Attitude and Heading Reference Systems (AHRS)

Overview

An Attitude and Heading Reference System (AHRS) is a sensor fusion system that estimates the 3D orientation (attitude) of an object using measurements from:

- **Gyroscope** – measures angular velocity
- **Accelerometer** – measures specific force (gravity + linear acceleration)
- **Magnetometer** – measures Earth's magnetic field (for yaw/heading estimation)

The AHRS provides orientation in terms of **Euler angles** (roll, pitch, yaw) or **quaternions**.

Motivation

- **Gyroscopes** provide accurate short-term angular rate measurements, but suffer from drift over time due to bias.
- **Accelerometers** measure gravity but are affected by dynamic motion.
- **Magnetometers** provide absolute heading but are sensitive to magnetic disturbances.

Combining these sensors allows robust and drift-compensated orientation estimation.

Common AHRS Algorithms

1. Complementary Filter

A simple and efficient method that combines gyroscope integration with accelerometer and magnetometer corrections. The filter blends high-frequency components of the gyroscope and low-frequency components of the accelerometer/magnetometer:

$$\theta_{\text{filtered}} = \alpha \cdot \theta_{\text{gyro}} + (1 - \alpha) \cdot \theta_{\text{acc/mag}}$$

Pros: Lightweight, real-time capable
Cons: Requires tuning, less optimal under rapid motion

2. Extended Kalman Filter (EKF)

A probabilistic filter that uses a nonlinear process model and a measurement update to optimally estimate orientation:

- Predicts orientation from gyroscope
- Corrects using accelerometer and magnetometer

Pros: Handles sensor noise and biases
Cons: Computationally heavier, requires good models

3. Madgwick Filter

A gradient-descent-based filter designed for fast and efficient orientation estimation using quaternions.

- Optimizes orientation to minimize the error between measured and expected sensor readings

Pros: Accurate and computationally efficient
Cons: Requires tuning of gain parameters

4. Mahony Filter

This filter uses a control-based approach with proportional and integral terms:

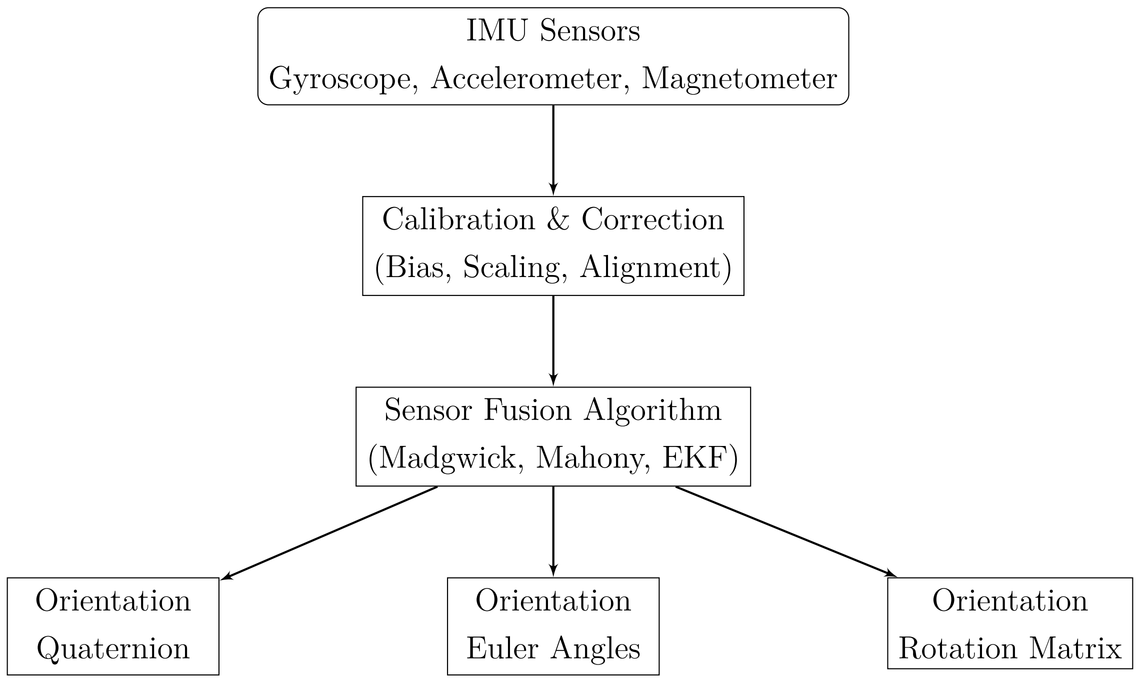
- Gyro integration is corrected using feedback from accelerometer and magnetometer

Pros: Efficient and robust
Cons: Still susceptible to magnetic interference

Applications

- Unmanned Aerial Vehicles (UAVs)
- Robotics and automation
- Augmented and Virtual Reality
- Automotive stability and navigation systems
- Smartphones and wearable devices

AHRS System Overview (Block Diagram)



Description:

- The IMU continuously provides raw measurements.
- The sensor fusion algorithm processes the data to estimate orientation.
- The output can be expressed as a quaternion or converted to roll–pitch–yaw (Euler angles) for control or display.

Coordinate Systems for Orientation and Navigation

Orientation and motion are always described relative to a coordinate system. Different disciplines adopt different conventions, which can lead to confusion if not clearly specified. This section summarizes the most commonly used coordinate systems.

1. ENU: East-North-Up

The **East-North-Up (ENU)** system is commonly used in robotics, mobile devices, and geographic coordinate conversions. It is a local tangent frame with:

- x-axis: points **east**
- y-axis: points **north**
- z-axis: points **upward** (away from the Earth)

Applications:

- GPS/INS integration in terrestrial vehicles
- Mapping and SLAM in robotics
- Android and mobile phone sensors

Right-handed system: $\hat{x} \times \hat{y} = \hat{z}$

2. NED: North-East-Down

The **North-East-Down (NED)** system is widely used in aerospace and marine applications. It defines a local frame with:

- x-axis: points **north**
- y-axis: points **east**
- z-axis: points **downward** (toward the center of the Earth)

Applications:

- UAV and aircraft dynamics
- Flight controllers and navigation software
- Marine navigation systems

Right-handed system: $\hat{x} \times \hat{y} = \hat{z}$

3. ECEF: Earth-Centered, Earth-Fixed

The **ECEF** frame is a global reference frame fixed to the Earth's surface:

- Origin: Earth's center of mass
- x-axis: through the intersection of the equator and prime meridian
- y-axis: 90° east along the equator
- z-axis: through the North Pole

Applications:

- Global navigation satellite systems (GNSS)
- Satellite positioning
- Transformations with geodetic coordinates (latitude, longitude, altitude)

4. Body Frame

The **body frame** is a moving coordinate system fixed to the rigid body (e.g., aircraft, UAV, robot). Its orientation and position vary over time.

Typical convention in aerospace:

- x-axis: points forward (nose)
- y-axis: points to the right (starboard)

- z-axis: points downward

Applications:

- Control and navigation of UAVs and vehicles
- Sensor mounting frames (IMU, cameras)

Note: Roll, pitch, and yaw are usually defined relative to the body frame axes.

5. Navigation vs. Sensor Frame

It is often necessary to distinguish between:

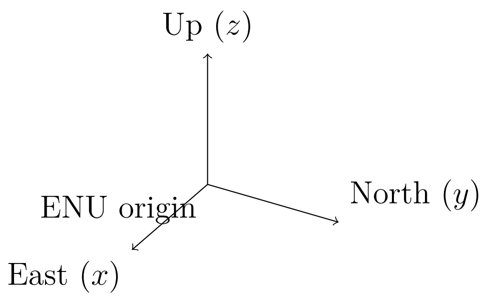
- **Navigation frame:** Earth-fixed (e.g., ENU, NED, or ECEF)
- **Sensor frame:** Fixed to the inertial measurement unit (IMU)

Transformations between these frames are typically handled via rotation matrices or quaternions. It is important to clearly define the frame in which a rotation or acceleration is expressed.

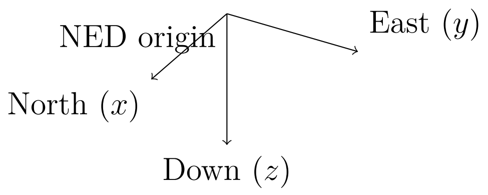
Frame	**x-axis**	**y-axis**	**z-axis**
ENU	East	North	Up
NED	North	East	Down
ECEF	Equator/Prime Meridian	Equator/90°E	North Pole
Body (Aircraft)	Forward	Right	Down

Coordinate System Diagrams

ENU Frame



NED Frame



Coordinate Frame Transformations

1. ENU $\leftrightarrow$ NED

The ENU and NED frames differ primarily in axis naming and orientation. The transformation matrix from ENU to NED is:

$$\mathbf{T}_{\text{ENU}\rightarrow\text{NED}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

To convert a vector $\mathbf{v}_{\text{ENU}}$ to NED:

$$\mathbf{v}_{\text{NED}} = \mathbf{T}_{\text{ENU}\rightarrow\text{NED}} \cdot \mathbf{v}_{\text{ENU}}$$

The inverse is identical:

$$\mathbf{T}_{\text{NED}\rightarrow\text{ENU}} = \mathbf{T}_{\text{ENU}\rightarrow\text{NED}}^{T} = \mathbf{T}_{\text{ENU}\rightarrow\text{NED}}$$

2. ECEF $\leftrightarrow$ ENU

Let λ be the latitude and ϕ the longitude of the local origin. The rotation matrix to convert from ECEF to ENU is:

$$\mathbf{T}_{\text{ECEF}\rightarrow\text{ENU}} = \begin{bmatrix} -\sin\phi & \cos\phi & 0 \\ -\sin\lambda\cos\phi & -\sin\lambda\sin\phi & \cos\lambda \\ \cos\lambda\cos\phi & \cos\lambda\sin\phi & \sin\lambda \end{bmatrix}$$

To convert a point $\mathbf{p}_{\text{ECEF}}$ to ENU:

$$\mathbf{p}_{\text{ENU}} = \mathbf{T}_{\text{ECEF}\rightarrow\text{ENU}} \cdot (\mathbf{p}_{\text{ECEF}} - \mathbf{p}_0)$$

Where $\mathbf{p}_0$ is the ECEF coordinate of the local reference origin.

3. Body $\leftrightarrow$ Navigation Frame (ENU or NED)

The orientation of the body frame with respect to the navigation frame is typically represented using a rotation matrix R_b^n or a quaternion q_b^n. For a vector expressed in the body frame:

$$\mathbf{v}_{\text{nav}} = R_b^n \cdot \mathbf{v}_{\text{body}} \quad \text{or} \quad \mathbf{v}_{\text{nav}} = q_b^n \otimes \mathbf{v}_{\text{body}} \otimes q_b^{n*}$$

Where:

- R_b^n: rotation matrix from body to navigation frame
- q_b^n: quaternion from body to navigation
- q_b^{n*}: conjugate of the quaternion

The inverse transformation (navigation to body frame) is given by $(R_b^n)^T$ or $(q_b^n)^*$.

Example: Transforming a Vector from Body to ECEF Frame

Suppose a UAV is flying at a location with:

- Latitude: $\lambda = 52°$
- Longitude: $\phi = 4°$
- Altitude: $h = 100\,\mathrm{m}$

Let the UAV measures an acceleration vector in the **body frame**:

$$\mathbf{a}_{\text{body}} = \begin{bmatrix} 0.1 \\ 0.0 \\ -9.7 \end{bmatrix} \quad (\mathrm{m/s^2})$$

where:

- x: forward
- y: right
- z: down

Assume the orientation of the UAV relative to ENU is given by a rotation matrix R_b^e (body to ENU):

$$R_b^e = \begin{bmatrix} 0.0 & 1.0 & 0.0 \\ 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0 \end{bmatrix}$$

This corresponds to a 90° rotation around the body's z-axis (yaw).

Step 1: Convert the vector to ENU frame

$$\mathbf{a}_{\text{ENU}} = R_b^e \cdot \mathbf{a}_{\text{body}} = \begin{bmatrix} 0.0 & 1.0 & 0.0 \\ 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & -1.0 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.0 \\ -9.7 \end{bmatrix} = \begin{bmatrix} 0.0 \cdot 0.1 + 1.0 \cdot 0.0 + 0.0 \cdot (-9.7) \\ 1.0 \cdot 0.1 + 0.0 \cdot 0.0 + 0.0 \cdot (-9.7) \\ 0.0 \cdot 0.1 + 0.0 \cdot 0.0 + (-1.0) \cdot (-9.7) \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.1 \\ 9.7 \end{bmatrix}$$

Step 2: Convert from ENU to ECEF

We use the transformation matrix $T_{\text{ENU}\to\text{ECEF}} = T^T_{\text{ECEF}\to\text{ENU}}$. First compute $T_{\text{ECEF}\to\text{ENU}}$ from latitude $\lambda = 52°$, longitude $\phi = 4°$:

$$\lambda = 0.907\,\text{rad}, \quad \phi = 0.0698\,\text{rad}$$

$$T_{\text{ECEF}\to\text{ENU}} = \begin{bmatrix} -\sin\phi & \cos\phi & 0 \\ -\sin\lambda\cos\phi & -\sin\lambda\sin\phi & \cos\lambda \\ \cos\lambda\cos\phi & \cos\lambda\sin\phi & \sin\lambda \end{bmatrix} \approx \begin{bmatrix} -0.0698 & 0.9976 & 0 \\ -0.7826 & -0.0547 & 0.6191 \\ 0.6182 & 0.0395 & 0.7842 \end{bmatrix}$$

Transpose it:

$$T_{\text{ENU}\to\text{ECEF}} = T^T_{\text{ECEF}\to\text{ENU}} \approx \begin{bmatrix} -0.0698 & -0.7826 & 0.6182 \\ 0.9976 & -0.0547 & 0.0395 \\ 0 & 0.6191 & 0.7842 \end{bmatrix}$$

Final conversion:

$$\mathbf{a}_{\text{ECEF}} = T_{\text{ENU}\to\text{ECEF}} \cdot \mathbf{a}_{\text{ENU}} = \begin{bmatrix} -0.0698 & -0.7826 & 0.6182 \\ 0.9976 & -0.0547 & 0.0395 \\ 0 & 0.6191 & 0.7842 \end{bmatrix} \begin{bmatrix} 0.0 \\ 0.1 \\ 9.7 \end{bmatrix}$$

$$= \begin{bmatrix} -0.7826 \cdot 0.1 + 0.6182 \cdot 9.7 \\ -0.0547 \cdot 0.1 + 0.0395 \cdot 9.7 \\ 0.6191 \cdot 0.1 + 0.7842 \cdot 9.7 \end{bmatrix} \approx \begin{bmatrix} 5.94 \\ 0.37 \\ 7.68 \end{bmatrix} \text{ m/s}^2$$

Conclusion: The UAV's acceleration vector, originally in the body frame, is now represented in the global ECEF frame, which is suitable for combining with global navigation data like satellite positions or velocity estimates.

Example: Transforming a Vector from ENU to NED Frame

Suppose a ground vehicle's velocity vector is expressed in the ENU frame as:

$$\mathbf{v}_{\text{ENU}} = \begin{bmatrix} 5.0 \\ 2.0 \\ -0.5 \end{bmatrix} \quad (\text{m/s})$$

This means:

- 5.0 m/s toward the **east**

- 2.0 m/s toward the **north**
- −0.5 m/s **upward**

We want to express this same vector in the NED frame, which has:

- x: north
- y: east
- z: down

Step 1: Use the ENU→NED transformation matrix

$$T_{\text{ENU}\to\text{NED}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Step 2: Apply the transformation

$$\mathbf{v}_{\text{NED}} = T_{\text{ENU}\to\text{NED}} \cdot \mathbf{v}_{\text{ENU}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 5.0 \\ 2.0 \\ -0.5 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 5.0 \\ 0.5 \end{bmatrix} \quad \text{(m/s)}$$

Interpretation:

- 2.0 m/s toward the **north**
- 5.0 m/s toward the **east**
- 0.5 m/s **downward**

This is consistent with the coordinate system definition of NED: the upward vertical velocity in ENU becomes a downward component in NED (sign flip on z).

Step 3: Verify invertibility

To convert back, use the same matrix:

$$\mathbf{v}_{\text{ENU}} = T_{\text{ENU}\to\text{NED}} \cdot \mathbf{v}_{\text{NED}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} 2.0 \\ 5.0 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 5.0 \\ 2.0 \\ -0.5 \end{bmatrix} \quad \checkmark$$

Orientation Representations and Conversions

The orientation of a rigid body in three-dimensional space can be described using multiple mathematical representations. Each has its own strengths and is suited for different applications. Below we describe the most common representations, including detailed formulas and symbol definitions.

1. Euler Angles

Euler angles represent orientation through a sequence of three rotations about specified axes. The most common sequence is `ZYX` (yaw–pitch–roll), where:

- ϕ: roll angle (rotation about body-fixed x-axis)
- θ: pitch angle (rotation about intermediate y-axis)
- ψ: yaw angle (rotation about inertial z-axis)

The overall rotation matrix is given by:

$$R = R_z(\psi) R_y(\theta) R_x(\phi)$$

With:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Properties:

- Uses 3 parameters (minimal representation)
- Intuitive for human interpretation
- Suffers from *gimbal lock* when $\theta = \pm 90°$

Inverse Conversion (Rotation Matrix to Euler angles - ZYX sequence):

$$\theta = \arcsin(-R_{31}), \quad \phi = \arctan 2(R_{32}, R_{33}), \quad \psi = \arctan 2(R_{21}, R_{11})$$

2. Rotation Matrix

A rotation matrix $R \in \mathbb{R}^{3\times 3}$ is an orthonormal matrix satisfying:

$$R^T R = I, \quad \det(R) = 1$$

Properties:

- Redundant: 9 elements but only 3 degrees of freedom
- Free from singularities
- Easy to apply to vectors via multiplication: $\mathbf{v}_{\text{rotated}} = R\mathbf{v}$

3. Quaternions

A unit quaternion is a 4D vector that encodes 3D orientation without singularities. Defined as:

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) \\ u_x \sin\left(\frac{\theta}{2}\right) \\ u_y \sin\left(\frac{\theta}{2}\right) \\ u_z \sin\left(\frac{\theta}{2}\right) \end{bmatrix}$$

Where:

- θ: rotation angle
- $\hat{\mathbf{u}} = [u_x, u_y, u_z]^T$: unit rotation axis

Quaternion to Rotation Matrix:

$$R = \begin{bmatrix} 1 - 2(q_2^2 + q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 1 - 2(q_1^2 + q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 1 - 2(q_1^2 + q_2^2) \end{bmatrix}$$

Rotation Matrix to Quaternion (trace method):

$$q_0 = \tfrac{1}{2}\sqrt{1 + R_{11} + R_{22} + R_{33}}, \quad q_1 = \frac{R_{32} - R_{23}}{4q_0}, \quad q_2 = \frac{R_{13} - R_{31}}{4q_0}, \quad q_3 = \frac{R_{21} - R_{12}}{4q_0}$$

Properties:

- 4 elements, 3 DoF (must be normalized)
- No gimbal lock
- Suitable for interpolation (e.g., slerp)
- Double cover: q and $-q$ represent the same rotation

4. Axis-Angle Representation

Describes a rotation by an axis $\hat{\mathbf{u}} = [u_x, u_y, u_z]^T$ and an angle θ. The axis is a unit vector, and the rotation occurs counterclockwise about $\hat{\mathbf{u}}$.

Rodrigues' Formula:

$$R = I + \sin\theta[\hat{u}]_\times + (1 - \cos\theta)[\hat{u}]_\times^2$$

where the skew-symmetric matrix $[\hat{u}]_\times$ is:

$$[\hat{u}]_\times = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix}$$

Properties:

- 3 parameters (minimal)
- Intuitive geometrical meaning
- Singular at $\theta = 0$

5. Exponential Coordinates (Rotation Vector)

The rotation vector is defined as $\boldsymbol{\omega} = \theta\hat{\mathbf{u}} \in \mathbb{R}^3$. The matrix exponential gives:

$$R = \exp([\boldsymbol{\omega}]_\times) = I + \frac{\sin\theta}{\theta}[\boldsymbol{\omega}]_\times + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\omega}]_\times^2$$

where $\theta = \|\boldsymbol{\omega}\|$, and $\hat{\mathbf{u}} = \boldsymbol{\omega}/\theta$.

Properties:

- Minimal and smooth
- Foundation for Lie group theory in SO(3)
- Requires matrix exponential/logarithm for conversion

6. Homogeneous Transformation Matrix

Combines rotation R and translation $\mathbf{t} \in \mathbb{R}^3$ into a single matrix:

$$T = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3)$$

Used to represent full pose (position + orientation) in robotics.

Summary of Key Symbols

- ϕ: roll (rotation about x-axis)
- θ: pitch (rotation about y-axis)
- ψ: yaw (rotation about z-axis)
- $\hat{\mathbf{u}}$: unit rotation axis
- θ: rotation angle

- $\boldsymbol{\omega}$: rotation vector (exponential coordinates)
- q_0, q_1, q_2, q_3: quaternion components
- R: rotation matrix
- $[\cdot]_\times$: skew-symmetric matrix operator

Complete Conversion Formulas Between Representations

Below is a summary of key conversion formulas between the major orientation representations.

1. Euler Angles $\leftrightarrow$ Rotation Matrix (ZYX sequence)

Euler to Rotation Matrix:

$$R = R_z(\psi)R_y(\theta)R_x(\phi)$$

Rotation Matrix to Euler (ZYX):

$$\theta = \arcsin(-R_{31})$$

$$\phi = \arctan 2(R_{32}, R_{33}), \quad \psi = \arctan 2(R_{21}, R_{11})$$

2. Rotation Matrix $\leftrightarrow$ Quaternion

Rotation Matrix to Quaternion (trace method):

$$\mathrm{tr} = R_{11} + R_{22} + R_{33}$$

$$q_0 = \frac{1}{2}\sqrt{1+\mathrm{tr}}, \quad q_1 = \frac{R_{32} - R_{23}}{4q_0}, \quad q_2 = \frac{R_{13} - R_{31}}{4q_0}, \quad q_3 = \frac{R_{21} - R_{12}}{4q_0}$$

Quaternion to Rotation Matrix:

$$R = \begin{bmatrix} 1 - 2(q_2^2 + q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & 1 - 2(q_1^2 + q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & 1 - 2(q_1^2 + q_2^2) \end{bmatrix}$$

3. Quaternion ↔ Axis-Angle

Quaternion to Axis-Angle:

$$\theta = 2\arccos(q_0), \quad \hat{\mathbf{u}} = \frac{1}{\sin(\theta/2)}[q_1, q_2, q_3]^T$$

Axis-Angle to Quaternion:

$$q_0 = \cos\left(\frac{\theta}{2}\right), \quad \mathbf{q}_v = \hat{\mathbf{u}}\sin\left(\frac{\theta}{2}\right)$$

4. Axis-Angle ↔ Rotation Matrix

Axis-Angle to Rotation Matrix (Rodrigues' formula):

$$R = I + \sin\theta[\hat{u}]_\times + (1-\cos\theta)[\hat{u}]_\times^2$$

Rotation Matrix to Axis-Angle:

$$\theta = \arccos\left(\frac{\mathrm{tr}(R)-1}{2}\right)$$

$$\hat{\mathbf{u}} = \frac{1}{2\sin\theta}\begin{bmatrix} R_{32}-R_{23} \\ R_{13}-R_{31} \\ R_{21}-R_{12} \end{bmatrix}$$

5. Quaternion ↔ Euler Angles (ZYX)

Quaternion to Euler:

$$\phi = \arctan 2(2(q_0q_1 + q_2q_3), 1 - 2(q_1^2 + q_2^2))$$

$$\theta = \arcsin(2(q_0q_2 - q_3q_1))$$

$$\psi = \arctan 2(2(q_0q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2))$$

Euler to Quaternion:

$$q_0 = \cos\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right)$$

$$q_1 = \sin\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) - \cos\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right)$$

$$q_2 = \cos\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right) + \sin\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right)$$

$$q_3 = \cos\left(\frac{\phi}{2}\right)\cos\left(\frac{\theta}{2}\right)\sin\left(\frac{\psi}{2}\right) - \sin\left(\frac{\phi}{2}\right)\sin\left(\frac{\theta}{2}\right)\cos\left(\frac{\psi}{2}\right)$$

6. Rotation Vector (Exponential Coordinates) $\leftrightarrow$ Rotation Matrix

Let $\boldsymbol{\omega} = \theta\hat{\mathbf{u}} \in \mathbb{R}^3$.

Rotation Vector to Matrix:

$$R = \exp([\boldsymbol{\omega}]_\times) = I + \frac{\sin\theta}{\theta}[\boldsymbol{\omega}]_\times + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\omega}]_\times^2$$

Rotation Matrix to Rotation Vector:

$$\theta = \arccos\left(\frac{\mathrm{tr}(R)-1}{2}\right), \quad \hat{\mathbf{u}} = \frac{1}{2\sin\theta}\begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix}$$

$$\boldsymbol{\omega} = \theta\hat{\mathbf{u}}$$

Inertial Measurement Units (IMUs)

An **Inertial Measurement Unit (IMU)** is an electronic device that measures and reports a body's motion and orientation using a combination of inertial sensors. It is the primary source of raw data used in Attitude and Heading Reference Systems (AHRS), navigation, robotics, and control applications.

1. Core Sensors in an IMU

Most IMUs contain the following types of sensors:

- **Gyroscope:** Measures angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$ (rad/s). Each axis output corresponds to the rate of rotation about the x, y, z axes of the sensor frame.
- **Accelerometer:** Measures specific force $\mathbf{f} \in \mathbb{R}^3$ (m/s^2), including gravity. It senses linear acceleration along each axis.
- **Magnetometer:** Measures the local magnetic field $\mathbf{B} \in \mathbb{R}^3$ (µT), which can be used to determine heading relative to the Earth's magnetic north.

An IMU that includes all three types is often called a **9-axis IMU**. Devices with only a gyroscope and accelerometer are called **6-axis IMUs**.

2. Sensor Frame and Mounting Alignment

IMU data is reported in its own body-fixed coordinate system, often called the **sensor frame**. To interpret IMU data correctly, one must know how this frame is oriented relative to the vehicle or robot's body frame.

- Misalignment between sensor and body axes can lead to incorrect orientation estimation.
- Calibration matrices or manual alignment must be applied if the IMU is not aligned with the vehicle axes.

3. IMU Sampling and Integration

IMUs typically sample data at high frequencies (100 Hz–1000 Hz or more). The gyroscope readings can be integrated over time to estimate orientation:

$$\mathbf{R}(t) = \mathbf{R}(t_0) \cdot \exp\left([\boldsymbol{\omega}(t)]_\times \cdot (t - t_0)\right)$$

Problem: Integration of angular velocity causes drift due to gyroscope bias and noise.

Solution: Combine gyroscope data with accelerometer and magnetometer using sensor fusion algorithms (e.g., Complementary Filter, Kalman Filter, Madgwick, Mahony).

4. Typical Error Sources

- **Bias:** Constant offset in sensor output (especially gyroscope). Causes slow drift over time if not corrected.
- **Scale factor errors:** Deviations in the sensitivity of the sensor.
- **Noise:** High-frequency fluctuations due to electronics or vibrations.
- **Non-orthogonality:** Slight axis misalignments inside the MEMS sensor.
- **Temperature sensitivity:** Readings vary with temperature if not compensated.
- **Magnetic disturbances:** Affect magnetometer accuracy (e.g., near motors or metal).

5. Practical Considerations for IMU Use

- **Calibration:** It is essential to calibrate all three sensors to correct for bias, scale, and misalignment.
- **Mounting location:** Preferably near the center of mass to reduce rotational coupling.
- **Sampling rate:** Must be high enough to capture fast dynamics (usually $\geq 100\,\text{Hz}$).
- **Filtering:** Use low-pass filters to remove noise and fusion algorithms to prevent drift.

- **Synchronization:** All sensors in the IMU should be synchronized; otherwise, fusion quality degrades.

6. Output Data Formats

IMUs typically provide:

- Raw 3D vectors for angular velocity, acceleration, and magnetic field
- Sometimes: temperature, pressure, orientation estimate (quaternion or Euler angles)
- In some devices: built-in fusion outputs via internal AHRS algorithms

7. Applications of IMUs

- UAV attitude estimation and flight control
- Mobile phone screen orientation and motion tracking
- VR/AR headsets and motion controllers
- Automotive inertial navigation systems (INS)
- Robot motion and localization

IMU Case Studies: MPU-9250 and LSM9DS1

This section presents two widely-used low-cost Inertial Measurement Units (IMUs): the **MPU-9250** by InvenSense and the **LSM9DS1** by STMicroelectronics. Each integrates a 3-axis gyroscope, 3-axis accelerometer, and 3-axis magnetometer.

1. MPU-9250

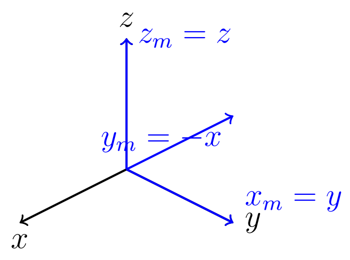
The MPU-9250 is a 9-axis MEMS IMU featuring:

- 3-axis gyroscope (± 250 to ± 2000 °/s)
- 3-axis accelerometer (± 2 to ± 16 g)
- 3-axis magnetometer (AK8963, ± 4800 µT)
- I^2C/SPI interface
- 16-bit ADCs, internal Digital Motion Processor (DMP)

Sensor Frame Alignment:

The sensor frame (looking top-down on the chip) is defined as:

- x: points to the right
- y: points forward (away from pin 1)
- z: points upward (out of the PCB)



Remarks:

- The magnetometer (AK8963) is internally mounted at a 90° rotation compared to the accelerometer/gyro.
- Sensor fusion requires realignment of the magnetometer frame to match the accelerometer/gyro frame.

Use Cases:

- Widely used in Pixhawk flight controllers and open-source autopilots (e.g., ArduPilot, PX4)
- Mobile robot navigation and localization
- Open-source projects: Arduino, Raspberry Pi, ESP32

2. LSM9DS1

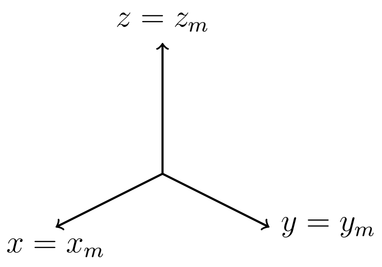
The LSM9DS1 is a 9-axis IMU with:

- 3-axis gyroscope (±245 to ±2000 °/s)
- 3-axis accelerometer (±2 to ±16 g)
- 3-axis magnetometer (±4 to ±16 gauss)
- Separate internal dies for accel/gyro and magnetometer
- I^{2}C/SPI interface

Sensor Frame Alignment:

ST defines the axes as follows (when looking top-down at the package):

- x: points right (parallel to long edge)
- y: points downward (perpendicular to chip top marking)
- z: points out of the PCB



Remarks:

- Gyroscope and accelerometer share the same frame; magnetometer axes are nominally aligned, but may have a small offset or inversion depending on board layout.
- Slight variation in axes may occur depending on breakout board manufacturer (e.g., Adafruit, SparkFun).

Use Cases:

- Sensor fusion in mobile robots
- Handheld and wearable devices
- Well-supported in platforms like Arduino, Teensy, STM32

Summary: IMU Frame Alignment Comparison

IMU	x	y	z
MPU-9250	Right	Forward	Up
LSM9DS1	Right	Down	Up

Note: Always consult the specific datasheet or breakout board schematic for the exact axis orientation and origin.

Transforming MPU-9250 Readings to the NED Frame

The MPU-9250 reports all sensor readings in its internal sensor frame. To use these values in control or estimation algorithms, they must be transformed into a global navigation frame such as **NED** (North–East–Down).

1. Sensor Frame Definition (MPU-9250)

Based on the standard orientation when viewed from the top (chip markings visible), the default sensor axes are:

- x_s: points right (to the right edge of the chip)
- y_s: points forward (out of the chip, away from pin 1)
- z_s: points up (out of the PCB)

This frame is right-handed: $\mathbf{x}_s \times \mathbf{y}_s = \mathbf{z}_s$

2. Target Frame: NED (North–East–Down)

The standard NED frame is defined as:

- x_n: points North
- y_n: points East
- z_n: points Down (toward Earth's center)

This is also a right-handed frame: $\mathbf{x}_n \times \mathbf{y}_n = \mathbf{z}_n$

3. Rotation Matrix: Sensor Frame to NED

To map a vector $\mathbf{v}_s$ from the sensor frame to the NED frame, we define the transformation matrix $R_{\text{NED}\leftarrow s}$ such that:

$$\mathbf{v}_{\text{NED}} = R_{\text{NED}\leftarrow s} \cdot \mathbf{v}_s$$

Assuming the following alignment between axes:

$$\begin{cases} x_s \to y_n \\ y_s \to x_n \\ z_s \to -z_n \end{cases} \quad \Rightarrow \quad R_{\text{NED}\leftarrow s} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

4. Applying to Each Sensor

Let the raw sensor readings from the MPU-9250 be:

$$\omega_s, \quad \mathbf{a}_s, \quad \mathbf{m}_s$$

representing angular velocity (gyroscope), specific force (accelerometer), and magnetic field (magnetometer), all in the sensor frame.

Then, to convert each to the NED frame:

$$\omega_{\text{NED}} = R_{\text{NED}\leftarrow s} \cdot \omega_s$$

$$\mathbf{a}_{\text{NED}} = R_{\text{NED}\leftarrow s} \cdot \mathbf{a}_s$$

Magnetometer adjustment:

The MPU-9250's internal magnetometer (AK8963) is rotated $+90°$ about the z_s axis relative to the accelerometer/gyro frame. Therefore, apply an extra rotation before transforming to NED.

$$R_{\text{gyro}\leftarrow\text{mag}} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \Rightarrow \quad \mathbf{m}_{\text{aligned}} = R_{\text{gyro}\leftarrow\text{mag}} \cdot \mathbf{m}_s$$

Then convert to NED:

$$\mathbf{m}_{\text{NED}} = R_{\text{NED}\leftarrow s} \cdot \mathbf{m}_{\text{aligned}}$$

5. Final Formulas

$$\boxed{\omega_{\text{NED}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \cdot \omega_s} \quad \boxed{\mathbf{a}_{\text{NED}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \cdot \mathbf{a}_s}$$

$$\boxed{\mathbf{m}_{\text{NED}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{m}_s}$$

This final matrix composition can be precomputed to yield:

$$\mathbf{m}_{\text{NED}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix} \cdot \mathbf{m}_s$$

6. Interpretation

The above transformations allow sensor readings from the MPU-9250 to be interpreted in a global NED frame, suitable for use in navigation, control, and state estimation. Proper alignment is essential to avoid orientation errors and heading drift.

Transforming LSM9DS1 Readings to the NED Frame

The LSM9DS1 reports all sensor readings in its internal sensor frame. To interpret these values in a global frame such as NED (North–East–Down), a coordinate transformation is required.

1. Sensor Frame Definition (LSM9DS1)

According to STMicroelectronics documentation and typical breakout board orientation:

- x_s: points right
- y_s: points forward (along the PCB)
- z_s: points up (out of the PCB)

This is a standard right-handed frame:

$$\mathbf{x}_s \times \mathbf{y}_s = \mathbf{z}_s$$

2. Target Frame: NED (North–East–Down)

As before, the NED frame has:

- x_n: North
- y_n: East
- z_n: Down

3. Rotation Matrix: Sensor Frame to NED

Assuming perfect alignment between the sensor and body axes, and the board is mounted such that:

$$\begin{cases} x_s \to y_n \\ y_s \to x_n \\ z_s \to -z_n \end{cases}$$

Then, the rotation matrix $R_{\text{NED}\leftarrow s}$ is:

$$R_{\text{NED}\leftarrow s} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

4. Applying to Each Sensor

Let:

$$\omega_s, \quad \mathbf{a}_s, \quad \mathbf{m}_s$$

be the gyroscope, accelerometer, and magnetometer vectors in the sensor frame.

Then:

$$\omega_{\text{NED}} = R_{\text{NED}\leftarrow s} \cdot \omega_s \quad \mathbf{a}_{\text{NED}} = R_{\text{NED}\leftarrow s} \cdot \mathbf{a}_s \quad \mathbf{m}_{\text{NED}} = R_{\text{NED}\leftarrow s} \cdot \mathbf{m}_s$$

5. Final Formulas

$$\boxed{\omega_{\text{NED}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \cdot \omega_s} \quad \boxed{\mathbf{a}_{\text{NED}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \cdot \mathbf{a}_s} \quad \boxed{\mathbf{m}_{\text{NED}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \cdot \mathbf{m}_s}$$

6. Notes

- These formulas assume the LSM9DS1 breakout board is mounted such that the sensor axes align as described.
- Some commercial breakout boards may invert one or more axes; always verify using the datasheet or test measurements (e.g., stationary gravity, rotation direction).
- If the board is mounted at a different orientation, a different rotation matrix must be derived based on that alignment.

Glossary of Terms

AHRS (Attitude and Heading Reference System): A system that estimates a body's 3D orientation using data from inertial and magnetic sensors.

IMU (Inertial Measurement Unit): A sensor module that measures angular velocity (gyroscope), linear acceleration (accelerometer), and sometimes magnetic field (magnetometer).

Orientation: The rotation of a body in 3D space relative to a reference frame.

Roll–Pitch–Yaw (Euler Angles): A minimal and human-readable representation of orientation as sequential rotations around body-fixed axes.

Quaternion: A 4-element vector used to represent orientation without singularities (no gimbal lock) and with smooth interpolation.

Rotation Matrix: A 3×3 orthonormal matrix representing a 3D rotation. Useful for applying orientation to vectors.

Skew-Symmetric Matrix: A matrix used to represent cross products and angular velocity in matrix form. Denoted $[\omega]_\times$.

NED (North–East–Down): A navigation coordinate frame commonly used in aerospace, where x = north, y = east, z = down.

ENU (East–North–Up): A local geographic frame commonly used in robotics and mobile applications.

ECEF (Earth-Centered, Earth-Fixed): A global geocentric frame used in satellite and GPS systems.

Bias: A constant offset in sensor readings (especially common in gyroscopes), which causes drift over time if uncorrected.

Drift: A slow deviation in estimated orientation due to bias accumulation in gyroscope integration.

Sensor Fusion: The process of combining data from multiple sensors to produce a more accurate estimate of orientation or motion.

Madgwick/Mahony Filter: Computationally efficient algorithms used to estimate orientation by fusing gyroscope, accelerometer, and magnetometer data.

Sampling Rate: The frequency at which the IMU updates its measurements, typically 100–1000 Hz.

DMP (Digital Motion Processor): A hardware component found in some IMUs (e.g., MPU-9250) that performs sensor fusion and outputs orientation directly.