Rotations – from complex numbers to quaternions

A Mohanraj, and S Ganga Prasath*

Department of Applied Mechanics and Biomedical Engineering,
Indian Institute of Technology Madras, Chennai 600 036.

Abstract

Rotation is one of the fundamental operations of geometry. It provides a way to transform vectors from one location to another while hinged with respect to a pre-defined axis. In this first part of the two part primer we introduce this simple yet often confusing geometric operation in 2D and 3D. We start with the complex number representation of the rotation operation in 2D and its connection to the standard matrix version. Then we discuss the Euler-angle representation of rotation of vector basis, their extrinsic and intrinsic forms. Ultimately we introduce quaternions, which are the natural extension of complex numbers to 3D. In the process, we also look at Gimbol locking, an intrinsic berrier in Euler-angle represetation. Each concept discussed is explained with a worked out example and a supplemental `python`–code that numerically implements these examples.

1 Introduction

In this two part series, we are interested in understanding the connection rotation and geometry of curves 3D. In the first part, we develop the notation and the mathematics behind rotations and in the second we will see how it forms the basis for geometry of curves in 3D. Though understanding geometry of 3D curves is our ultimate objective, we take the 2D route to make the concepts approachable before extending them to 3D. Besides its application to 3D curves, rotation has a variety of applications such as in the kinematics and dynamics of rigid bodies in physics (useful in mechanics of machines, robotics, and flight/rocket/satellite trajectories), animation & graphic design and so on. In this primer, we start with the rotation operation in 2D, its representation using complex numbers and an example by applying it to transform a vector. We then move to rotations in 3D which is considerably complex than its 2D counterpart. In 3D, we introduce rotation about an arbitrary orientation, often called Rodrigues rotation, the ideas of intrinsic and extrinsic rotation of basis vectors and show how to transform between two arbitrary basis vectors. We end with a generalization of complex numbers to 3D i.e. the quaternions and use it to perform rotations. This primer borrows heavily from these [1, 2, 3] excellent approachable references.

*sgangaprasath@smail.iitm.ac.in

2 Rotation in 2D

We will illustrate rotation in 2D with 3 examples: (a) rotation of a vector represented in cartesian coordinate basis, (b) rotation of the coordinate basis vectors (which will become relevant in sec. 3), (c) rotation of a triangle about the origin. We also show that the complex number system provides an intuitive and simple way to perform rotations. Most of what we see in this section is taught in high-school. Nevertheless, we leave it here to ensure completeness.

2.1 Complex plane representation

A complex number $z \in \mathbb{C}$ is defined as a number of the form $z = a + ib$, where $a, b \in \mathbb{R}$, and i is the imaginary unit with the property $i^2 = -1$. Complex numbers can be graphically represented in a cartesian plane with a representing the magnitude along $\hat{\mathbf{e}}_x$ and b along $\hat{\mathbf{e}}_y$ with $\hat{\mathbf{e}}_{x,y}$ being the unit vectors along x, y-axes (see Fig. 1(a)). This complex number z corresponds to the point with coordinates (a, b).

Polar form of complex numbers

In addition to the rectangular form, complex numbers can also be expressed in polar notation. The complex number z can be written as $z = re^{i\theta} = a + ib$, where r is the magnitude (or modulus) of the complex number with $r > 0$ and θ is the argument (or angle) of the complex number, shown schematically in Fig. 1(a), taking values in the range, $\theta \in [0, 2\pi]$, measured from the positive real axis. The term $e^{i\theta}$ is defined using Euler's formula:

$$e^{i\theta} = \cos\theta + i\sin\theta.$$

This representation is particularly useful for performing rotations in the complex plane. In polar notation, the multiplication of two complex numbers results in the multiplication of their magnitudes and the addition of their angles, thus providing a straightforward geometric interpretation of complex multiplication as rotation and scaling.

Multiplication as rotation and scaling

When you multiply two complex numbers, you perform a rotation and scaling in the complex plane. The multiplication can be expressed as:

$$z_1 \cdot z_2 = (a_1 + ib_1)(a_2 + ib_2).$$

This multiplication changes the magnitude and rotates the point in the complex plane. In polar notation, multiplication of two complex numbers given by, $z_1 = r_1 e^{i\theta_1}$ and $z_2 = r_2 e^{i\theta_2}$, where r_1 and r_2 are the magnitudes, and θ_1 and θ_2 are the angles of z_1 and z_2, can be written as,

$$z_1 \cdot z_2 = r_1 e^{i\theta_1} \cdot r_2 e^{i\theta_2} = r_1 r_2 e^{i(\theta_1+\theta_2)}. \tag{1}$$

This result implies that the magnitudes of the complex numbers are multiplied, and their angles (or arguments) are added. Therefore, multiplication in polar form corresponds to a rotation (given by the sum of angles) and a scaling (given by the product of magnitudes) in the complex plane.

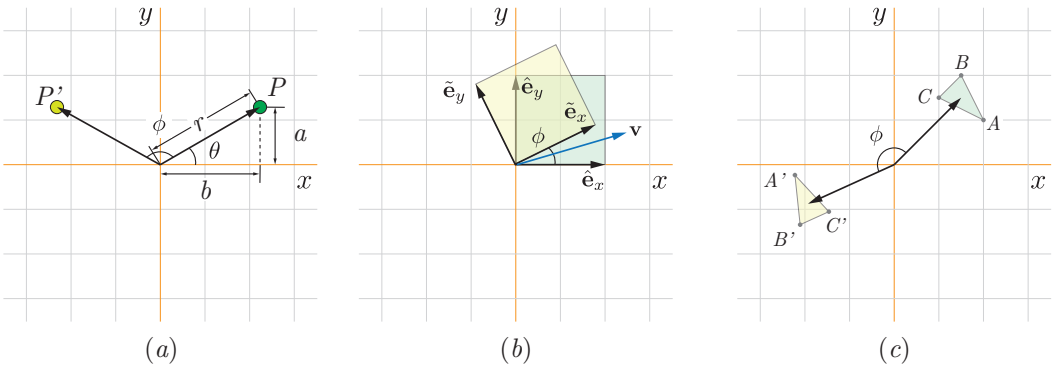

Figure 1: Schematic showing rotation of 3 different objects in two-dimensions: (a) a vector P located at a distance r and an angle θ (with $a = r\sin\theta, b = r\cos\theta$) with respect to the origin which when rotated by an angle ϕ transforms to P', (b) a coordinate basis, $\hat{\mathbf{e}}_x, \hat{\mathbf{e}}_y$ when rotated by an angle ϕ transforms to $\tilde{\mathbf{e}}_x, \tilde{\mathbf{e}}_y$, (c) a triangle ABC when rotated by an angle ϕ reaches $A'B'C'$. Refer `python`-notebook for implementation.

2.2 Vector components in a rotated frame

We briefly move away from the complex plane representation of rotation to the cartesian plane and connect them by the end of this sub-section. Let us consider two basis vectors $\hat{\mathbf{e}}_x, \hat{\mathbf{e}}_y$ pointing along x, y–axes (see Fig. 1(b)) which when rotated by an angle ϕ in the counter-clockwise direction gives $\tilde{\mathbf{e}}_x, \tilde{\mathbf{e}}_y$. It is easy to see that $\hat{\mathbf{e}}_x \cdot \tilde{\mathbf{e}}_x = \hat{\mathbf{e}}_y \cdot \tilde{\mathbf{e}}_y = \cos\phi$ and $\hat{\mathbf{e}}_y \cdot \tilde{\mathbf{e}}_x = -\hat{\mathbf{e}}_x \cdot \tilde{\mathbf{e}}_y = \sin\phi$. Now a vector $\mathbf{v}$ with components $\{v_1, v_2\}$ in the $\hat{\mathbf{e}}_i$ basis transforms to components $\{\tilde{v}_1, \tilde{v}_2\}$ in the rotated basis $\tilde{\mathbf{e}}_i$ i.e., $\mathbf{v} = v_1\hat{\mathbf{e}}_x + v_2\hat{\mathbf{e}}_y$ and $\tilde{\mathbf{v}} = \tilde{v}_1\tilde{\mathbf{e}}_x + \tilde{v}_2\tilde{\mathbf{e}}_y$. A compact way of transforming between these two basis is to write $\mathbf{v} = \mathcal{Q}^{\mathsf{T}}\tilde{\mathbf{v}}$ and $\tilde{\mathbf{v}} = \mathcal{Q}\mathbf{v}$ where $\mathcal{Q} = \{\tilde{\mathbf{e}}_x, \tilde{\mathbf{e}}_y\}$ with $\mathcal{Q}$ being the rotation tensor. The rotation tensor $\mathcal{Q}$ can be written explicitly as

$$\mathcal{Q} = \begin{pmatrix} \tilde{\mathbf{e}}_x \cdot \hat{\mathbf{e}}_x & \tilde{\mathbf{e}}_x \cdot \hat{\mathbf{e}}_y \\ \tilde{\mathbf{e}}_y \cdot \hat{\mathbf{e}}_x & \tilde{\mathbf{e}}_y \cdot \hat{\mathbf{e}}_y \end{pmatrix} = \begin{pmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{pmatrix}.$$

This matrix $\mathcal{Q}$ is indeed the transpose of the standard rotation matrix $\mathcal{R}$ i.e., $\mathcal{Q} = \mathcal{R}^{\mathsf{T}}$ for a counter-clockwise rotation, confirming that rotating the basis $\hat{\mathbf{e}}_i$ by a counter-clockwise angle of ϕ rotates the vector $\mathbf{v}$ by a clockwise angle ϕ in the transformed frame $\tilde{\mathbf{e}}_i$. This rotation tensor, $\mathcal{Q}$ is fundamental to what we will see in the second part of the primer. We hope you are holding your breath for the arrival of part 2.

Rotation in complex plane vs cartesian plane

We can express a complex number $z = re^{i\theta}$ in cartesian coordinates as vector $\mathbf{r} = \{x, y\}$ with $x = r\cos\theta$ and $y = r\sin\theta$ being the x and y component of the vector. When this vector is rotated counter-clockwise by an angle ϕ, the new orientation is simply,

$$\tilde{z} = re^{i(\theta+\phi)} = r\{\cos(\theta+\phi) + i\sin(\theta+\phi)\}.$$

Example 1: Rotating a point using complex number

Consider a point P represented by the complex number $z = 1 + i\sqrt{3}$ (which corresponds to the point $(1, \sqrt{3})$ in the 2D plane, forming an angle of $\frac{\pi}{3}$ with the positive real axis). To rotate this point by 120 degrees (or $\frac{2\pi}{3}$ radians), we multiply it by $e^{i\frac{2\pi}{3}}$ (which represents a rotation of $\frac{2\pi}{3}$ radians in the complex plane). The multiplication is as follows:

$$z' = z \times e^{i\frac{2\pi}{3}} = (1 + i\sqrt{3}) \times e^{i\frac{2\pi}{3}} = (1 + i\sqrt{3}) \times \left(-\frac{1}{2} + i\frac{\sqrt{3}}{2}\right).$$

Expanding this product, we get:

$$z' = \left(1 \times -\frac{1}{2}\right) + \left(1 \times i\frac{\sqrt{3}}{2}\right) + \left(i\sqrt{3} \times -\frac{1}{2}\right) + \left(i\sqrt{3} \times i\frac{\sqrt{3}}{2}\right) = -\frac{1}{2} - \frac{3}{2} + i\left(\frac{\sqrt{3}}{2} - \frac{\sqrt{3}}{2}\right) = -2.$$

After the rotation, the new position z' of the point P in the complex plane is -2, which corresponds to the point $(-2, 0)$ in the 2D plane.

The rotation process can also be expressed in matrix form with the original vector $z \equiv \mathbf{r} = \{x, y\}$ transforming to $\tilde{z} \equiv \tilde{\mathbf{r}} = \{\tilde{x}, \tilde{y}\}$ through the relation

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \underbrace{\begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}}_{\text{Rotation matrix, } \mathcal{R}} \begin{pmatrix} x \\ y \end{pmatrix}.$$

2.3 Rotating an object

We have so far seen how to rotate a vector with respect to origin and how vectors behave in rotated coordinate systems. The last application of rotation matrix is to rotate an object such as a square or a triangle. The rotation of the object translates essentially to rotation of all the vectors defined within the boundary of the object. In the schematic shown in Fig. 1(c), the triangular object is defined by the region bounded by the edges connecting the vertices A, B, C. Rotating this translates to rotating all the vectors inside this boundary. However, because of the properties of the rotation matrix (discussed in the ensuing sub-section), it is sufficient to rotate the vertices along. Thus rotating A to A', B to B', C to C' by an angle ϕ and connecting them by straight lines is equivalent to rotating the triangle. See the python–code for an implementation of this rotation.

2.4 Properties of rotation matrix in 2D

Rotation matrices, $\mathcal{R}$ in 2D satisfy the following important properties:

- All rotation matrices are orthonormal matrices satisfying $\mathcal{R}\mathcal{R}^{\mathsf{T}} = \mathcal{R}^{\mathsf{T}}\mathcal{R} = \mathbb{I}$ with $\det(\mathcal{R}) = 1$. Here $\mathbb{I}$ is the identity matrix.
- Set of all orthonormal matrices with determinant 1 forms a group known as the special orthogonal group SO(2).
- Since the eigen values of $\mathcal{R}$ are always ± 1, $\mathcal{R}$ does not contribute to any stretching of the vector on which it acts.

3 Rotation in 3D

In this section, we extend our exploration of rotations from the 2D plane to three-dimensional space. While 2D rotations are confined to a single plane and can be effectively described using complex numbers, 3D rotations are inherently more intricate due to the additional degree of freedom. The Euler angle representation is one of the most common methods for describing these rotations. Despite its widespread use, Euler angles come with their own set of challenges, such as Gimbal lock, which we shall discuss later in this section. We will start with the Euler angle representation of rotating coordinate basis and objects in 3D. This sets the stage for the introduction of quaternions in sec. 4, which provide a powerful and efficient way to represent and compute rotations in 3D while simultaneously subverting the constraints of Euler angles.

3.1 Euler angle representation

Euler angles are a set of 3 angles, often represented by $\{\psi, \theta, \phi\}$, that denote the orientation of a rigid object with respect to a pre-defined coordinate axes. It is often used to transform any orthonormal coordinate basis from an initial configuration to a target configuration. An initial coordinate basis $\hat{\mathbf{e}}_i$, shown in Fig. 2, can be transformed into $\tilde{\mathbf{e}}_i$ by performing a sequence of 3 rotations with angles $\{\psi, \theta, \phi\}$. There are two ways by which this basis rotation can be achieved: (i) using extrinsic form, where rotations are performed with respect to a fixed global coordinate axes, $\hat{\mathbf{e}}_i$; (ii) intrinsic form, where rotation is performed with respect to the local reference frame of the object, $\hat{\mathbf{e}}'_i$ that gets modified with each rotation operation. These two frames are analogous to Eulerian and Lagrangian frames one might be familiar from classical mechanics. In the following section we will derive the rotation matrix for rotation about different fixed axis and look at how to put them together to achieve the extrinsic and the intrinsic forms of basis rotation with Euler angle representation.

3.2 Rotation around $\hat{\mathbf{e}}_i$

We will now derive the rotation matrix, $\mathcal{R}_i$ for $i = \{x, y, z\}$ that performs rotation around different axis, $\hat{\mathbf{e}}_i$ when acted on a vector represented in this basis. Fig. 2 shows schematically the effect of this rotation on a rigid object. When $\mathcal{R}_i$ acts on a vector or object, it leaves the component along the i-axis fixed while rotating the other two components. In essence, $\mathcal{R}_i$ performs 2D rotation with respect to i-axis.

- **Rotation about $\hat{\mathbf{e}}_x$**: The rotation matrix about $\hat{\mathbf{e}}_x$ by an angle ψ, denoted as $\mathcal{R}_x(\psi)$, is given by
$$\mathcal{R}_x(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi) & -\sin(\psi) \\ 0 & \sin(\psi) & \cos(\psi) \end{pmatrix}.$$
Here positive values of ψ denote rotation in the counter-clockwise direction when looking along $\hat{\mathbf{e}}_x$. This is shown schematically as the first step in extrinsic rotation in Fig. 2.

- **Rotation about $\hat{\mathbf{e}}_y$**: Rotation about $\hat{\mathbf{e}}_y$ by an angle θ can be performed by the matrix $\mathcal{R}_y(\theta)$ given by
$$\mathcal{R}_y(\theta) = \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix}.$$

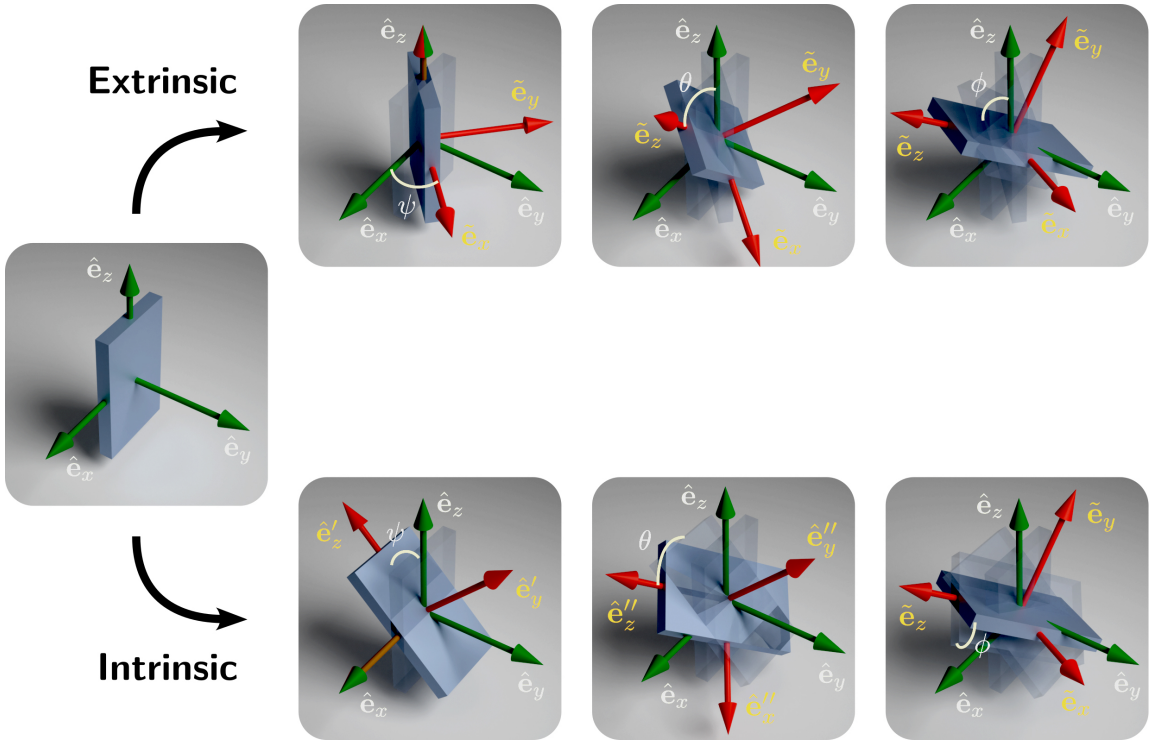

Figure 2: Schematic shows 2 ways of performing consecutive rotations to transform from one orientation to the other i.e., extrinsic and intrinsic rotations. Extrinsic rotation is performed about a fixed global axes, while intrinsic rotation about a moving axes that transforms with the object. It is important to note that rotations performed in extrinsic and intrisic form provide the same result (however the order of performing the operation is reversed, as discussed in the main text). Please see the linked python–code for an example.

- **Rotation about $\hat{\mathbf{e}}_z$**: Rotation about $\hat{\mathbf{e}}_z$, denoted as $\mathcal{R}_z(\phi)$, is

$$\mathcal{R}_z(\phi) = \begin{pmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Sequential rotation of a vector v

Consider a vector $\mathbf{v}$ with components in $\hat{\mathbf{e}}_i$ coordinate basis $\{v_1, v_2, v_3\}$. Performing a sequence of rotations of this vector will modify its orientation but the order of rotation determines the final orientation of $\mathbf{v}$. This is because rotation is intrinsically a non-cummutative operation, which is equivalent to saying that the order of the operation matters. When we rotate $\mathbf{v}$ around $\hat{\mathbf{e}}_x$ by ψ followed by rotation around $\hat{\mathbf{e}}_y$ by θ does not necessarily result in rotation around $\hat{\mathbf{e}}_y$ by θ followed around $\hat{\mathbf{e}}_x$ by ψ. This can be easily seen by the fact that matrix multiplication is not commutative. We can write this sequential operation and the non-cummutivity mathematically

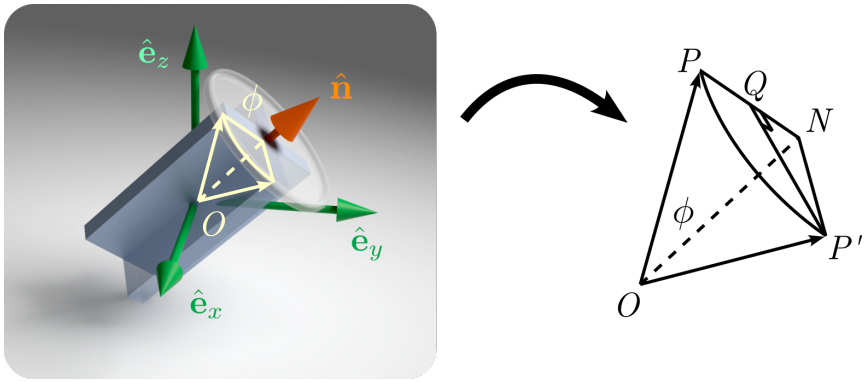

Figure 3: Schematic describes the rotation of vector $\mathbf{v}$ represented by $\vec{OP}$ about $\hat{\mathbf{n}}$ or $\vec{ON}$ by an angle ϕ thereby becoming the new vector $\tilde{\mathbf{v}}/\vec{OP'}$. This approach is also used to rotate a rigid body as shown here and the matrix to perform such a rotation in Eq. 8 is given the name Rodrigues formula.

as, $\mathcal{R}_x(\psi)\mathcal{R}_y(\theta) \neq \mathcal{R}_y(\theta)\mathcal{R}_x(\psi)$. No two sequences produce the same transformation unless they represent the same rotation.

3.3 Rotation around arbitrary direction

We have seen that simple rotations about $\hat{\mathbf{e}}_i$ can be performed by $\mathcal{R}_i$, as it extends 2D rotations. Now we look at how to perform rotation around a specified direction, $\hat{\mathbf{n}}$. This not only generalizes rotations around $\hat{\mathbf{e}}_i$ but is pivotal for transforming an object's configuration in three-dimensional space.

Let us consider a vector $\mathbf{v}$, shown schematically in Fig. 3, as $\overrightarrow{OP}$, that is to be rotated around another unit vector $\hat{\mathbf{n}}$ along $\overrightarrow{ON}$ by an angle ϕ to reach $\tilde{\mathbf{v}}/\overrightarrow{OP'}$. From Fig. 3 we can see that the rotated vector $\tilde{\mathbf{v}}$ can be expressed as the sum of three components,

$$\tilde{\mathbf{v}}/\overrightarrow{OP'} = \hat{\mathbf{n}} + \overrightarrow{NQ} + \overrightarrow{QP'}, \tag{2}$$

where $\overrightarrow{ON}$ is the component of $\mathbf{v}$ along the rotation axis $\hat{\mathbf{n}}$, $\overrightarrow{NQ}$ is the component of $\overrightarrow{NP'}$ perpendicular to $\hat{\mathbf{n}}$ within the plane containing $\hat{\mathbf{n}}$ and $\mathbf{v}$, and $\overrightarrow{QP'}$ is the component of $\overrightarrow{NP'}$ that is perpendicular to $\hat{\mathbf{n}}$ and $\mathbf{v}$. It is intuitive to see that the component along the axis, $\hat{\mathbf{n}}$ is along $\overrightarrow{ON} = (\mathbf{v}\cdot\hat{\mathbf{n}})\hat{\mathbf{n}}$. And the perpendicular component in the plane, $\overrightarrow{NP} = \mathbf{v} - \overrightarrow{ON}$. The rotated perpendicular component is then $\overrightarrow{NQ} = \cos(\phi)\overrightarrow{NP}$. The perpendicular component to the plane ultimately is $\overrightarrow{QP'} = \sin(\phi)(\hat{\mathbf{n}}\times\mathbf{v})$. Summing up these different components then gives the relation between $\mathbf{v}, \hat{\mathbf{n}}, \tilde{\mathbf{v}}$ and ϕ as,

$$\tilde{\mathbf{v}} = \cos(\phi)\mathbf{v} + [1-\cos(\phi)](\hat{\mathbf{n}}\cdot\mathbf{v})\hat{\mathbf{n}} + \sin(\phi)(\hat{\mathbf{n}}\times\mathbf{v}). \tag{3}$$

This can be reformulated using matrix notation as,

$$\tilde{\mathbf{v}} = \left[\cos(\phi)\mathbb{I} + (1-\cos(\phi))\hat{\mathbf{n}}\hat{\mathbf{n}}^{\mathrm{T}} + \sin(\phi)\boldsymbol{\epsilon}\hat{\mathbf{n}}\right]\mathbf{v}, \tag{4}$$

Example 2: Rotation using Rodrigues' formula

Consider a scenario where we have a vector $\mathbf{v} = (1, 0, 0)$. Our goal is to rotate this vector by an angle of $\frac{\pi}{4}$ radians around a unit vector $\mathbf{n} = \frac{1}{\sqrt{3}}(1, 1, 1)$.
First, we calculate the components of the rotation matrix $\mathcal{R}(\phi, \hat{\mathbf{n}})$ using Rodrigues' formula. With $\phi = \frac{\pi}{4}$ and $\hat{\mathbf{n}} = \frac{1}{\sqrt{3}}(1, 1, 1)$, the rotation matrix is computed as:

$$\mathcal{R}(\frac{\pi}{4}, \hat{\mathbf{n}}) = \frac{1}{2}\begin{pmatrix} 1+\frac{1}{\sqrt{3}} & 1-\frac{1}{\sqrt{3}} & 1-\frac{1}{\sqrt{3}} \\ 1-\frac{1}{\sqrt{3}} & 1+\frac{1}{\sqrt{3}} & 1-\frac{1}{\sqrt{3}} \\ 1-\frac{1}{\sqrt{3}} & 1-\frac{1}{\sqrt{3}} & 1+\frac{1}{\sqrt{3}} \end{pmatrix}.$$

Applying this matrix to the vector $\mathbf{v}$, we find the rotated vector $\tilde{\mathbf{v}}$ is,

$$\tilde{\mathbf{v}} = \mathcal{R}(\frac{\pi}{4}, \hat{\mathbf{n}})\mathbf{v} = \frac{1}{2}\begin{pmatrix} 1+\frac{1}{\sqrt{3}} \\ 1-\frac{1}{\sqrt{3}} \\ 1-\frac{1}{\sqrt{3}} \end{pmatrix}.$$

This rotated vector $\tilde{\mathbf{v}}$ represents the new orientation of $\mathbf{v}$ after the rotation.

where $\mathbb{I}$ is the 3×3 identity matrix, $\boldsymbol{\epsilon}\hat{\mathbf{n}}$ is the skew-symmetric matrix corresponding to the vector $\hat{\mathbf{n}}$, defined as

$$\boldsymbol{\epsilon}\hat{\mathbf{n}} = \begin{pmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{pmatrix}, \tag{5}$$

with $\boldsymbol{\epsilon} \equiv \epsilon_{ijk}$ being the third order anti-symmetric tensor with the property

$$\epsilon_{ijk} = \begin{cases} +1 & \text{if } (i,j,k) \text{ are even permutations of (1, 2, 3),} \\ -1 & \text{if } (i,j,k) \text{ are odd permutations of (1, 2, 3),} \\ 0 & \text{if } i = j \text{ or } j = k \text{ or } i = k. \end{cases}$$

and the expression $\boldsymbol{\epsilon}\hat{\mathbf{n}} = \sum_{k=1}^{3} \epsilon_{ijk} n_k$. Thus, the rotation matrix $\mathcal{R}(\phi, \hat{\mathbf{n}})$ is simply

$$\mathcal{R}(\phi, \hat{\mathbf{n}}) = \cos(\phi)\mathbb{I} + (1 - \cos(\phi))\hat{\mathbf{n}}\hat{\mathbf{n}}^{\mathsf{T}} + \sin(\phi)\boldsymbol{\epsilon}\hat{\mathbf{n}}, \tag{6}$$

and the rotated vector $\tilde{\mathbf{v}}$ is obtained simply by acting this matrix on $\mathbf{v}$, i.e., $\tilde{\mathbf{v}} = \mathcal{R}\mathbf{v}$.
The unit vector defining the axis of rotation $\hat{\mathbf{n}}$ in $\hat{\mathbf{e}}_i$ basis can be written as

$$\hat{\mathbf{n}} = n_1\hat{\mathbf{e}}_x + n_2\hat{\mathbf{e}}_y + n_3\hat{\mathbf{e}}_z, \tag{7}$$

with $||\hat{\mathbf{n}}||^2 = n_1^2 + n_2^2 + n_3^2 = 1$. The rotation matrix $\mathcal{R}(\phi, \hat{\mathbf{n}})$ explicitly reads as

$$\mathcal{R}(\phi, \hat{\mathbf{n}}) = \begin{pmatrix} n_1^2(1-\mathsf{c}(\phi)) + \mathsf{c}(\phi) & n_1 n_2 \cdot (1-\mathsf{c}(\phi)) - n_3\mathsf{s}(\phi) & n_1 n_3(1-\mathsf{c}(\phi)) + n_2\mathsf{s}(\phi) \\ n_1 n_2(1-\mathsf{c}(\phi)) + n_3\mathsf{s}(\phi) & n_2^2(1-\mathsf{c}(\phi)) + \mathsf{c}(\phi) & -n_1\mathsf{s}(\phi) + n_2 n_3(1-\mathsf{c}(\phi)) \\ n_1 n_3(1-\mathsf{c}(\phi)) - n_2\mathsf{s}(\phi) & n_1\mathsf{s}(\phi) + n_2 n_3 \cdot (1-\mathsf{c}(\phi)) & n_3^2(1-\mathsf{c}(\phi)) + \mathsf{c}(\phi) \end{pmatrix}, \tag{8}$$

where we have used the short-hand notation $\mathsf{c}(\bullet) \equiv \cos(\bullet), \mathsf{s}(\bullet) \equiv \sin(\bullet)$. This form of $\mathcal{R}(\phi, \hat{\mathbf{n}})$ is sometimes given the name Rodrigues' rotation formula.

3.4 Extrinsic rotation from $\hat{\mathbf{e}}_i$ basis to $\tilde{\mathbf{e}}_i$ basis

In this section we look at one of the most important functions of rotations, which is to transform a coordinate axes from an initial configuration to a target configuration. As we have mentioned briefly earlier and shown schematically in Fig. 2, rigid objects' orientation or coordinate basis can be transformed from initial to target configuration by either using a global fixed reference frame, referred to as extrinsic rotation or by using the local reference frame which transforms with every rotation operation, referred to as intrinsic form.

We will first look at the more intuitive extrinsic form of rotation between two coordinate basis denoted by A, the initial configuration and B, the target configuration shown in Fig. 2. The global coordinate basis is A which is aligned with global axes $\hat{\mathbf{e}}_i$ and it must be rotated to reach target frame B with axes $\tilde{\mathbf{e}}_i$. We can do this by performing three rotations: first around the first axis of frame A that is rotation about $\hat{\mathbf{e}}_z$, followed by rotation about $\hat{\mathbf{e}}_y$ and lastly by rotating about $\hat{\mathbf{e}}_x$. As it must be clear, this rotation sequence is performed in the extrinsic form as the global coordinate basis, $\hat{\mathbf{e}}_i$ is used for the transformation. It is important to note that this sequence of rotations, starting with the global x-axis, followed by the global y-axis, and then the global z-axis, has equivalent intrinsic rotations. In order to do this in intrinsic form, in Fig. 2 we show that we can traverse from A by first rotating with respect to $\hat{\mathbf{e}}_x$ to reach an intermediate frame A' (with axes $\{\hat{\mathbf{e}}'_x, \hat{\mathbf{e}}'_y, \hat{\mathbf{e}}_z\}$), and second rotation around $\hat{\mathbf{e}}'_y$ leading to frame A'' (with axes $\{\hat{\mathbf{e}}''_x, \hat{\mathbf{e}}''_y, \hat{\mathbf{e}}''_z\}$), and third rotation around $\hat{\mathbf{e}}''_z$ to arrive at frame B.

This particular sequence of rotation starting with the global z-axis followed by global y-axis then by global x-axis is called 321 rotation. The 321 rotation can be represented as composition of rotation matrices, $\mathcal{R}_i$ we have seen in the earlier section. We can perform the rotation about global z, y, x-axes by angles ψ, θ, ϕ using product of 3 $\mathcal{R}_i$ matrices given by,

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

In the context of 3D rotations, yaw (ψ), pitch(θ), and roll(ϕ) are commonly used terms for 321 rotation sequence:

$$\mathcal{R}_{\text{final}}(\psi, \theta, \phi) = \begin{pmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi) & \cos(\phi) \end{pmatrix}.$$

This results in the following matrix for $\mathcal{R}_{\text{final}}$:

$$\mathcal{R}_{\text{final}}(\psi, \theta, \phi) = \begin{pmatrix} \mathsf{c}(\psi)\mathsf{c}(\theta) & -\mathsf{s}(\psi)\mathsf{c}(\phi) + \mathsf{c}(\psi)\mathsf{s}(\theta)\mathsf{s}(\phi) & \mathsf{s}(\psi)\mathsf{s}(\phi) + \mathsf{c}(\psi)\mathsf{s}(\theta)\mathsf{c}(\phi) \\ \mathsf{s}(\psi)\mathsf{c}(\theta) & \mathsf{c}(\psi)\mathsf{c}(\phi) + \mathsf{s}(\psi)\mathsf{s}(\theta)\mathsf{s}(\phi) & -\mathsf{c}(\psi)\mathsf{s}(\phi) + \mathsf{s}(\psi)\mathsf{s}(\theta)\mathsf{c}(\phi) \\ -\mathsf{s}(\theta) & \mathsf{c}(\theta)\mathsf{s}(\phi) & \mathsf{c}(\theta)\mathsf{c}(\phi) \end{pmatrix}.$$

313 Rotation Sequence

Just like we can transform between $\hat{\mathbf{e}}_i$ to $\tilde{\mathbf{e}}_i$ via 321 rotations, we can do this transformation via 11 other sequences (which we will discuss soon). Another such sequence is the 313 rotation sequence for which the rotation matrix $\mathcal{R}$ is the product of rotations about $\hat{\mathbf{e}}_z$, followed by $\hat{\mathbf{e}}_x$, and again $\hat{\mathbf{e}}_z$. The rotation matrix $\mathcal{R}$ for the 313 sequence is given by,

$$\mathcal{R}(\psi, \theta, \phi) = \mathcal{R}_z(\psi)\mathcal{R}_x(\theta)\mathcal{R}_z(\phi),$$

Example 3: Rotating from initial basis $\hat{\mathbf{e}}_i$ to target basis $\tilde{\mathbf{e}}_i$

Consider a scenario where we need to rotate a coordinate system from an initial basis $\hat{\mathbf{e}}_i = \{\hat{\mathbf{e}}_x, \hat{\mathbf{e}}_y, \hat{\mathbf{e}}_z\}$ to a target basis $\tilde{\mathbf{e}}_i = \{\tilde{\mathbf{e}}_x, \tilde{\mathbf{e}}_y, \tilde{\mathbf{e}}_z\}$. Let's assume the rotation involves a yaw of $\frac{\pi}{6}$, a pitch of $\frac{\pi}{4}$, and a roll of $\frac{\pi}{3}$. Using the 321 rotation sequence, the final rotation matrix is calculated as follows:

$$\mathcal{R}_{\text{final}}(\psi,\theta,\phi) = \begin{pmatrix} \mathsf{c}(\frac{\pi}{6})\mathsf{c}(\frac{\pi}{4}) & \mathsf{s}(\frac{\pi}{3})\mathsf{s}(\frac{\pi}{4})\mathsf{c}(\frac{\pi}{6}) - \mathsf{s}(\frac{\pi}{6})\mathsf{c}(\frac{\pi}{3}) & \mathsf{s}(\frac{\pi}{3})\mathsf{s}(\frac{\pi}{6}) + \mathsf{s}(\frac{\pi}{4})\mathsf{c}(\frac{\pi}{3})\mathsf{c}(\frac{\pi}{6}) \\ \mathsf{s}(\frac{\pi}{6})\mathsf{c}(\frac{\pi}{4}) & \mathsf{s}(\frac{\pi}{3})\mathsf{s}(\frac{\pi}{6})\mathsf{c}(\frac{\pi}{4}) + \mathsf{c}(\frac{\pi}{3})\mathsf{c}(\frac{\pi}{6}) & \mathsf{s}(\frac{\pi}{6})\mathsf{s}(\frac{\pi}{4})\mathsf{c}(\frac{\pi}{3}) - \mathsf{s}(\frac{\pi}{3})\mathsf{c}(\frac{\pi}{6}) \\ -\mathsf{s}(\frac{\pi}{4}) & \mathsf{s}(\frac{\pi}{3})\mathsf{c}(\frac{\pi}{4}) & \mathsf{c}(\frac{\pi}{3})\mathsf{c}(\frac{\pi}{4}) \end{pmatrix}.$$

This matrix represents the combined effect of yaw, pitch, and roll rotations, transforming the initial basis $\hat{\mathbf{e}}_i$ to the target basis $\tilde{\mathbf{e}}_i$. The columns of this rotation matrix can be interpreted as follows:

- The first column represents the target basis vector $\tilde{\mathbf{e}}_x$ written in terms of the initial basis vectors $\hat{\mathbf{e}}_i$.
- The second column represents the target basis vector $\tilde{\mathbf{e}}_y$ written in terms of the initial basis vectors $\hat{\mathbf{e}}_i$.
- The third column represents the target basis vector $\tilde{\mathbf{e}}_z$ written in terms of the initial basis vectors $\hat{\mathbf{e}}_i$.

where $\mathcal{R}_z(\psi)$ and $\mathcal{R}_z(\phi)$ are the rotation matrices about the z-axis, and $\mathcal{R}_x(\theta)$ is the rotation matrix about the x-axis.
The final rotation matrix $\mathcal{R}_{\text{final}}(\psi,\theta,\phi)$ for the 313 rotation sequence can be written as,

$$\mathcal{R}_{\text{final}}(\psi,\theta,\phi) = \begin{pmatrix} -\mathsf{s}(\phi)\mathsf{s}(\psi)\mathsf{c}(\theta) + \mathsf{c}(\phi)\mathsf{c}(\psi) & -\mathsf{s}(\phi)\mathsf{c}(\psi) - \mathsf{s}(\psi)\mathsf{c}(\phi)\mathsf{c}(\theta) & \mathsf{s}(\psi)\mathsf{s}(\theta) \\ \mathsf{s}(\phi)\mathsf{c}(\psi)\mathsf{c}(\theta) + \mathsf{s}(\psi)\mathsf{c}(\phi) & -\mathsf{s}(\phi)\mathsf{s}(\psi) + \mathsf{c}(\phi)\mathsf{c}(\psi)\mathsf{c}(\theta) & -\mathsf{s}(\theta)\mathsf{c}(\psi) \\ \mathsf{s}(\phi)\mathsf{s}(\theta) & \mathsf{s}(\theta)\mathsf{c}(\phi) & \mathsf{c}(\theta) \end{pmatrix}.$$

In three-dimensional space, like the 321, 313 rotation sequence we have seen above, Euler angle rotations can be represented via a total of 12 distinct sequences. These sequences are derived from permutations of rotations about the three principal axes, $\hat{\mathbf{e}}_i$. For each sequence, the first and last rotations must be about different axes, and the middle rotation is about the axis not used in the first rotation. This rule leads to 12 unique combinations: 321, 323, 313, 312, 231, 213, 212, 232, 123, 132, 121, 131.

We have seen so far that the rotation between two different coordinate basis $\hat{\mathbf{e}}_i$ and $\tilde{\mathbf{e}}_i$ can be achieved either by using Rodrigues' formula with $\hat{\mathbf{n}}, \phi$ in Eq. 8 or through 3 Euler angles, $\{\psi,\theta,\phi\}$. In the former case of it might seem as though we require 4–degrees of freedom, $\{n_i,\phi\}$, the unit-vector constraint of $\hat{\mathbf{n}}$ i.e. $||\hat{\mathbf{n}}||^2 = 1$, however, reduces the required variables to 3. The attached `python`–code has details of such an implementation.

Connection between extrinsic and intrinsic transformations

We have seen that in the extrinsic form, each rotation is performed with respect to a fixed coordinate basis, $\hat{\mathbf{e}}_i$. These rotations are easier to visualize as they relate to a stationary frame of reference. In contrast, intrinsic rotations in Fig. 2 are performed with respect to the rotating coordinate system. The key to connection between extrinsic and intrinsic transformations lies in the order of applied rotations. An extrinsic rotation sequence can be converted to an intrinsic sequence by reversing the order of rotations and vice versa. This is because rotating an object first about one axis and then about another in a fixed coordinate system (extrinsic) is equivalent to rotating it about the second axis and then the first in its own coordinate system (intrinsic).

For example, an extrinsic rotation sequence about the 123 ($\hat{\mathbf{e}}_z \to \hat{\mathbf{e}}_y \to \hat{\mathbf{e}}_x$) is equivalent to an intrinsic rotation sequence 321 ($\hat{\mathbf{e}}_x \to \hat{\mathbf{e}}'_y \to \hat{\mathbf{e}}''_x$). Though the order of applied sequence is reversed,

the rotation matrix for both these sequences is the same. This relationship allows for the translation of a series of rotations from one perspective to another. In order to gain a deeper understanding of this connection, we refer the reader to this link.

Vector components in different coordinate basis

In sec. 2.2 we saw that the vector components in 2D transform under the rule, $\mathbf{v} = \mathcal{Q}^{\mathsf{T}}\tilde{\mathbf{v}}$ and $\tilde{\mathbf{v}} = \mathcal{Q}\mathbf{v}$. This transformation is valid also in 3D where the matrix $\mathcal{Q} = \{\tilde{\mathbf{e}}_x, \tilde{\mathbf{e}}_y, \tilde{\mathbf{e}}_z\}$, which can be written explicitly as

$$\mathcal{Q} = \begin{pmatrix} \tilde{\mathbf{e}}_x \cdot \hat{\mathbf{e}}_x & \tilde{\mathbf{e}}_x \cdot \hat{\mathbf{e}}_y & \tilde{\mathbf{e}}_x \cdot \hat{\mathbf{e}}_z \\ \tilde{\mathbf{e}}_y \cdot \hat{\mathbf{e}}_x & \tilde{\mathbf{e}}_y \cdot \hat{\mathbf{e}}_y & \tilde{\mathbf{e}}_y \cdot \hat{\mathbf{e}}_z \\ \tilde{\mathbf{e}}_z \cdot \hat{\mathbf{e}}_x & \tilde{\mathbf{e}}_z \cdot \hat{\mathbf{e}}_y & \tilde{\mathbf{e}}_z \cdot \hat{\mathbf{e}}_z \end{pmatrix}.$$

Let us consider three basis vectors $\hat{\mathbf{e}}_x, \hat{\mathbf{e}}_y, \hat{\mathbf{e}}_z$ pointing along the x, y, z-axis respectively. Similar to our 2D calculations, When the vectors $\hat{\mathbf{e}}_x, \hat{\mathbf{e}}_y$ are rotated by an angle ϕ in the counter-clockwise direction around the $\hat{\mathbf{e}}_z$ axis, they give rise to $\tilde{\mathbf{e}}_x, \tilde{\mathbf{e}}_y$. It is straightforward to see that $\hat{\mathbf{e}}_x \cdot \tilde{\mathbf{e}}_x = \hat{\mathbf{e}}_y \cdot \tilde{\mathbf{e}}_y = \cos\phi$ and $\hat{\mathbf{e}}_y \cdot \tilde{\mathbf{e}}_x = -\hat{\mathbf{e}}_x \cdot \tilde{\mathbf{e}}_y = \sin\phi$. The third basis vector $\hat{\mathbf{e}}_z$ remains unchanged during this rotation, so $\hat{\mathbf{e}}_z \cdot \tilde{\mathbf{e}}_z = 1$. The rotation tensor $\mathcal{Q}$ in 3D then reduces to

$$\mathcal{Q} = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Now, a vector $\mathbf{v}$ with components $\{v_1, v_2, v_3\}$ in the $\hat{\mathbf{e}}_i$ basis can be transformed to components $\{\tilde{v}_1, \tilde{v}_2, \tilde{v}_3\}$ in the rotated basis $\tilde{\mathbf{e}}_i$ using $\tilde{\mathbf{v}} = \mathcal{Q}\mathbf{v}$ and the inverse operation can be done using $\mathbf{v} = \mathcal{Q}^{\mathsf{T}}\tilde{\mathbf{v}}$. The `python`–code for an implementation of this to transform the basis vectors is available here.

Properties of rotation matrix in 3D

Apart from orthonormal property of rotation matrix (which is valid also in 3D), $\mathcal{R}$ has the following properties:

- A 3D rotation matrix $\mathcal{R}$ is an orthonormal matrix, meaning $\mathcal{R}^{\mathsf{T}}\mathcal{R} = \mathcal{R}\mathcal{R}^{\mathsf{T}} = \mathbb{I}$, where $\mathcal{R}^{\mathsf{T}}$ is the transpose of $\mathcal{R}$ and $\mathbb{I}$ is the identity matrix($\mathcal{R}^{-1} = \mathcal{R}^{\mathsf{T}}$.). The determinant of $\mathcal{R}$ in the special orthogonal group SO(3) is $+1$, indicating that the rotation preserves the orientation of a right-handed coordinate system: $\det(\mathcal{R}) = +1$.
- The set of all orthonormal matrices with determinant $+1$ forms the special orthogonal group SO(3).
- A 3D rotation matrix always has one real eigenvalue, which is $+1$, with the corresponding eigenvector being the axis of rotation. The other two eigenvalues are complex conjugates of each other and lie on the unit circle in the complex plane.
- Every orthonormal matrix in 3D corresponds to rotation about a specific axis by a certain angle, known as the axis-angle representation. The rotation axis, $\hat{\mathbf{n}}$ is the eigenvector associated with the eigenvalue $+1$ of the matrix, and the rotation angle ϕ can be calculated from the matrix's trace: $\phi = \arccos\left[\dfrac{\mathrm{tr}(\mathcal{R}) - 1}{2}\right]$.

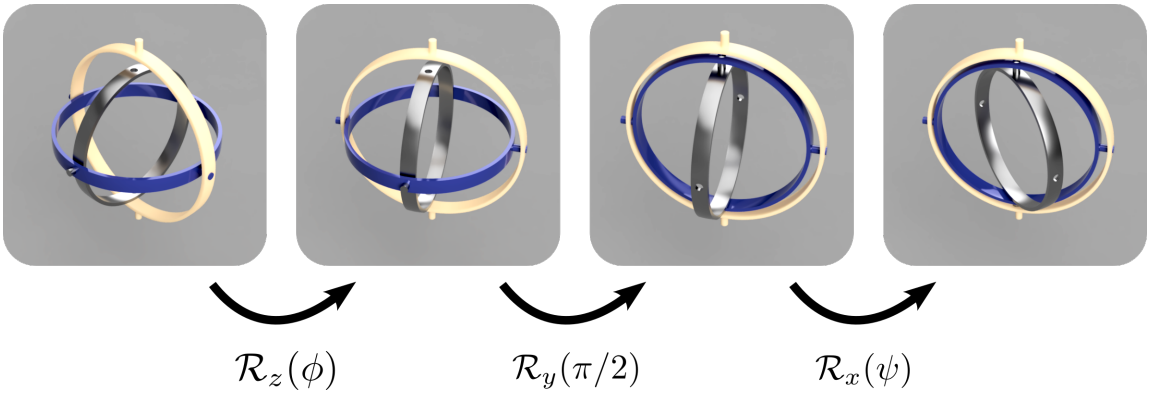

Figure 4: Schematic showing the phenomenon of Gimbal locking occuring given the initial configuration is transformed by the 321 rotation sequence. Such a locking situation arises when rotating the axis, $\hat{\mathbf{e}}_i$ by $\mathcal{R}_z(\phi)$ followed by $\mathcal{R}_y(\pi/2)$ then by $\mathcal{R}_x(\psi)$. It is worth noting that Gimbal locking occurs with all the 12 Euler angle sequences discussed in sec. 3.4.

3.5 Gimbal lock

Gimbal is a structure used to provide pivoted support to an object so that the object can be oriented in any specified direction (see Fig. 4). Gimbal lock is a phenomenon in which the connection of the rings pointing towards three directions lose a degree-of-freedom as two rings align themselves. This can occur in a sequence of rotations where one rotation leads to an alignment of two axes. For example, in a yaw-pitch-roll system (321-rotation sequence) shown in , if the pitch angle (θ) is rotated by $\pm 90°$, the yaw and the roll axes align. This alignment causes the loss of one degree of freedom, making it impossible to distinguish between yaw and roll motions.

$$\mathcal{R}_{\text{final}}(\psi,\theta,\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \mathsf{c}(\phi) & -\mathsf{s}(\phi) \\ 0 & \mathsf{s}(\phi) & \mathsf{c}(\phi) \end{pmatrix} \begin{pmatrix} \mathsf{c}(\theta) & 0 & \mathsf{s}(\theta) \\ 0 & 1 & 0 \\ -\mathsf{s}(\theta) & 0 & \mathsf{c}(\theta) \end{pmatrix} \begin{pmatrix} \mathsf{c}(\psi) & -\mathsf{s}(\psi) & 0 \\ \mathsf{s}(\psi) & \mathsf{c}(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

When $\theta = 90°$ or $\theta = \dfrac{\pi}{2}$ radians, the pitch rotation matrix simplifies to,

$$\mathcal{R}_y(\pi/2) = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{pmatrix}.$$

Multiplying these matrices together with $\theta = 90°$, we get the final rotation matrix as,

$$\mathcal{R}_{\text{final}}(\psi,\pi/2,\phi) = \mathcal{R}_x(\psi)\mathcal{R}_y(\pi/2)\mathcal{R}_z(\phi) = \begin{pmatrix} 0 & 0 & 1 \\ \mathsf{s}(\psi+\phi) & \mathsf{c}(\psi+\phi) & 0 \\ -\mathsf{c}(\psi+\phi) & \mathsf{s}(\psi+\phi) & 0 \end{pmatrix}. \tag{9}$$

This matrix shows that the rotation about the third axis is now the only distinct motion. The rotation about the yaw and roll axes ($\hat{\mathbf{e}}_z$ and $\hat{\mathbf{e}}_x$) becomes dependent on each other, represented by the combination of ψ and ϕ in the second and third rows. Essentially when $\theta = \pi/2$, the yaw

(ψ) and roll (ϕ) rotations combine, resulting in the loss of one degree of freedom. This is the phenomenon of gimbal lock, where the system cannot differentiate between yaw and roll rotations independently. Though we see that Gimbal lock is an issue seen in a Gimbal parameterised using Euler angles, it points to a fundamental issue with Euler angle representation of rotation. We see in Eq. 9 that we can no more distinguish between change in ψ or ϕ but rather only $(\psi + \phi)$. This happens to all rotation matrices, $\mathcal{R}$ parameterised with Euler angles at particular values of ψ or θ or ϕ.

The implications of gimbal lock are significant in fields that rely on precise 3D rotations, such as aerospace and robotics. In these fields, gimbal lock can lead to unpredictable behavior and loss of control. For instance, in an aircraft, if gimbal lock occurs, the pilot may lose the ability to control the aircraft's orientation accurately. One common method to overcome gimbal lock is to use a different representation for 3D rotations, such as quaternions. Quaternions do not suffer from gimbal lock because they represent rotations in four dimensions, avoiding the alignment issue inherent in three-dimensional Euler angles which we shall elucidate in the next section.

4 Quaternion

We have so far seen two forms of rotations, one using complex numbers in 2D and the standard matrix form acting on vectors in 2D and 3D. In this last section, we will see how to extend the ideas of complex numbers from 2D into 3D using quaternions. Quaternions form an interesting algebra, which is beyond the scope of this primer, nevertheless, the interested reader can look at [1, 2] for detailed discussions.

Quaternions are an extension of complex numbers, represented with one real part and three imaginary parts. A quaternion q is expressed as,

$$q = q_0 + q_1 i + q_2 j + q_3 k,$$

where $q_i \in \mathbb{R}$ for $i = \{0, \ldots, 3\}$, and i, j, k are the fundamental quaternion units. These units, similar to the imaginary constant in complex numbers, have specific multiplication rules that are crucial to their behaviour. They are,

$$\begin{aligned} i^2 = j^2 = k^2 = ijk = -1, \\ ij = k, \quad ji = -k, \\ jk = i, \quad kj = -i, \\ ki = j, \quad ik = -j. \end{aligned}$$

The non-commutative nature of quaternion multiplication (e.g., $ij \neq ji$) is a key feature that distinguishes them from complex numbers and allows for their unique application. Quaternions are also represented in 4–vector form as $q = \{q_0, \mathbf{q}\}$ where $\mathbf{q}$ is a vector in 3D with components $\mathbf{q} = \{q_1, q_2, q_3\}$. Quaternion multiplication is a key operation that combines the effects of two rotations. Given two quaternions $p = p_0 + p_1 i + p_2 j + p_3 k$ and $q = q_0 + q_1 i + q_2 j + q_3 k$, their product $p \star q$ is not commutative and can be expressed as

$$\begin{aligned} p \star q = (p_0 q_0 - p_1 q_1 - p_2 q_2 - p_3 q_3) + \\ (p_0 q_1 + p_1 q_0 + p_2 q_3 - p_3 q_2) i + \\ (p_0 q_2 - p_1 q_3 + p_2 q_0 + p_3 q_1) j + \\ (p_0 q_3 + p_1 q_2 - p_2 q_1 + p_3 q_0) k. \end{aligned} \tag{10}$$

This result can be rearranged into a matrix multiplication form, often used in numerical implementations, where the quaternion is represented as a 4×1 column vector, and the coefficients form a 4×4 matrix. The matrix representation of the quaternion multiplication is given by,

$$\begin{pmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{pmatrix} \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} p_0 q_0 - p_1 q_1 - p_2 q_2 - p_3 q_3 \\ p_0 q_1 + p_1 q_0 + p_2 q_3 - p_3 q_2 \\ p_0 q_2 - p_1 q_3 + p_2 q_0 + p_3 q_1 \\ p_0 q_3 + p_1 q_2 - p_2 q_1 + p_3 q_0 \end{pmatrix}.$$

We can write this multiplication operation in terms of vector operations as $p \star q = (p_0 q_0 - \mathbf{p} \cdot \mathbf{q}, p_0 \mathbf{q} + q_0 \mathbf{p} + \mathbf{p} \times \mathbf{q})$. Here $\cdot$, $\times$ operations are the standard dot–product and cross–product used for 3–vectors. Quaternions also have conjugates, $\bar{\mathbf{q}} = \{q_0, -q_1, -q_2, -q_3\}$, similar to complex numbers, and with this the multiplication becomes, $q \star \bar{q} = \{q \cdot q, 0, 0, 0\} = \{q_0^2 + \mathbf{q} \cdot \mathbf{q}, 0, 0, 0\} = \{\sum_{i=0}^{3} q_i^2, 0, 0, 0\}$.

4.1 Rotation about arbitrary direction

In order to capture rotation with quaternions, we first restrict them to be of unit magnitude, i.e. $q \star \bar{q} = 1$. Then we define the axis of rotation using quaternions and construct the rotation quaternion, followed by the action on the vector that must be rotated.

Let the unit vector denoting the axis of rotation be $\hat{\mathbf{n}} = n_i \hat{\mathbf{e}}_i$ with $\hat{\mathbf{n}} \cdot \hat{\mathbf{n}} = \sum_{i=1}^{3} n_i^2 = 1$ and the rotation quaternion, $q = q_0 + q_1 i + q_2 j + q_3 k$ be defined as,

$$\begin{aligned} q_0 &= \cos\left(\frac{\phi}{2}\right), \\ q_1 &= n_1 \sin\left(\frac{\phi}{2}\right), \\ q_2 &= n_2 \sin\left(\frac{\phi}{2}\right), \\ q_3 &= n_3 \sin\left(\frac{\phi}{2}\right), \end{aligned}$$

where ϕ is the angle of rotation. We can write this rotation quaternion compactly in 4–vector form as $q = \{\cos(\phi/2), \hat{\mathbf{n}} \sin(\phi/2)\}$. With this definition, the Rodrigues' formula for rotation matrix, $\mathcal{R}(\phi, \hat{\mathbf{n}})$ around $\hat{\mathbf{n}}$ by an angle ϕ in Eq. 8 can be written in terms of rotation quaternion components, q_0, q_1, q_2, and q_3 as,

$$\mathcal{R}(q) \equiv \mathcal{R}(\phi, \hat{\mathbf{n}}) = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix}. \tag{11}$$

Rotation of a vector

Given a vector $\mathbf{v}$, we know from sec. 3.3 that we can rotate a vector by an angle ϕ by constructing $\mathcal{R}(\phi, \hat{\mathbf{n}})$ and obtained the transformed vector by simply performing the matrix multiplication operation, $\tilde{\mathbf{v}} = \mathcal{R}(\phi, \hat{\mathbf{n}})\mathbf{v}$. However, in order to rotate a vector $\mathbf{v}$ using quaternions, we need to construct two quaternions: (i) rotation quaternion, $q = \{\cos(\phi/2), \hat{\mathbf{n}} \sin(\phi/2)\}$, (ii) 4–vector form

of $\mathbf{v}$ i.e. $p = \{0, \mathbf{v}\}$ (a pure quaternion, with vanishing first component i.e. $p_0 = 0$). The rotation operation is then simply, $(q \star p \star \bar{q})$. This can be expanded explicitly as

$$\tilde{p} = q \star \{0, \mathbf{v}\} \star \bar{q} = \{0, \mathcal{R}(q)\mathbf{v}\},$$

where the quaternion multiplication rule in Eq. 10 is used and $\tilde{p} = \{0, \tilde{\mathbf{v}}\}$.

Connecting Euler angles

We have seen that the rotation matrix, $\mathcal{R}(\phi, \hat{\mathbf{n}})$ can be mapped to a quaternion $q = \{\cos(\phi/2), \hat{\mathbf{n}}\sin(\phi/2)\}$. We can also do a similar transformation from Euler angle rotation $\mathcal{R}(\psi, \theta, \phi)$ to quaternion by first choosing the rotation sequence and then describe the rotation using a quaternion. Let us say we would like to follow 313 sequence, $R_{313}(\psi, \theta, \phi)$ can be written as quaternion q with components,

$$\begin{aligned}
q_0 &= \cos\left(\frac{\theta}{2}\right)\cos\frac{1}{2}\Big(\psi + \phi\Big), \\
q_1 &= \sin\left(\frac{\theta}{2}\right)\sin\frac{1}{2}\Big(\phi - \psi\Big), \\
q_2 &= \sin\left(\frac{\theta}{2}\right)\cos\frac{1}{2}\Big(\phi - \psi\Big), \\
q_3 &= \cos\left(\frac{\theta}{2}\right)\sin\frac{1}{2}\Big(\psi + \phi\Big).
\end{aligned}$$

4.2 Extension of complex numbers

As we have mentioned earlier, quaternions generalize complex numbers from 2D to 3D. In order to see this in action, we can write the rotation quaternion as $q = \exp(\{0, \hat{\mathbf{n}}\phi/2\})$. With this definition, we can see that the exponential of a pure quaternion (of unit modulus) satisfies,

$$\exp(\{0, \hat{\mathbf{n}}\phi/2\}) = \cos(\phi/2) + \hat{\mathbf{n}}\sin(\phi/2).$$

One easy way to check the validity is to set $\hat{\mathbf{n}} = \{1, 0, 0\}$ and see that the expression reduces to the standard complex number, $q = \cos(\phi/2) + i\sin(\phi/2)$.
Continuing, we have seen in sec. 2.1 that multiplying two complex numbers is simpler in the polar form. Multiplying quaternions in polar form follows a similar behaviour. Consider two quaternions in polar form, $p = \{r_1, \hat{\mathbf{n}}_1\phi_1/2\}, q = \{r_2, \hat{\mathbf{n}}_2\phi_2/2\}$, then the multiplication rule is simply,

$$p \star q = \{r_1, \hat{\mathbf{n}}_1\phi_1/2\} \star \{r_2, \hat{\mathbf{n}}_2\phi_2/2\} = \{r_1 r_2, \hat{\mathbf{m}}\phi_{12}/2\},$$

where $\cos(\phi_{12}/2) = p_0 q_0 - \mathbf{p}\cdot\mathbf{q}$ and

$$\hat{\mathbf{m}} = \frac{p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p}\times\mathbf{q}}{||p_0\mathbf{q} + q_0\mathbf{p} + \mathbf{p}\times\mathbf{q}||}.$$

It is easy to verify that $p \star q$ reduces to Eq. 1 when $\hat{\mathbf{n}}_1 = \hat{\mathbf{n}}_2 = \{1, 0, 0\}$.

Properties of quaternions

Quaternions are a compact way to represent rotations as one needs only 4 components of q instead of 9 components of $\mathcal{R}$. Further, all unit quaternions represent rotation and captures a space of S(3) representing a 3–sphere. Unlike Euler angle form of parameterising rotation matrix, $\mathcal{R}$ the unit quaternion, q avoids Gimbal locking because there are no singularities associated with them. This has been leveraged for interpolation in animation for camera movement. Nevertheless, quaternions are complex mathematically and are less intuitive in terms of what the values represent as they must be converted to $\phi, \hat{\mathbf{n}}$ to gain insights.

5 Conclusion

In summary, we have seen in this primer some of the properties of rotation in 2D and 3D with examples. We have seen how to rotate vectors, coordinate basis, and rigid objects. Finally, we introduced quaternions which are a compact way of representing rotations in 3D. The `python`–code accompanying this primer has the numerical implementation of the various methods discussed here. In the second part of this primer, we will extend ideas from this primer to curves. Rotations are intimately tied to curvature and twist about which we will explore in detail there.

References

[1] Andrew J Hanson. Visualizing quaternions. In *ACM SIGGRAPH 2005 Courses*, pages 1–es. 2005.

[2] John Vince. *Geometric algebra for computer graphics*. Springer Science & Business Media, 2008.

[3] John Vince and John A Vince. *Mathematics for computer graphics*, volume 251. Springer, 2006.