ROBOTICS

MEEC

Orientation Estimation and Trajectory Analysis Using Sensor Data

Authors:

Vladimiro José de Medeiros Roque (98589)	vladimiro.roque@tecnico.ulisboa.pt
José Pedro da Cunha Rodrigues (113234)	jose.p.rodrigues@tecnico.ulisboa.pt
Miguel Rodrigues Ferreira (113289)	miguel.r.ferreira@tecnico.ulisboa.pt
Pedro Alexandre Ferreira Dias Lopes (103194)	pedroafdlopes@tecnico.ulisboa.pt

Group 9

2024/2025 – 1º Semester, P2

Contents

1 Group Members Contribution

- **Task 1:** José Rodrigues
- **Task 2:** José Rodrigues
- **Task 3:** Pedro Lopes
- **Task 4:** Vladimiro Roque
- **Task 5:** Miguel Ferreira
- **Task 6:** Miguel Ferreira
- **Task 7:** Pedro Lopes
- **Task 8:** José Rodrigues/Vladimiro Roque
- **Python/Matlab scripts:** All group members

2 Introduction

This laboratory assignment focuses on two core objectives: understanding the estimation of orientation using data from a rate-gyro sensor and an accelerometer, and demonstrating the application of an industrial-grade serial manipulator. By addressing these objectives, students will gain practical experience in processing sensor data, applying mathematical models, and reconstructing trajectories in both Cartesian and orientation spaces. The work begins with the analysis of sensor data, provided in unique datasets for each group, containing approximately 20 seconds of measurements. The initial 5 seconds represent a static phase where the sensor remains stationary, providing a baseline for filtering and processing. These datasets include accelerometer readings (in milli-g) and rate-gyro readings (in degrees per second), formatted to facilitate computational analysis. The assignment is divided into multiple tasks, starting with data visualization, filtering, and theoretical trajectory reconstruction equations. Further tasks involve the graphical representation of reconstructed trajectories and their interpretation. In the later stages, the focus shifts to the use of the Scorbot VII manipulator. Here, students will derive the robot's direct kinematics equations and assess whether the reconstructed trajectories can be executed by the manipulator. By combining theoretical concepts with practical implementation, this lab offers an in-depth understanding of sensor data processing and robotic manipulator operations.

3 Tasks

3.1 Task 1

Task 1 involves visualizing sensor data by plotting the components of the accelerometer and rate-gyro measurements along their respective axes. This step is critical for understanding the behavior of the sensor during the data collection process and identifying any initial patterns or anomalies in the raw data. The dataset provided contains time-stamped measurements from an accelerometer and a rate-gyro, with values distributed across three axes: x, y, and z. The accelerometer data (measured in milli-g) reflects linear acceleration along each axis, while the rate-gyro data (measured in degrees per second) provides angular velocity along the same axes. The first few seconds of the dataset capture the sensor in a static configuration, offering a baseline for comparison against later movements. Through the visualization of these components in a combined plot, we aim to:

- Distinguish the data trends for each axis.
- Identify any irregularities or noise in the data.
- Provide a foundation for further processing in subsequent tasks.
- This initial analysis will serve as the starting point for understanding the sensor's performance and the nature of the motion captured.

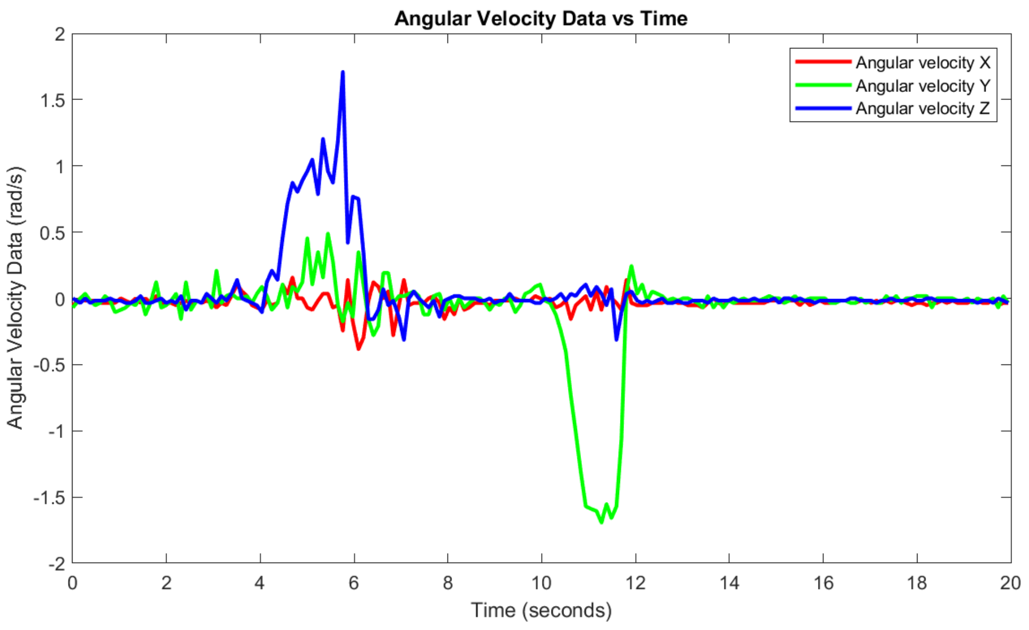

Figure 1: Angular velocity vs time.

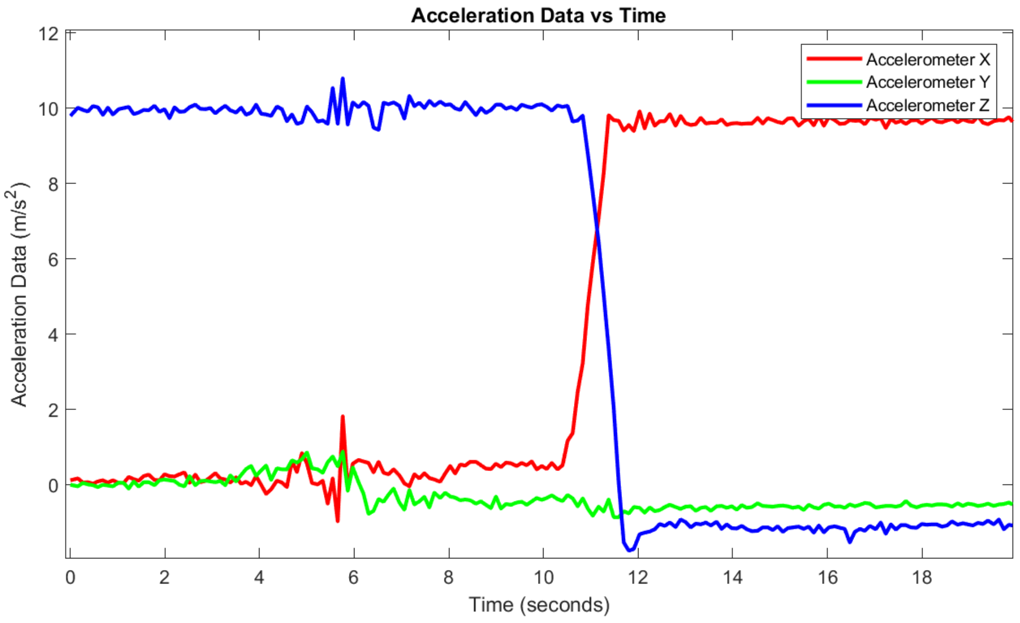

Figure 2: Acceleration vs time

Fixing the y-Axis

When the y-axis is fixed, a rotation around this axis (ω_y) results in the x- and z-axes exchanging roles in the coordinate system. Mathematically, this can be expressed using a rotation matrix around the y-axis:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Here:

- x', y', z' are the new coordinates after rotation
- x, y, z are the original coordinates
- θ is the angle of rotation around the y-axis.

This transformation shows that:

- The acceleration along the x-axis (a_x) is projected onto the z-axis (a'_z) after rotation,
- Similarly, the acceleration along the z-axis (a_z) is projected onto the x-axis (a'_x).

This axis "swap" is most noticeable when the angular velocity around y (ω_y) increases, as this corresponds to the y-axis starting to rotate significantly. In the acceleration graph, this behavior can be observed when the acceleration of a_x and a_z begin to swap. We can see that the accelerations (a_x) and (a_z) start to swap between the time interval of $10s$ to $12s$ wich corresponds to the peak in the y-axis in the angular velocity graph between 10 and $12s$ wich shows that we started to rotate around y-axis.

Dynamic Behavior and Angular Velocity ω_y

During the rotation around y, the angular velocity ω_y increases as the x- and z-axes dynamically swap roles. This is further supported by observing:

- A clear increase in ω_y in the angular velocity graph,
- Symmetric oscillatory patterns in the accelerations a_x and a_z, indicating their dynamic exchange.

As the angular velocity ω_y increases, the system exhibits significant rotational motion in the xz plane, while the accelerations are projected between the two axes. The relationship can be described as:

$$a'_x = a_x \cos\theta + a_z \sin\theta,$$

$$a'_z = -a_x \sin\theta + a_z \cos\theta,$$

where θ represents the instantaneous angle of rotation around the y-axis.

Key Observations

From the graphs:

- When $\omega_y \approx 0$, the accelerations a_x and a_z remain largely independent, with minimal projection effects.
- As ω_y increases, the projections of a_x onto a_z and vice versa become prominent, resulting in the observed axis swapping.
- The y-axis rotation introduces dynamic correlations between the accelerations in x and z, explaining their exchange during periods of high ω_y.

3.2 Task 2

Preprocessing sensor data is a vital step to ensure accurate and meaningful analysis. Raw data, collected from accelerometers and rate-gyros, often contain noise and occasional outliers caused by sensor imperfections, environmental factors, or abrupt movements. These irregularities can obscure the true motion dynamics and negatively affect subsequent analyses. In this case, a median filter was applied to clean the data effectively. This filtering technique is well-suited for handling outliers and reducing noise, as it replaces each data point with the median of its neighbors within a defined window. Unlike other filtering methods, the median filter preserves sharp transitions and edges in the data while removing unwanted spikes, making it ideal for dynamic motion data. The results achieved with the median filter demonstrated significant improvements, yielding smooth and reliable datasets without distortions. This preprocessing ensures that the data is well-prepared for reconstructing accurate trajectories and performing further analysis with confidence in the validity of the underlying patterns.

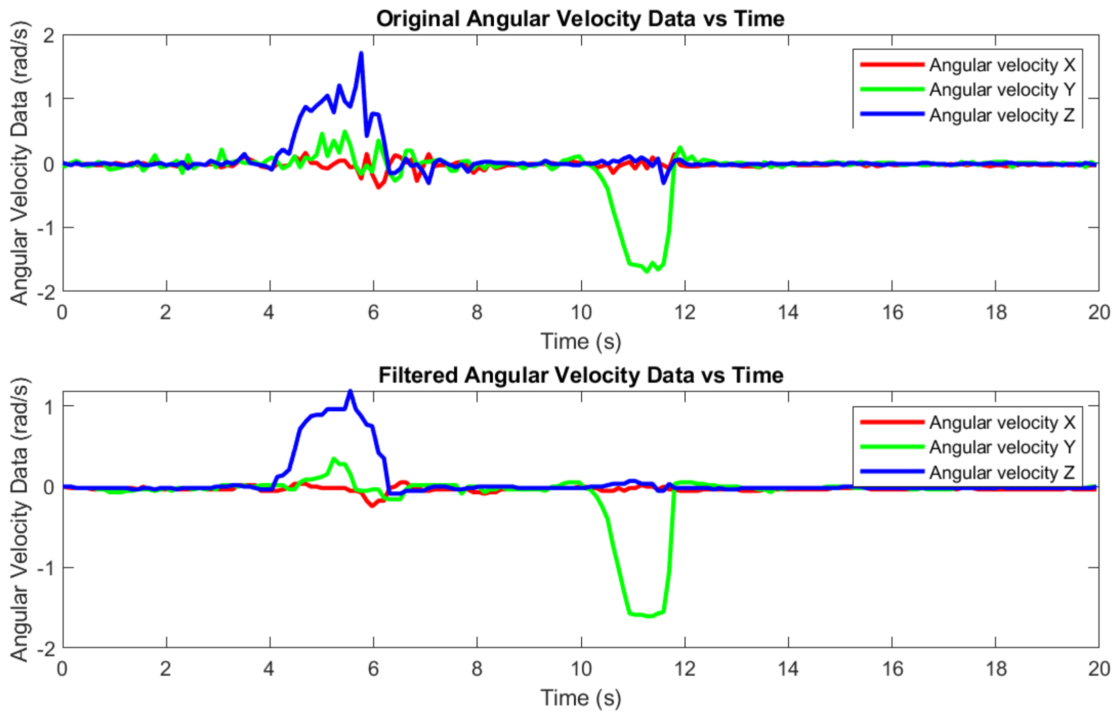

Figure 3: Filtered angular velocity vs original (using median filter).

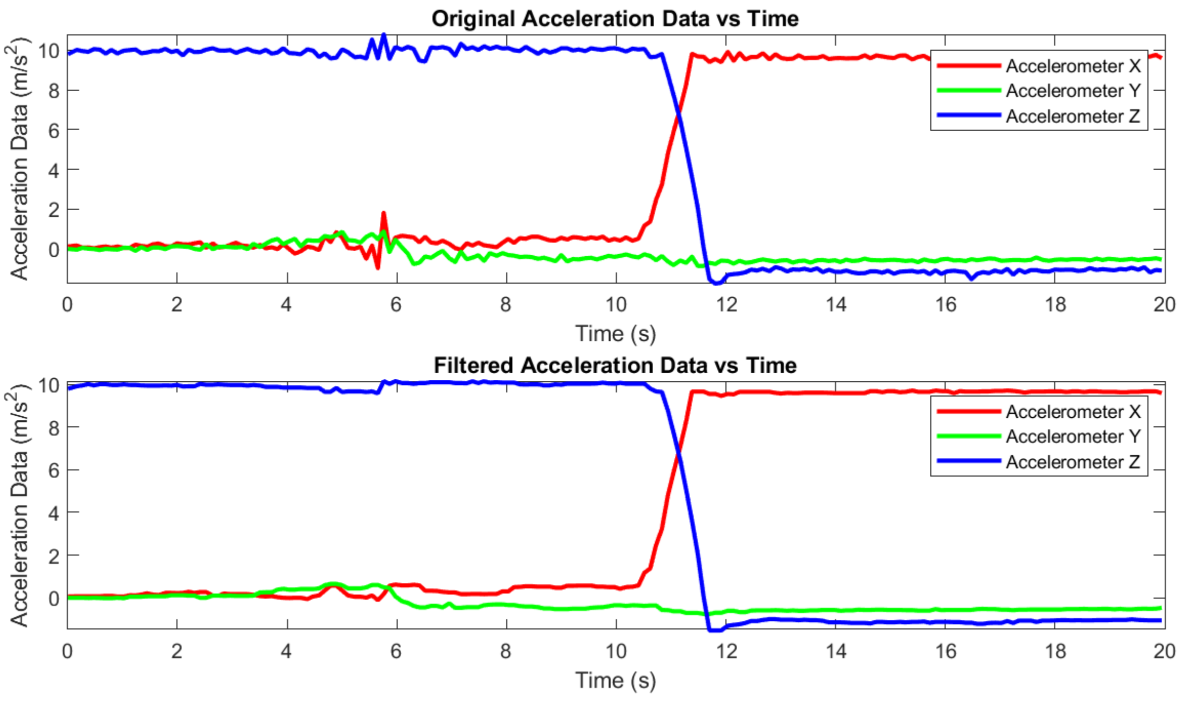

Figure 4: Filtered acceleration vs original (using median filter)

Comparison of Filters

1. Sliding Window Average/Median Filters

- **Average Filter:** Calculates the average of data points within a sliding window, effectively reducing high-frequency noise. However, it is sensitive to outliers, as extreme values can significantly skew the average.
- **Median Filter:** Replaces each point with the median of its neighbors within the sliding window. This makes it more robust than the average filter, as it can effectively handle outliers without distorting the signal.

2. Gaussian Filters

- Smooths data by applying a weighted average, where weights follow a Gaussian distribution. This reduces noise while preserving overall trends better than simple averaging.
- It is less robust to outliers compared to the median filter, as the weights are still influenced by extreme values.
- Requires the selection of a standard deviation parameter (σ), which controls the level of smoothing, making it less straightforward for general applications.

3. Savitzky–Golay Filters

- Fits a polynomial to a subset of data points within a sliding window, then replaces the center point with the value predicted by the polynomial.
- Effective at preserving sharp transitions and trends while reducing noise, making it ideal for smooth but non-linear signals.
- Sensitive to outliers, as the polynomial fit can be distorted by extreme values within the window.
- Requires careful tuning of the polynomial order and window size for optimal results.

4. Smooth Filters

- General-purpose filters that reduce noise by averaging data points or applying weighted smoothing.
- These filters can smooth data effectively but often blur sharp transitions, making them less suitable for dynamic systems like robotic arms.
- They do not specifically address outliers, which can still influence the smoothed result.

5. Outliers Filters

- Specifically designed to detect and remove outliers based on statistical thresholds (e.g., Z-Score or Interquartile Range methods).
- While effective at identifying and removing anomalies, these filters often leave gaps in the data by replacing outliers with `NaN`, requiring additional interpolation steps.
- Do not inherently smooth the data or reduce noise in non-outlier regions.

Why Use the Median Filter?

The Median Filter stands out compared to these alternatives due to its:

- **Robustness:** Unlike the Average, Gaussian, or Savitzky–Golay filters, the Median Filter is highly robust against outliers and extreme values, as it is unaffected by the magnitude of data points outside the sliding window.
- **Preservation of Trends:** The Median Filter preserves sharp transitions and dynamic changes better than smooth or Gaussian filters, which tend to over-smooth the data.
- **Simplicity:** Requires only a window size as a parameter, making it simpler to implement than Gaussian or Savitzky–Golay filters, which require tuning of additional parameters (e.g., σ or polynomial order).
- **Completeness:** Unlike Outliers Filters, the Median Filter does not leave gaps in the data and simultaneously reduces noise and removes anomalies.

3.3 Task 3

To reconstruct the sensor position in Cartesian coordinates (x, y, z) using accelerometer data, we analyze the relationship between acceleration data $(\ddot{x}, \ddot{y}, \ddot{z})$ and the velocity data we get from integrating the acceleration $(\dot{x}, \dot{y}, \dot{z})$.

3.3.1 Theory and Mathematical Formulation

- **X** — Position on the global x axis.
- **Y** — Position on the global y axis.
- **Z** — Position on the global z axis.

Parameterization To reconstruct the position accurately, we define the following parameters:

- x, y, z: Position components in the Cartesian space.
- $\dot{x}, \dot{y}, \dot{z}$: Velocity components, which are the first derivatives of position (first derivatives).
- $\ddot{x}, \ddot{y}, \ddot{z}$: Acceleration components, which are the second derivatives of position (second derivatives).
- Δt: Sampling period of the acceleration data.

3.3.2 Effect of Gravity

The acceleration due to gravity must be taken into account when reconstructing the sensor position in the Cartesian coordinates. The gravitational acceleration is approximated as $g = 9.8\,\mathrm{m/s}^2$ in the downward direction along the global z-axis. The total acceleration vector in the world frame, $\mathbf{A}$, is the final accelerations only due to the motion of the sensor.

The equation for the total acceleration vector is:

$$\mathbf{A} = \mathbf{R} \cdot \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} - 9.8 \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Where: - $\mathbf{R}$ is the transformation matrix, calculated in **Orientation Representation**, that relates the accelerometer's local frame to the world frame. - $\ddot{x}, \ddot{y}, \ddot{z}$ are the accelerations along the x, y, and z axes, respectively. - The second term $-9.8 \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ represents the gravitational acceleration in the world frame.

This equation allows us to separate the accelerometer's linear motion from the gravitational effect, enabling a more accurate reconstruction of the sensor's position.

Position and Velocity Update

General Formulas

In general, the position update equations for discrete system can be written as:

$$\mathbf{p}(t + \Delta t) = \mathbf{p}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)(\Delta t)^2$$

The velocity update equation is:

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \mathbf{a}(t)\Delta t$$

Matrix Notation for x, y, z Components

Using matrix notation, the updates become:

$$\begin{bmatrix} x(t+\Delta t) \\ y(t+\Delta t) \\ z(t+\Delta t) \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} + \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} \Delta t + \frac{1}{2} \begin{bmatrix} \ddot{x}(t) \\ \ddot{y}(t) \\ \ddot{z}(t) \end{bmatrix} (\Delta t)^2$$

For velocity:

$$\begin{bmatrix} \dot{x}(t+\Delta t) \\ \dot{y}(t+\Delta t) \\ \dot{z}(t+\Delta t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix} + \begin{bmatrix} \ddot{x}(t) \\ \ddot{y}(t) \\ \ddot{z}(t) \end{bmatrix} \Delta t$$

3.4 Task 4

The objective of Task 4 is to derive the equations required to reconstruct the trajectory of the sensor in the orientation space defined by Euler angles (α, β, γ). Using the data from the rate-gyro sensor, the task focuses on relating the angular velocities measured in the body frame $(\omega_x, \omega_y, \omega_z)$ to the time derivatives of the Euler angles $(\dot{\alpha}, \dot{\beta}, \dot{\gamma})$, and integrating these equations to obtain the sensor's trajectory in orientation space.

3.4.1 Parameterization

To reconstruct the orientation accurately, we define the following parameters:

- α, β, γ: Euler angles representing the sensor orientation in 3D space.
- $\omega_x, \omega_y, \omega_z$: Angular velocities provided by the rate-gyro, indicating rotational speeds around the sensor local axes.
- $\Omega_{\text{body}} = [\omega_x, \omega_y, \omega_z]^T$: Angular velocity vector in the body frame.
- $\mathrm{T}(\alpha, \beta)$: Transformation matrix that relates the Euler angle rates to the angular velocities.
- Δt: Time interval between consecutive measurements.

In the ZYX convention, the angles represent rotations applied in a specific order:

- **Yaw** (α) — rotation about the global Z-axis. This initial rotation aligns the sensor to face a desired target or orientation.
- **Pitch** (β) — rotation about the new Y-axis after the yaw rotation. This step adjusts the sensor's vertical alignment relative to the target.
- **Roll** (γ) — rotation about the new X-axis following the yaw and pitch rotations. This final adjustment ensures precise alignment of the sensor or manipulator's tool for interaction with the target.

3.4.2 Orientation Representation Using Rotation Matrix

The orientation of the sensor is represented by a rotation matrix R. Using the ZYX Euler convention, the rotation matrix R is constructed as the product of three elementary rotation matrices:

$$R = R_z(\alpha) R_y(\beta) R_x(\gamma), \tag{1}$$

where each elementary rotation matrix corresponds to a rotation about one of the principal axes:

$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}, \tag{3}$$

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}. \tag{4}$$

By multiplying 2, 3 and 4, 1 becomes:

$$R = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}. \tag{5}$$

3.4.3 Relating Orientation and Angular Velocity

The relationship between the time derivative of the rotation matrix $\dot{R}$ and the angular velocity vector $\mathbf{\Omega}_{\text{body}} = [\omega_x, \omega_y, \omega_z]^T$ is given by:

$$\dot{R} = \Omega R, \tag{6}$$

where Ω is the skew-symmetric matrix of angular velocities:

$$\Omega = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{7}$$

Using the relationship 6, we expand $\dot{R}$ in terms of the Euler angle derivatives to identify how Ω relates to the angular velocity components. This leads to the expression:

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = T(\alpha, \beta) \begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix}, \tag{8}$$

where $T(\alpha, \beta)$ is the Jacobian matrix.

3.4.4 Angular Velocity in the Body Frame

The angular velocity vector in the body frame, $\mathbf{\Omega}_{\text{body}} = [\omega_x, \omega_y, \omega_z]^T$, is related to the time derivatives of the Euler angles $(\dot{\alpha}, \dot{\beta}, \dot{\gamma})$ through the Jacobian Matrix $T(\alpha, \beta)$.

Each angular velocity component is derived as follows:

- The yaw rate $(\dot{\alpha})$ contributes directly along the Z-axis of the inertial frame.
- The pitch rate $(\dot{\beta})$ contributes along the rotated Y'-axis after the yaw rotation.
- The roll rate $(\dot{\gamma})$ contributes along the twice-rotated X''-axis, incorporating the effects of both yaw and pitch rotations.

To transform these contributions into the body frame, the transpose rotation matrices are applied, resulting the following decomposition:

$$\mathbf{\Omega}_{\text{body}} = \begin{bmatrix} \dot{\alpha} \\ 0 \\ 0 \end{bmatrix}_{\text{inertial}} + R_z^T \begin{bmatrix} 0 \\ \dot{\beta} \\ 0 \end{bmatrix}_{\text{inertial}} + (R_y R_z)^T \begin{bmatrix} 0 \\ 0 \\ \dot{\gamma} \end{bmatrix}_{\text{inertial}} . \tag{9}$$

By summing these contributions, the angular velocity vector in the body frame is expressed as:

$$\mathbf{\Omega}_{\text{body}} = \begin{bmatrix} \dot{\alpha} + \sin\alpha\dot{\beta} - \sin\beta\cos\alpha\dot{\gamma} \\ \cos\alpha\dot{\beta} + \sin\alpha\sin\beta\dot{\gamma} \\ \cos\beta\dot{\gamma} \end{bmatrix} . \tag{10}$$

3.4.5 Reconstructing the Trajectory of the Sensor in Orientation Space

From 10, we can identify the Jacobian matrix $T(\alpha, \beta)$ by grouping the coefficients of $\dot{\alpha}$, $\dot{\beta}$, and $\dot{\gamma}$:

$$T(\alpha, \beta) = \begin{bmatrix} 1 & \sin\alpha & -\sin\beta\cos\alpha \\ 0 & \cos\alpha & \sin\alpha\sin\beta \\ 0 & 0 & \cos\beta \end{bmatrix} . \tag{11}$$

To calculate the time derivatives of the Euler angles, the inverse of 11 is used:

$$T(\alpha, \beta)^{-1} = \begin{bmatrix} 1 & 0 & -\sin\beta \\ 0 & \cos\alpha & \sin\alpha\cos\beta \\ 0 & -\sin\alpha & \cos\alpha\cos\beta \end{bmatrix} . \tag{12}$$

Thus, the time derivatives of the Euler angles are computed as:

$$\begin{bmatrix} \dot{\alpha} \\ \dot{\beta} \\ \dot{\gamma} \end{bmatrix} = T(\alpha, \beta)^{-1} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} . \tag{13}$$

From this, we derive the equations to reconstruct the trajectory of the sensor in the orientation space (Euler angles α, β, γ) using rate-gyro data:

$$\dot{\alpha} = \omega_x - \sin(\beta)\omega_z, \tag{14}$$

$$\dot{\beta} = \cos(\alpha)\omega_y + \sin(\alpha)\cos(\beta)\omega_z, \tag{15}$$

$$\dot{\gamma} = -\sin(\alpha)\omega_y + \cos(\alpha)\cos(\beta)\omega_z. \tag{16}$$

3.4.6 Numerical Integration of Euler Angles

The Euler angles are updated using numerical integration based on the Euler explicit method:

$$\alpha(t+\Delta t) = \alpha(t) + \dot{\alpha}(t)\Delta t, \tag{17}$$

$$\beta(t+\Delta t) = \beta(t) + \dot{\beta}(t)\Delta t, \tag{18}$$

$$\gamma(t+\Delta t) = \gamma(t) + \dot{\gamma}(t)\Delta t. \tag{19}$$

This method assumes a sufficiently small Δt to minimize integration errors. For requiring higher precision or long-term stability, more advanced integration methods, such as Runge-Kutta or sensor fusion techniques (e.g., Kalman filter) may be considered.

3.5 Task 5

We implemented the calculation of the 3D orientation trajectory using angular velocity data.We started by defining the initial rotational angles ($\theta_x = 0$, $\theta_y = 0$, $\theta_z = 0$) at the initial time $t = 0$. The angular velocities are derived from filtered velocity data and are used in the motion equations to compute the orientation changes along each axis.The orientation angles are updated at each time step and stored in arrays for later visualization. Once the euler angles were calculated, we plotted the orientation trajectory in 3D space. This plot shows how the object rotates over time.

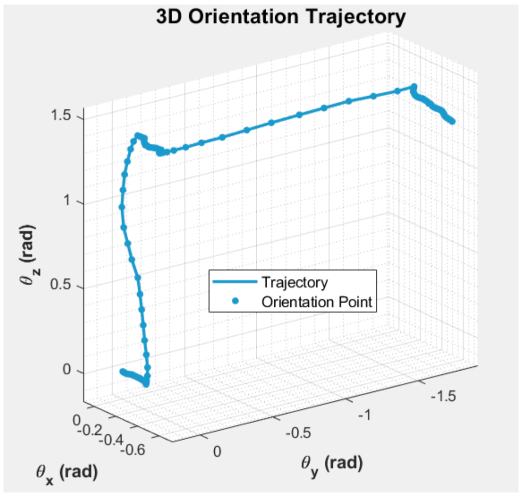

Figure 5: 3D Orientation

To better visualize the results, we plotted the evolution of the Euler angles θ_x, θ_y, and θ_z over time, providing a clear representation of the rotational dynamics.

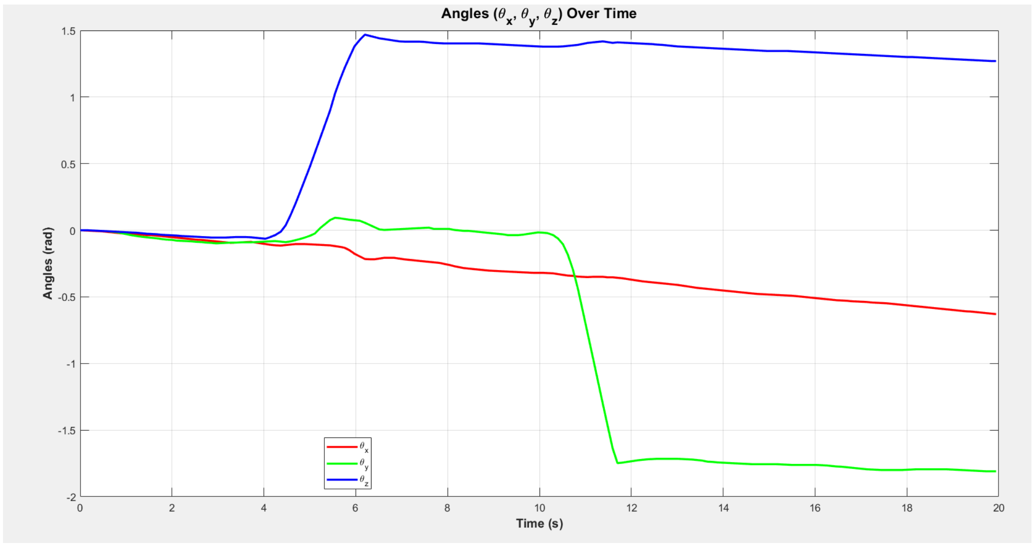

Figure 6: Euler Angles evolution

The graph shows the evolution of the Euler angles θ_x, θ_y, and θ_z over time, representing rotations about the X, Y, and Z axes, respectively.The results reveal significant rotations about the Z-axis (θ_z) and Y-axis (θ_y) with minimal variation in the X-axis (θ_x). Variations in the X-axis (θ_x) could have been caused by several factors, even though only rotations about the Z-axis (θ_z) and Y-axis (θ_y) were expected. One possibility is small measurement errors or noise in the angular velocity data, another factor could be unintended coupling between the rotational axes due to imperfections in the system's dynamics or control inputs and inaccuracies in the integration process used to compute the angles from the angular velocity data could contribute to slight variations. Despite these small deviations, the primary motion aligns with the expected rotations about the Z and Y axes.

3.6 Task 6

We reconstructed the 3D trajectory using kinematic equations of motion, for that we defined the following initial conditions:

- **Velocities:** All initial velocities $(v_{x,0}, v_{y,0}, v_{z,0})$ were set to zero.
- **Positions:** The initial position $(p_{x,0}, p_{y,0}, p_{z,0})$ was also set to zero.
- **Time:** The time (t_0) was initialized to zero.

We computed the filtered accelerations from the sensor frame to the global coordinates using a ZY rotation matrix. Additionally, gravity is subtracted to isolated the acceleration on the object.In this process, we considered only the rotations around the Z and Y axes, as there was no significant rotation about the X axis. This simplification was made to minimize noise and improve the accuracy of the transformation. By excluding the X axis rotation, we reduced the potential impact of small errors or fluctuations in the sensor data, which could otherwise propagate through the calculations and affect the overall trajectory estimation.

Subsequently, the algorithm calculates the velocity and position iteratively. Using a time step derived from the difference between successive timestamps. To ensure accuracy, we computed the velocities and positions using the cumptrapz function which performs numerical integration. We plotted both methods for comparison.

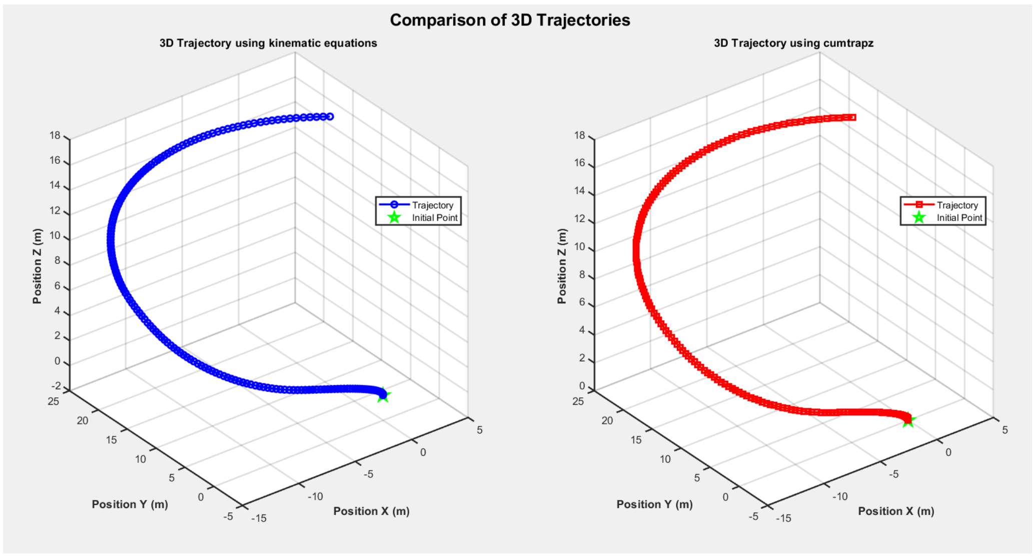

Figure 7: Comparison of 3D Trajectories: Numerical and using function `cumtrapz`

The comparison between the trajectories shows that the method based on numerical iterative integration based on kinematic equations was valid and correctly applied. The close agreement with the `cumtrapz` results confirms that the step-by-step integration accurately reconstructed the 3D trajectory.

Additionally, even after filtering, IMU (Inertial Measurement Unit) accelerometer data can still contain noise or calibration issues, leading to incorrect accelerations and unrealistic position estimates. The gyroscopic data within the IMUs can also have errors, which may further affect the accuracy of angular velocity calculations. The assumption of zero initial velocity and position may not have accurately reflected the true starting conditions, contributing to discrepancies in the trajectory. Numerical integration errors can accumulate over time, causing the trajectory to drift, which is common in iterative methods where small inaccuracies grow with each step. Moreover, the absence of corrections from external sensors such as position sensors or GPS, combined with relying on IMUs alone (without external calibration), can lead to errors in both acceleration and angular velocity estimates, further affecting the accuracy of the trajectory reconstruction.

We designed a 3D of the trajectory, overlaid with local orientation vectors (X, Y, Z axes), was generated. The orientations were represented as arrows at discrete points along the trajectory, calculated using the rotation matrices derived from the orientation angles.

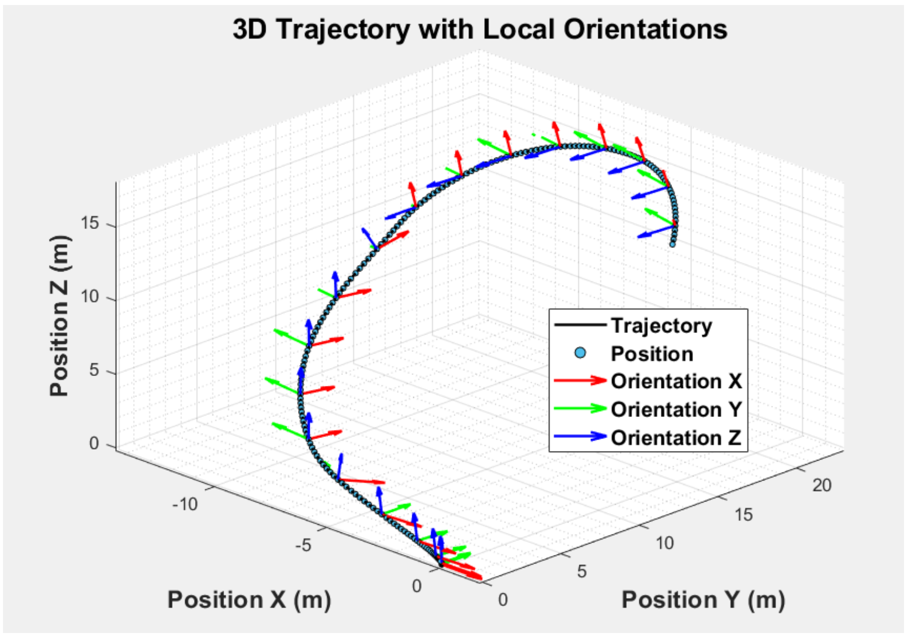

Figure 8: 3D plot of trajectory with orientation vectors

The plot in Figure 8 helps visualize the trajectory and the corresponding orientation vectors. This type of representation is particularly useful in understanding the relationship between the object's path and its orientation over time.

3.7 Task 7

In order to describe the direct kinematics of the Scorbot-ER VII robot we will use the Denavit-Hartenberg convention. Through this approach we will be able to model the relationship between the joints and links of the robot in order to determine the robot's hand position and orientation in world frame.

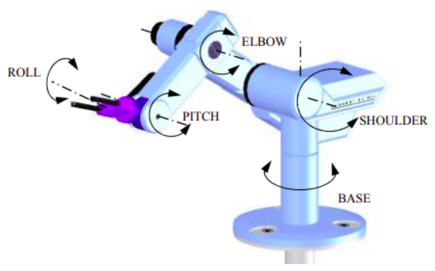

Figure 9: The Scorbot VII robot

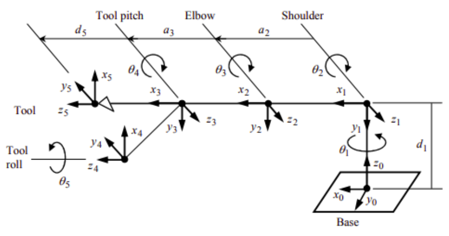

Figure 10: Kinematic diagram of the Scorbot VII robot

3.7.1 Denavit-Hartenberg Parameters and Transformation Matrix Generic Formulas for each Joint

In order to use the convention we need to define four parameters for each joint in the manipulator, which are used to define the transformation matrix between consecutive coordinate frames:

- a_i: Link length (distance along the X-axis).
- α_i: Link twist (angle about the X-axis).
- d_i: Link offset (distance along the Z-axis).
- θ_i: Joint angle (rotation about the Z-axis).

The transformation matrix T_i^{i-1} from frame $i-1$ to frame i is given by:

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

To find the position and orientation of the robot's hand, we need to multiply the transformation matrices from each joint:

$$T_{total} = T_1^0 \cdot T_2^1 \cdot T_3^2 \cdot T_4^3 \cdot T_5^4$$

The final transformation matrix T_{total} will give us the position and orientation of the robot's hand in the world frame. This final matrix is also equal to:

$$T_{total} = \begin{bmatrix} R_{total} & \mathbf{P} \\ 0 & 1 \end{bmatrix}$$

where R_{total} is the rotation matrix and $\mathbf{P} = [x, y, z]^T$ is the position vector of the robot's hand in the world frame.

3.7.2 Substituting the Parameters for the Scorbot-ER VII

From the Scorbot-ER VII manual we obtain the following values:

Joint	θ_i (Angle)(rad)	a_i (Link Length)(m)	α_i (Link Twist)(rad)	d_i (Link Offset)(
Joint 1 (Base)	θ_1	$0,05$	$+\frac{\pi}{2}$	$0,3855$
Joint 2 (Shoulder)	θ_2	0.3	0	0
Joint 3 (Elbow)	θ_3	$0,35$	0	0
Joint 4 (Wrist pitch)	θ_4	0	$-\frac{\pi}{2}$	0
Joint 5 (Wrist roll)	θ_5	0	0	$0,251$

Table 1: Parameters for the Scorbot-ER VII Robot

Now we substitute these values in the matrices for each joint:

$$T_1^0 = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0,05\cos(\theta_1) \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0,05\sin(\theta_1) \\ 0 & 0 & 1 & 0,3855 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^1 = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & 0.3\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & 0.3\sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & 0,35\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0,35\sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^3 = \begin{bmatrix} \cos(\theta_4) & -\sin(\theta_4) & 0 & 0 \\ \sin(\theta_4) & \cos(\theta_4) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_5^4 = \begin{bmatrix} \cos(\theta_5) & -\sin(\theta_5) & 0 & 0 \\ \sin(\theta_5) & \cos(\theta_5) & 0 & 0 \\ 0 & 0 & 1 & 0,251 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

By multiplying all the transformation matrices, we can obtain the total transformation matrix T_{total}, which will, as said before, provide the position and orientation of the robot's hand in the world frame.

3.8 Task 8

Direct Kinematics

To execute the orientation trajectory derived in Task 5, the Scorbot VII requires the reconstructed Euler angles to be converted into encoder values that match its joint-space representation. This involves mapping the calculated orientation angles into the robot's specific actuation system to ensure that the motion aligns with its mechanical configuration.

However, the angular velocity data used to compute the Euler angles suffered from noise and integration errors, introducing inaccuracies. Without advanced filtering and calibration techniques, such as Kalman filters or complementary filters, these errors could propagate into the encoder values.

Inverse Kinematics

The trajectory reconstructed in Task 6 cannot be executed by the Scorbot VII robot due to unrealistic position estimates obtained from the IMU sensors. As noted in Task 6, the IMU data exhibited significant noise, calibration issues, and numerical integration errors. These problems resulted in position values far exceeding the robot's physical capabilities, such as trajectories extending up to 15 meters, which are far beyond the Scorbot VII's operational workspace.

The lack of external calibration or position sensors, such as GPS or optical systems, further amplified these inaccuracies. Without absolute positional references, the trajectory suffered from drift over time, making it unfeasible for execution. As such, the reconstructed trajectory in Cartesian space cannot be achieved by the robot. When the Cartesian coordinates were input into the Scorbot VII, the robot's control system flagged collision warnings and did not move. This behavior indicates that the reconstructed trajectory was not only outside the robot's workspace but also included configurations that violated safety constraints.

However, the orientation of the robot, derived from angular velocity data, remains valid and provides a reliable representation.

Enhancements and Recommendations

The Scorbot VII, in its current state, cannot execute the full trajectory reconstructed in Task 6. The unrealistic and inaccurate position estimates make the Cartesian trajectory unachievable. However, the reconstructed orientation trajectory in task 5, while more reliable, requires further refinement and proper mapping to the robot's joint space to be practically executable. To improve the execution of trajectories and ensure the robot's capability to follow them:

- **Use of Position Sensors:** Adding GPS, optical tracking systems, or other absolute position sensors would complement the IMU data and reduce cumulative errors during integration.
- **Advanced Filtering Techniques:** Employing sensor fusion methods like Extended Kalman Filters (EKF) or complementary filters can combine IMU data with external sensors to improve accuracy.
- **Realistic Constraints:** Incorporating the robot's physical constraints into the trajectory reconstruction process would help in discarding unrealistic positions and focusing on achievable movements.
- **Improved Calibration:** Regular calibration of IMUs and other sensors would reduce systematic errors and enhance the accuracy of motion estimates.

This analysis highlights the importance of accurate and well-calibrated sensor data for both inverse and direct kinematics, underscoring the challenges and limitations of relying solely on noisy IMU data for trajectory and orientation reconstruction.

4 Conclusion

The Scorbot VII robot, in its current configuration, cannot fully execute the reconstructed trajectory derived in Task 6. The Cartesian trajectory is unrealistic and unachievable due to significant inaccuracies in the IMU sensor data, such as noise, calibration issues, and integration drift. These inaccuracies result in position estimates far beyond the robot's physical workspace, making direct implementation infeasible.

However, the reconstructed orientation trajectory in task 5 provides a more reliable representation of the robot's motion. For practical execution, this trajectory would need to be refined, filtered, and converted into the robot's joint-space representation using encoder values.

To enhance trajectory execution feasibility, the integration of position sensors, improved filtering techniques, adherence to physical constraints, and sensor recalibration are recommended. These improvements would ensure more accurate and realistic trajectory data, enabling the Scorbot VII to effectively follow both orientation and positional commands in future experiments.