ME 5751: 3D Robot Kinematics

The Rosetta Stone: Translating Between Pose Representations

From Redundancy to Intuition

Recap of the Rotation Matrix

The 3x3 Rotation Matrix, ${}^{A}_{B}R$, is the computational foundation of kinematics. It uses nine parameters to describe a three-degree-of-freedom orientation, constrained by six orthonormality conditions ($R^T R = I, \det(R) = +1$).

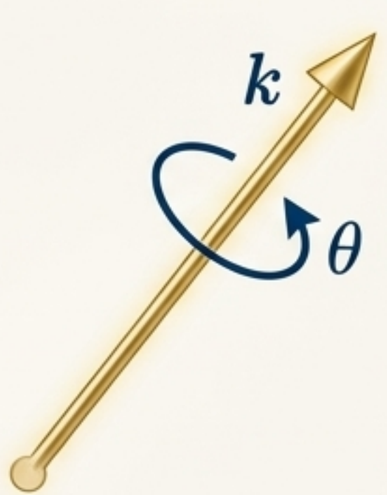
This representation is redundant and non-intuitive for specifying or visualizing an orientation.

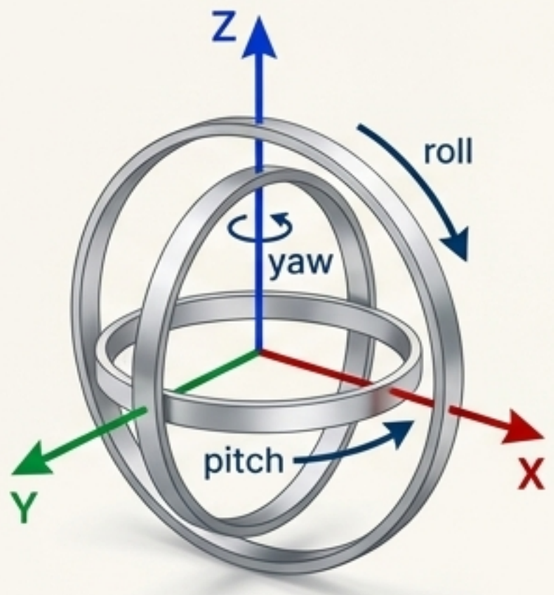
$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Vs.

The Core Problem

How can we describe orientation with just three, intuitive numbers?

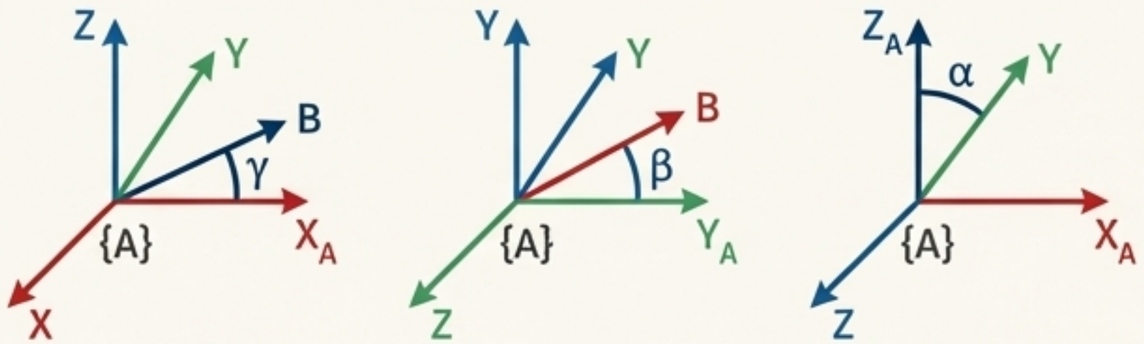




A Tale of Two Conventions: Intrinsic vs. Extrinsic Rotations

Extrinsic (Fixed) Angles

A sequence of three rotations about the axes of a *fixed* reference frame {A}.

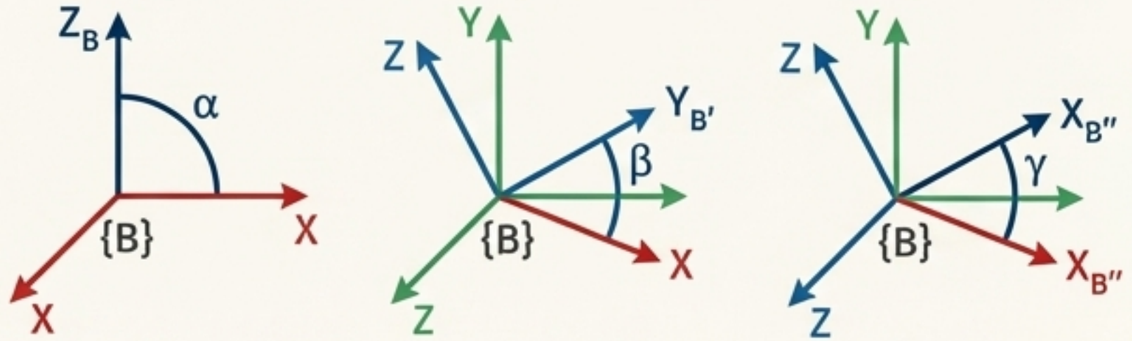


The final rotation is found by pre-multiplying the rotation matrices. For an X-Y-Z sequence:

$${}^{A}_{B}R = R_Z(\alpha)\,R_Y(\beta)\,R_X(\gamma)$$

Intrinsic (Euler) Angles

A sequence of three rotations about the axes of the *moving* body frame.



The final rotation is found by post-multiplying. For a Z'-Y"-X"" sequence:

$${}^{A}_{B}R = R_{Z'}(\alpha)\,R_{Y''}(\beta)\,R_{X'''}(\gamma)$$

An extrinsic X-Y-Z rotation is mathematically equivalent to an intrinsic Z-Y-X rotation.

The Twelve Flavors of Euler Angles

Proper Euler Angles

The first and third rotation axes are the same. (**e.g.**, Z-Y-Z, X-Y-X).
Used in rigid body dynamics.

Tait-Bryan Angles

The three rotation axes are distinct. (**e.g.**, X-Y-Z, Z-Y-X).
Common in aerospace and robotics as "roll, pitch, yaw".

Example: X-Y-Z Fixed Angles (equivalent to Z''-Y''-X''' Intrinsic)

Example: $$ {}^{A}_{B}R_{XYZ}(\gamma, \beta, \alpha) = R_Z(\alpha)R_Y(\beta)R_X(\gamma) $$

$$ \begin{bmatrix} c\alpha c\beta & c\alpha s\beta s\gamma - s\alpha c\gamma & c\alpha s\beta c\gamma + s\alpha s\gamma \\ s\alpha c\beta & s\alpha s\beta s\gamma + c\alpha c\gamma & s\alpha s\beta c\gamma - c\alpha s\gamma \\ -s\beta & c\beta s\gamma & c\beta c\gamma \end{bmatrix} $$

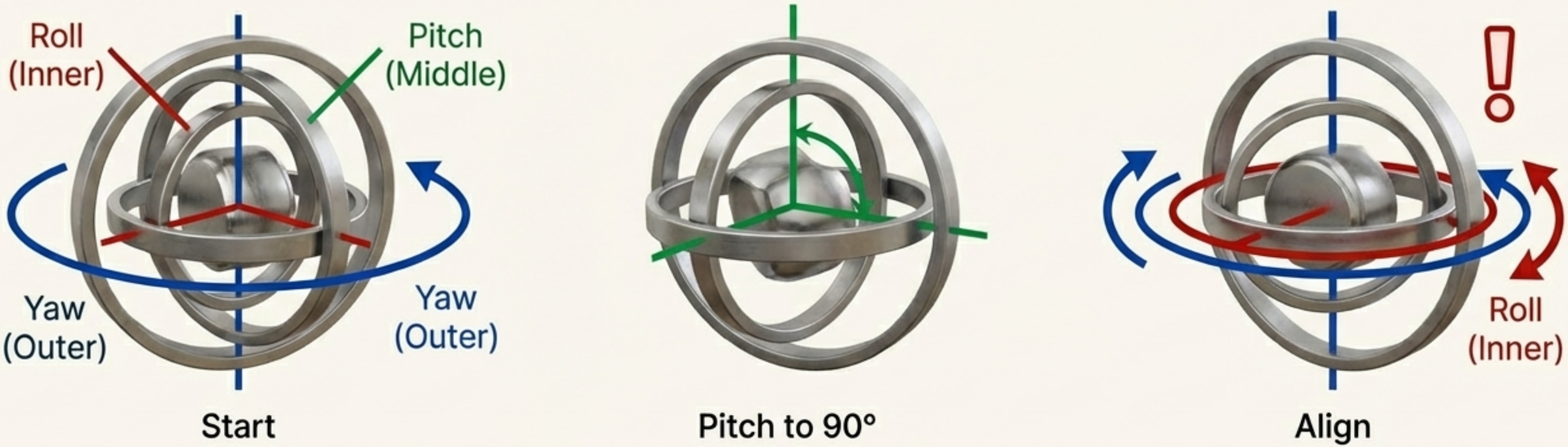
cα = cos(α), sα = sin(α)

The Problem with Simplicity: Representational Singularities

Gimbal Lock

When the middle rotation angle (e.g., β in an X-Y-Z sequence) becomes $\pm 90°$, the first and third axes of rotation align. This causes a loss of one degree of freedom, as "roll" and "yaw" produce the same motion.

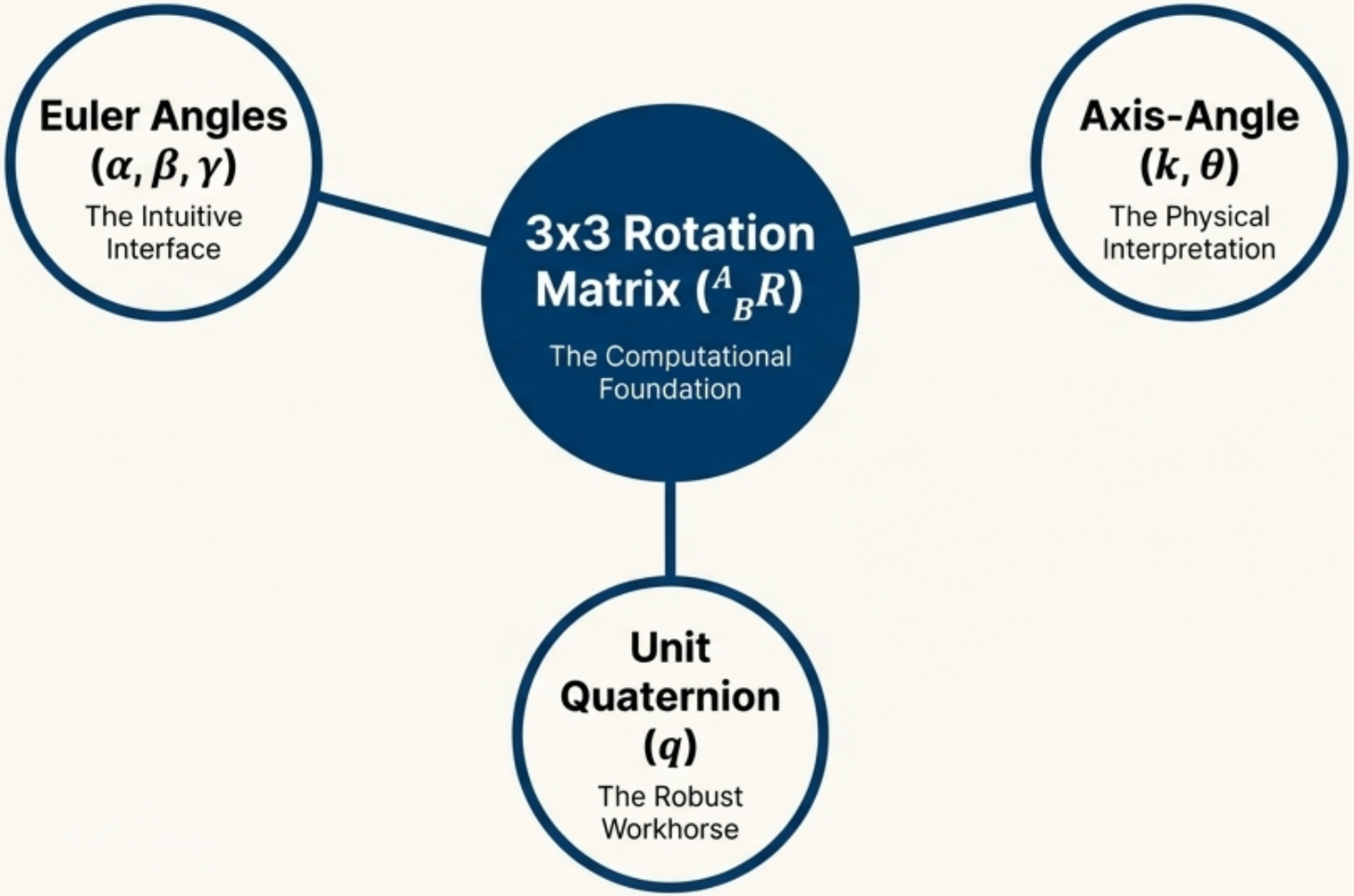
At this singularity, there are infinite solutions for the first and third angles. This is catastrophic for control and interpolation.



Start | Pitch to 90° | Align

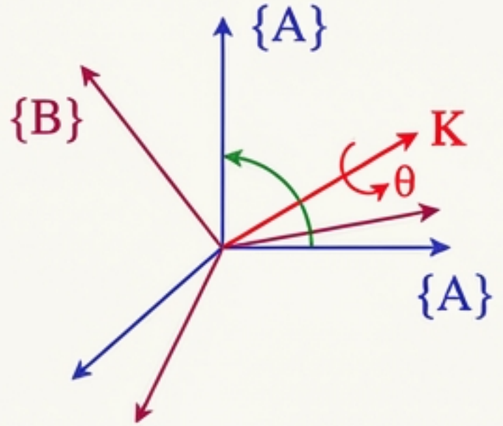
A Unified Framework for Orientation: The Kinematic "Rosetta Stone"

To overcome the limitations of any single representation, we must be able to convert between all of them. The Rotation Matrix serves as our central hub for translation, providing a computationally robust foundation.



Path 1: Rotation Matrix ⇔ Axis-Angle

Recap: Euler's Rotation Theorem: Any orientation can be described by a single rotation (θ) about a single axis ($\boldsymbol{k}$).



From Physical Meaning to Matrix

Rodrigues' Rotation Formula

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

where $[\boldsymbol{k}]_\times$ is the skew-symmetric matrix of the unit vector $\boldsymbol{k}$.

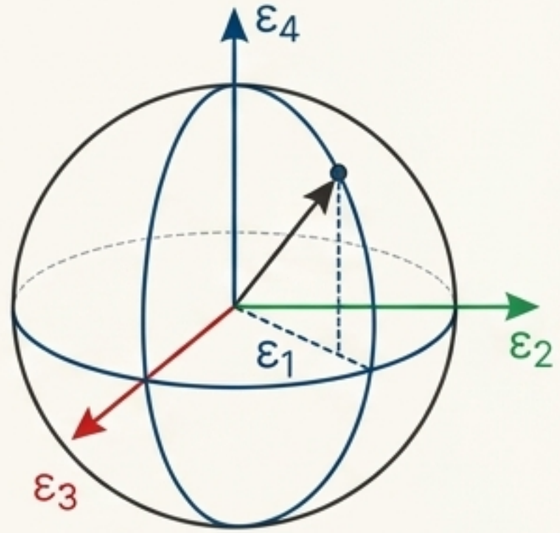
From Matrix to Physical Meaning

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

$$\boldsymbol{k} = \frac{1}{2\sin\theta}\begin{bmatrix} r_{32}-r_{23} \\ r_{13}-r_{31} \\ r_{21}-r_{12} \end{bmatrix}$$

Path 2: Axis-Angle ⇔ Quaternion

Recap: Unit Quaternions (Euler Parameters): a four-parameter representation that avoids gimbal lock and is computationally efficient for interpolation and composition.



Axis-Angle to Quaternion

$$\epsilon_1 = k_x \sin(\theta/2)$$
$$\epsilon_2 = k_y \sin(\theta/2)$$
$$\epsilon_3 = k_z \sin(\theta/2)$$
$$\epsilon_4 = \cos(\theta/2)$$

Constraint: $\epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2 + \epsilon_4^2 = 1$

Quaternion to Axis-Angle

$$\theta = 2 \arccos(\epsilon_4)$$

$$k = \frac{1}{\sqrt{1 - \epsilon_4^2}} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix}$$

(valid for $\sin(\theta/2) \neq 0$)

Path 3: Rotation Matrix <=> Quaternion

From Quaternion to Matrix

$$R = \begin{bmatrix} 1-2(\epsilon_2^2+\epsilon_3^2) & 2(\epsilon_1\epsilon_2-\epsilon_3\epsilon_4) & 2(\epsilon_1\epsilon_3+\epsilon_2\epsilon_4) \\ 2(\epsilon_1\epsilon_2+\epsilon_3\epsilon_4) & 1-2(\epsilon_1^2+\epsilon_3^2) & 2(\epsilon_2\epsilon_3-\epsilon_1\epsilon_4) \\ 2(\epsilon_1\epsilon_3-\epsilon_2\epsilon_4) & 2(\epsilon_2\epsilon_3+\epsilon_1\epsilon_4) & 1-2(\epsilon_1^2+\epsilon_2^2) \end{bmatrix}$$

From Matrix to Quaternion

$$\epsilon_4 = \frac{1}{2}\sqrt{1+r_{11}+r_{22}+r_{33}}$$

$$\epsilon_1 = (r_{32}-r_{23})/(4\epsilon_4)$$
$$\epsilon_2 = (r_{13}-r_{31})/(4\epsilon_4)$$
$$\epsilon_2 = (r_{11}-r_{31})/(4\epsilon_4)$$
$$\epsilon_3 = (r_{21}-r_{12})/(4\epsilon_4)$$

Practical Implementation

```
from scipy.spatial.transform import Rotation as R

# Given a matrix, get the quaternion
q = R.from_matrix(mat).as_quat()  # [x, y, z, w]

# Given a quaternion, get the matrix
mat = R.from_quat(q).as_matrix()
```

Extracting Intuition from the Matrix: The Inverse Problem

Given a rotation matrix ${}^{A}_{B}R$, find a set of X-Y-Z fixed angles (γ, β, α) that generate it by comparing the given matrix $[r_{ij}]$ to the symbolic X-Y-Z fixed angle matrix.

Case 1: No Singularity ($\cos\beta \neq 0$)

Two Distinct Solutions

$$\beta = \operatorname{atan2}\left(-r_{31}, \sqrt{r_{11}^2 + r_{21}^2}\right)$$

$$\alpha = \operatorname{atan2}(r_{21}/\cos\beta, r_{11}/\cos\beta)$$

$$\gamma = \operatorname{atan2}(r_{32}/\cos\beta, r_{33}/\cos\beta)$$

The second solution is found using $\beta' = \pi - \beta$, which yields a different set of α' and γ'.

Case 2: Singularity ($\cos\beta = 0 \Rightarrow \beta = \pm 90°$)

The Mathematical Gimbal Lock

This condition represents the singularity we visualized earlier. An infinite number of solutions exist as only the sum or difference of α and γ is fixed.

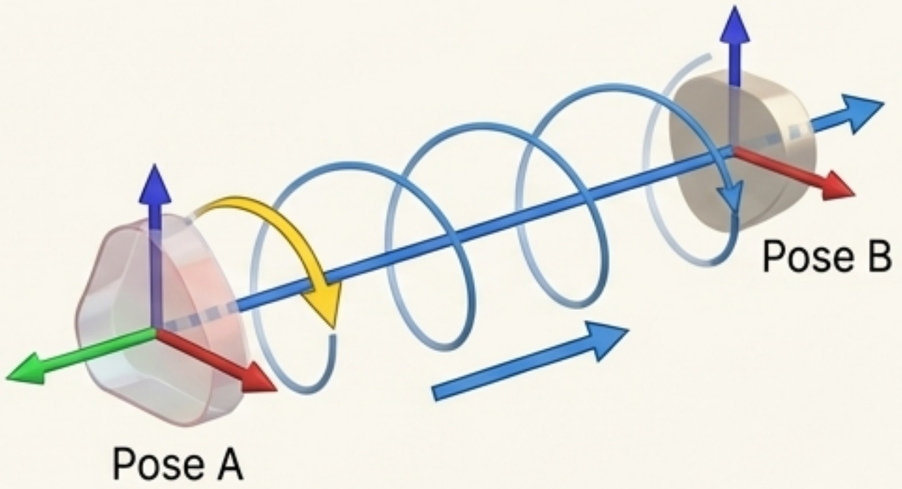
If $\beta = +90°$: $\gamma - \alpha = \operatorname{atan2}(r_{12}, r_{13})$

If $\beta = -90°$: $\gamma + \alpha = \operatorname{atan2}(-r_{12}, -r_{13})$

The Grand Unification: From Pose to Screw Motion

Recap

Chasles' Theorem: Any rigid body displacement can be produced by a rotation about an axis combined with a translation parallel to that same axis. This is a screw displacement.



Main Content: Side-by-Side Conversion Formulas

Deconstructing a Pose

Given $T = \begin{bmatrix} R & \mathbf{p} \\ 0 & 1 \end{bmatrix}$, we find the screw axis $S = (\boldsymbol{\omega}, \mathbf{v})$ and rotation magnitude θ.

1. Find rotation axis $\boldsymbol{\omega}$ and angle θ from the rotation matrix R (as on Slide 7).
2. The full screw motion is then described by the matrix exponential $T = e^{[\mathcal{S}]\theta}$.

Constructing a Pose

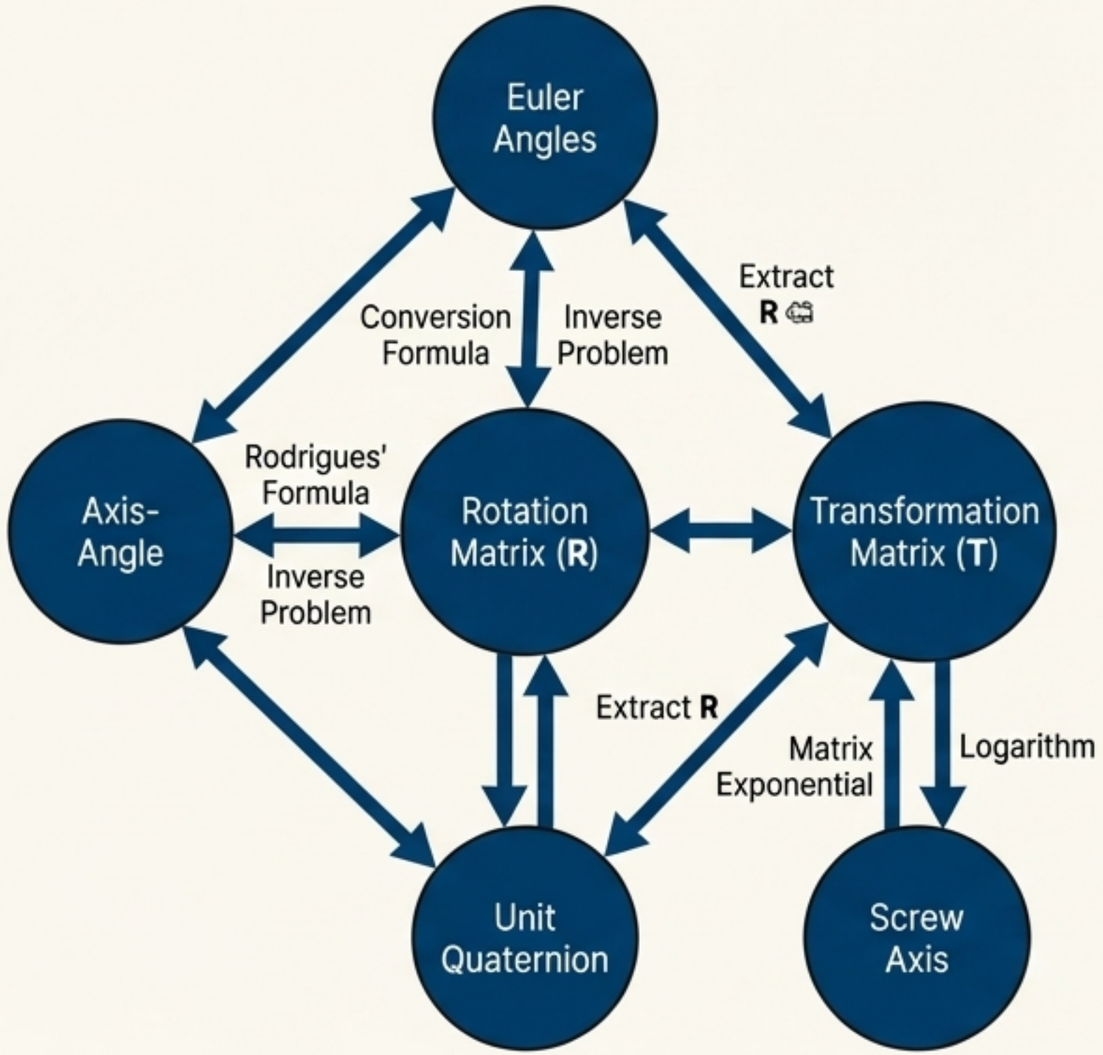
The Matrix Exponential: $T = e^{[\mathcal{S}]\theta}$ where $[\mathcal{S}] = \begin{bmatrix} [\boldsymbol{\omega}]_\times & \mathbf{v} \\ 0 & \mathbf{0} \end{bmatrix}$

This yields Rodrigues' formula for transformations:

$$R = e^{[\omega]_\times \theta}$$

$$\boldsymbol{p} = (I\theta + (1 - \cos\theta)[\boldsymbol{\omega}]_\times + (\theta - \sin\theta)[\boldsymbol{\omega}]_\times^2)\mathbf{v}$$

The Full Picture: A Map of Kinematic Representations



Representation	# Params	Pros	Cons	Primary Use Case
Euler Angles	3	Intuitive, minimal	Gimbal lock, ambiguous	Human interfaces, specification
Axis-Angle	4 (3+1)	Physical meaning	Not for composition	Physics, visualization
Rotation Matrix	9 (3 DoF)	Robust, for composition	Redundant, non-intuitive	Core computation, chaining rotations
Unit Quaternion	4 (3 DoF)	No singularities, efficient	Less intuitive	Interpolation (SLERP), control
Transformation Matrix	16 (6 DoF)	Unified pose operator	Redundant	Forward kinematics, chaining poses
Screw Axis	7 (6 DoF)	Fundamental, compact	Abstract, complex math	Motion analysis, advanced planning

When to Use What: A Practitioner's Guide

Human Input / Visualization

Tool: Euler Angles (α, β, γ)

Why: Best for GUIs with sliders for roll, pitch, and yaw. Easy for a human operator to understand and specify a desired orientation.

Interpolating Rotations (Motion Planning)

Tool: Unit Quaternions (q)

Why: Spherical Linear Interpolation (SLERP) provides the shortest, smoothest rotational path and avoids the singularities and jerky motions of Euler angle interpolation.

Chaining Transformations (Forward Kinematics)

Tool: Rotation/Transformation Matrices (R, T)

Why: Simple matrix multiplication is the most efficient and robust way to compose successive rotations and translations along a kinematic chain.

Analyzing Physical Motion

Tool: Axis-Angle (k, θ) or Screw Axis (S)

Why: They provide the most direct physical insight into a displacement—what is the net axis of rotation and translation?

Mastering the Language of Pose

We can now represent optimes and pose in the most future of a crrnieficta1 to now represent orientation and experimeents as roburt, orientation, whatter representatiting field to bet worker roests.

Key Takeaways

- We can now represent orientation and pose in multiple, intuitive, and robust ways.
- We have built a “Rosetta Stone” to translate fluently between these representations.
- Understanding the pros and cons of each tool is essential for effective robot programming.

Upcoming Topics

Now that we can fluently *describe* any pose, our next lecture will address how to *calculate* the joint angles required to achieve it: **Inverse Kinematics**.