3D Rigid Body Motion

Lin

Abstract

This note simply concludes some basic knowledge about three dimensonal rigid body motion.

1 Preliminary

1.1 Points, Vectors, Coordinate Frames

When it comes to localization and motion in 3D space, a rigid has two attribute: position and orientation. Position reflects a rigid's exact location with respect to a specific reference and is often descirbed by its coordinate. Orientation reflects the direction it faces towards.

Points and vectors are the most basic components in space. Points don't have length and volume. Connecting two points forms a vector. A vector is an existence in space and is independent of its coordinates. Therefore, a vector $\mathbf{v}$ does not need to associated with several real numbers. Only when we specify a coordinate frame in this 3D space can we talk about the vectors coordinates in this system,finding several real numbers corresponding to this vector.

A coordinate frame can be specified by a set of base $(\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3})$. Given this, we can use a coordinate (v_1, v_2, v_3) to describe an arbitrary vector $\mathbf{v}$.

$$\mathbf{v} = \begin{bmatrix} \mathbf{e_1} & \mathbf{e_2} & \mathbf{e_3} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

In the following sections, we regard vectors as column vectors by default.

1.2 Inner Product (Dot Product)

Given two vectors $\mathbf{a}, \mathbf{b}$ and their respective coordinate. The inner product of $\mathbf{a}, \mathbf{b}$ defines as

$$\begin{aligned}
\mathbf{a} \cdot \mathbf{b} &= \mathbf{a^T b} \\
&= \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} \mathbf{e_1} \\ \mathbf{e_2} \\ \mathbf{e_3} \end{bmatrix} \begin{bmatrix} \mathbf{e_1} & \mathbf{e_2} & \mathbf{e_3} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \\
&= \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} \mathbf{e_1} \cdot \mathbf{e_1} & \mathbf{e_1} \cdot \mathbf{e_2} & \mathbf{e_1} \cdot \mathbf{e_3} \\ \mathbf{e_2} \cdot \mathbf{e_1} & \mathbf{e_2} \cdot \mathbf{e_2} & \mathbf{e_2} \cdot \mathbf{e_3} \\ \mathbf{e_3} \cdot \mathbf{e_1} & \mathbf{e_3} \cdot \mathbf{e_2} & \mathbf{e_3} \cdot \mathbf{e_3} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \\
&= \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \mathbf{I} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \\
&= \sum a_i b_i
\end{aligned}$$

In addition, according to $c^2 = a^2 + b^2 + 2abcos\theta$, it can be proven that

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos \langle \mathbf{a}, \mathbf{b} \rangle$$

In geometry, the inner product can describe the projection relationship between vectors.

$$\Pi_b(a) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|b\|}$$

1.3 Outer Product (Cross Product)

Given two vectors $\mathbf{a}, \mathbf{b}$ and their respective coordinate. The outer product of $\mathbf{a}, \mathbf{b}$ which are expressed in the same reference frame defines as

$$\begin{aligned}
\mathbf{a} \times \mathbf{b} &= \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} \mathbf{e_1} \\ \mathbf{e_2} \\ \mathbf{e_3} \end{bmatrix} \times \begin{bmatrix} \mathbf{e_1} & \mathbf{e_2} & \mathbf{e_3} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \\
&= \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} \mathbf{e_1} \times \mathbf{e_1} & \mathbf{e_1} \times \mathbf{e_2} & \mathbf{e_1} \times \mathbf{e_3} \\ \mathbf{e_2} \times \mathbf{e_1} & \mathbf{e_2} \times \mathbf{e_2} & \mathbf{e_2} \times \mathbf{e_3} \\ \mathbf{e_3} \times \mathbf{e_1} & \mathbf{e_3} \times \mathbf{e_2} & \mathbf{e_3} \times \mathbf{e_3} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \\
&= \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix} \begin{bmatrix} 0 & \mathbf{e_3} & -\mathbf{e_2} \\ -\mathbf{e_3} & 0 & \mathbf{e_1} \\ \mathbf{e_2} & -\mathbf{e_1} & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{e_1} & \mathbf{e_2} & \mathbf{e_3} \end{bmatrix} \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{e_1} & \mathbf{e_2} & \mathbf{e_3} \end{bmatrix} a^\wedge b \\
&= \mathbf{a}^\wedge \mathbf{b}
\end{aligned}$$

where the fact that the basis vectos are orthogonal and arranged in a dextral fashion has been exploited. We refer matrix $a^\wedge$ as skew-symmetric matrix. Obviously,skew-symmetric matrix allows us to perform outer product by a linear opertaion.

In addition,

$$\mathbf{a} \times \mathbf{b} = \begin{Vmatrix} \mathbf{e_1} & \mathbf{e_2} & \mathbf{e_3} \\ a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \end{Vmatrix}$$

reminds us that the result of the outer product is a vector whose direction is perpendicular to the two vectors and length is equal to the area formed by the two vector.

$$\|\mathbf{a} \times \mathbf{b}\| = \|\mathbf{a}\| \, \|\mathbf{b}\| \sin \langle \mathbf{a}, \mathbf{b} \rangle$$

You may have the question: why the length of result vector is equal to the area fromed by the two vector? Please refer to `https://www.zhihu.com/question/22902370/answer/1803099726`

2 Coordinate Frame Transformation

2.1 Euclidean Transform

What is the definition of rigid motion? A rigid motion consists of a rotation and a translation. Commonly, we consider the motion of moving robot is rigid motion. Building a global map from the observation of a moving robot involves the transformation between two coordinate frames, the stationary inertial coordinate frame(or world coordinate frame) and the moving coordinate frame.

In the context of rigid motion, this kind of coordinate frame transformation refers as euclidean transform and it only involves rotation and translation.

2.2 Rotation Matrix

It is obvious that a vector is invarient during the corrdinate frame transformation.

$$\begin{bmatrix} \mathbf{e_1} & \mathbf{e_2} & \mathbf{e_3} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \mathbf{e'_1} & \mathbf{e'_2} & \mathbf{e'_3} \end{bmatrix} \begin{bmatrix} v'_1 \\ v'_2 \\ v'_3 \end{bmatrix}$$

left multiply $\begin{bmatrix} \mathbf{e_1^T} \\ \mathbf{e_2^T} \\ \mathbf{e_3^T} \end{bmatrix}$ on the both sides of equation.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \mathbf{e_1^T e_1'} & \mathbf{e_1^T e_2'} & \mathbf{e_1^T e_3'} \\ \mathbf{e_2^T e_1'} & \mathbf{e_2^T e_2'} & \mathbf{e_2^T e_3'} \\ \mathbf{e_3^T e_1'} & \mathbf{e_3^T e_2'} & \mathbf{e_3^T e_3'} \end{bmatrix} \begin{bmatrix} v_1' \\ v_2' \\ v_3' \end{bmatrix} = \mathbf{R v'}$$

We refer $\mathbf{R}$ as the Rotation Matrix. Because the length of basis vectors is equal to one, the result of their inner product is actually equal to the cosine of the angle between them. Therefore, we also refer $\mathbf{R}$ as the direction cosine matrix.

A rotation matrix is an orthogonal matrix with determinant of 1. Conversely, an orthogonal matrix with deferminant of 1 is a rotation matrix.Therefore, we can define a set to describe n dimensonal rotation matrices.

$$\mathrm{SO}(n) = \{\mathbf{R} \in \mathbb{R}^{n\times n} | \mathbf{R R^T} = \mathbf{I}, det(\mathbf{R}) = 1\}$$

$\mathrm{SO}(n)$ refers to the special orthogonal group. Specifically, $\mathrm{SO}(3)$ indicates the rotation matrices in 3D space.

Thanks to the orthogonality of rotation matrix, we can easily calculate the reverse rotation with respect to the rotation $\mathbf{v} = \mathbf{R v'}$ as following

$$\mathbf{v'} = \mathbf{R^{-1} v} = \mathbf{R^T v}$$

Specially, when considering rotations about one basis vector, we can derive principal rotation matrices. When rotation is about $\mathbf{e_3}$, the rotation matrix is

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

When rotation is about $\mathbf{e_2}$, the rotation matrix is

$$\begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}$$

When rotation is about $\mathbf{e_1}$, the rotation matrix is

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix}$$

2.3 Transform Matrix and Homogeneous Coordinates

Given rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$, we can describe euclidean transform as

$$\mathbf{b} = \mathbf{R a} + \mathbf{t}$$

Suppose we have two consecutive euclidean transforms $\mathbf{c} = \mathbf{R_2 b} + \mathbf{t_2}, \mathbf{b} = \mathbf{R_1 a} + \mathbf{t_1}$, then

$$\mathbf{c} = \mathbf{R_2 R_1 a} + \mathbf{R_2 t_1} + \mathbf{t_2}$$

which is not convenient for computation.Therefore, it's necessary to introduce homogeneous coordinates and transformation matrices. Rewriting the above description,

$$\begin{bmatrix} \mathbf{a'} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0^T} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} \mathbf{a} \\ 1 \end{bmatrix}$$

We refer $\mathbf{T}$ as transform matrix.

By homogeneous coordinates and transformation matrices, we can describe two consecutive euclidean transforms concisely.

$$\mathbf{c} = \mathbf{T_2 T_1 a}$$

Similiarly, we can define a set of transform matrices

$$\mathrm{SE}(n) = \{\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0^T} & 1 \end{bmatrix} \in \mathbb{R}^{(n+1)\times(n+1)} | \mathbf{R} \in \mathrm{SO(n)}, \mathbf{t} \in \mathbb{R}^n\}$$

$\mathrm{SE}(n)$ refers to Special Euclidean Group.

3 Alternative Rotation Representaion

Until now, we've introduced rotation matrix. However, using rotation matrix to represent rotation has following disadvantages:

(1) $SO(3)$ has matrices with 9 quantities, which are redundant to 3D rotation which only has 3 degrees of freedom.

(2) Rotation matrices must to be orthogonal matrices with determinant of 1. These constraints make estiamtion and optimization more difficult.

Therefore, we expect a more compact representation of rotation. It must be clear that there doesn't exsit a perfect way to represent rotation. The representations that have more than three parameters must have constraints to limit the number of degrees of freedom to three. On the other hand, the representations that exactly have three parameters have singularities. There's always a trade-off between compacity and redundancy.

3.1 Rotation Vector

It's obvious that a rotation can be described by a rotation axis and a rotation angle. So, we can use a vector whose has the same direction as the rotation axis and has length equal to the rotation angle. This kind of vector is called rotation vector(or angle-axis/axis-angle).

$$\mathbf{w} = \theta \mathbf{n}$$

where $\mathbf{n}$ is a unit-length vector which is parallel with the rotation axis.

By Rodrigues's Formula, we can evaluate transformation between rotation matrix and rotation vector.

$$\mathbf{R} = \cos\theta \mathbf{I} + (1 - \cos\theta)\mathbf{n}\mathbf{n}^{\mathbf{T}} + \sin\theta \mathbf{n}^{\wedge}$$

Take trace of the both sides of the formula,

$$\begin{aligned} tr(\mathbf{R}) &= \cos\theta tr(\mathbf{I}) + (1 - \cos\theta) tr(\mathbf{n}\mathbf{n}^{\mathbf{T}}) + \sin\theta tr(\mathbf{n}^{\wedge}) \\ &= 3\cos\theta + (1 - \cos\theta) \\ &= 1 + 2\cos\theta \end{aligned}$$

Therefore,

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

Because vectors which are parallel with rotaion axis are invarient after the rotation, we have

$$\mathbf{n} = \mathbf{R}\mathbf{n}$$

Hence, $\mathbf{n}$ is the eigenvector corresponding to the matrix $\mathbf{R}$'s eigenvalue 1.

3.2 Euler Angle

Neither rotation matrix or rotation vector is friendly for human to understand how the rotation is happening. Here comes the euler angle.

Euler angle uses three primal axis to decompose the rotation into three independent procedures. There's no a uniform definition of euler angle. People can be free to choose different decompostion orders, rotating around the fixed axis, rotating around the axis after rotation, and so on.

Here introduce the most commonly used Euler angle, the yaw-pitch-roll angle, which is equivalent to the rotation of the ZYX axis. Please refer to figure 1 to get a concrete understanding.

As above mentioned, the representations that exactly have three parameters have singularities. The singularity problem of euler angle is called Gimbal Lock. The rigorous mathematical proof of Gimbal Lock is in Appendix A.2 .

3.3 Quaternion

Before introducing quaternion, understanding the relationship between complex number and 2D rotation is beneficial for us to figure out how quaternion is asscoiated with 3D rotation.

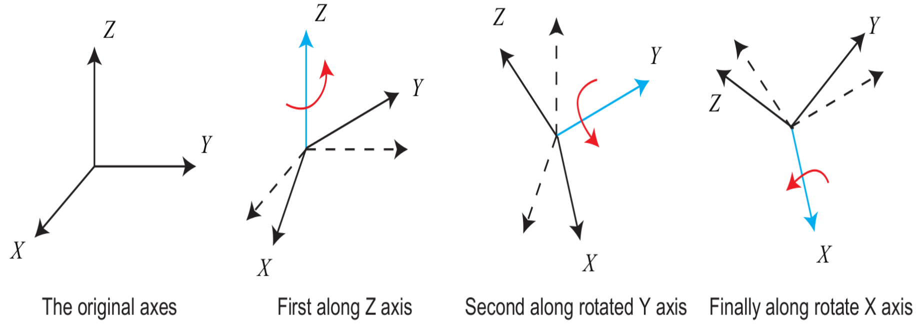


Figure 1: yaw-pitch-roll

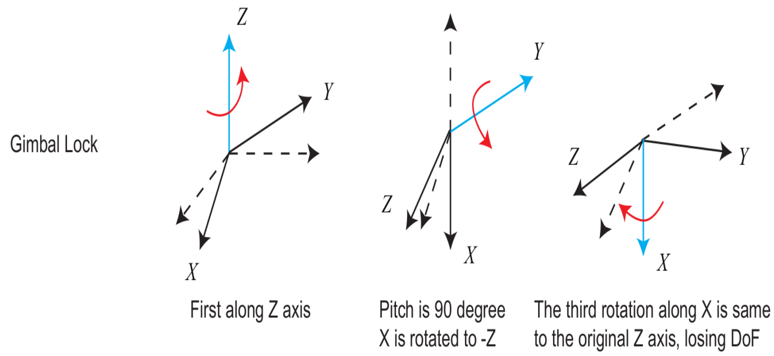


Figure 2: Gimbal Lock

3.3.1 Complex Number and 2D Rotation

As we all known, a complex number consists of a real number and a imaginary number.

$$z_i = a + bi$$

The multiplication between complex numbers follows commutative law and associative law.

$$\begin{aligned} z_1 z_2 &= (a_1 + b_1 i)(a_2 + b_2 i) \\ &= (a_1 a_2 - b_1 b_2) + (a_1 b_2 + a_2 b_1) i \end{aligned}$$

In addition, the multiplication between complex numbers can be interpreted as matrix multiplication.

$$z_1 z_2 = \begin{bmatrix} a_1 & -b_1 \\ b_1 & a_1 \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \end{bmatrix}$$

Let's refer $\begin{bmatrix} a & -b \\ b & a \end{bmatrix}$ as the left multiplicator of complex number. Making a simple conversion,

$$\begin{bmatrix} a & -b \\ b & a \end{bmatrix} = \sqrt{a^2 + b^2} \begin{bmatrix} \frac{a}{\sqrt{a^2+b^2}} & \frac{-b}{\sqrt{a^2+b^2}} \\ \frac{b}{\sqrt{a^2+b^2}} & \frac{a}{\sqrt{a^2+b^2}} \end{bmatrix} = \|z\| \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

which means that left multiplying a complex number is equivalent to scaling with the magnitude of the complex number and rotating anti clockwisely around original point by θ angle in complex plane. And if $\|z\| = 1$, it's a pure rotation.

3.3.2 Introduction of quaternion

The general form to express quaternions is

$$q = q_0 + q_1 i + q_2 j + q_3 k \qquad q_0, q_1, q_2, q_3 \in \mathbb{R}$$

where q_0 is the real part of quaternion and q_1, q_2, q_3 are the imaginary parts of quaternion. These three imaginary parts satify the following relationship

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

which is similiar with the cross product.

An alternative expression of quaternion is to use a scalar and a vector,

$$\mathbf{q} = [s, \mathbf{v}] \quad s \in \mathbb{R} \quad \mathbf{v} = [xi, yj, zk] \quad x, y, z \in \mathbb{R}$$

where s is the real part of the quaternion and $\mathbf{v}$ is the imaginary part of the quaternion. If $s = 0$, $\mathbf{q}$ refers as the imaginary quaternion, If $\mathbf{v} = \mathbf{0}$, $\mathbf{q}$ refers as the real quaternion.

Addition and Subtraction The addition and subtraction of the quaternion q_a, q_b is

$$q_a \pm q_b = [s_a \pm s_b, \mathbf{v_a} \pm \mathbf{v_b}]$$

Multiplication The multiplication of the quaternion q_a, q_b is

$$\begin{aligned} q_a q_b &= [s_a, \mathbf{v_a}][s_b, \mathbf{v_b}] \\ &= (s_a + x_a i + y_a j + z_a k)(s_b + x_b i + y_b j + z_b k) \\ &= s_a s_b + s_a x_b i + s_a y_b j + s_a z_b k \\ &+ x_a s_b i + x_a x_b i^2 + x_a y_b ij + x_a z_b ik \\ &+ y_a s_b j + y_a x_b ji + y_a y_b j^2 + y_a z_b jk \\ &+ z_a s_b k + z_a x_b ki + z_a y_b kj + z_a z_b k^2 \\ &= (s_a s_b - x_a s_b - y_a y_b - z_a z_b) \\ &+ (s_a x_b + x_a s_b + y_a z_b - z_a y_b) i \\ &+ (s_a y_b - x_a z_b + y_a s_b + z_a x_b) j \\ &+ (s_a z_b + x_a y_b - y_a x_b + z_a s_b) k \\ &= [s_a s_b - \mathbf{v_a} \mathbf{v_b^T}, s_a \mathbf{v_b} + s_b \mathbf{v_a} + \mathbf{v_a} \times \mathbf{v_b}] \end{aligned}$$

$[s_a s_b - \mathbf{v_a} \mathbf{v_b^T}, s_a \mathbf{v_b} + s_b \mathbf{v_a} + \mathbf{v_a} \times \mathbf{v_b}]$ also is called Graβmann Product of q_a, q_b.

Because of the existence of the cross product in the result, the multiplication of quaternions doesn't follow the commutative law.

Length The length of a quaternion is defined as

$$\|q\| = \sqrt{s^2 + x^2 + y^2 + z^2}$$

It can be verified that

$$\|q_a q_b\| = \|q_a\| \, \|q_b\|$$

Conjugate The conjugate of a quaternion is defined as

$$q^* = [s, -\mathbf{v}]$$

In addition, $qq^* = \|q\|^2$

Inverse The inverse of a quaternion is defined as

$$q^{-1} = \frac{q^*}{\|q\|^2}$$

Scalar Multiplication The scalar multiplication of a quaternion is defined as

$$k[s, \mathbf{v}] = [ks, k\mathbf{v}]$$

3.3.3 Quaternion and 3D Rotation

First of all, we extend the 3D point to an imaginary quaternion

$$\mathbf{p} = [0, x, y, z] = [0, \mathbf{v}]$$

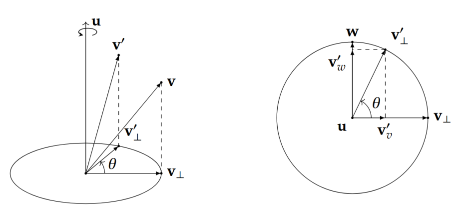
Given a vector $\mathbf{v}$ denotes the original vector and a vector $\mathbf{v}'$ denotes the vector after the rotation around $\mathbf{u}$ by θ angle, decompose $\mathbf{v}$ into $\mathbf{v}_{\|}, \mathbf{v}_{\perp}$,and decompose $\mathbf{v}'$ into $\mathbf{v}_{\|}^{'}, \mathbf{v}_{\perp}^{'}$. Then, use quaternions to express these vectors respectively.

$$\begin{aligned} v_{\|} &= [0, \mathbf{v}_{\|}] \\ v_{\perp} &= [0, \mathbf{v}_{\perp}] \\ v_{\|}^{'} &= [0, \mathbf{v}_{\|}^{'}] \\ v_{\perp}^{'} &= [0, \mathbf{v}_{\perp}^{'}] \\ u &= [0, \mathbf{u}] \end{aligned}$$

The rotation of $\mathbf{v}_{\perp}$ Obviously, a vector $\mathbf{v}_{\perp}$ which is perpendicular with $\mathbf{u}$ rotates around $\mathbf{u}$ anti clockwisely by θ angle, the result vector $\mathbf{v}_{\perp}^{'}$ is

$$\mathbf{v}_{\perp}^{'} = \cos\theta \mathbf{v}_{\perp} + \sin\theta \mathbf{u} \times \mathbf{v}_{\perp}$$

According to Graβmann Product of two quaternions, we have



$$uv_{\perp} = [0, \mathbf{u} \times \mathbf{v}_{\perp}]$$

Therefore,

$$\begin{aligned} v_{\perp}^{'} &= \cos\theta v_{\perp} + \sin\theta u v_{\perp} \\ &= (\cos\theta + \sin\theta u) v_{\perp} \end{aligned}$$

Let $q = \cos\theta + \sin\theta u = [\cos\theta, \sin\theta \mathbf{u}]$, we have

$$v_{\perp}^{'} = q v_{\perp}$$

which means that rotating $v_{\perp}$ is equivalent to left multiplying a unit quaternion whose imaginary part contains the rotation vector $\mathbf{u}$.

The rotation of $\mathbf{v}_{\|}$ As we all known, $\mathbf{v}_{\|}$ is invarient with the rotation.

The rotation of $\mathbf{v}$ Combining the above discussion, we have

$$\begin{aligned} v^{'} &= v_{\|}^{'} + v_{\perp}^{'} \\ &= v_{\|} + q v_{\perp} \end{aligned}$$

Lemma 1. *If $q = [\cos\theta, \sin\theta \mathbf{u}]$ and $\mathbf{u}$ is a unit vector, $q^2 = qq = [\cos(2\theta), \sin(2\theta)\mathbf{u}]$*

Proof.

$$\begin{aligned} [\cos\theta, \sin\theta \mathbf{u}][\cos\theta, \sin\theta \mathbf{u}] &= [\cos^2(\theta) - \sin^2(\theta)\|u\|^2, 2\cos(\theta)\sin(\theta)\mathbf{u} + \sin^2(\theta)\mathbf{u} \times \mathbf{u}] \\ &= [\cos(2\theta), sin(2\theta)\mathbf{u}] \end{aligned}$$

□

Lemma 2. *If* $\mathbf{v}$ *is parallel with* $\mathbf{u}$*,* $p = [s_1, \mathbf{v}]$ *and* $q = [s_2, \mathbf{u}]$*,* $pq = qp$

Proof.

$$\begin{aligned} pq &= [s_1, \mathbf{v}][s_2, \mathbf{u}] \\ &= [s_1 s_2 - \mathbf{v^T u}, s_1\mathbf{u} + s_2\mathbf{v} + \mathbf{v} \times \mathbf{u}] \\ &= [s_1 s_2 - \mathbf{v^T u}, s_1\mathbf{u} + s_2\mathbf{v}] \end{aligned}$$

On the other hand,

$$\begin{aligned} qp &= [s_2, \mathbf{u}][s_1, \mathbf{v}] \\ &= [s_1 s_2 - \mathbf{v^T u}, s_1\mathbf{u} + s_2\mathbf{v} + \mathbf{u} \times \mathbf{v}] \\ &= [s_1 s_2 - \mathbf{u^T v}, s_1\mathbf{u} + s_2\mathbf{v}] \end{aligned}$$

□

Lemma 3. *If* $\mathbf{v}$ *is perpendicular with* $\mathbf{u}$*,* $p = [0, \mathbf{v}]$ *and* $q = [s_2, \mathbf{u}]$*,* $qp = pq^*$

Proof.

$$\begin{aligned} qp &= [s_2, \mathbf{u}][0, \mathbf{v}] \\ &= [0 - \mathbf{v^T u}, s_2\mathbf{v} + \mathbf{u} \times \mathbf{v}] \\ &= [0, s_2\mathbf{v} + \mathbf{u} \times \mathbf{v}] \end{aligned}$$

On the other hand,

$$\begin{aligned} pq^* &= [0, \mathbf{v}][s_2, -\mathbf{u}] \\ &= [0 + \mathbf{v^T u}, s_2\mathbf{v} - \mathbf{v} \times \mathbf{u}] \\ &= [0, s_2\mathbf{v} + \mathbf{u} \times \mathbf{v}] \end{aligned}$$

□

Given Lemma 1,2,3 and let $pp = q$, we have

$$\begin{aligned} v^{'} &= v_{\|} + qv_{\perp} \\ &= pp^{-1}v_{\|} + ppv_{\perp} \\ &= pp^*v_{\|} + ppv_{\perp} \\ &= p(v_{\|} + v_{\perp})p^* \\ &= pvp* \end{aligned}$$

where $p = \left[\cos(\frac{\theta}{2}), \sin(\frac{\theta}{2})\mathbf{u}\right]$

The conversion between quaternion and axis-angle emerges.

$$\theta = 2 \arccos s$$

$$\mathbf{n} = \frac{\mathbf{v}}{\sin(\theta/2)}$$

3.3.4 Matrix Multiplication Representation of Quaternion Rotation

Given quaternion $q = [s, \mathbf{v}]$, define q^+ and $q^\oplus$ as

$$q^+ = \begin{bmatrix} s & -\mathbf{v^T} \\ \mathbf{v} & s\mathbf{I} + \mathbf{v}^\wedge \end{bmatrix} \quad q^\oplus = \begin{bmatrix} s & -\mathbf{v^T} \\ \mathbf{v} & s\mathbf{I} - \mathbf{v}^\wedge \end{bmatrix}$$

Then

$$q_a q_b = q_a^+ q_b = \begin{bmatrix} s_a & -\mathbf{v_a^T} \\ \mathbf{v_a} & s_a\mathbf{I} + \mathbf{v_a^\wedge} \end{bmatrix} \begin{bmatrix} s_b \\ \mathbf{v_b} \end{bmatrix} = \begin{bmatrix} s_a s_b - \mathbf{v_a^T v_b} \\ s_a\mathbf{v_b} + s_b\mathbf{v_a} + \mathbf{v_a} \times \mathbf{v_b} \end{bmatrix}$$

and

$$q_a q_b = q_b^{\oplus} q_a = \begin{bmatrix} s_b & -\mathbf{v_b^T} \\ \mathbf{v_b} & s_a\mathbf{I} - \mathbf{v_b^{\wedge}} \end{bmatrix} \begin{bmatrix} s_a \\ \mathbf{v_a} \end{bmatrix} = \begin{bmatrix} s_a s_b - \mathbf{v_a^T v_b} \\ s_a \mathbf{v_b} + s_b \mathbf{v_a} + \mathbf{v_a} \times \mathbf{v_b} \end{bmatrix}$$

Therefore,

$$\begin{aligned} v' &= pvp^* \\ &= p^+ (p^*)^{\oplus} v \end{aligned}$$

In addition,

$$p^+ (p^*)^{\oplus} = \begin{bmatrix} s & -\mathbf{v^T} \\ \mathbf{v} & s\mathbf{I} + \mathbf{v}^{\wedge} \end{bmatrix} \begin{bmatrix} s & \mathbf{v^T} \\ -\mathbf{v} & s\mathbf{I} + \mathbf{v}^{\wedge} \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0^T} & \mathbf{vv^T} + s^2\mathbf{I} + 2s\mathbf{v}^{\wedge} + (\mathbf{v}^{\wedge})^2 \end{bmatrix}$$

Hence, the conversion between rotaion matrix and quaternion is

$$\mathbf{R} = \mathbf{vv^T} + s^2\mathbf{I} + 2s\mathbf{v}^{\wedge} + (\mathbf{v}^{\wedge})^2$$

References

[1] Gao, Xiang, et al. "14 lectures on visual SLAM: from theory to practice." Publishing House of Electronics Industry, Beijing (2017).

[2] Barfoot, Timothy D. State estimation for robotics. Cambridge University Press, 2017.

[3] krasjet. Quaternion and 3D Rotation. Quaternion. Retrieved July 20, 2022, from https://krasjet.github.io/quaternion/

[4] Jeremiah, et al. "Understanding Quaternions." 3D Game Engine Programming, 10 May 2019, www.3dgep.com/understanding-quaternions/.

A Appendix

A.1 Rodrigues's Formula

Rodrigues's Formula has two forms:

$$\mathbf{R} = \cos\theta\mathbf{I} + (1 - \cos\theta)\mathbf{nn^T} + \sin\theta\mathbf{n}^{\wedge}$$

or

$$\mathbf{R} = \mathbf{I} + (1 - \cos\theta)(\mathbf{n}^{\wedge})^{\mathbf{2}} + \sin\theta\mathbf{n}^{\wedge}$$

Let's proof the second form.

Proof. Suppose we have two vector $\mathbf{v}$ and $\mathbf{v_{rot}}$. And $\mathbf{v}$ rotates around $\mathbf{k}$ by θ angle. Before starting the proof, it's necessary to understand what we are expecting. Our anticipation is to find a matrix $\mathbf{R}$ to establish such a equation.

$$\mathbf{v_{rot}} = \mathbf{Rv}$$

Let's start proofing.

First, decompose $\mathbf{v}$ into $\mathbf{v}_{\|}$ and $\mathbf{v}_{\perp}$, and decompose $\mathbf{v_{rot}}$ into $\mathbf{v_{rot\|}}$ and $\mathbf{v_{rot\perp}}$. $\mathbf{v}_{\|}$ and $\mathbf{v_{rot\|}}$ is parallel with rotation axis $\mathbf{k}$, and they are the same vector.

Secondly, decompose $\mathbf{v_{rot\perp}}$ into vector $\mathbf{a}, \mathbf{b}$.

As shown in figure 1, $\mathbf{a} = \sin(\pi - \theta)\mathbf{v_{rot\perp}}$, and therefore $\|a\| = |\sin\theta| \, \|\mathbf{v_{rot\perp}}\|$. And, it's obvious that $\mathbf{a}$ is parallel with $\mathbf{k} \times \mathbf{v}$. In addition, $\|\mathbf{k} \times \mathbf{v}\| = |\sin\alpha| \, \|\mathbf{v}\| \, \|\mathbf{k}\| = \|\mathbf{v}_{\perp}\|$, and $\|\mathbf{v}_{\perp}\| = \|\mathbf{v_{rot\perp}}\|$. Hence, $\|a\| = |\sin\theta| \, \|\mathbf{k} \times \mathbf{v}\|$. Therefore, we can conclude that

$$\mathbf{a} = \sin\theta\mathbf{k} \times \mathbf{v}$$

By the same way, we can derive

$$\mathbf{b} = -\mathbf{k} \times \mathbf{k} \times \mathbf{v} \cos\theta$$

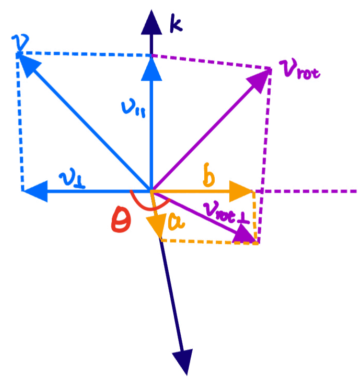


Figure 3: rotation example

Ultimately, we can obtain following equation,

$$\begin{aligned}
\mathbf{v_{rot}} &= \mathbf{v}_{\|} + \mathbf{a} + \mathbf{b} \\
&= (\mathbf{v} + \mathbf{k} \times \mathbf{k} \times \mathbf{v}) + \sin\theta \mathbf{k} \times \mathbf{v} - \mathbf{k} \times \mathbf{k} \times \mathbf{v} \cos\theta \\
&= \mathbf{v} + (\mathbf{1} - \cos\theta)\mathbf{k} \times \mathbf{k} \times \mathbf{v} + \sin\theta \mathbf{k} \times \mathbf{v} \\
&= \mathbf{v} + (\mathbf{1} - \cos\theta)(\mathbf{k}^{\wedge})^{\mathbf{2}}\mathbf{v} + \sin\theta \mathbf{k}^{\wedge}\mathbf{v} \\
&= (\mathbf{I} + (\mathbf{1} - \cos\theta)(\mathbf{k}^{\wedge})^{\mathbf{2}} + \sin\theta \mathbf{k}^{\wedge})\mathbf{v}
\end{aligned}$$

Therefore,

$$\mathbf{R} = \mathbf{I} + (\mathbf{1} - \cos\theta)(\mathbf{k}^{\wedge})^{\mathbf{2}} + \sin\theta \mathbf{k}^{\wedge}$$

□

A.2 Gimbal Lock

Here's the mathematical proof of Gimbal Lock.

Proof.

$$\begin{aligned}
\mathbf{R_z}(\alpha)\mathbf{R_y}(\frac{\pi}{\mathbf{2}})\mathbf{R_x}(\beta) &= \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{bmatrix} \\
&= \begin{bmatrix} 0 & \sin\alpha & -\cos\alpha \\ 0 & \cos\alpha & \sin\alpha \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{bmatrix} \\
&= \begin{bmatrix} 0 & \sin(\alpha+\beta) & -\cos(\alpha+\beta) \\ 0 & \cos(\alpha+\beta) & \sin(\alpha+\beta) \\ 1 & 0 & 0 \end{bmatrix} \\
&= \mathbf{R_y}(\frac{\pi}{\mathbf{2}})\mathbf{R_x}(\alpha+\beta)
\end{aligned}$$

□