TÉCNICO LISBOA

# 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** **2**

**2 Introduction** **2**

**3 Tasks** **3**
3.1 Task 1 . . . 3
3.2 Task 2 . . . 6
3.3 Task 3 . . . 8
3.3.1 Theory and Mathematical Formulation . . . 9
3.3.2 Effect of Gravity . . . 9
3.4 Task 4 . . . 10
3.4.1 Parameterization . . . 10
3.4.2 Orientation Representation Using Rotation Matrix . . . 11
3.4.3 Relating Orientation and Angular Velocity . . . 11
3.4.4 Angular Velocity in the Body Frame . . . 12
3.4.5 Reconstructing the Trajectory of the Sensor in Orientation Space . . . 12
3.4.6 Numerical Integration of Euler Angles . . . 13
3.5 Task 5 . . . 13
3.6 Task 6 . . . 13
3.7 Task 7 . . . 13
3.8 Task 8 . . . 13

**4 Conclusion** **13**

**Appendices** **14**

# 1 Group Members Contribution

# 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.

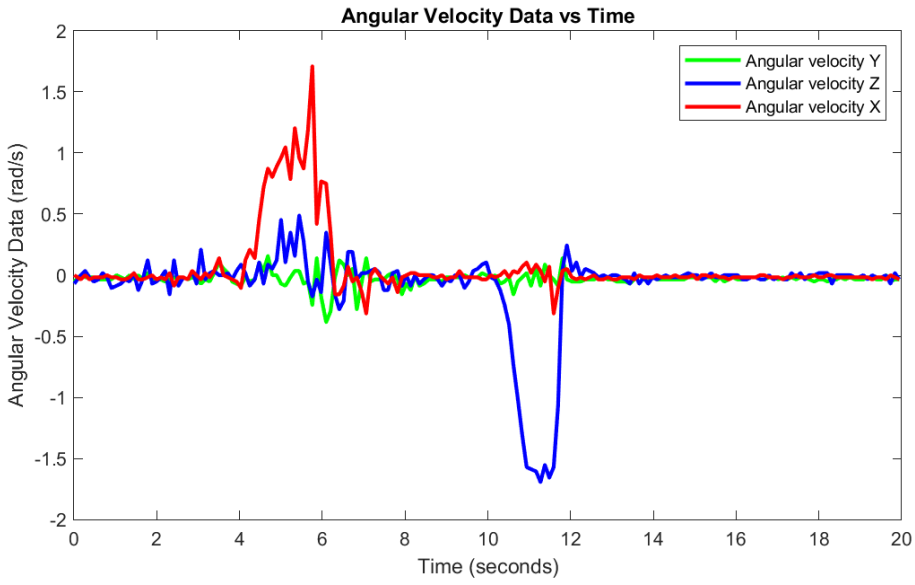


**Figure 1:** Angular velocity vs time.

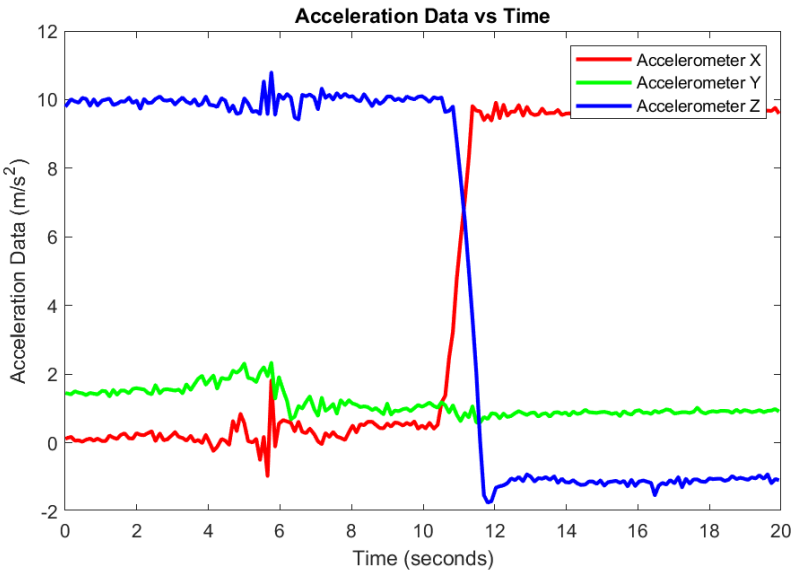


**Figure 2:** Acceleration vs time

## Z-Axis (Blue Line)

In the acceleration graph, the blue line starts near $10\,\mathrm{m/s^2}$, which corresponds to the gravitational acceleration. This confirms that the blue line represents the $z$-axis, as it aligns with gravity during the static phase of the robotic arm.

In the angular velocity graph, the blue line represents $\omega_z$ (angular velocity around the $z$-axis). As expected, there is a strong correlation between the changes in acceleration and angular velocity along this axis:

- For example, at around 12 seconds, both acceleration and angular velocity show a significant negative dip, confirming the $z$-axis mapping.

## X-Axis (Red Line)

In the acceleration graph, the red line shows values near $0\,\mathrm{m/s^2}$ during the static phase, confirming that it is not aligned with gravity. However, during the dynamic phase (after $\sim 5$ seconds), it shows significant positive and negative variations due to movement along this axis.

In the angular velocity graph, the red line represents $\omega_x$ (angular velocity around the $x$-axis). We observe that:

- When the angular velocity ($\omega_x$) increases positively (e.g., around 6 seconds), the corresponding acceleration ($a_x$) also shows a positive variation.
- This synchronized behavior validates that the red lines in both graphs represent the $x$-axis.

## Y-Axis (Green Line)

In the acceleration graph, the green line remains near a low positive value during the static phase, indicating that it is neither aligned with gravity nor orthogonal to it. During the dynamic phase, it exhibits moderate oscillations due to translational and rotational effects.

In the angular velocity graph, the green line represents $\omega_y$ (angular velocity around the $y$-axis). Observing the dynamic phase:

- Positive peaks in $\omega_y$ (e.g., around 6 seconds) correspond to similar positive peaks in $a_y$, confirming the mapping of the green line to the $y$-axis.

## Alignment in Variation

For all three axes ($x$, $y$, and $z$), we observe that when the angular velocity increases positively, the corresponding acceleration also increases positively. Similarly, when the angular velocity decreases or becomes negative, the corresponding acceleration exhibits a similar negative variation. This consistent relationship occurs because:

### Rotational Dynamics Influence Linear Dynamics

- In a robotic arm, each joint's motion generates a combination of tangential and centripetal accelerations, which are directly tied to the angular velocity and its rate of change.
- The tangential acceleration depends on the rate of change of angular velocity ($\dot{\omega}$), while the centripetal acceleration depends on the square of angular velocity ($\omega^2$) and acts perpendicular to the direction of rotation.

### Controlled System Dynamics

- Robotic arms are designed for precise and coordinated movements. Each joint is controlled by motors that apply predictable torques, resulting in synchronized changes in angular velocity and acceleration.
- When a joint rotates positively (e.g., $\omega_x > 0$), it induces a linear acceleration along the same axis as part of the controlled rotational motion.

### Fixed Axis of Rotation

- Unlike free-moving objects, the axes of rotation in a robotic arm are fixed, ensuring that the measured angular velocities and accelerations align in their respective directions without interference from external or random forces.

This predictable alignment between angular velocity and acceleration highlights the consistency of the robotic arm's motion and validates the observed variations in the provided graphs.

## 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.

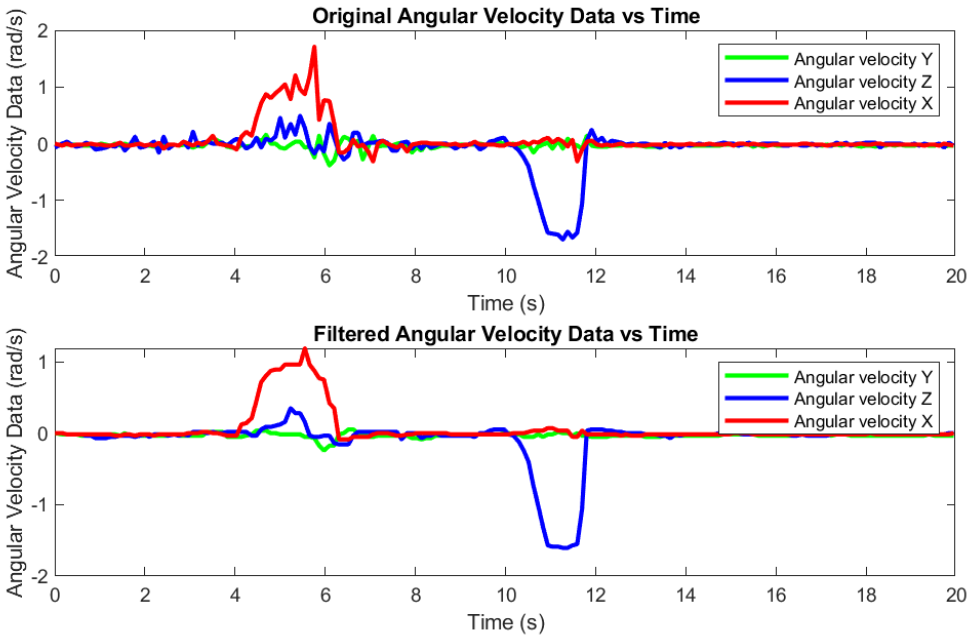


**Figure 3:** Filtered angular velocity vs original (using median filter).

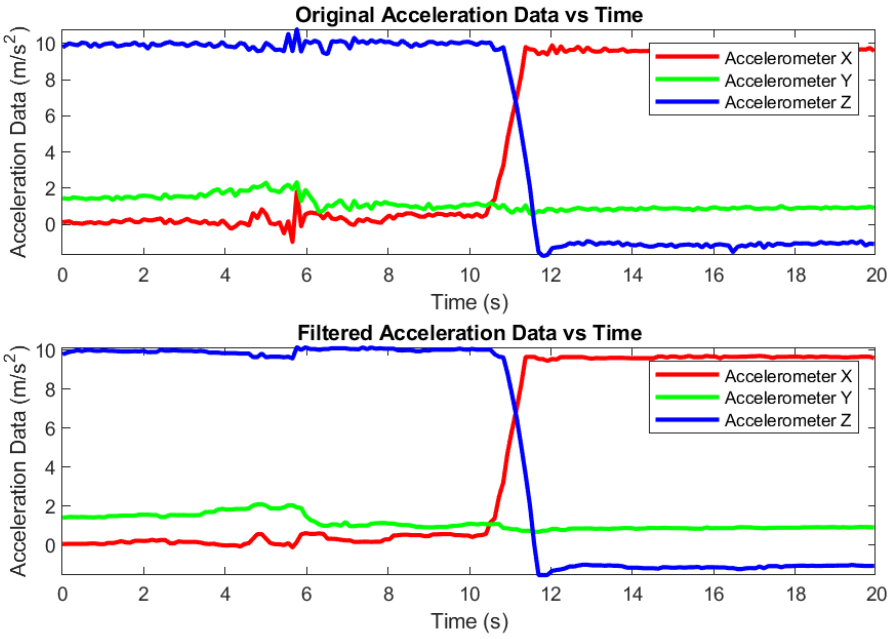


**Figure 4:** Filtered acceleration vs original (using median filter)

## 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 ($\sigma$), 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., $\sigma$ 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

- $\mathbf{X}$ — Position on the global x axis.
- $\mathbf{Y}$ — Position on the global y axis.
- $\mathbf{Z}$ — 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).
- $\Delta t$: 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 $(\alpha, \beta, \gamma)$. 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:

- $\alpha, \beta, \gamma$: 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.
- $\Delta t$: Time interval between consecutive measurements.

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

- **Yaw** $(\alpha)$ — rotation about the global Z-axis. This initial rotation aligns the sensor to face a desired target or orientation.
- **Pitch** $(\beta)$ — rotation about the new Y-axis after the yaw rotation. This step adjusts the sensor's vertical alignment relative to the target.
- **Roll** $(\gamma)$ — 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 $\boldsymbol{\Omega}_{\text{body}} = [\omega_x, \omega_y, \omega_z]^T$ is given by:

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

where $\Omega$ 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 $\Omega$ 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, $\boldsymbol{\Omega}_{\text{body}} = [\omega_x, \omega_y, \omega_z]^T$, 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:

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

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

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

### 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 $\alpha, \beta, \gamma$) 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 $\Delta t$ 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

### 3.6 Task 6

### 3.7 Task 7

### 3.8 Task 8

## 4 Conclusion

# Appendices