

Applications of Quaternions in 3D Rotation

Martin Velikov

To what extent can quaternions supersede rotation matrices constructed from Euler angles in representing 3D rotation?

Contents

1	Introduction	2
2	Rotation Matrices	3
2.1	In two dimensions	3
2.2	In three dimensions	7
2.3	Construction from Euler angles	8
2.3.1	Concept	8
2.3.2	Deriving rotation matrices for each angle	9
2.3.3	Multiplication	10
2.4	The gimball lock problem	11
3	Improving on Euler angles with Quaternions	12
3.1	Standard Form	12
3.2	Basic Quaternions	13
3.2.1	Multiplying by Real Numbers	13
3.2.2	Multiplication by other basic quaternions	13
3.2.3	Associativity	14
3.3	Quaternion Operations	14
3.3.1	Multiplication of non-basic quaternions	14
3.3.2	Conjugation	14
3.3.3	Multiplication by Conjugate	15
3.3.4	Inverse	15
3.4	Proof that qpq^{-1} returns a pure quaternion	15

1 Introduction

Historically, mathematicians have endeavoured to describe the world around us using mathematics. While mathematicians often create simplifications or models in lower dimensions, as is the case with fields of mathematics like trigonometry and complex numbers, we are ultimately inhabitants of a three dimensional world. In the modern day, demand for working in three dimensions is ever increasing, with scientific applications in aerodynamic modeling, astrophysics, weather modelling and computational chemistry, in addition to numerous uses in computer generated imagery (CGI). In all of these use cases, there is a need for a rigorous mathematical framework to represent and manipulate the 3D world.

One of the intuitively simplest, but surprisingly complex aspects of transitioning from a two dimensional space to three dimensional is the problem of rotation. Indeed, there currently exist several competing mathematical models for 3D rotation in common use which bear their own strengths and weaknesses. Of these, some of the most commonly seen constructs are 3D "rotation matrices", which are usually constructed from "Euler angles", and Quaternions.

This investigation aims to compare and contrast these methods of rotation in order to draw conclusions about the usefulness and applicability of Quaternions in 3D computer graphics and scientific simulations compared to rotation matrices constructed from Euler angles.

As an aspiring computer animator and game developer, this question is of real significance to me as it will prove to be invaluable experience in the underlying concepts of my field of work.

2 Rotation Matrices

Rotation matrices are considered one of the most versatile ways of doing 3D rotation. They are often used in computer animation and praised for their computational efficiency and versatility. Mathematically, they represent specific a 3D linear transformation which, when applied to a set of points, describes a 3D rotation.

2.1 In two dimensions

It is worthwhile conceptually exploring what happens in two dimensions first, as this will introduce valuable concepts which will help with understanding how three dimensional rotation matrices work.

Let 2D points be represented as a vector of the form $\vec{v} = \begin{bmatrix} x \\ y \end{bmatrix}$ where x and y represent the x and y coordinates of the point respective to the origin.

By representing the points as vectors, rotation of a point around the origin would be equivalent to rotating the vector. Consider the rotation of point $(1, 0)$ by 45 degrees anti-clockwise:

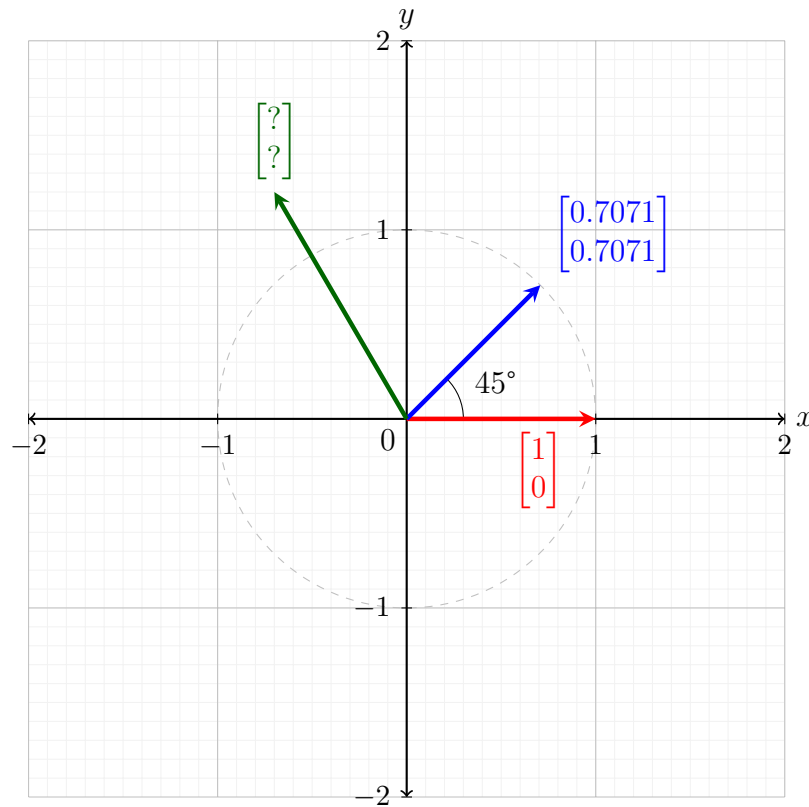


Figure 1: Apparent rotation of red vector $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ by 45° anti-clockwise to blue vector

From Figure 1, it may seem like calculating this rotation is trivial by using the trigonometric functions: for any angle θ , the rotation of $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ is equal to $\begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix}$.

Indeed, this is the case for the simple example above, but the applicability of this model quickly breaks down once the complexity is increased. For example, rotating the green vector about a point

different to the origin by using this model is difficult to imagine. Furthermore, if one was to go beyond rotation, how could the red vector be transformed into the green vector?

Linear transformation using matrices elegantly deals with these problems by introducing one concept - instead of transforming each individual vector, transform the axes themselves.

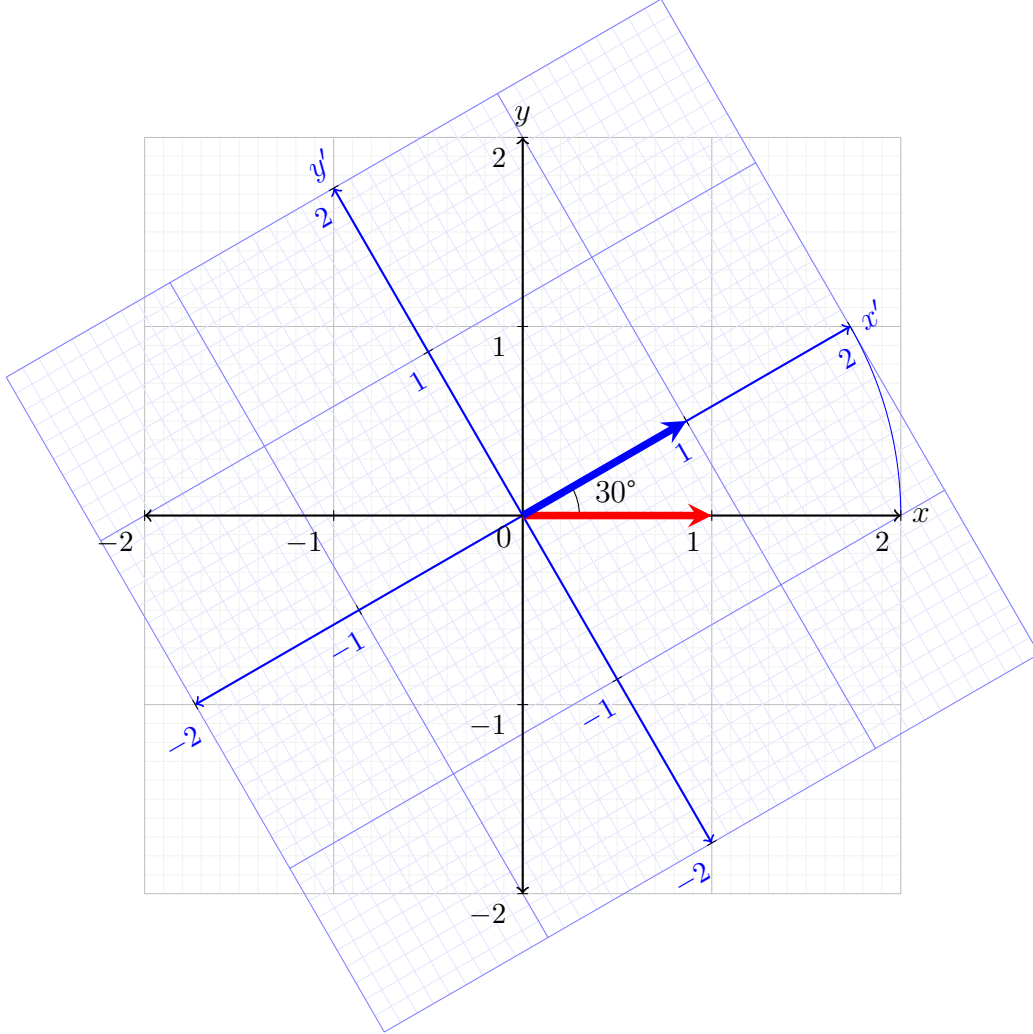


Figure 2: Overlay of new cartesian plane (blue) with axes x' and y' rotated 30° anti-clockwise relative to the original.

In Figure 2, the blue vector's coordinates on the rotated axes are identical to the red vector's coordinates on the original - both are $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$. In fact, any one point on the original axis has a counterpart on the rotated axis which corresponds to the rotation of that point about the origin by 30° . If one examines the coordinates of the blue vector b on the original axes, it becomes evident that $\vec{b} = \begin{bmatrix} 0.8660 \dots \\ 0.5 \end{bmatrix} = \begin{bmatrix} \cos 30^\circ \\ \sin 30^\circ \end{bmatrix}$, which aligns with findings from the trigonometric model.

Another invaluable benefit of this method is that one may also choose to rotate the axes about a different point to the origin, and observe the correct off-centric rotations for vectors without any changes to how the model is evaluated. Extrapolating this to three dimensions yields immeasurable savings in complexity and extra calculations which are expensive for computers to do.

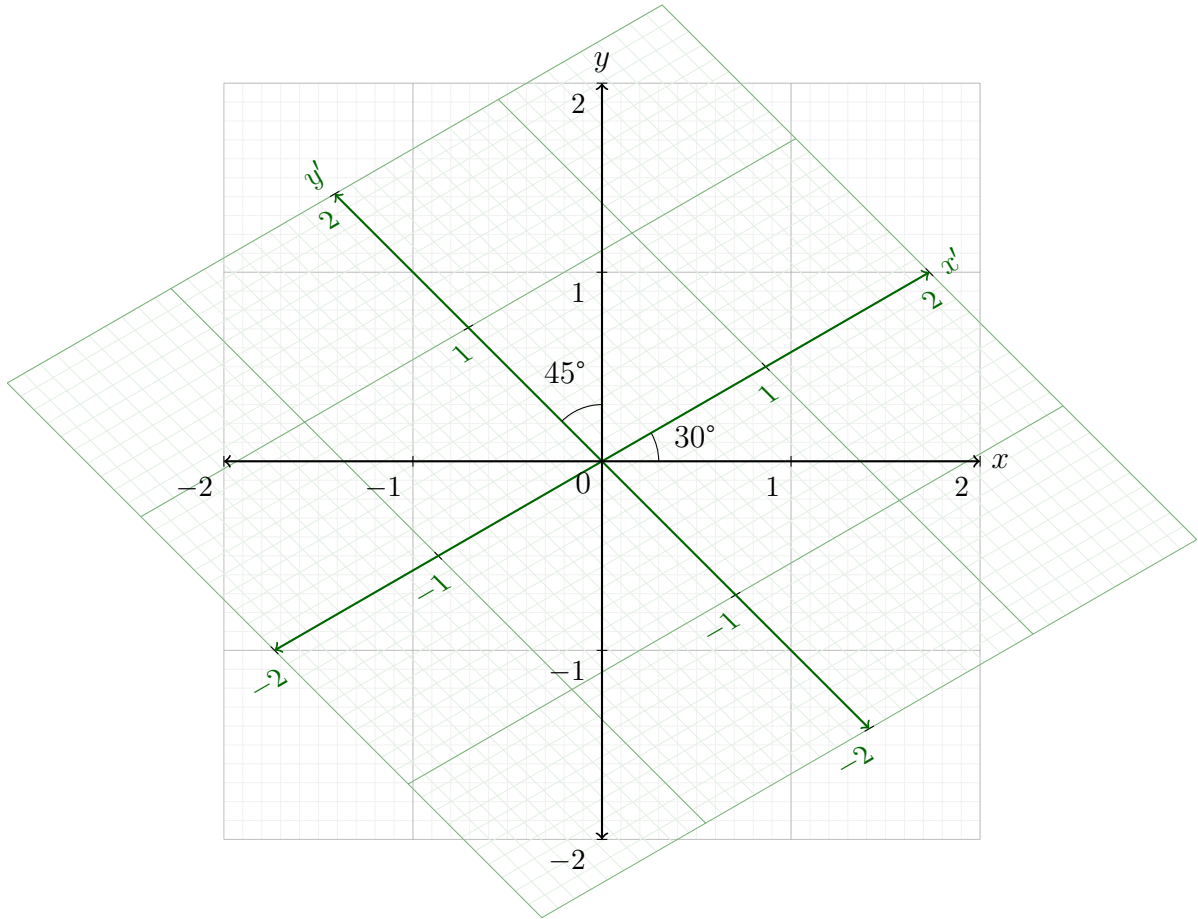


Figure 3: Overlay of new cartesian plane (green) with axes x' and y' rotated 30° anti-clockwise and 45° anti-clockwise respectively.

Figure 3 depicts what is known as a "shear". It occurs when each axis is rotated by a different angle, and results in warping of the transformed vectors, evident by the asymmetric appearance of the graph. If you were interested in going beyond rotation, there is an opportunity to translate, dilate and shear the axes to achieve various effects like the one depicted.

The elegance in this model comes from how one such transformation is actually computed. The first step to this is determining the direction and dilation of the transformed axes by examining what happens to the vectors $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ after the transformation. If the magnitude of any one of these vectors changes, that means that the respective axis has been dilated by a factor of the magnitude. If the direction changes, that means that that axis has been rotated. By knowing the final position of these two vectors after a transformation, the transformation that took place can be reconstructed and applied to more points.

The extremely useful tool that bridges the gap between this theory of axis shifting and mathematics is called a transformation matrix. 2D Transformation matrices are constructed by taking the vectors found above and storing them in a single 2×2 matrix:

Let x be the post-transform position of $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$, where $\vec{x} = \begin{bmatrix} x_x \\ x_y \end{bmatrix}$, and let y be the post-transform position of $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$, where $\vec{y} = \begin{bmatrix} y_x \\ y_y \end{bmatrix}$. The transformation matrix M is described by $M = \begin{bmatrix} x_x & y_x \\ x_y & y_y \end{bmatrix}$.

The transformation $T_M(\vec{v})$ of \vec{v} by matrix M can then be described as

$$T_M(\vec{v}) = M\vec{v} = \begin{bmatrix} x_x & y_x \\ x_y & y_y \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} x_x v_x + y_x v_y \\ x_y v_x + y_y v_y \end{bmatrix}$$

If \vec{x} and \vec{y} do not change after a transformation, they would be equal to $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ respectively. From this, a transformation matrix which does nothing (called the "identity") M_0 can be constructed in the form of $M_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. In this case, $T_{M_0}(\vec{v}) = \begin{bmatrix} (1)v_x + (0)v_y \\ (0)v_x + (1)v_y \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \vec{v}$

Considering the point $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ on a unit circle, a rotation by θ would be represented as $\begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix}$. Likewise, the rotation of the point $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ is equivalent to $\begin{bmatrix} -\sin \theta \\ \cos \theta \end{bmatrix}$. Hence, a 2D transformation matrix $R(\theta)$ can be constructed so that it rotates the 2D axes by any angle θ . This is what is known as a rotation matrix.

$$R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}$$

You can combine different linear transformations by multiplying a vector by multiple transformation matrices sequentially. For example, you might choose to apply a shear or a scale to your points before rotating them. This property applies to both two and three dimensional transformation matrices. Because of this versatility, transformation matrices are very easy to work with and integrate into more complex operations.

It is important to note that matrix multiplication is not commutative, and hence it is important to always multiply matrices in a right-to-left order: if one wanted to apply the linear transformations $T_1 \rightarrow T_2 \rightarrow T_3$ on vector \vec{v} , the correct multiplication would be $T_3 T_2 T_1 \vec{v}$.

2.2 In three dimensions

Now that we have established basic intuition on what 2D matrices actually do, we can extend the established observations to three dimensions.

In 3D, there are now three axes which must be kept track of:

- X (left \leftrightarrow right)
- Y (up \updownarrow down)
- Z (forward \swarrow back)

The identity vectors \vec{x}_0 , \vec{y}_0 and \vec{z}_0 in three dimensions, like we saw with 2D, have a magnitude of 1 and lie on the axis they represent.

$$\vec{x}_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \vec{y}_0 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \vec{z}_0 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

The 3D transformation defines the transition from the identity vectors into ones (\vec{x} , \vec{y} and \vec{z}) with potentially different magnitude and direction, signifying reoriented axes.

$$\vec{x} = \begin{bmatrix} x_x \\ y_x \\ z_x \end{bmatrix} \quad \vec{y} = \begin{bmatrix} x_y \\ y_y \\ z_y \end{bmatrix} \quad \vec{z} = \begin{bmatrix} x_z \\ y_z \\ z_z \end{bmatrix}$$

Changing the magnitude and/or direction of these vectors results in various effects on the 3D space, similar to how this works in two dimensions. Individually changing any one or two of these vectors results in the three dimensional equivalent of shear called "deformation". This property is often used in 3D animation to compress or stretch objects – a bouncing ball, for example.

As before, constructing a matrix out of these vectors allows us to leverage matrix multiplication to transform points. A 3×3 matrix M can now be constructed to represent a 3D transformation using the above axes. Let the identity of M be M_0 .

$$M = \begin{bmatrix} x_x & y_x & z_x \\ x_y & y_y & z_y \\ x_z & y_z & z_z \end{bmatrix} \quad \therefore \quad M_0 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

For a transformation $T_M(\vec{v})$ on $\vec{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}$ by matrix M ,

$$T_M(\vec{v}) = M\vec{v} = \begin{bmatrix} x_x & y_x & z_x \\ x_y & y_y & z_y \\ x_z & y_z & z_z \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} x_x v_x + y_x v_y + z_x v_z \\ x_y v_x + y_y v_y + z_y v_z \\ x_z v_x + y_z v_y + z_z v_z \end{bmatrix}$$

In our case, we seek to perform a rotation on a set of 3D points, meaning that we would need a form for a 3D rotation matrix. Like observed in the 2D case, a rotation could affect more than one axis, and so it is important to consider the matrix as a whole. Historically, many different ways of constructing rotation matrices have been shown, each with their benefits and downsides. In a sense, rotation matrices are not an entirely independent method of performing rotation, but are rather an underlying infrastructure for other methods to use.

2.3 Construction from Euler angles

2.3.1 Concept

In real life, it is practical to represent rotations with easily measurable metrics, like angles. Aircraft and watercraft make extensive use of yaw, pitch and roll angles in instrumentation to aid their pilots in judging the orientation of the craft. Yaw indicates a horizontal swivel, or a "heading", pitch indicates the angle of the nose relative to the horizon, and roll indicates how much the plane is turned about the forward axis. Figure 4 demonstrates this in terms of an x , y and z axis. In aircraft, y axis here can be considered as the "front" of the craft.

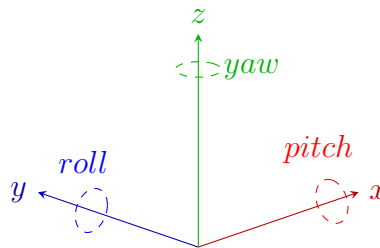


Figure 4: Yaw pitch and roll

Euler angles are based around this concept. A set of Euler angles is a 3D vector containing the yaw, pitch and roll of an object around a reference frame. In the example of aircraft, the reference frame is usually the surface of the Earth. In space, it is usually a prominent constellation or celestial object.

As this system of representing rotation is simple, measurable and intuitive for humans to use, one of the main ways of constructing a rotation matrix is through Euler angles.

2.3.2 Deriving rotation matrices for each angle

Consider rotating the 3D identity axes \vec{x}_0 , \vec{y}_0 and \vec{z}_0 by θ around the X axis. Let the matrix that describes this rotation in terms of θ be $R_x(\theta)$. After the transformation, let the identity vectors' new positions be \vec{x}_r , \vec{y}_r and \vec{z}_r .

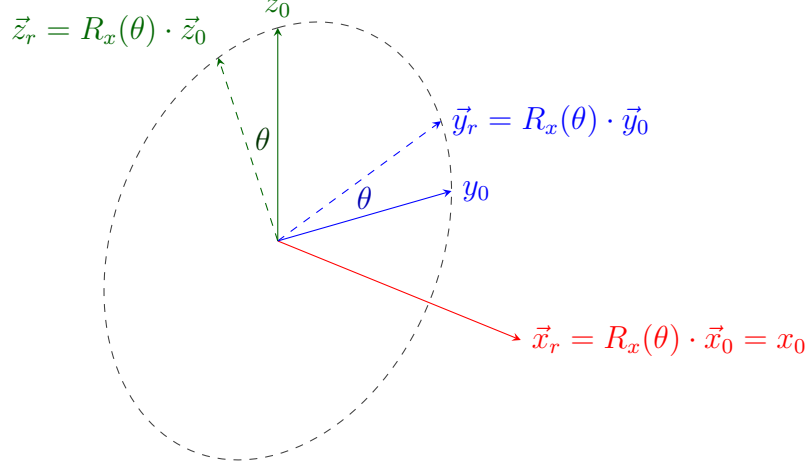


Figure 5: Rotation around the X-axis by angle θ . \vec{x}_r , \vec{y}_r and \vec{z}_r represent the identity vectors after the transformation by $R_x(\theta)$ where $\vec{x}_0 \rightarrow \vec{x}_r$, $\vec{y}_0 \rightarrow \vec{y}_r$ and $\vec{z}_0 \rightarrow \vec{z}_r$.

In Figure 5, the rotation $R_x(\theta)$ could be represented as

$$R_x(\theta) = \begin{bmatrix} x_{rx} & y_{rx} & z_{rx} \\ x_{ry} & y_{ry} & z_{ry} \\ x_{rz} & y_{rz} & z_{rz} \end{bmatrix}$$

From Figure 5 it becomes apparent that points colinear with \vec{x}_0 do not move at all, because in this case the rotation is happening along the X-axis. Hence,

$$\vec{x}_r = \vec{x}_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Points colinear with the Y axis get rotated around a unit circle in the YZ plane (pictured in Figure 5). Hence, \vec{y}_r is described with the trigonometric functions $\cos \theta$ and $\sin \theta$ as its Y and Z components. The rotation of \vec{z}_r can be derived similarly.

$$\vec{y}_r = \begin{bmatrix} 0 \\ \cos \theta \\ \sin \theta \end{bmatrix} \quad \vec{x}_r = \begin{bmatrix} 0 \\ -\sin \theta \\ \cos \theta \end{bmatrix}$$

Combining \vec{x}_r , \vec{y}_r and \vec{z}_r yields the rotation matrix $R_x(\theta)$ for rotations around the X-axis.

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

A similar process can be followed to derive the rotation matrices by the Y and Z axes, so it has been omitted for brevity. The only thing that changes in the derivation process is that the plane of the circle follows the two axes perpendicular to the rotation axis.

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

2.3.3 Multiplication

For the Euler angles α , β and γ representing yaw, pitch and roll respectively, it is now possible to compute a general rotation matrix by first constructing each individual matrix, then multiplying them. Effectively, this applies the rotation $R_x(\alpha)$ followed by $R_y(\beta)$ and finally $R_z(\gamma)$.

$$\begin{aligned} R(\alpha, \beta, \gamma) &= R_z(\gamma)R_y(\beta)R_x(\alpha) \\ &= \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix} \\ &= \begin{bmatrix} \cos \beta \cos \gamma & \sin \alpha \sin \beta \cos \gamma - \cos \alpha \sin \gamma & \cos \alpha \sin \beta \cos \gamma + \sin \alpha \sin \gamma \\ \cos \beta \sin \gamma & \sin \alpha \sin \beta \sin \gamma + \cos \alpha \cos \gamma & \cos \alpha \sin \beta \sin \gamma - \sin \alpha \cos \gamma \\ -\sin \beta & \sin \alpha \cos \beta & \cos \alpha \cos \beta \end{bmatrix} \end{aligned}$$

As matrix multiplication does not commute, it is not always the case that rotating by an axis A, then rotating by B would be the same as rotating first by axis B, then by A. Hence, it is important to note that the order of multiplication of R_z , R_y and R_x produces a unique final rotation. For the rest of the investigation, the $x \rightarrow y \rightarrow z$ rotation order seen above will be used. This is alternatively called the ZYX Tait-Bryan order.

2.4 The gimball lock problem

There is one large problem with constructing rotation matrices from Euler angles. Suppose you rotated an object with the Euler angles α , β and γ , where $\beta = \frac{\pi}{2}$

$$\begin{aligned}
 R(\alpha, \frac{\pi}{2}, \gamma) &= \begin{bmatrix} \cos \frac{\pi}{2} \cos \gamma & \sin \alpha \sin \frac{\pi}{2} \cos \gamma - \cos \alpha \sin \gamma & \cos \alpha \sin \frac{\pi}{2} \cos \gamma + \sin \alpha \sin \gamma \\ \cos \frac{\pi}{2} \sin \gamma & \sin \alpha \sin \frac{\pi}{2} \sin \gamma + \cos \alpha \cos \gamma & \cos \alpha \sin \frac{\pi}{2} \sin \gamma - \sin \alpha \cos \gamma \\ -\sin \frac{\pi}{2} & \sin \alpha \cos \frac{\pi}{2} & \cos \alpha \cos \frac{\pi}{2} \end{bmatrix} \\
 &= \begin{bmatrix} 0 & \sin \alpha \cos \gamma - \cos \alpha \sin \gamma & \cos \alpha \cos \gamma + \sin \alpha \sin \gamma \\ 0 & \sin \alpha \sin \gamma + \cos \alpha \cos \gamma & \cos \alpha \sin \gamma - \sin \alpha \cos \gamma \\ -1 & 0 & 0 \end{bmatrix} \\
 &= \begin{bmatrix} 0 & \sin(\alpha - \gamma) & \cos(\alpha - \gamma) \\ 0 & \cos(\alpha - \gamma) & -\sin(\alpha - \gamma) \\ -1 & 0 & 0 \end{bmatrix}
 \end{aligned}$$

Let $\alpha - \gamma = \theta$. The rotation matrix R now becomes $R(\theta) = \begin{bmatrix} 0 & \sin \theta & \cos \theta \\ 0 & \cos \theta & -\sin \theta \\ -1 & 0 & 0 \end{bmatrix}$.

By inspecting this matrix, we find that:

- \vec{x}_0 will always transform to $-\vec{z}_0$. This means that this is the axis around which rotation occurs by angle θ .
- \vec{y} and \vec{z} revolve around a unit circle in the YX plane by angle θ .
- This matrix is entirely in terms of θ . It does not matter whether the α or γ is changed, the resulting rotation will always be around the $-\vec{z}_0$ axis.

Hence, this shows that when $\beta = \frac{\pi}{2}$, the rotation matrix actually loses one degree of rotational freedom entirely as α and γ describe rotation about the same axis. In fact, the same can be shown for the case where one of α , β or γ is equal to $\frac{a\pi}{2}$ where $a = (2b + 1)$, $b \in \mathbb{Z}$.

This phenomenon is known as "gimball lock". It is widely considered to be one of the largest issues with using Euler angles for rotation. It is significant because it means that for specific angles of rotation, the system can actually lose the ability to freely rotate, and may even be unable to escape the gimball lock.

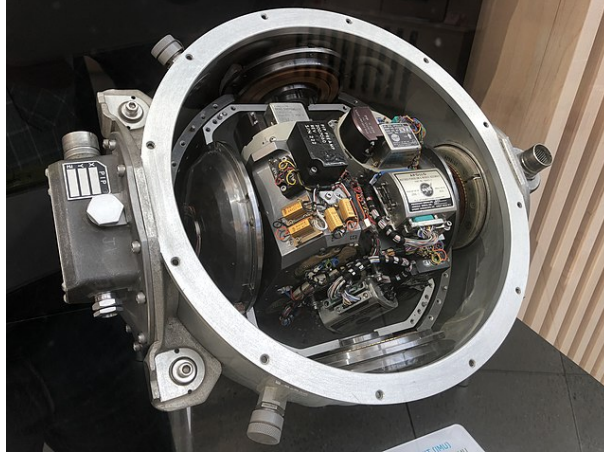


Figure 6: The Apollo 11 "Inertial Measurement Unit" responsible for guidance & orientation of the spacecraft in space.

One of the most famous examples of gimbal lock in action was during the 1969 Apollo 11 manned mission to the moon. The Apollo 11 spacecraft was equipped with an "Inertial Measurement Unit" which was responsible for measuring the orientation of the spacecraft in space. This was done using three gyroscopes, each of which measured the rotation of the spacecraft in terms of yaw, pitch and roll relative to the craft's starting position. Mechanically, these gyroscopes, too undergo gimbal lock when the spacecraft is oriented at specific angles resulting in complete disorientation of the guidance unit. This happened in-flight during the actual mission, meaning that the unit had to be manually moved out of the gimbal lock position using the stars as a reference, in order to regain the lost degree of freedom.

This fundamental flaw with Euler angles demonstrates how despite their simplicity and intuitiveness, they clearly leave a lot to be desired for rigorously representing 3D rotations.

3 Improving on Euler angles with Quaternions

In 1843, Irish mathematician Sir William Rowan Hamilton proposed an alternative way to represent 3D rotation by using four dimensional imaginary numbers called "quaternions". Quaternions allow us to rotate any point vector around any axis by a certain angle. This method of rotation is not anything special in particular - in fact, rotating around the X, Y and Z axes individually has already been explored in section 2.3.2.

3.1 Standard Form

Quaternions, like complex numbers, are defined with "real" and "imaginary" parts, and are essentially an extension of the complex numbers to four dimensions. The set of all quaternions is known as \mathbb{H} , named after the last initial of the Irish mathematician William Rowan Hamilton. Algebraically, a quaternion q can be defined in terms of the coefficients of its terms:

$$q = a + bi + cj + dk \quad a, b, c, d \in \mathbb{R}$$

i , j and k , called "basic quaternions", do not have an explicit definition of their value but are rather defined expressly in terms of the way they interact with each other in that they must satisfy the equality

$$i^2 = j^2 = k^2 = ijk = -1$$

3.2 Basic Quaternions

3.2.1 Multiplying by Real Numbers

For any $n \in \mathbb{R}$, it is defined that $in = ni$, $jn = nj$ and $kn = nk$. Hence, for any $q \in \mathbb{H}$, $qn = nq$. Quaternion multiplication by real numbers does, in fact, commute.

3.2.2 Multiplication by other basic quaternions

Hamilton's quaternion definition can then be used to derive the multiplicative interactions between i , j and k :

$$\begin{array}{ll}
 ijk = k^2 & ijk = i^2 \\
 ijk^2 = k^2k & i^2jk = i \cdot i^2 \\
 ij(-1) = (-1)k & (-1)jk = i(-1) \\
 ij = k & jk = i \\
 \\
 i = jk & k = ij \\
 ji = j^2k & kj = ij^2 \\
 ji = -k & kj = -i \\
 \\
 ij = k & \\
 kij = k^2 & k = ij \\
 kij^2 = k^2j & ik = i^2j \\
 ki(-1) = (-1)j & ik = -j \\
 ki = j &
 \end{array}$$

An important concept becomes apparent from the above calculations - quaternion multiplication by other quaternions is not commutative, that is, it can be the case that $q_1q_2 \neq q_2q_1$ where $q_1, q_2 \in \mathbb{H}$. For example, it is seen above that while $ij = k$, $ji = -k$.

The multiplication table of basic quaternions is hence formed:

\times	i	j	k
i	-1	$-k$	j
j	k	-1	$-i$
k	$-j$	i	-1

Table 1: Basic quaternion noncommutative multiplication table

3.2.3 Associativity

Quaternion multiplication is associative in that $(q_1q_2)q_3 = q_1(q_2q_3)$ where $q_1, q_2, q_3 \in \mathbb{H}$. The same property applies for addition, $(q_1 + q_2) + q_3 = q_1 + (q_2 + q_3)$ where $q_1, q_2, q_3 \in \mathbb{H}$.

This associativity allows for application of useful algebraic techniques like the distributive law.

3.3 Quaternion Operations

3.3.1 Multiplication of non-basic quaternions

Let $q_1 = a_1 + b_1i + c_1j + d_1k$ and $q_2 = a_2 + b_2i + c_2j + d_2k$ where $a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2 \in \mathbb{R}$. The multiplication q_1q_2 can be computed using the distributive law.

$$\begin{aligned} q_1q_2 &= (a_1 + b_1i + c_1j + d_1k)(a_2 + b_2i + c_2j + d_2k) \\ &= a_1a_2 + a_1b_2i + a_1c_2j + a_1d_2k \\ &\quad + b_1a_2i + b_1b_2i^2 + b_1c_2ij + b_1d_2ik \\ &\quad + c_1a_2j + c_1b_2ji + c_1c_2j^2 + c_1d_2jk \\ &\quad + d_1a_2k + d_1b_2ki + d_1c_2kj + d_1d_2k^2 \end{aligned}$$

Applying the basic quaternion rules then

factoring out the real part and i , j , and k ,

$$\begin{aligned} &= a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2 \\ &\quad + (a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2)i \\ &\quad + (a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2)j \\ &\quad + (a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2)k \end{aligned}$$

3.3.2 Conjugation

For any quaternion $q = a + bi + cj + dk$ $a, b, c, d \in \mathbb{R}$, its conjugate q^* is defined as $q^* = a - bi - cj - dk$.

3.3.3 Multiplication by Conjugate

Suppose one was to multiply a quaternion q by its conjugate q^* ,

$$\begin{aligned} qq^* &= (a + bi + cj + dk)(a - bi - cj - dk) \\ &= a^2 - abi - acj - adk \\ &\quad + bia - (bi)^2 - bicj - bidk \\ &\quad + cja - cjbi - (cj)^2 - cjdk \\ &\quad + dka - dkbi - dkci - (dk)^2 \end{aligned}$$

Reordering real coefficients and expanding

$$\begin{aligned} &= a^2 - abi - acj - adk \\ &\quad + abi - b^2i^2 - bcij - bdik \\ &\quad + acj - bcji - c^2j^2 - cdjk \\ &\quad + adk - bdk i - cdkj - d^2k^2 \end{aligned}$$

Applying basic quaternion rules

$$\begin{aligned} &= a^2 - abi - acj - adk \\ &\quad + abi + b^2 - bck + bdj \\ &\quad + acj + bck + c^2 - cdi \\ &\quad + adk - bdj + cdi + d^2 \end{aligned}$$

$$a, b, c, d \in \mathbb{R}$$

$$\therefore a^2 + b^2 + c^2 + d^2 \in \mathbb{R}$$

Hence, if you multiply a quaternion by its conjugate, the result will always be a real number and equal to the sum of the squares of its coefficients.

3.3.4 Inverse

The inverse q^{-1} of a quaternion $q = a + bi + cj + dk$ exists such that $qq^{-1} = 1$, effectively "undoing" any multiplication caused by q . The inverse of a quaternion can be computed using the previously established rules.

$$\begin{aligned} qq^{-1} &= 1 \Rightarrow q^{-1} = \frac{1}{q} \\ \frac{1}{q} \cdot \frac{q^*}{q^*} &= \frac{q^*}{qq^*} = \frac{q^*}{a^2 + b^2 + c^2 + d^2} \end{aligned}$$

3.4 Proof that qpq^{-1} returns a pure quaternion

Whenever a pure quaternion is multiplied by a rotation quaternion, the transformation inevitably distorts it into the fourth dimension.

$$\begin{aligned}
qp &= (a_q a_p - b_q b_p - c_q c_p - d_q d_p) \\
&+ (a_q b_p + b_q a_p + c_q d_p - d_q c_p)i \\
&+ (a_q c_p - b_q d_p + c_q a_p + d_q b_p)j \\
&+ (a_q d_p + b_q c_p - c_q b_p + d_q a_p)k
\end{aligned}$$

Let:

$$\begin{aligned}
a_{qp} &= a_q a_p - b_q b_p - c_q c_p - d_q d_p, \\
b_{qp} &= a_q b_p + b_q a_p + c_q d_p - d_q c_p, \\
c_{qp} &= a_q c_p - b_q d_p + c_q a_p + d_q b_p, \\
d_{qp} &= a_q d_p + b_q c_p - c_q b_p + d_q a_p
\end{aligned}$$

As q is of norm one, $q^{-1} = q^*$

$$\begin{aligned}
\therefore qpq^{-1} &= qpq^* \\
&= a_{qp}a_q + b_{qp}b_q + c_{qp}c_q + d_{qp}d_q \\
&+ (-a_{qp}b_q + b_{qp}a_q - c_{qp}d_q + d_{qp}c_q)i \\
&+ (-a_{qp}c_q + b_{qp}d_q + c_{qp}a_q - d_{qp}b_q)j \\
&+ (-a_{qp}d_q - b_{qp}c_q + c_{qp}b_q + d_{qp}a_q)k
\end{aligned}$$

In order for qpq^* to be pure, its real part must be equal to zero.

Let a_{qpq^*} be the real part of qpq^* .

$$\begin{aligned}
a_{qpq^*} &= a_{qp}a_q + b_{qp}b_q + c_{qp}c_q + d_{qp}d_q \\
&= a_q(a_q a_p - b_q b_p - c_q c_p - d_q d_p) \\
&+ b_q(a_q b_p + b_q a_p + c_q d_p - d_q c_p) \\
&+ c_q(a_q c_p - b_q d_p + c_q a_p + d_q b_p) \\
&+ d_q(a_q d_p + b_q c_p - c_q b_p + d_q a_p) \\
&= a_q^2 a_p - a_q b_q b_p - a_q c_q c_p - a_q d_q d_p \\
&+ b_q a_q b_p + b_q^2 a_p + b_q c_q d_p - b_q d_q c_p \\
&+ c_q a_q c_p - c_q b_q d_p + c_q^2 a_p + c_q d_q b_p \\
&+ d_q a_q d_p + d_q b_q c_p - d_q c_q b_p + d_q^2 a_p \\
&= a_q^2 a_p + b_q^2 a_p + c_q^2 a_p + d_q^2 a_p \\
&= a_p(a_q^2 + b_q^2 + c_q^2 + d_q^2) \\
&= a_p = 0 \\
\therefore qpq^{-1} &\text{ is pure for } \|q\|^2 = 1, \Re(p) = 0
\end{aligned}$$