Cooperative Robotics

Authors: Piccinini Davide, Porta Francesco and Gotelli Andrea EMAILs: 4343879@studenti.unige.it, 4404040@studenti.unige.it Date: 21/12/2020

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.

Use the following template when you need to discuss the hierarchy of tasks of a given action or set of actions:

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 A	I	1		1
Task B	I	2	1	
Task C	E		2	2

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: What is the Jacobian relationship for the Vehicle Position control task? How was the task reference computed?

The Jacobian relationship for the Vehicle Position control task need to take into account the linear velocities of the vehicle. In the configuration vector y the velocities are expressed in the vehicle frame so the Jacobian has to project them in the reference frame. In fact, the task reference was computed taking into account the position of the goal and the vehicle expressed in the reference frame.

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.

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

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? Report the final hierarchy of tasks (and their priorities, see the template table in the introduction) which makes up the Safe Waypoint Navigation action.

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 new hierarchy of tasks of the Safe Waypoint Navigation and their priorities. Comment how you choose the priority level for the minimum altitude.

1.2.2 Q2: What is the Jacobian relationship for the Minimum Altitude control task? Report the formula for the desired task reference generation, and the activation thresholds.

1.2.3 Q3: Try imposing a minimum altitude of 1, 5, 10 m respectively. What is the behaviour? Does the vehicle reach its final goal in all cases?

1.2.4 Q4: How was the sensor distance processed to obtain the altitude measurement? Does it work in all cases or some underlying assumptions are implicitly made?

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 to implement the Landing Action. Comment how you choose the priority level for the altitude control task.

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

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

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.

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

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 hierarchy of tasks used and their priorities in each action. Comment the behaviour.

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

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? Could you devise a solution that is gain-independent guaranteeing that the landing is accomplished aligned to the target?

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. If, after landing, the nodule is not in the manipulator's workspace, how would you solve this problem to guarantee it?

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 hierarchy of tasks used and their priorities in each action. At which priority level did you add the constraint task?

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

4.1.3 Q3: Suppose that the vehicle is 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 hierarchy of tasks used and their priorities in each action. At which priority level did you add the joint limits task?

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

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 hierarchy of tasks used and their priorities in each action. 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?

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

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?

In order to obtain the desired behavior we have introduced two kinds of disturbances, one linear and the other as angular velocity disturbance, expressed in the inertial frame. The vehicle suffers from these velocities, as they are added before integration.

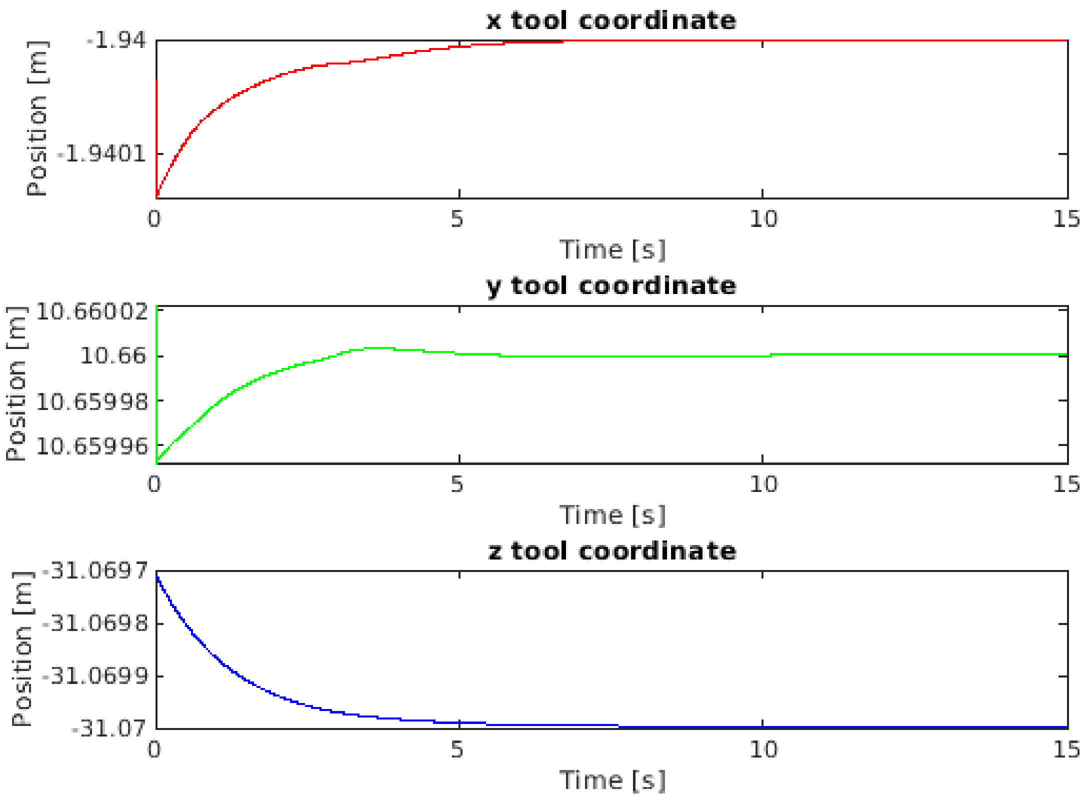
```
%        Adding the disturbances
disturb = [0 0.025 0]';
disturb_ang = [0 0 0]';
disturb_ang = disturb_ang*0.5*sin(0.5*t*2*pi);
uvms.p_dot(1:3) = uvms.wTv(1:3,1:3)*disturb;
uvms.p_dot(4:6) = uvms.wTv(1:3,1:3)*disturb_ang;

% beware: p_dot should be projected on <v>
uvms.p = integrate_vehicle(uvms.p, uvms.p_dot, deltat);
```

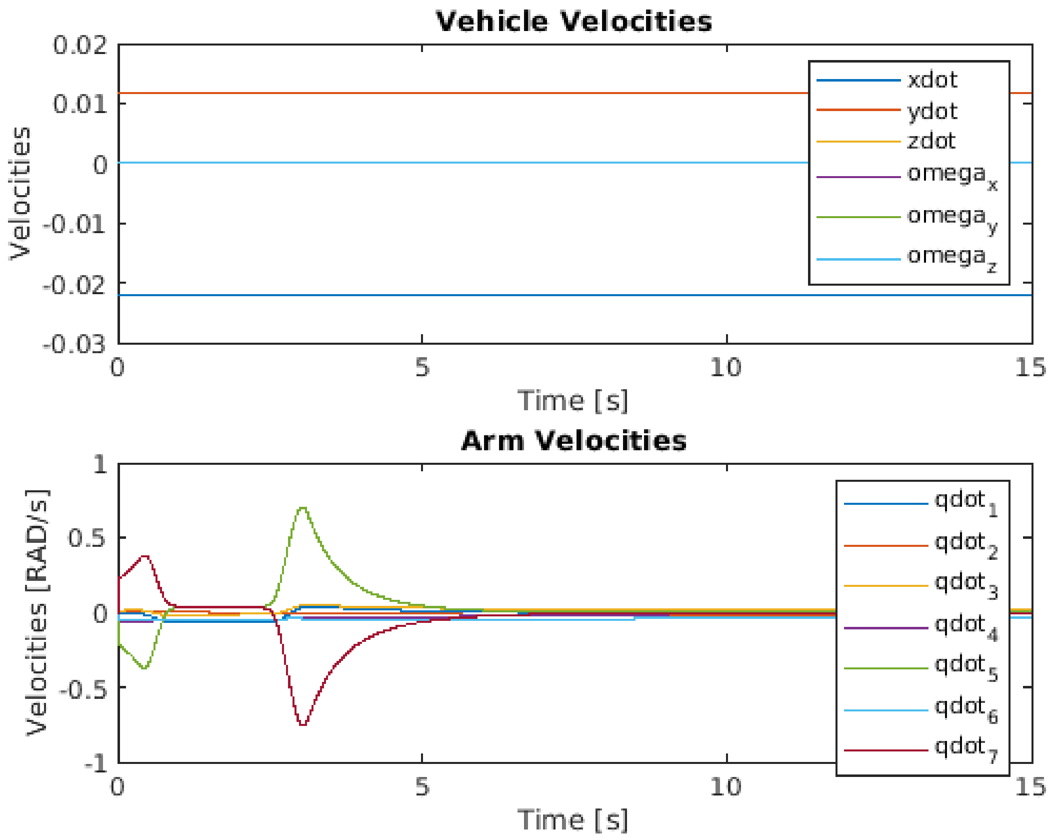
This task objective is to keep the end effector position stable while the vehicle moves. As a result, the arm and the vehicle result as two different systems; specifically, the first could cope with incugruences in the second, but only in a limited way. In order to cope with this disturbances we defined a new task called: armVehiCoord. This task is placed with the highest hierarchy, limiting the manifold of solution to the ones accounting for the vehicle velocities. In fact, in order to decouple the two systems, we need to feed the arm with the vehicle velocities, the jacobian matrix accounts only for the vehicle velocities, disregarding the arm velocities. The task takes as a reference the velocities of the vehicle, as this is the input the arm has to compensate. Concerning the activation funtion, this is defined as an identity matrix, as the task is an equality one.

6.1.2 Q2: Show the plot of the position of the end-effector, showing that it is constant. Show also a plot of the velocities of the vehicle and of the arm.

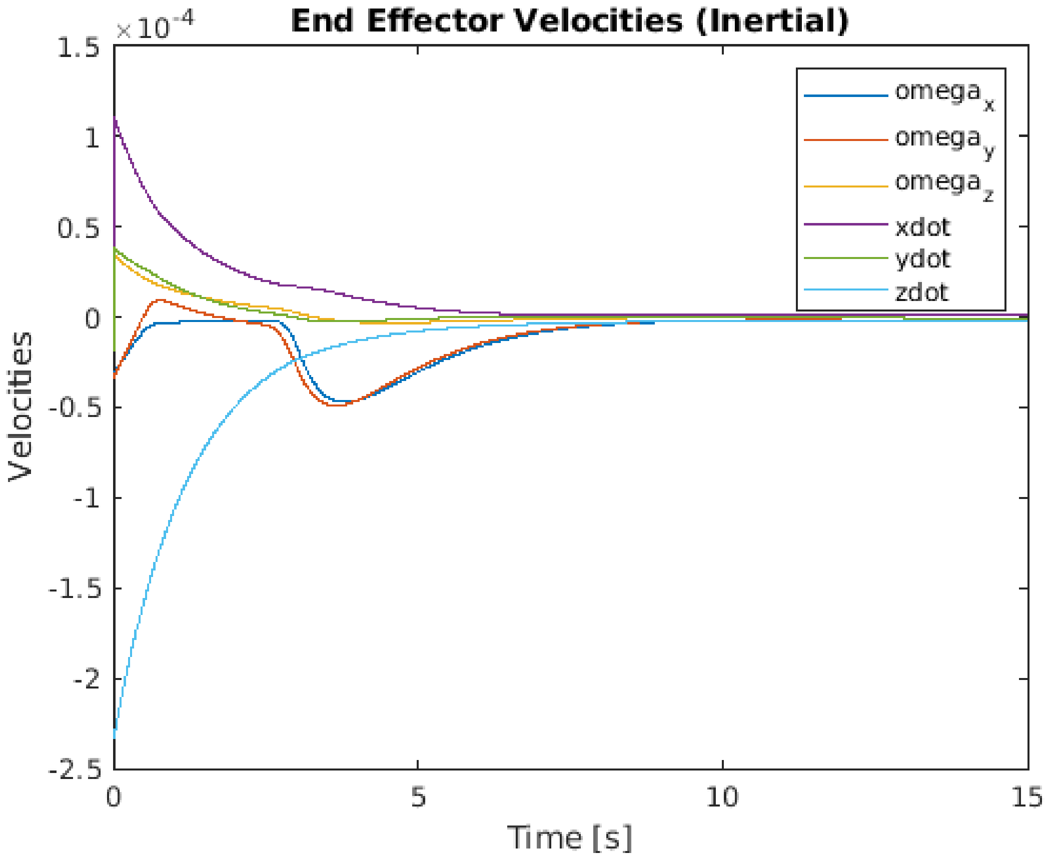
In the following images, the end effector position is plotted, showing the three cartesian components. The scale of the plots clearly states that the end effector position could be assumed as constant, as it only suffers from oscillation smaller than a millimiter.



The following image compares the vehicle velocities, expressed in the vehicle frame, and the velocites of the arm joints. The vehicle linear velocities show the effect of the linear disturbance, which brings a change in the vehicle velocities on x and y. The arm velocities are the resultant for the control of the end effector in its original position.



In this last image, we would like to show the evolution of the end effector velocities expressed in the inertial frame. The velocities start with a nonzero infinitesimal value, but they converge to zero in few seconds.

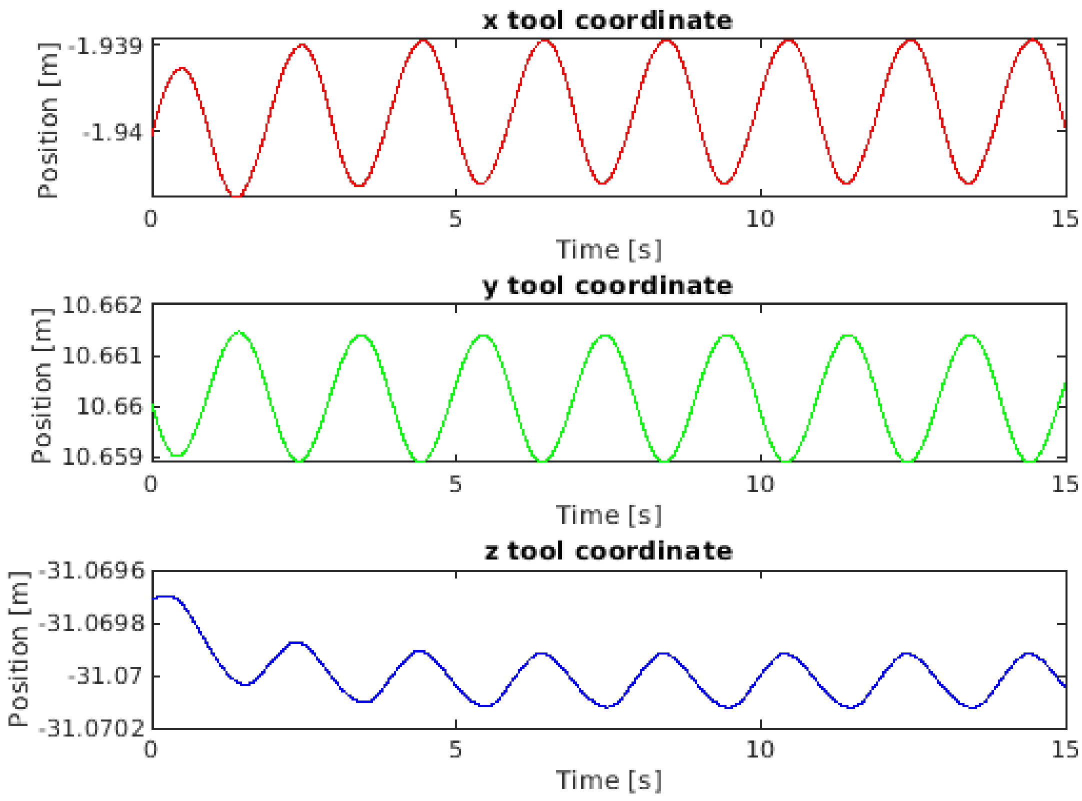


6.1.3 Q3: What happens if the sinusoidal disturbance becomes too big? Try increasing the saturation value of the end-effector task if it is too low.

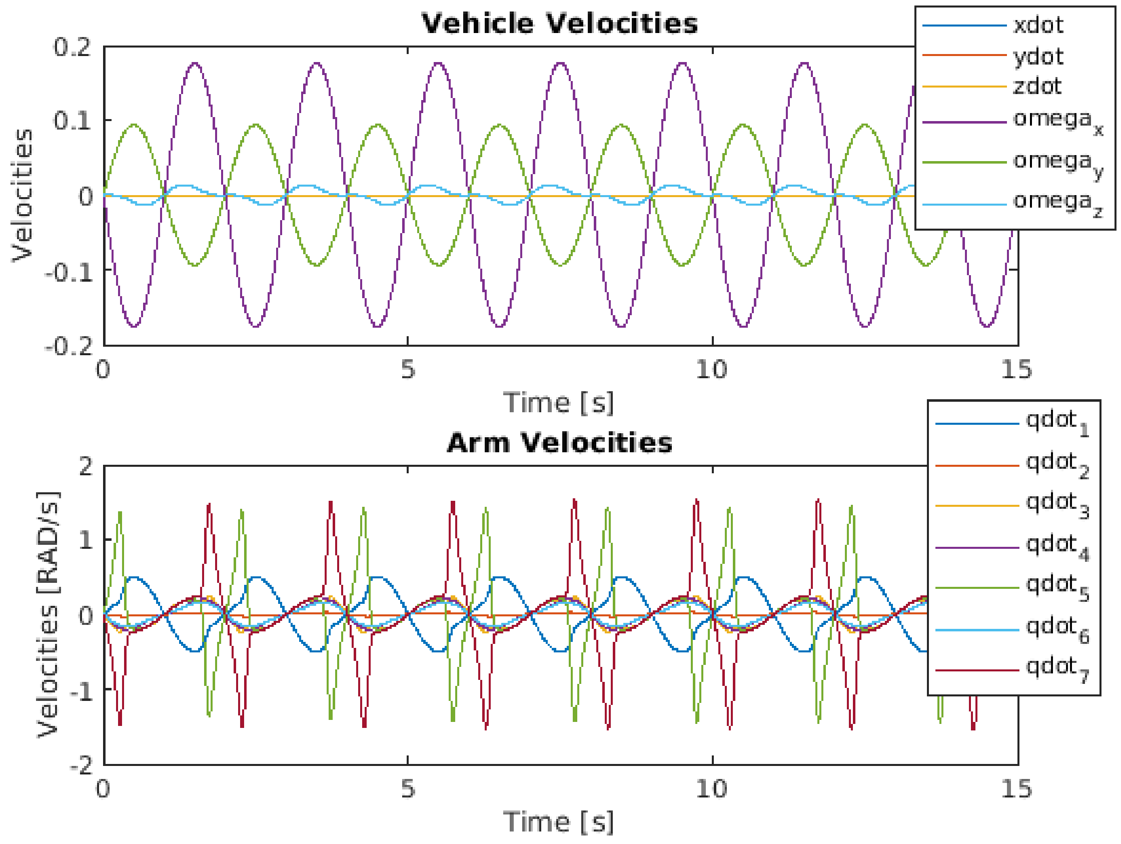
If we add a sinusoidal disturbance, it deteriorates the ability of keeping the end effector still. The disturbance is added as described below.

```
%         Adding the disturbances
disturb = [0  0.0  0]';
disturb_ang = [0  1  0]';
disturb_ang = disturb_ang*0.2*sin(0.5*t*2*pi);
uvms.p_dot(1:3) = uvms.wTv(1:3,1:3)*disturb;
uvms.p_dot(4:6) = uvms.wTv(1:3,1:3)*disturb_ang;
```

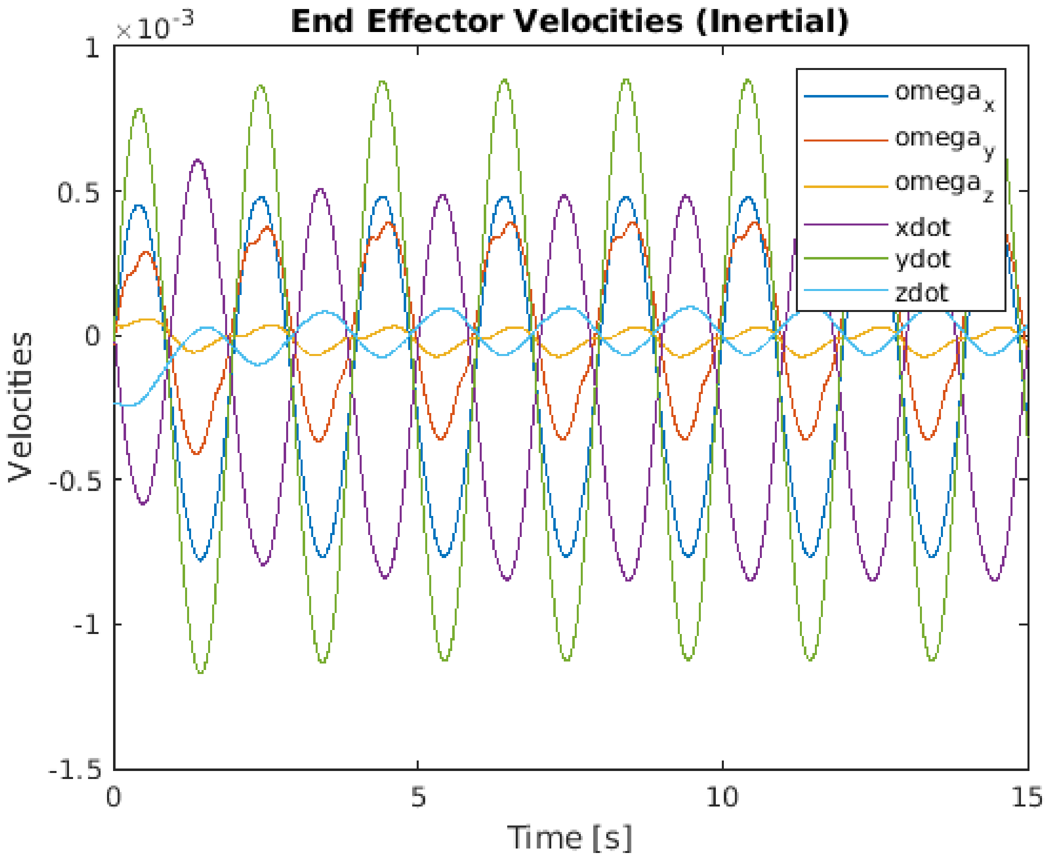
However, as it can be seen in the following images, the end effector does not moves much. The following figure shows that the oscillations of the tool position are of the order of one millimiter, which can be considered as a acceptable result.



The following images compares the vehicle velocities with the joint rotational speed.



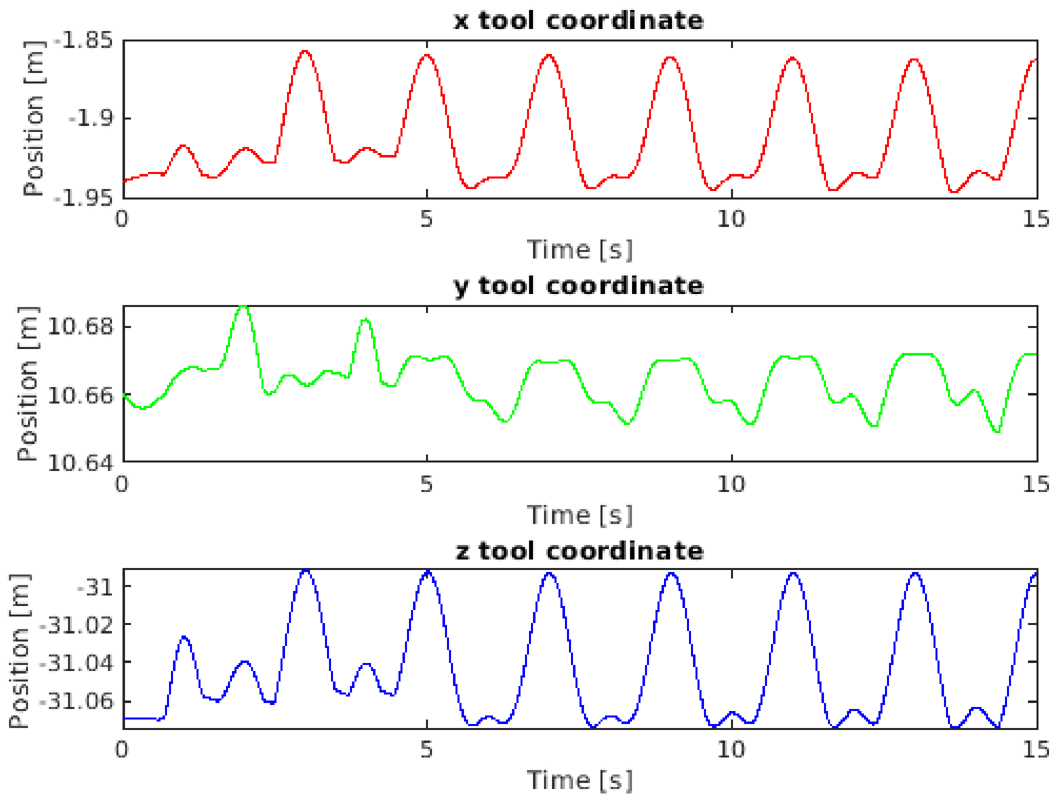
Finally, this last image shows that the end effector velocities remains reasonable small.

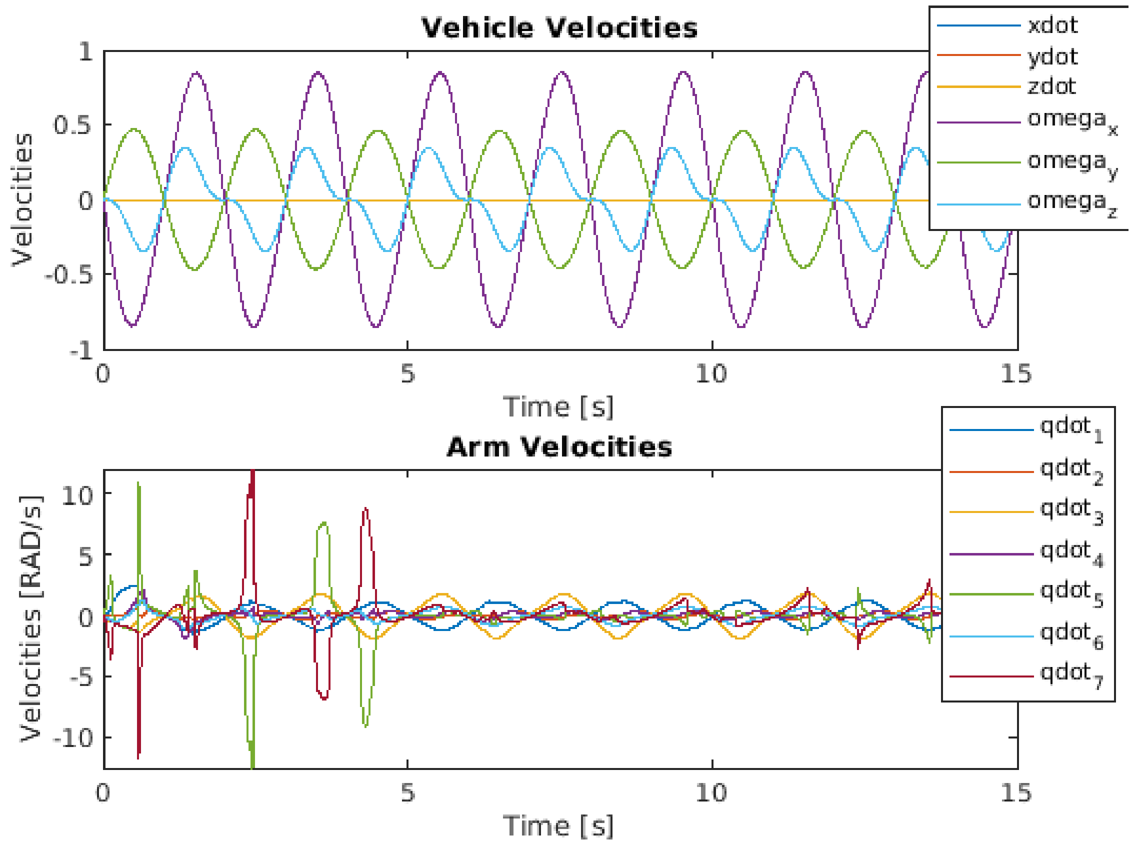


However, if we increase the disturbance too much, the arms cannot cope with the disturbance in the vehicle. As a result, the displacement of the end effer are not negletable. Even if we increase the gain and the saturation of the tool control task, this is not sufficient to cancel the disturbannce. The disturbance is added as described below.

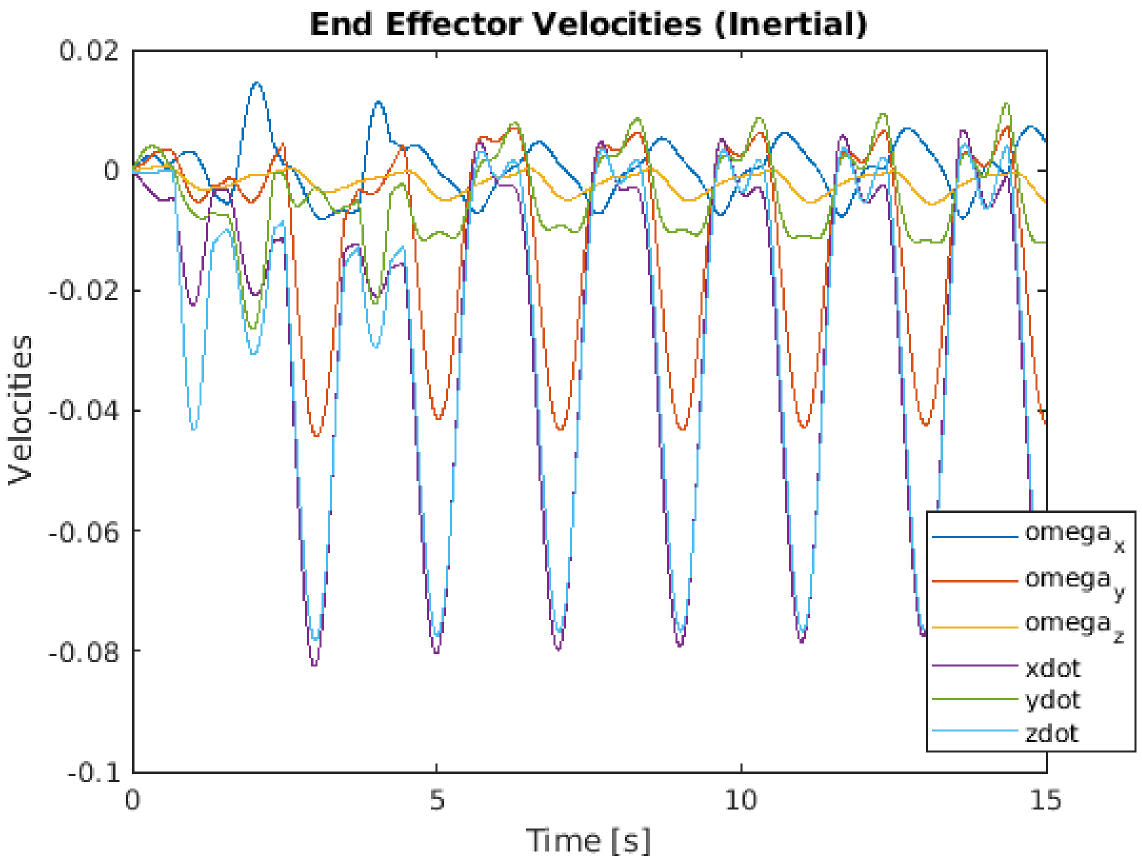
```
%         Adding the disturbances
     disturb = [0  0.0  0]';
     disturb_ang = [0  1  0]';
     disturb_ang = disturb_ang*1*sin(0.5*t*2*pi);
     uvms.p_dot(1:3)  = uvms.wTv(1:3,1:3)*disturb;
     uvms.p_dot(4:6)  = uvms.wTv(1:3,1:3)*disturb_ang;
```

The results of such a disturbance as shown in the images below. First from this first image we can see that the end effecor is clearly moving from the desired position. The displacemnts on x reach almost 10 cm.





Comparing the vehicle velocities with either the arm joints or end effector velocities, it is clear that in this case the task fails in its purpose.


End Effector Velocities (Inertial)
Velocities
Time [s]
omega$_x$
omega$_y$
omega$_z$
xdot
ydot
zdot