the target position is so far.
- **5 m:** The same as before, the activation is active at the first instant because again the initial position is below the minimum altitude. This control task is a safety one, so every time the UVMS is under the minimum altitude is active and it has the highest priority. Again, it seems no reasonable to perform such big gap with the seafloor (Reported graph, Figure 6).
- **1 m:** This time the problem is the opposite. The control task is no more active at the starting instant. The UVMS starts to accomplish the Vehicle Position Control task, when the seafloor present an hill the UVMS start to follow the pattern (Figure 7). However, there are some troubles during the navigation because the "noise" of the robot, with only 1 meter of distance (and no Δ that could prevent this), could touch the soil because the sensor it's on the center of the vehicle.

1.2.4 Q4: How was the sensor distance processed?

${}^{v}d_{sensor}$ is the distance vector measured by the sensor, so we need to projected it on the world frame, to do this we apply a rotation:

$$ {}^{w}\boldsymbol{d}_{sensor} = {}^{w}\boldsymbol{R}_{v}\, {}^{v}\boldsymbol{d}_{sensor} \tag{5} $$

Again we are only interested on the z axis. The $\boldsymbol{d}_{sensor}$ are the vectors build as: $\begin{bmatrix} 0 & 0 & d_{seafloor} \end{bmatrix}^{\top}$

```
Code for Exercise:
TO DO
TO DO
```

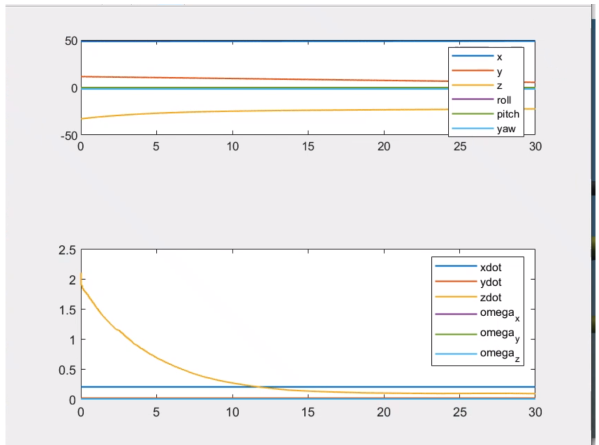

Figure 5: Resulted graph during the Minimum Altitude Vehicle task imposed 10m distance

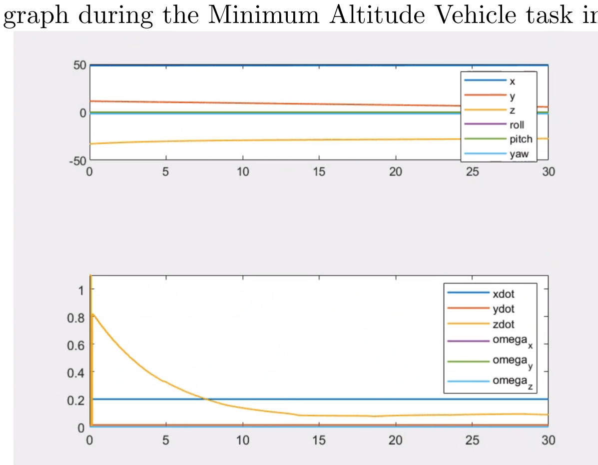

Figure 6: Resulted graph during the Minimum Altitude Vehicle task imposed 5m distance

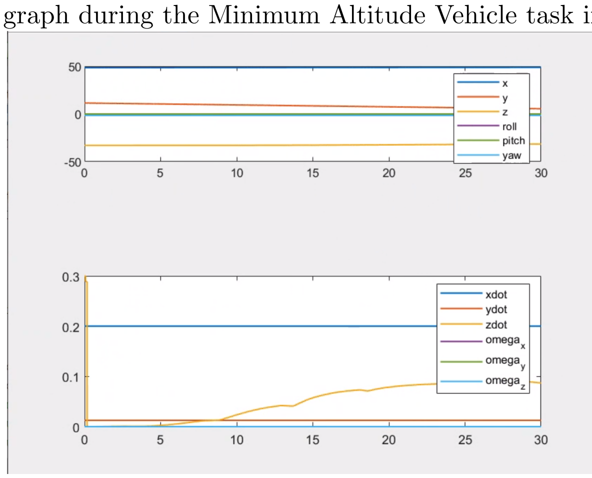

Figure 7: Resulted graph during the Minimum Altitude Vehicle task imposed 1m distance

2 Exercise 2: Implement a Basic "Landing" Action.

2.1 Adding an altitude control objective

Initialize the vehicle at the position:

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

Goal: add a control task to regulate the altitude to zero.

2.1.1 Q1: Report the hierarchy of task used and their priorities. Comment how you choose the priority level for the altitude control task.

The new hierarchy of tasks is:

- Manipulability
- Horizontal attitude
- Landing

Landing [R, E, AD], is the new task. It has the same priority as the Vehicle position task. The task Minimum Altitude is not enable now, because (of course) we need, only, to land.

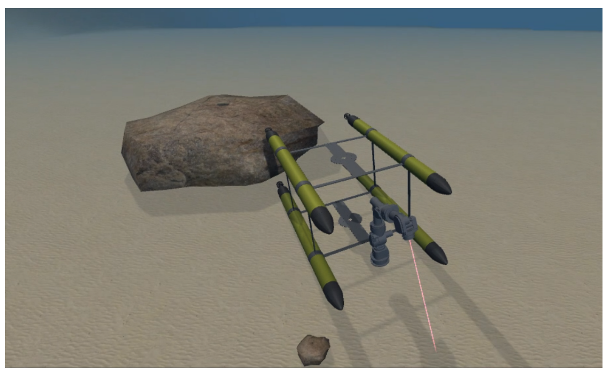

Figure 8: Implemented basic "Landing" action

2.1.2 Q2: What is the Jacobian relationship for the Altitude control task? How was the task reference computed?

$$\boldsymbol{J}_{land} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \underset{6\times 7}{\mathbf{0}} & \begin{matrix} \underset{3\times 3}{\mathbf{0}} & \underset{3\times 3}{{}^{w}\boldsymbol{R}_v} \\ \underset{3\times 3}{{}^{w}\boldsymbol{R}_v} & \underset{3\times 3}{\mathbf{0}} \end{matrix} \end{bmatrix} \tag{6}$$

As before the Jacobian take into account only the z component (using the pre-multiplication with the [0 0 0 0 0 1] matrix).

The task reference is computed as the difference between the seafloor and the measured distance projected on the world frame:

$$^{w}\dot{\boldsymbol{x}}_{land} = k(seafloor - ^{w} d_{sensor})) \tag{7}$$

- k is the control gain.
- $seafloor$ is simply 0 (where is the seafloor in which we will land).
- $^{w}d_{sensor}$ is the distance vector measured by the sensor and projected on the world frame.

Code for Exercise:
```
TO DO
TO DO
```

2.1.3 Q3: how does this task differs from a minimum altitude control task?

There are two major difference, the first one is that this task it isn't a safety task, it's a Action-Defining task. The Minimum Altitude it's used to avoid crash with the seafloor, the Landing task it's like the Vehicle Position task, and, as this one, it has low priority. The second big difference is that the Minimum altitude is an Inequality task rather than the Landing that is and equality. For this reason, as visible on the

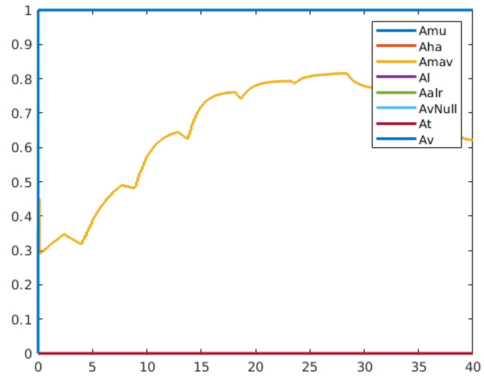

Figure 9: Graph MAV

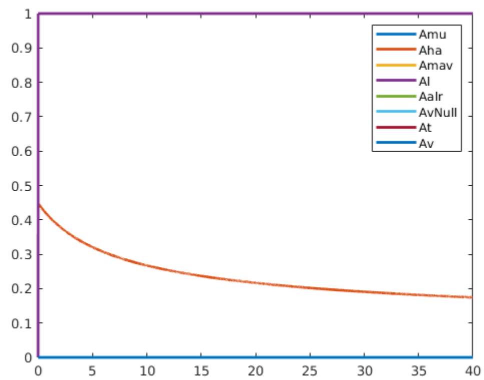

Figure 10: Graph Landing

2.2 Adding mission phases and change of action

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:

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

When the position has been reached, land on the seafloor using the basic "landing" action.

2.2.1 Q1: Report the unified hierarchy of tasks used and their priorities.

Control Task	Action A	Action B
MAV	Active	Inactive
Manipulabity	Active	Active
Horizonatal Attitude	Active	Active
Vehicle Position	Active	Inactive
Landing	Inactive	Active

Code for Exercise:
```
TO DO
TO DO
```

2.2.2 Q2: How did you implement the transition from one action to the other?

From Action A to Action B, we compute the cartesian error between the goal position and the actual. When the error is below our threshold (0.15m) the mission phase changes.

When we want to Inactivate a running task, we perform a Decreasing Bell Shape function in order to perform a smooth transition (like Minimum Altitude and Vehicle Position task, both Inactivated in the same method). In the same way, we also perform an Increasing Bell Shape function to perform a smooth activation of the Landing task. We perform the bell shaped function based on the mission phase time, this help us to fix the time when we switch from an Action to the other one, and perform shape from 0 to 0.5 seconds, to avoid spike to the motors.

We found this more adeguate that the previous attempt, in which we performed the shape based on the error distance.

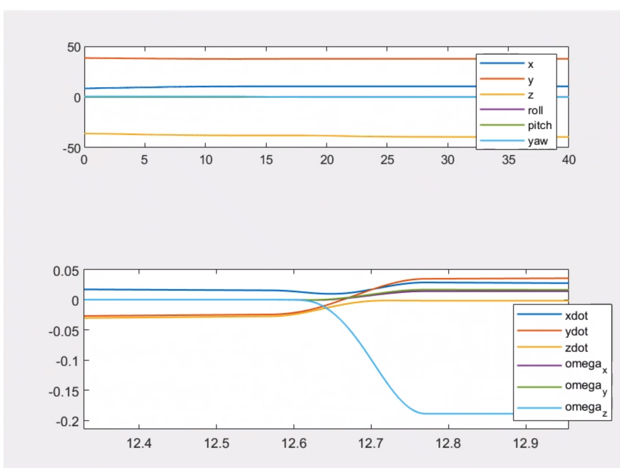

Figure 11: Zoom of the smooth change

3 Exercise 3: Improve the "Landing" Action

3.1 Adding an alignment to target control objective

If we use the landing action, there is no guarantee that we land in from of the nodule/rock. We need to add additional constraints to make the vehicle face the nodule. The position of the rock is contained in the variable `rock_center`.

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.

Goal: Add an alignment task between the longitudinal axis of the vehicle (x axis) and the nodule target. 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.

3.1.1 Q1: Report the unified hierarchy of tasks used and their priorities. Comment the behaviour.

Control Task	Action A	Action B	Action C
MAV	Active	Active	Inactive
Manipulabity	Active	Active	Active
Horizonatal Attitude	Active	Active	Active
Vehicle Position	Active	Inactive	Inactive
Alignment to the rock	Inactive	Active	Active
Landing	Inactive	Inactive	Active

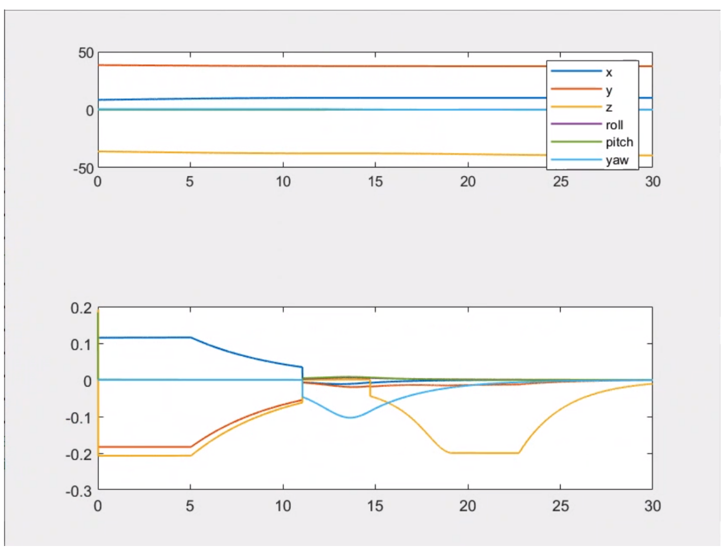

Figure 12: Final graph exercise

Code for Exercise:
```
TO DO
TO DO
```

3.1.2 Q2: What is the Jacobian relationship for the Alignment to Target control task? How was the task reference computed?

COPIARE JACOBIANA

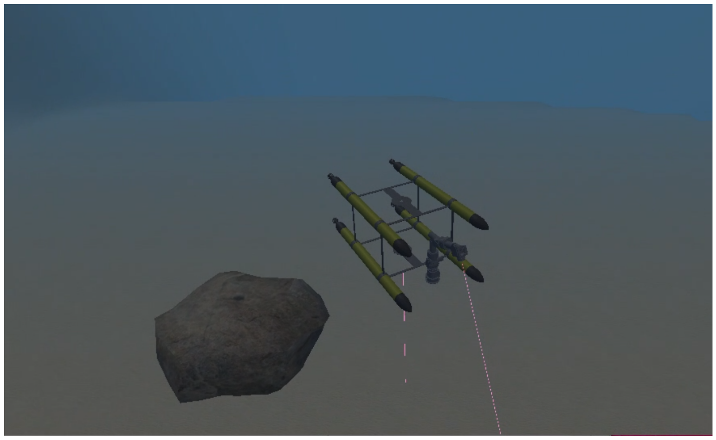

Figure 13: Action A: Navigation till the target position

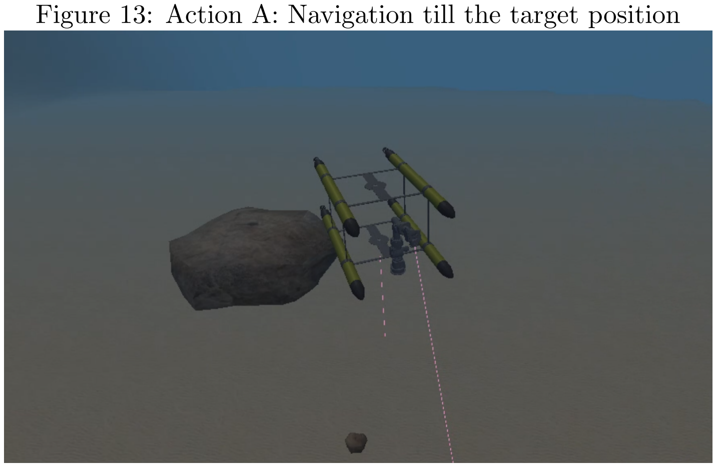

Figure 14: Action B: Alignment with the rock center

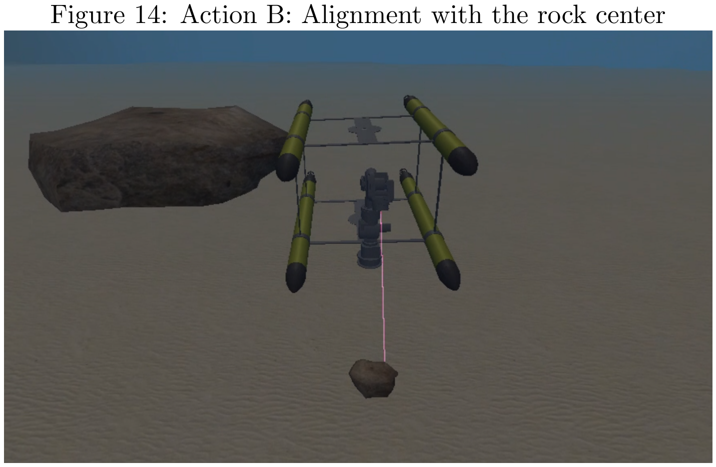

Figure 15: Action C: Landing, final action

3.1.3 Q3: Try changing the gain of the alignment task. Try at least three different values, where one is very small. What is the observed behaviour?

3.1.4 Q4: 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.

Ci arriva ma il vehicle lo aiuta un po' invece che sta femmo. A quanto pare serve una non-reactive task che lo tengo fermo li .

4 Exercise 4: Implementing a Fixed-base Manipulation Action

4.1 Adding non-reactive tasks

To manipulate as a fixed based manipulator, we need to constraint the vehicle to not move, otherwise the tool frame position task will make the vehicle move.

Goal: Add a constraint task that fixes the vehicle velocity to zero. Land on the seafloor. Try reaching the rock position with the end-effector, and observe that the vehicle does not move.

4.1.1 Q1: Report the unified hierarchy of tasks used and their priorities. At which priority level did you add the constraint task?

Control Task	Action A	Action B	Action C	Action D
Vehicle Null Velocity	Inactive	Inactive	Inactive	Active
MAV	Active	Active	Inactive	Inactive
Manipulabity	Active	Active	Active	Active
Horizontal Attitude	Active	Active	Active	Active
Vehicle Position	Active	Inactive	Inactive	Inactive
Alignment to the rock	Inactive	Active	Active	Inactive
Landing	Inactive	Inactive	Active	Inactive

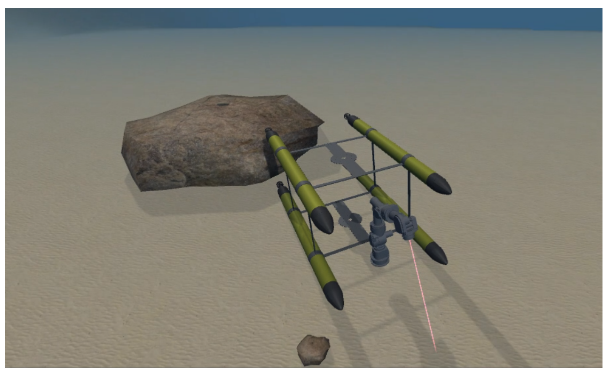

Figure 16: Implemented basic "Landing" action

4.1.2 Q2: What is the Jacobian relationship for the Vehicle Null Velocity task? How was the task reference computed?

$$\boldsymbol{J}_{vehNull} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \underset{6\times 7}{\mathbf{0}} & \begin{matrix} \underset{3\times 3}{\mathbf{0}} & \underset{3\times 3}{{}^{w}\boldsymbol{R}_{v}} \\ \underset{3\times 3}{{}^{w}\boldsymbol{R}_{v}} & \underset{3\times 3}{\mathbf{0}} \end{matrix} \end{bmatrix} \tag{8}$$

4.1.3 Q3: Suppose the vehicle floating, i.e. not landed on the seafloor. What would happen, if due to currents, the vehicle moves?

4.2 Adding a joint limit task

Let us now constrain the arm with the actual joint limits. The vector variables `uvms.jlmin` and `uvms.jlmax` contain the maximum and minimum values respectively.

Goal: Add a joint limits avoidance task. Land on the seafloor. Try reaching the rock position with the end-effector, and observe that the vehicle does not move and that all the joints are within their limits.

4.2.1 Q1: Report the unified hierarchy of tasks used and their priorities. At which priority level did you add the constraint task?

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

$$\boldsymbol{J}_{jointLimit} = \begin{bmatrix} \underset{7\times 7}{\boldsymbol{I}} & \underset{7\times 6}{\boldsymbol{0}} \end{bmatrix} \tag{9}$$

5 Exercise 5: Floating Manipulation

5.1 Adding an optimization control objective

Use the DexROV simulation for this exercise.

The goal is to try to optimize the joint positions, if possible, to keep the first four joints in a "preferred shape", represented by the following vector

$$\begin{bmatrix} -0.0031 & 1.2586 & 0.0128 & -1.2460 \end{bmatrix}^\top$$

Goal: Add an optimization objective to keep the first four joints of the manipulator in the preferred shape. Observe the behaviour with and without the task

5.1.1 Q1: Report the unified hierarchy of tasks used and their priorities. At which priority level did you add the optimization task?

5.1.2 Q2: What is the Jacobian relationship for the Joint Preferred Shape task? How was the task reference computed?

$$\boldsymbol{J}_{vehNull} = \begin{bmatrix} -0.0031 & 1.2586 & 0.0128 & -1.2460 \end{bmatrix}^\top \begin{bmatrix} \underset{4\times 4}{\boldsymbol{I}} & \underset{4\times 9}{\boldsymbol{0}} \end{bmatrix} \tag{10}$$

5.1.3 Q3: What is the difference between having or not having this objective?

Figure 17: Preferred Shape Task Active

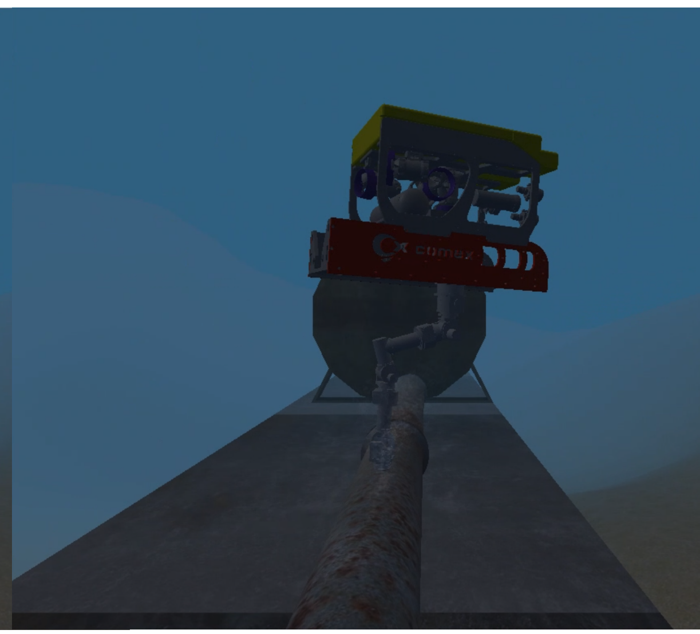

Figure 18: Preferred Shape Task Inactive

5.2 Adding mission phases

Let us now structure the mission in more than one phase. In the first phase, exploit the previous exercises, and implement a safe waypoint navigation. Move the vehicle to a location close to the current defined end-effector goal position, just slightly above it. Then, trigger a change of action and perform floating manipulation.

Goal: introduce mission phases in the floating manipulation scenario. Observe the difference.

5.2.1 Q1: Report the unified hierarchy of tasks used and their priorities. Which task is active in which phase/action?

5.2.2 Q2: What is the difference with the previous simulation (still in exercise 5), where only one action was used?

6 Exercise 6: Floating Manipulation with Arm-Vehicle Coordination Scheme

6.1 Adding the parallel arm-vehicle coordination scheme

Let us now see how the two different subsystems (arm and vehicle) can be properly coordinate. Introduce in the simulation a sinusoidal velocity disturbance acting on the vehicle, and assume the actual vehicle velocity measurable. To do so, add a constant (in the inertial frame) velocity vector to the reference vehicle velocity before integrating it in the simulator.

Goal: modify the control part to implement the parallel arm-vehicle coordination scheme. Observe that, even with a disturbance acting on the vehicle, the end-effector can stay in the required constant position.

6.1.1 Q1: Which tasks did you introduce to implement the parallel coordination scheme?

6.1.2 Q2: What happens if the sinusoidal disturbance becomes too big?