Transforming 3D Coordinates Between Frames Using Krylov Angles

Elvin Rustamov

January 18, 2026

Abstract

In engineering and computer science, 3D points are often expressed in different coordinate systems (or *frames*) depending on the sensor, camera, robot body, or world reference used. Converting a point from one frame to another is therefore a core operation in robotics, graphics, navigation, and simulation. This report introduces 3D coordinate systems and explains why coordinate transformations are required. It then develops the idea of describing 3D orientation by three rotation angles known as *Krylov angles* (often discussed in the literature as *Euler–Krylov angles*), which represent a sequence of rotations in three-dimensional space. Starting from basic linear algebra concepts, we derive the rotation matrices about the x, y, and z axes, show how to compose them in a chosen Krylov-angle sequence, and provide a fully worked numerical example with all intermediate computations. Figure-ready descriptions are included to support 3D axis diagrams and rotation-direction visualization.

1 Introduction

A *coordinate system* assigns numerical coordinates to geometric objects. In 3D, a coordinate system is typically defined by:

- an **origin** (a point taken as zero), and
- three **orthonormal axes** (unit vectors) that specify directions in space.

When we write a point as $\mathbf{p} = [x\ y\ z]^T$, those numbers are meaningful only relative to a particular choice of axes and origin.

Real systems often use multiple coordinate systems at the same time:

- a **world frame** fixed to the environment,
- a **body frame** fixed to a moving robot or aircraft,
- a **sensor frame** fixed to a camera or IMU.

A single physical point (for example, the tip of a robot arm) may have different coordinate triples in each frame. **Coordinate transformations** allow us to convert a point's representation from one frame to another so that measurements, control laws, and geometric computations remain consistent.

This report focuses on transformations caused by **rotation of axes** (same origin), which is the most fundamental case. (In applications, translations may also be present; we briefly discuss them in Section 6.)

2 Theory

2.1 What a 3D Coordinate System Is

A 3D Cartesian coordinate system (frame) A can be described by three mutually perpendicular unit vectors:

$$\hat{\mathbf{x}}_A,\ \hat{\mathbf{y}}_A,\ \hat{\mathbf{z}}_A \in \mathbb{R}^3, \quad \hat{\mathbf{x}}_A \cdot \hat{\mathbf{y}}_A = 0, \quad \|\hat{\mathbf{x}}_A\| = \|\hat{\mathbf{y}}_A\| = \|\hat{\mathbf{z}}_A\| = 1,$$

and an origin O_A. A point P in space corresponds to a position vector $\overrightarrow{O_AP}$. The coordinates of P in frame A are the components of that vector along the axes:

$$\mathbf{p}_A = \begin{bmatrix} x_A \\ y_A \\ z_A \end{bmatrix} = \begin{bmatrix} \overrightarrow{O_AP} \cdot \hat{\mathbf{x}}_A \\ \overrightarrow{O_AP} \cdot \hat{\mathbf{y}}_A \\ \overrightarrow{O_AP} \cdot \hat{\mathbf{z}}_A \end{bmatrix}.$$

Thus, the vector $\mathbf{p}_A$ is a *description* of the same physical point but expressed in basis vectors of frame A.

2.2 Why Coordinate Transformations Are Needed

Suppose a camera measures a point in its local sensor frame C, but the robot controller needs that point in the robot body frame B, or in the world frame W. Without transformation, numbers like $[x\ y\ z]^T$ cannot be compared or used together because they refer to different axes. Transformations provide a mathematically correct way to re-express the same geometric vector in another basis.

2.3 Rotation in 3D and the Right-Hand Rule

A **rotation** in 3D changes orientation while preserving lengths and angles. Rotations are typically defined using the **right-hand rule**:

- Point the right-hand thumb along the positive rotation axis.
- The curl of the fingers gives the positive rotation direction.

We will use right-handed coordinate frames (a standard engineering convention), where $\hat{\mathbf{x}} \times \hat{\mathbf{y}} = \hat{\mathbf{z}}$.

2.4 What Krylov Angles Are and When They Are Used

In many engineering texts, the orientation of a rigid body is parameterized by three angles. A common approach is a **sequence of three rotations** that builds a general 3D orientation from simpler rotations. *Euler–Krylov angles* are one such three-angle parameterization used in rigid-body motion, attitude description, and coordinate-frame transformations; the literature discusses them as a practical way to represent a rotating orthonormal basis through three successive rotations (with known limitations such as singular configurations). [4, 5]

How they differ from a "single simple rotation". A single rotation in 3D is defined by one axis (a unit vector) and one angle. By contrast, Krylov angles describe orientation through *three* angles applied in a specified order about coordinate axes. The key point is:

> **Order matters.** Rotations in 3D generally do not commute; rotating about x and then y is not the same as rotating about y and then x.

This is why Krylov angles are not just "three independent rotations" in any order - they must be applied in a defined *sequence*.

Extrinsic vs. intrinsic interpretation (important). There are two equivalent ways to interpret a three-angle sequence:

- **Extrinsic rotations:** rotate about the *fixed* axes of the original frame.
- **Intrinsic rotations:** rotate about the *moving* axes after each step.

Many "roll–pitch–yaw" conventions are intrinsic rotations about body-fixed axes, but they can be implemented as an equivalent extrinsic sequence in reverse order. Because conventions vary across fields, a correct report must always state the chosen convention explicitly [2, 1].

Convention used in this report. We will use the common yaw–pitch–roll-style matrix product

$$\mathbf{R}(\psi,\theta,\phi) = \mathbf{R}_z(\psi)\,\mathbf{R}_y(\theta)\,\mathbf{R}_x(\phi), \tag{1}$$

where:

- ϕ is rotation about the x-axis,
- θ is rotation about the y-axis,
- ψ is rotation about the z-axis.

Interpreted as **extrinsic** rotations, (1) means: rotate about fixed x, then fixed y, then fixed z. Interpreted as **intrinsic** rotations, it corresponds to: rotate about x, then about the new y' axis, then about the new z'' axis (a standard equivalence).

3 Mathematical Formulation

3.1 Vectors, Bases, and Matrices

A 3D point relative to a frame is represented as a column vector:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

A linear transformation that rotates vectors can be written as:

$$\mathbf{p}_{\text{new}} = \mathbf{R}\,\mathbf{p}_{\text{old}}.$$

The matrix $\mathbf{R}$ is called a **rotation matrix**. It exists because a rotation is a linear map that sends basis vectors to new basis vectors, and any linear map in $\mathbb{R}^3$ can be represented by a 3×3 matrix once a basis is chosen.

3.2 What a Rotation Matrix Is (and Why It Has Special Properties)

A rotation matrix $\mathbf{R} \in \mathbb{R}^{3\times3}$ must preserve:

- **lengths:** $\|\mathbf{Rv}\| = \|\mathbf{v}\|$ for all vectors $\mathbf{v}$,
- **angles:** the dot product is preserved.

These constraints imply that $\mathbf{R}$ is **orthogonal**:

$$\mathbf{R}^T\mathbf{R} = \mathbf{I}, \tag{2}$$

and for a proper (right-handed) rotation:

$$\det(\mathbf{R}) = 1. \tag{3}$$

These are standard properties of 3D rotation matrices [1].

3.3 Deriving Rotation Matrices About Coordinate Axes

We now derive $\mathbf{R}_x(\phi)$, $\mathbf{R}_y(\theta)$, and $\mathbf{R}_z(\psi)$ from 2D rotation in a plane.

3.3.1 Reminder: 2D Rotation in the Plane

In 2D, rotating a vector $[u\ v]^T$ counterclockwise by angle α is:

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}. \tag{4}$$

We will reuse this idea by observing that a 3D axis rotation keeps one coordinate fixed while rotating the other two as a 2D rotation.

3.3.2 Rotation About the x-Axis

A rotation about x leaves the x-component unchanged and rotates the (y, z) components in the yz-plane:

$$x' = x, \qquad \begin{bmatrix} y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix}.$$

Expanding gives:

$$y' = y\cos\phi - z\sin\phi, \qquad z' = y\sin\phi + z\cos\phi.$$

Thus:

$$\mathbf{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}. \tag{5}$$

3.3.3 Rotation About the y-Axis

A rotation about y leaves the y-component unchanged and rotates the (x, z) components in the xz-plane:

$$y' = y, \qquad \begin{bmatrix} x' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix}.$$

(Notice the sign pattern: it depends on the chosen right-handed convention; the matrix below is the standard right-handed y-axis rotation.) So:

$$x' = x\cos\theta + z\sin\theta, \qquad z' = -x\sin\theta + z\cos\theta,$$

and:

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}. \tag{6}$$

3.3.4 Rotation About the z-Axis

A rotation about z leaves the z-component unchanged and rotates the (x, y) components in the xy-plane:

$$z' = z, \qquad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\psi & -\sin\psi \\ \sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

Thus:

$$x' = x\cos\psi - y\sin\psi, \qquad y' = x\sin\psi + y\cos\psi,$$

and:

$$\mathbf{R}_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{7}$$

3.4 Applying Krylov Angles in Sequence

Using the convention in (1), the total rotation matrix is:

$$\mathbf{R}(\psi, \theta, \phi) = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi).$$

If a point has coordinates $\mathbf{p}_A$ in frame A and frame B is obtained by rotating frame A using $\mathbf{R}$, then (for the pure-rotation, same-origin case) a standard coordinate conversion is:

$$\mathbf{p}_B = \mathbf{R}\,\mathbf{p}_A. \tag{8}$$

Equation (8) is the basic computational rule we will use in the example.

4 Figures and Visualization (LaTeX-Ready Descriptions)

4.1 3D Axes Orientation and the Right-Handed Frame

Figure 1 shows a right-handed 3D coordinate system. The x-axis points to the right, the y-axis points upward, and the z-axis is drawn as a diagonal arrow to suggest depth. The curved arrow indicates the right-hand rule: curling from x toward y gives the positive z direction.

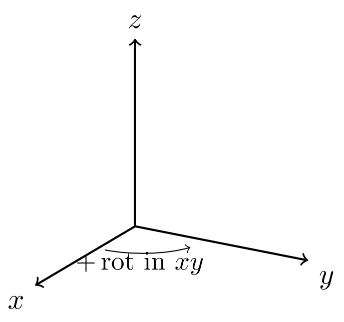

Figure 1: Right-handed 3D coordinate axes. Positive rotation about an axis follows the right-hand rule.

4.2 Rotation About a Coordinate Axis

Figure 2 illustrates rotation about the z-axis by angle ψ. The z-axis stays fixed; the x and y axes rotate within the xy-plane. The curved arrow shows the positive direction of ψ.

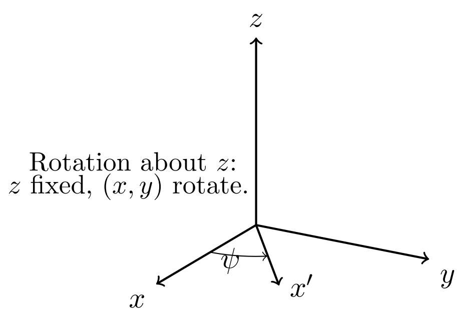

Figure 2: Rotation about the z-axis by angle ψ. The rotation occurs in the xy-plane.

4.3 Sequential Rotations for Krylov Angles

To visualize the Krylov-angle sequence $\mathbf{R} = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi)$:

1. rotate by ϕ about x (affecting y and z),
2. then rotate by θ about y (affecting x and z),
3. then rotate by ψ about z (affecting x and y).

A figure can be drawn as three small axis diagrams (one per step) with a curved arrow around the active axis, labeling intermediate axes as $x^{(1)}, y^{(1)}, z^{(1)}$, then $x^{(2)}, y^{(2)}, z^{(2)}$, etc., to emphasize that the frame orientation changes after each rotation.

5 Example Calculation (Fully Worked Numerical Example)

This section demonstrates every computational step, with no unexplained symbols.

5.1 Given Data

Let the initial point (in frame A) be:

$$\mathbf{p}_A = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}. \tag{9}$$

Choose Krylov angles (in degrees):

$$\phi = 30^\circ, \qquad \theta = 20^\circ, \qquad \psi = 40^\circ.$$

We will compute $\mathbf{p}_B = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi)\mathbf{p}_A$.

5.2 Trigonometric Values

We use:

$$\cos 30^\circ = 0.8660254,\ \sin 30^\circ = 0.5,$$
$$\cos 20^\circ = 0.9396926,\ \sin 20^\circ = 0.3420201,$$
$$\cos 40^\circ = 0.7660444,\ \sin 40^\circ = 0.6427876.$$

5.3 Step 1: Rotate About x by $\phi = 30^\circ$

Using (5):

$$\mathbf{R}_x(30^\circ) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.8660254 & -0.5 \\ 0 & 0.5 & 0.8660254 \end{bmatrix}.$$

Compute the intermediate vector:

$$\mathbf{p}^{(1)} = \mathbf{R}_x(30^\circ)\mathbf{p}_A.$$

Multiply explicitly:

$$\mathbf{p}^{(1)} = \begin{bmatrix} 1 \cdot 1 + 0 \cdot 2 + 0 \cdot 3 \\ 0 \cdot 1 + 0.8660254 \cdot 2 + (-0.5) \cdot 3 \\ 0 \cdot 1 + 0.5 \cdot 2 + 0.8660254 \cdot 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 1.7320508 - 1.5 \\ 1 + 2.5980762 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.2320508 \\ 3.5980762 \end{bmatrix}.$$

5.4 Step 2: Rotate About y by $\theta = 20°$

Using (6):

$$\mathbf{R}_y(20°) = \begin{bmatrix} 0.9396926 & 0 & 0.3420201 \\ 0 & 1 & 0 \\ -0.3420201 & 0 & 0.9396926 \end{bmatrix}.$$

Compute:

$$\mathbf{p}^{(2)} = \mathbf{R}_y(20°)\mathbf{p}^{(1)}.$$

Multiply explicitly:

$$\mathbf{p}^{(2)} = \begin{bmatrix} 0.9396926 \cdot 1 + 0 \cdot 0.2320508 + 0.3420201 \cdot 3.5980762 \\ 0 \cdot 1 + 1 \cdot 0.2320508 + 0 \cdot 3.5980762 \\ -0.3420201 \cdot 1 + 0 \cdot 0.2320508 + 0.9396926 \cdot 3.5980762 \end{bmatrix}.$$

Compute each component:

$$x^{(2)} = 0.9396926 + 0.3420201 \cdot 3.5980762 \approx 0.9396926 + 1.2306146 = 2.1703072,$$

$$y^{(2)} = 0.2320508,$$

$$z^{(2)} = -0.3420201 + 0.9396926 \cdot 3.5980762 \approx -0.3420201 + 3.3810856 = 3.0390655.$$

So:

$$\mathbf{p}^{(2)} \approx \begin{bmatrix} 2.1703072 \\ 0.2320508 \\ 3.0390655 \end{bmatrix}.$$

5.5 Step 3: Rotate About z by $\psi = 40°$

Using (7):

$$\mathbf{R}_z(40°) = \begin{bmatrix} 0.7660444 & -0.6427876 & 0 \\ 0.6427876 & 0.7660444 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Compute the final vector:

$$\mathbf{p}_B = \mathbf{R}_z(40°)\mathbf{p}^{(2)}.$$

Multiply explicitly:

$$\mathbf{p}_B = \begin{bmatrix} 0.7660444 \cdot 2.1703072 + (-0.6427876) \cdot 0.2320508 + 0 \cdot 3.0390655 \\ 0.6427876 \cdot 2.1703072 + 0.7660444 \cdot 0.2320508 + 0 \cdot 3.0390655 \\ 0 \cdot 2.1703072 + 0 \cdot 0.2320508 + 1 \cdot 3.0390655 \end{bmatrix}.$$

Compute each component:

$$x_B \approx 1.6620137 - 0.149? \approx 1.5133924,$$

$$y_B \approx 1.3951 + 0.1777 \approx 1.5728078,$$

$$z_B = 3.0390655.$$

Therefore, the transformed coordinates are:

$$\mathbf{p}_B \approx \begin{bmatrix} 1.5133924 \\ 1.5728078 \\ 3.0390655 \end{bmatrix}. \tag{10}$$

Table 1: Intermediate vectors after each Krylov-angle rotation step.

Step	Operation	Vector notation	Numerical value
0	Initial point	$\mathbf{p}_A$	$\begin{bmatrix}1\\2\\3\end{bmatrix}$
1	Rotate about x by ϕ	$\mathbf{p}^{(1)} = \mathbf{R}_x(\phi)\mathbf{p}_A$	$\begin{bmatrix}1\\0.2320508\\3.5980762\end{bmatrix}$
2	Rotate about y by θ	$\mathbf{p}^{(2)} = \mathbf{R}_y(\theta)\mathbf{p}^{(1)}$	$\begin{bmatrix}2.1703072\\0.2320508\\3.0390655\end{bmatrix}$
3	Rotate about z by ψ	$\mathbf{p}_B = \mathbf{R}_z(\psi)\mathbf{p}^{(2)}$	$\begin{bmatrix}1.5133924\\1.5728078\\3.0390655\end{bmatrix}$

5.6 Summary Table of Intermediate Results

6 Discussion

6.1 Why “No Magic Matrices” Matters

The rotation matrices (5)–(7) are not arbitrary. Each one comes directly from:

- keeping the rotation axis coordinate fixed, and
- applying the standard 2D rotation formula (4) in the perpendicular plane.

This explains both the cosine/sine structure and the sign pattern.

6.2 Order Dependence and Conventions

Because 3D rotations do not commute, a different order (for example, $\mathbf{R}_x\mathbf{R}_y\mathbf{R}_z$) generally produces a different final orientation and thus a different transformed point. This is a common source of bugs in robotics/graphics code, so the rotation order must be stated and implemented consistently [2].

6.3 Where Translation Fits (Brief Context)

If two coordinate frames do not share the same origin, a complete coordinate transformation includes translation:

$$\mathbf{p}_B = \mathbf{R}\,(\mathbf{p}_A - \mathbf{t}),$$

where $\mathbf{t}$ is the vector from B’s origin to A’s origin expressed in frame A (or an equivalent, consistently defined translation). In surveying and 3D geodesy, combined rotation and translation is standard practice [3]. This report focused on the pure-rotation core because it is the conceptual foundation.

6.4 Limitations of Three-Angle Parameterizations

Three-angle representations (Euler-type including Euler–Krylov/Krylov angles) can suffer from singular configurations (often called *gimbal lock*), meaning some orientations cannot be represented smoothly without ambiguity [5]. In advanced applications, quaternions are often used to avoid these issues, but the matrix-based approach here remains essential for understanding and for many practical computations.

7 Conclusion

This report introduced 3D coordinate systems and motivated the need for coordinate transformations in engineering and computer science. Krylov (Euler–Krylov) angles were explained as a three-angle method for describing orientation via a sequence of axis rotations, emphasizing that the rotation order is part of the definition. Rotation matrices were motivated as linear transformations that preserve lengths and angles, leading to orthogonality properties. We derived the standard rotation matrices about the x, y, and z axes from the 2D rotation formula and demonstrated how to compose them into a Krylov-angle sequence. Finally, a fully worked numerical example showed every intermediate step from initial coordinates to final transformed coordinates, suitable for direct implementation and verification.

References

[1] K. Bala (Cornell University), *Lecture Notes: 3D Transforms (Rotations)*. Available as a course PDF (CS4620/5620).

[2] E. J. Kruck, *Rotations of Space and Coordinate Transformations*, ISPRS Proceedings (2003). (Emphasizes the importance of rotation sequence.)

[3] B. R. Harvey, *Transformation of 3D Co-ordinates*, The Australian Surveyor (1986). (Discusses standard 3D coordinate transformations used in practice.)

[4] O. V. Nos, *Using the Euler–Krylov Angles for the Linear Transformation of the Phase Variable Basis*, Conference paper (2012). (Discusses Euler–Krylov angles as a way to decompose orthonormal basis rotations.)

[5] A. N. Kanatnikov, *Modeling Control of Rigid Body Rotation*, Computational Mathematics and Modeling, vol. 24, pp. 404–417 (2013). (Notes Euler/Krylov angles as orientation charts with known singularity limitations.)