Instituto Superior Técnico

Mestrado em Engenharia Eletrotécnica e de Computadores

First Lab Assignment

IMU sensor Data Analysis and Exemplification

Rodrigo Coimbra, nº 100078 rodrigo.coimbra@tecnico.ulisboa.pt
Rodrigo Nobre, nº 100080 rodrigonobrepereira2002@tecnico.ulisboa.pt
Jorge Contente, nº 102143 jorge.contente@tecnico.ulisboa.pt
Joaquin Navarro , nº 113083 joaqanape@gmail.com

2nd period 2024/25

Índice

Contributions

As a group, we tackled tasks 1 through 4 together, as they were the most complex both conceptually and in a programming context. As for the last four tasks, Rodrigo Coimbra and Joaquin Navarro were assigned tasks 5 and 6, while Rodrigo Nobre and Jorge Contente adressed the final two tasks. This being said, everyone helped each other out throughout the whole assignment.

Introduction

This report focuses on the analysis of data obtained from an IMU sensor, which includes an accelerometer and a rate gyro. The sensor provides acceleration and angular velocity values at specific points in time and the objective of this assignment is to carefully analyze the data to reconstruct the trajectory and, ultimately, recreate it on the SCORBOT VII.

This robotic arm is an unit commonly used in education, particularly in the fields of robotics, where its design allows it to mimic the ways other similar but more industrial units behave. It has 5 degrees of freedom, as expressed below, making it a versatile but simple tool to use in this context, which also can be programmed using different interfaces, in our case, that being ACL (Advanced Control Language). This method allows us to use commands which can provide both direct use of the robot's axis and programming the system with, for example, a set of instructions.

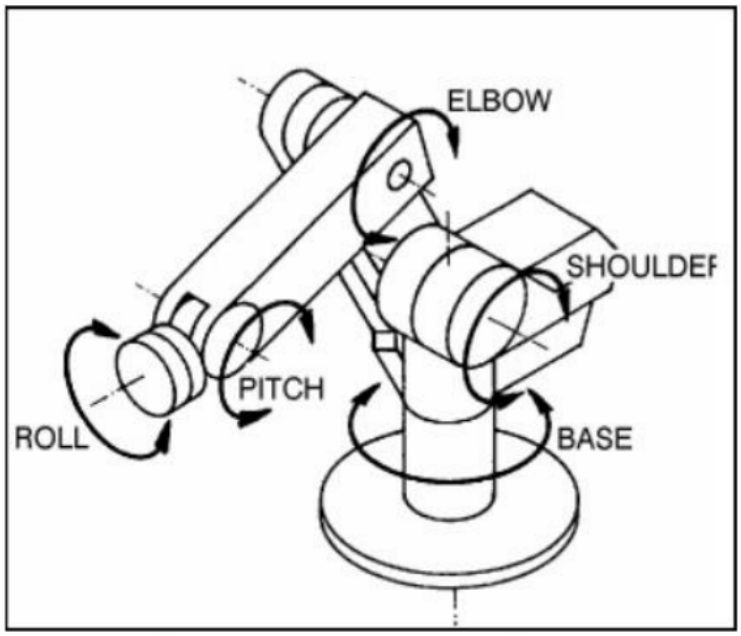


Fig. 1: SCORBOT VII joints

Task 1 - Plotting and Assigning Axis

To begin analyzing the trajectory, we must first examine and understand the data. Initially, we have a text file containing 20 seconds of discrete-time data, divided into two datasets: one for acceleration and the other for angular velocity. The plot corresponding to this is represented in the figure 2.

Identify each component

We start by analyzing the data from the first 5 seconds, when the sensors are stationary. The accelerometer always detects the effect of gravitational acceleration. Based on this, we can infer that the dataset representing acceleration corresponds to the z-axis. Since the acceleration is positive, we can also conclude that the IMU is upside down during this period. Moreover, we observe a rotation on one axis between approximately 4.5 and 7.5 seconds, resulting in about 180^{o} of rotation on the z-axis. This is indicated by the negative gravitational acceleration measured on one axis and the positive acceleration observed on another axis. Such a positive rotation could only occur around the x-axis, corresponding to a roll of 180^{o}. Finally, since the gravitational acceleration remains unchanged during the last rotation, it must have occurred around the z-axis.

Plots

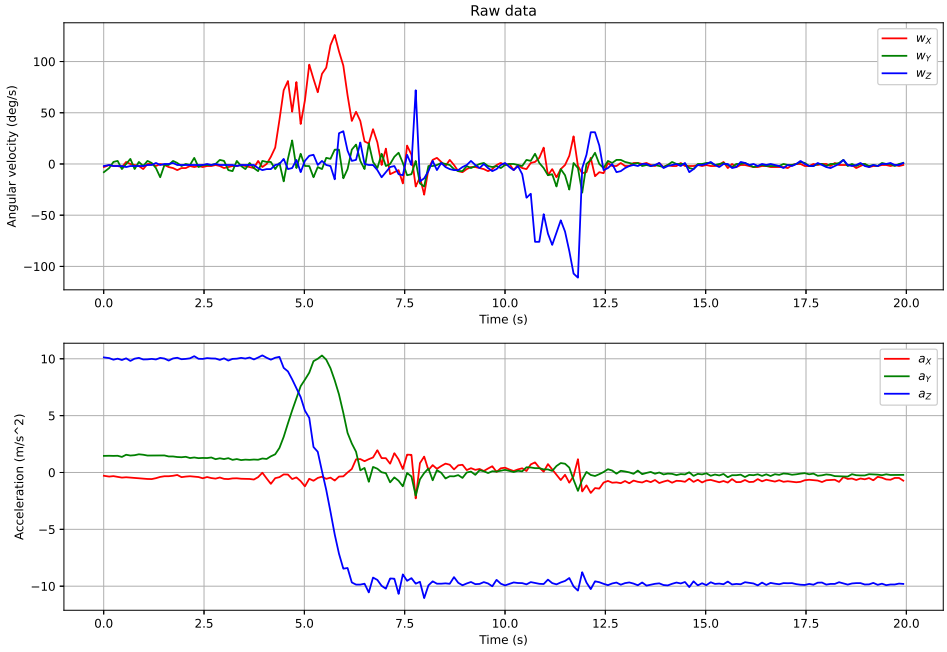


Fig. 2: Raw data of the angular velocity and acceleration from the IMU sensor

Acceleration

As mentioned earlier, the acceleration value on the z-axis is positive, which indicates that the IMU is upside down. Due to its rotation around the x-axis, the y-axis starts to show movement as gravity is now divided between the z and y axes. By the 6th second, the z-axis acceleration approaches the value corresponding to gravitational acceleration, suggesting that the IMU has stopped rotating.

Task 2 - Filtering Data

Filters

The raw data presented in Task 1 can be classified as very noisy and containing outlying points, making it somewhat unreliable. To address this, we experimented with filters such as the mean and median filters to prepare the data for plotting and analysis.

Before applying these filters, we reduced the noise by establishing a minimum threshold that measurements must exceed. Any measurement below this threshold was set to 0. To determine the threshold bounds, we measured the standard deviation of the sensor readings while the sensor was stationary. Using this value, we defined a range of $[-3\sigma, 3\sigma]$ as the threshold. This process is illustrated in figure 3, and the resulting data was used as input for the filters.

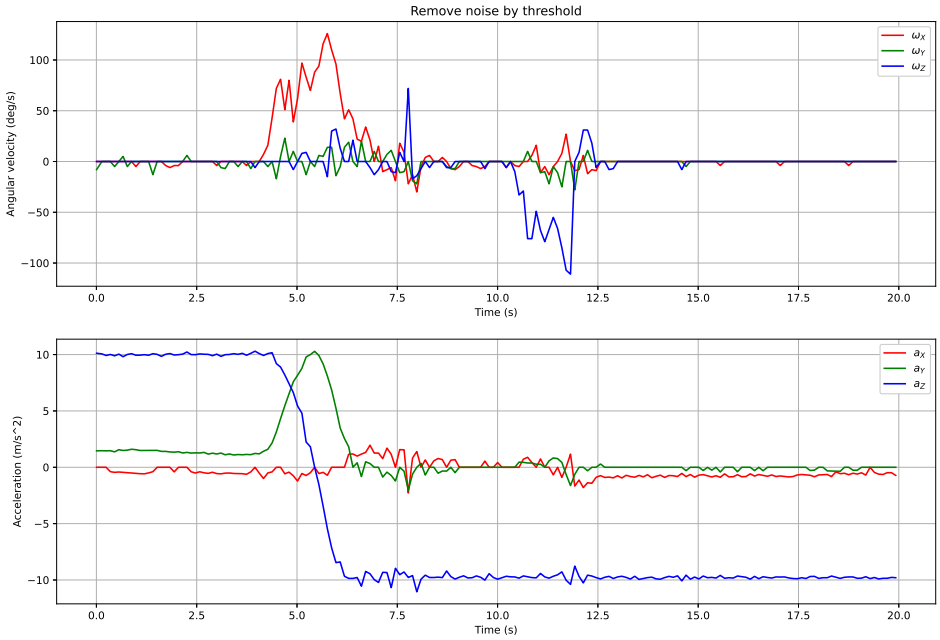


Fig. 3: Removed noise by threshold

Mean filter

The mean filter is a type of linear filter that replaces the point with the average of the n points in it's surroundings. The size of this window was experimented through trial and error, where $n = 5$ was found to have a good output. It's a simple filter to compute that reduces noise but can leave to distortions on rapid changes of raw data.

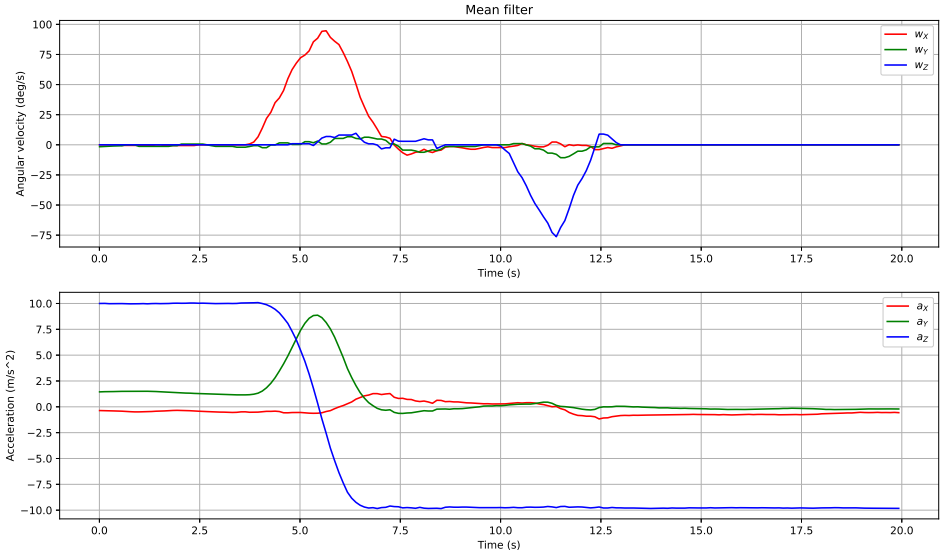


Fig. 4: Filtered angular velocity (top) and acceleration (bottom) data

Median filter

The median filter utilizes a similar concept to the mean filter, but uses the median of the surrounding n-points as a replacement for the point in question. Like the before-mentioned filter, the size of this window was chosen through trial and error, where a value of $n = 10$ was deemed suitable to use throughout comparisons with different filters.

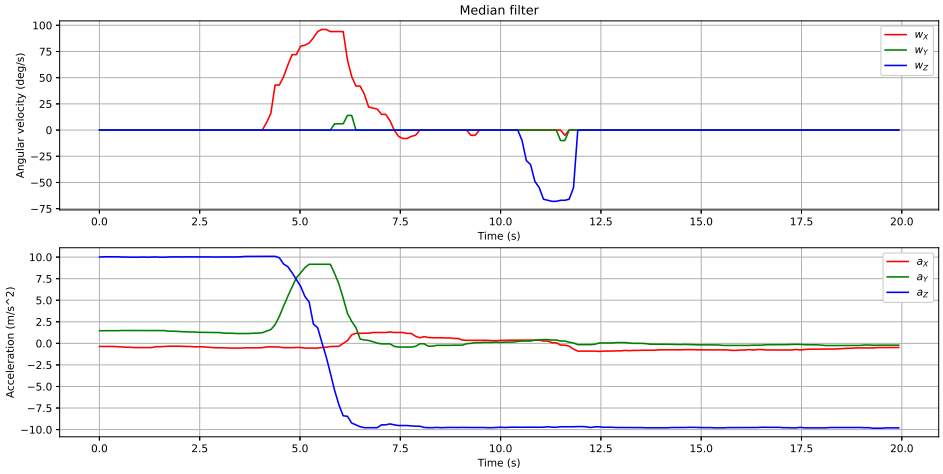


Fig. 5: Filtered angular velocity (top) and acceleration (bottom) data

After some analysis on the graphs shown previously, we found that the median filter with $n = 10$ was the better candidate for the next tasks and thus we will be using the median-filtered data to discuss them.

Task 3 - Reconstructing the Translation

For this task, we had to reconstruct the trajectory of the sensor in the Cartesian space (x,y,z) based on the data of the accelerometer, shown in the lower plot of the figure 2. The sensor's movement was continuous, however, sensors must discretize measurements, resulting, in this case, in data sampled approximately every 0.1s. Since we lack information between these timestamps and the measurement interval is relatively short, we assume that the acceleration remained constant during these intervals. Additionally, gravitational acceleration is always present in the accelerometer measurements. However, since the sensor takes measurements in its own frame as this task focuses solely on reconstructing the trajectory without accounting for orientation, we cannot remove the gravitational component. Removing it would require knowing which direction it's pointing, which is unavailable because the measurements are taken in the sensor frame rather than in a fixed world frame.

Taking all this into account, we need to integrate the acceleration twice to obtain the position of the sensor. Since the measurements are discrete, the integration is expressed as a sum over time, resulting in the following equations:

$$\Delta t_k = t_k - t_{k-1}, \tag{1a}$$

$$v_k = \sum_{k=1}^{k} a_k \Delta t_k + v_0, \tag{1b}$$

,

$$p_k = \sum_{k=1}^{k} v_k \Delta t_k + p_0, \tag{1c}$$

where $a_k = [a_{x_k}, a_{y_k}, a_{z_k}]$ is the acceleration in the instant k in m/s^2, $v_k = [v_{x_k}, v_{y_k}, v_{z_k}]$ is the velocity in the instant k in m/s, t_k is the time at instant k in seconds and $p_k = [x_k, y_k, z_k]$ is the position in 3D Cartesian space of the point in the instant k. Finally, p_0 and v_0 are the initial velocity and position, respectively. In this case, we consider $v_0 = [0, 0, 0]$ and $p_0 = [0, 0, 0]$.

Task 4 - Reconstructing the Rotation

For this task, we had to reconstruct the trajectory in the orientation space of the sensor based on the data from the rate-gyro, shown in the upper plot of the figure 2. As in the previous task, the sensor has discrete measurements, and therefore we assume a constant angular velocity between measurements. Additionally, the sensor measures all the information in its own frame. Therefore, we can multiply the rotation matrices associated with the rotation of each timestamp from the start until the desired instant, to obtain the rotation matrix at that instant. Finally, we just need to convert the rotation matrix to Euler angles.

Based on this, we start by calculating the change in angular position between timestamps, which can be achieved with the following:

$$\Delta\theta_k = \omega_k \cdot \Delta t_k, \tag{2}$$

where Δt_k, given by equation (1a), is expressed in seconds, $\omega_k = [\omega_{x_k}, \omega_{y_k}, \omega_{z_k}]$ is the angular velocity in radians per second and θ_k is the angular position in radians for each

component. These are then converted to a rotation matrix from this frame to the previous one using

$$R = \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix}, \tag{3}$$

where γ is the rotation around the x-axis ($\Delta\theta_x$), β is the rotation around the y-axis ($\Delta\theta_y$) and α is the rotation around the z-axis ($\Delta\theta_z$). Moreover, the c and s are abbreviations for cosine and sine, respectively.

Having the rotation matrix defined, we just need to, at each timestamp, calculate the rotation matrix from the current frame of the sensor to the world frame. The first rotation matrix, which represents the initial position of the sensor, is aligned with the world frame but rotated $\frac{pi}{2}$ along the x-axis, to be in accordance with the gravitational acceleration observed in figure 2, which was along the z-axis and positive. The following rotations can be obtained by multiplying the matrices as follows:

$${}^0_kR = \prod_{i=1}^{k} {}^{i-1}_iR \tag{4}$$

Having the rotation matrices from the sensor's frame to the world frame for all instances, we just need to convert them to Euler angles.

Task 5 - Plotting Trajectory in the orientation space

Applying what was discussed in task 4 to our dataset, we obtain figures 6, 7, 8 which illustrate how much rotation occurred along each axis, with values constrained to the range $[-\pi, \pi]$. This is evident at the beginning of both figures 6, 8, in the yaw representation, which shouldn't be mistaken for full rotations around the z-axis. In these plots, although the median filter produces a smoother change in orientation, the raw data still reveals the general trend of each orientation, which is plausible, as opposed to the position obtained by the acceleration data, as we will see later. These plots, like the angular velocity plot, allow us to see the two major rotations done. However, these let us extend that observation to the approximate rotation of each of them, where one is approximately π radians, and the other $\frac{\pi}{2}$ radians.

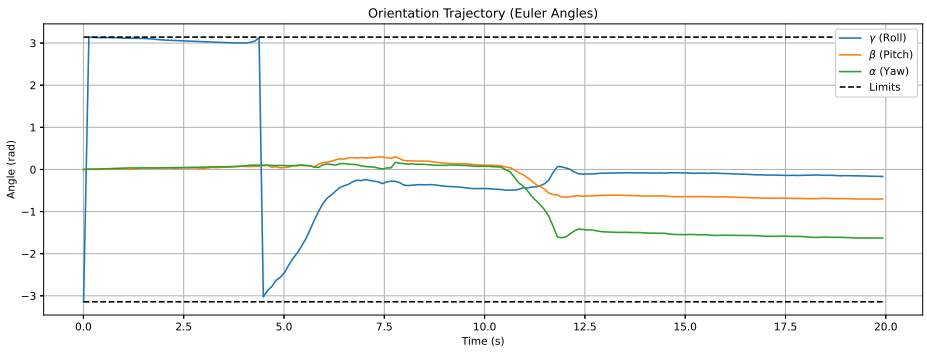


Fig. 6: Orientation space trajectory for the unfiltered data

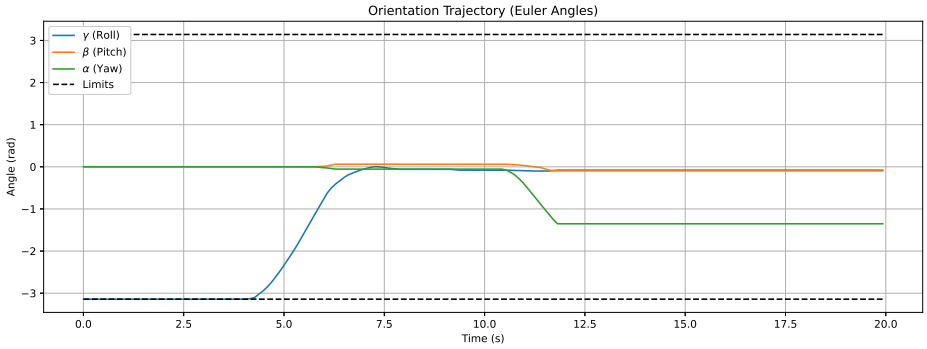


Fig. 7: Orientation space trajectory for the data filtered by a median filter

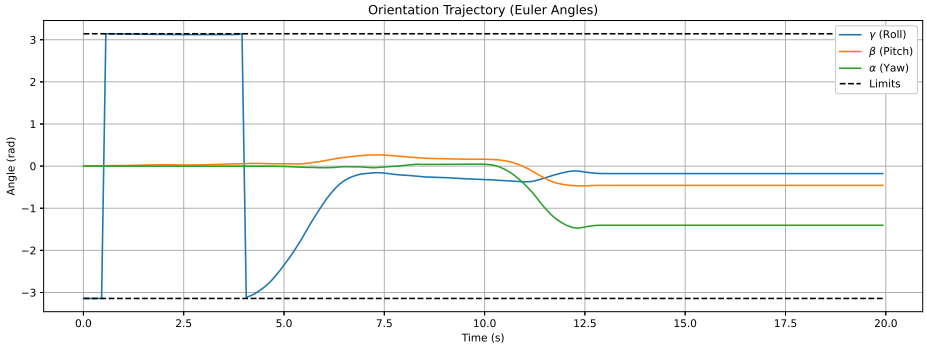


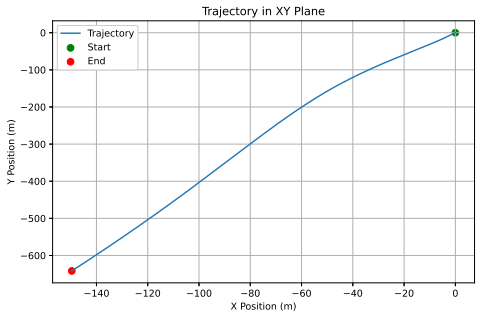
Fig. 8: Orientation space trajectory for the data filtered by a rolling mean filter

Task 6 - Plotting Trajectory in 3D

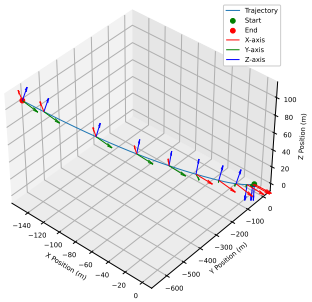
To plot the trajectory in 3D Cartesian space, we first need to combine both acceleration and angular velocity data, as these are necessary to reconstruct the trajectory.

To achieve this, we start by computing the rotation matrix from the sensor's frame to the world frame using the equations described in task 4. With this rotation matrix, we transform the acceleration data, obtaining the acceleration in the world frame. This transformation enables the reconstruction of the trajectory in Cartesian space using the equations presented in Task 3. Additionally, we compensate for the gravitational acceleration measured by the sensor, as it does not correspond to actual movement. This compensation is performed by adding a component to the transformed acceleration, pointing in the positive z-axis direction and with a magnitude equal to the gravitational acceleration.

Figures 9 and 10 show both the 3D trajectory and the trajectory in the XY plane. The additional XY-plane projection is included to better illustrate this specific plane's trajectory, which may be difficult to discern in this 3D perspective.

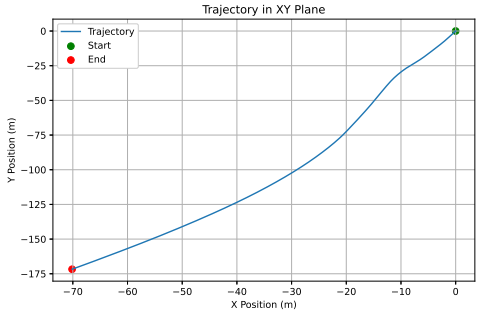


(a) Trajectory in XY Plane

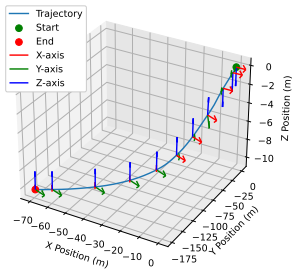


(b) Trajectory in 3D Cartesian space

Fig. 9: Unfiltered Data



(a) Trajectory in XY Plane



(b) Trajectory in 3D Cartesian space

Fig. 10: Median-Filter Data

The figures 9a and 9b show the trajectories using directly the raw data captured by the sensor. The 3D trajectory also includes the orientation of the sensor throughout the movement. As observed, the IMU measurements from the rate gyro are more plausible than the positions derived from acceleration data, which result in distances on the order of hundreds of meters. This difference in precision is partly due to the fact that rate-gyro measurements involve single integration, whereas acceleration data requires double integration, which propagates even more the errors measured by the accelerometer. After applying a median-filter, we obtained figures 10a and 10b. This was the only filter presented here, as it produced the best trajectory. In these results, we observe that the estimated movement has been significantly reduced. However, the distances are still unrealistically high and do not accurately represent the real trajectory. Given that the movement was performed with an arm, this values shouldn't exceed 1 meter approximately.

Task 7 - Direct Kinematics of SCORBOT VII

For task 7, we were tasked to write down the direct kinematics of the SCORBOT VII. To be able to do this, we need to analyze its degrees of freedom and its dimensions.

Dimensions

The dimensions of the robot are important for us due to their influence on our transformation matrix and in turn, the direct kinematics of the robot. To know this, we resorted to the measurements of the SCORBOT-ER VII Guide provided by the teachers, aswell as direct measurements on the unit using a measuring tape. We have 5 degrees of freedom, therefore we we have 4 links, that are between each joint.

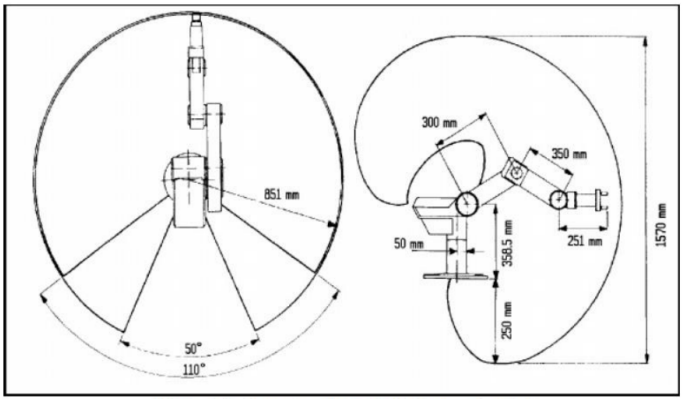


Fig. 11: Guide's measurements of SCORBOT and it's operating range

	x(mm)	**y(mm)**	**z(mm)**
L1	*50*	*0*	*358.5*
L2	*0*	*-37*	*300*
L3	*0*	*0*	*350*
L4	*0*	*0*	*251*

Tab. 1: Links dimensions

Direct Kinematics Matrix

To get out Direct Kinematics matrix, first we needed to decide a convention for our reference frames to use, ultimatly sticking with the Denavit-Hartenberg convention as it was not only widely discussed on the theoretical classes but also since there are a lot of references and case studies online about the matter. The D-H convention is compromised of four different matrices, two for both translation, $Trans$, and rotation, Rot, of the two links involved on the transformation (see (5)). Each matrix uses a specific parameter:

- Rot_1: θ, the angle about z_1 from x_1 to x_2
- $Trans_1$: d, the offset along z_1 to the common normal
- $Trans_2$: a, the length of the common normal. Assuming a revolute joint, this is the radius about z_1.
- Rot_2: α, angle about common normal, from z_1 to z_2

Thus, our matrix for a determined link transformation, T_i, maybe written as such:

$$T_i = Rot_z(\theta_i)Trans_z(d_i)Trans_x(a_i)Rot_x(\alpha_i) \tag{5}$$

$$T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow T_i = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i)cos(\alpha_i) & sin(\theta_i)sin(\alpha_i) & a \cdot cos(\theta_i) \\ sin(\theta_i) & cos(\theta_i)cos(\alpha_i) & -cos(\theta_i)sin(\alpha_i) & a \cdot sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Where the direct kinematics matrix is shown as $\prod_{i=1}^{k} T_i$, giving us the final position and orientation of an endpoint in relation to the world frame (in our case, the world frame and the base frame coincide).

Task 8 - Executing the Orientation Trajectory

In this final task, we get to apply our work to a practical environment using the SCORBOT, executing the orientation trajectory that we carefully analyzed throughout this laboratory work, using the previously stated ACL interface. Before doing this though, it is important to understand how this unit will behave given our ACL instructions and also how we need to handle its programming as a whole.

As previously stated on our laboratory introduction, ACL allows us to send commands to the SCORBOT (via PuTTY software on a serial port) that can in turn dispose the unit's end-factor in a valid position, using the already built-in inverse kinematics. With this in mind, to represent our movement, the starting position of the robot's end-point needed to be one that the arm, once moving, didn't transverse with any position that either damaged the unit (i.e the arm colliding with the surrounding environment) and that the positions would be inside the machine's operating range (see figure 11), meaning we should remain above a certain height at all times. However, since our pitch did not change much, we only had to guarantee that the initial position was high enough to avoid collisions. For our yaw, we move almost $-\frac{\pi}{2}$ radians, which means that starting with a small positive rotation would ensure safety. Lastly, we had a roll of π radians, which means that our end point would need to be capable of rotating that much, which it was, not causing any problems in relation to limits or the path.

With these concerns in mind, our initial position was described with the following joint angles:

$$\theta = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 \end{bmatrix} = \begin{bmatrix} \frac{\pi}{4} & \frac{\pi}{2.4} & -\frac{\pi}{1.6} & \frac{\pi}{4} & 0 \end{bmatrix}$$

To represent the reference frames of each joint, we used the Denavit-Hartenberg convention and the previous section's notions on the direct kinematics of the robot. To visualize the reference frames, the joints at the defined initial position was represented in MATLAB, as shown in 12.

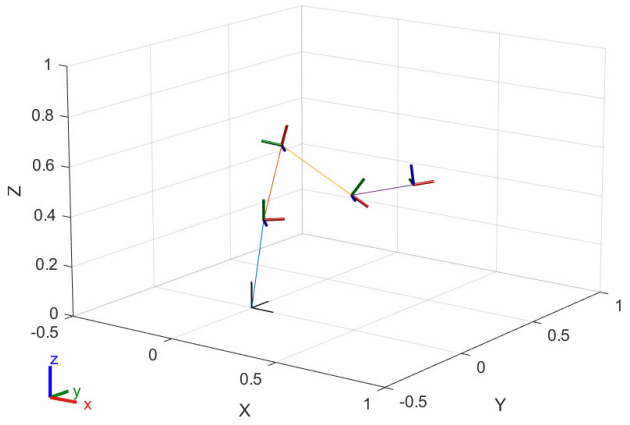


Fig. 12: Reference frames of SCORBOT VII's joints

With said convetion with the measured parameters, we arrived at the following matrices used to calculate the final position of the endpoint of the robot. Each row of the table 2 represents a, α and d for each link, from the base (first row) up to the wrist roll (last row), that the respective T_i matrix is using depending on the link it represents.

	a_i	α_i	d_i	Final
1	0.05	$\frac{\pi}{2}$	0.3585	θ_1
2	0.3	0	-0.035	θ_2
3	0.35	0	0	θ_3
4	0.251	$\frac{\pi}{2}$	0	θ_4
5	0	0	0	θ_5

Tab. 2: Denavit-Hartenberg Parameters

Figures 13 and 14 show a possible trajectory to replicate our orientation through inputting the direct kinematics with the orientation desired to each joint. In this case, the base joint (seen in figure 1) is used to produced the desired yaw movement. About the pitch, since it's a minimal movement, we assigned it just to the wrist pitch joint, which was more than capable of handling that movement. Finally, the wrist roll joint was used to replicate the roll of our original data. To validate our trajectory produced using direct kinematics, we used the roboticstoolbox, which, given the parameters of table 3, produced the same trajectory.

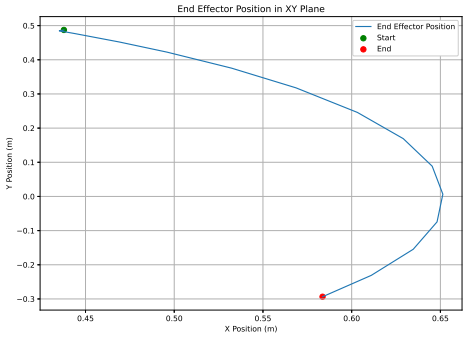


Fig. 13: Trajectory in the XY plane based on the orientation

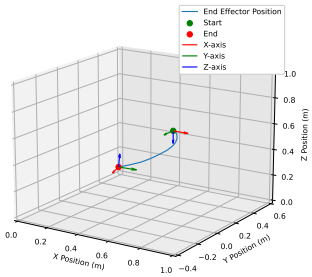


Fig. 14: Trajectory in the 3D Cartesian space based on the orientation

Following what was said previously about the safety and trajectory concerns, our arbitrary initial position and the final one can be seen on the table 3. The final testing of the robot involved creating two positions based on the previously calculated coordinates, moving from one to another, as figures 15 and 16 demonstrate.

	Initial	Final
X	4378	5836
Y	4873	-2935
Z	4680	4799
R	-1799	-49.73

Tab. 3: Initial and final robot coordinates

Fig. 15: Initial position

Fig. 16: Final position

Conclusion

This assignment was compromised with 8 tasks consisting of analyzing and visualizing accelerometer and rate-gyro in order to efficiently understand and replicate the movement of our data. From it, we managed to get a good sense of how to handle this type of data by, for example, filtering the noise in a satisfactory way, as well as getting practice on the behavior of real sensors, instead of just idealistic data. Following that, we found that

the IMU-sensor, due to the nature of the data that it measures (acceleration and angular velocity), is more suitable for obtaining the orientation of the movement comparing to the actual positioning of the endpoint. This is due to the fact that extracting the positioning from the discrete data of the accelerometer can lead to larger errors since we integrate twice, relative to once on the orientation, as discussed on task 3.

Replicating the movement was by far the most interesting part of the assignment, since we could test on a real-life device our results, which, as said on task 8, align with relevant python function's output and therefore we consider that the assignment was a success.