3D Rigid Body Motion

1 Rigid body transformation

In robotics and computer vision, we often need to choose multiple coordinate frames to reflect the motion of cameras, different links reference to a common selected world frame, aka inertial frame. The transformation between these frames is denoted as a transformation matrix $\mathbf{T}$,

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{1}$$

its inverse is

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top\mathbf{t} \\ \mathbf{0}^\top & \mathbf{1} \end{bmatrix} \tag{2}$$

where $\mathbf{R}$ is a rotation matrix, $\mathbf{t}$ is a translation vector. The transformation descrtibe a relative motion, so we have to specify the motion is in relative to which frame. For example, we define the pose of the camera frame C expressed in the world frame W as $\mathbf{T}_{WC}$, which means freame C seen from frame W. Assume a robot moves with a onboard camera. Robot Body frame is B, the pose of the camera frame C expressed in the robot body frame B is $\mathbf{T}_{BC}$. The robot pose can be expressed as,

$$\mathbf{T}_{WB} = \mathbf{T}_{WC}\mathbf{T}_{BC}^{-1} \tag{3}$$

The pose indicating trasnformation from body frame B seen from world frame W, which indicating the motion of the robot body, is a transform of a 3D landmark point $\mathbf{X}$ from the body frame B to the world frame W.

$$\mathbf{X}_W = \mathbf{T}_{WB}\mathbf{X}_B \tag{4}$$

And the transformation from world frame W to body frame B is the inverse of the above transformation,

$$\mathbf{X}_B = \mathbf{T}_{WB}^{-1}\mathbf{X}_W \tag{5}$$

We will give a illustration of the inverse of the pose in the next section.

2 Rotation matrix

Let's start with a 2D example, 2-dimensional rotation matrix is denoted as:

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

rotation $\mathbf{R}$ rotates vectors in the plane by θ degrees counterclockwise.
Extend to 3-dimensional rotation matrix:

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

$\mathbf{R}$ is orthogonal matrix and its determinant equals to 1.

$$\mathbf{R}\mathbf{R}^\top = \mathbf{I} \quad \text{hence} \quad \mathbf{R}^{-1} = \mathbf{R}^\top \tag{8}$$

$$\det(\mathbf{R}) = 1 \tag{9}$$

Try to prove the determinant of $\mathbf{R}$ is 1?

$$\det(\mathbf{R}) = \det(\mathbf{R}_x\mathbf{R}_y\mathbf{R}_z) = \det(\mathbf{R}_x)\det(\mathbf{R}_y)\det(\mathbf{R}_z) = 1 \times 1 \times 1 = 1 \tag{10}$$

If the determinant of a matrix is 1, it means the transformation preserves the volume of the linear space — it does not *scale* it.
These properties make rotation matrix very unique, it is not just a matrix, but actually a manifold, more specifically, a Lie Manifold and also a Lie Group. So $\mathbf{R}$ is a n dimensional Special Orthogonal (SO) Group, which is,

$$\mathrm{SO}(n) = \left\{\mathbf{R} \in \mathbf{R}^{n\times n} | \mathbf{R}\mathbf{R}^\top = \mathbf{I}, \det(\mathbf{R}) = 1\right\} \tag{11}$$

The rigid body trasnform $\mathbf{T}$ contains both rotation and translation, we form a Lie Manifold which is Special Euclidean Group, SE(3).

$$\mathrm{SE}(3) = \left\{\mathbf{T} \in \mathbf{R}^{4\times 4} | \mathbf{T} \text{ is a rigid body transform}\right\} \tag{12}$$

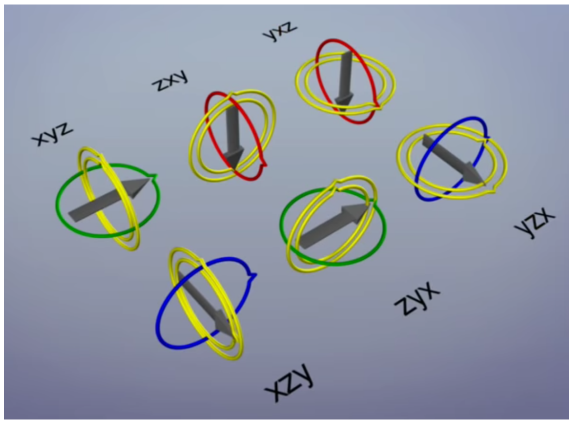
We will talk about more details about Lie Group and Lie Algebra in the next lecture.

2.1 Problem of Euler angle and Introduction to Quternion

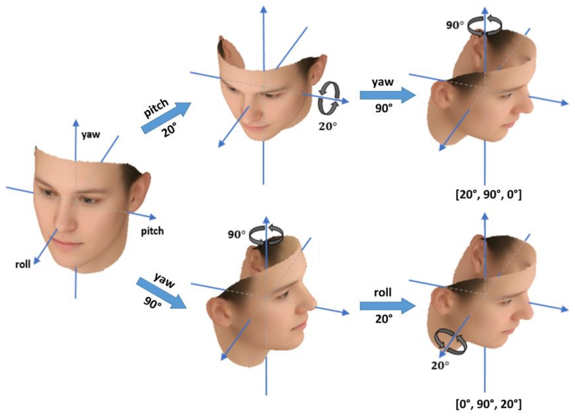
So can we just use a 3D vector which contains the rotation angles around x, y, z axes, to represent the rotation? The answer is no since it will bring the problem of *Gimbal Lock*, shown in the following figure.
Although it is OK to represent the 3D rotation by 3 by 3 rotation matrix, it is not the best way to do so since we are literrally using 9D vector to represent the a 3D rotation. To solve this problem, we can use a 4D vector to represent the rotation, which is called *Quaternion*, which in

$$\mathbf{q} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \tag{13}$$

where q_x, q_y, q_z are the imaginary parts, or called *vector part*, and q_w is the real part, or called *scalar part*.



(a) Gimbal Lock 1



(b) Gimbal Lock 2

Figure 1: Illustration of the problem of using Euler angle to bring the problem of Gimbal Lock.

3 Homogeneous coordinate

if a point $\mathbf{P}$, its coordinate is transformed by $\mathbf{R}_1$ to $\mathbf{P}_1$, then by $\mathbf{R}_2$ to $\mathbf{P}_2$, without translation, we have,

$$\mathbf{P}_2 = \mathbf{R}_2\mathbf{P}_1 = \mathbf{R}_2\mathbf{R}_1\mathbf{P} \tag{14}$$

if a point $\mathbf{P}$, its coordinate is transformed by by $\mathbf{t}_1$ to $\mathbf{P}_1$, then by $\mathbf{t}_2$ to $\mathbf{P}_2$, without rotation, we have,

$$\mathbf{P}_2 = \mathbf{t}_2 + \mathbf{P}_1 = \mathbf{t}_2 + \mathbf{t}_1 + \mathbf{P} \tag{15}$$

Suppose a point $\mathbf{P}$, its coordinate is transformed by $[\mathbf{R}_1, \mathbf{t}_1]$ to $\mathbf{P}_1$, Then transformed by $[\mathbf{R}_2, \mathbf{t}_2]$ to $\mathbf{P}_2$.

$$\mathbf{P}_1 = \mathbf{R}_1\mathbf{P} + \mathbf{t}_1 \tag{16}$$

$$\mathbf{P}_2 = \mathbf{R}_2\mathbf{P}_1 + \mathbf{t}_2 \tag{17}$$

$$\mathbf{P}_2 = \mathbf{R}_2(\mathbf{R}_1\mathbf{P} + \mathbf{t}_1) + \mathbf{t}_2 = \mathbf{R}_2\mathbf{R}_1\mathbf{P} + \mathbf{R}_2\mathbf{t}_1 + \mathbf{t}_2 \neq \mathbf{R}_2\mathbf{R}_1\mathbf{P} + \mathbf{t}_1 + \mathbf{t}_2 \tag{18}$$

Multiple transformations cannot be described in a uniform way. How do we solve this problem?
The answer is to represent $\mathbf{P}$ in **homogeneous coordinates**.

$$\begin{bmatrix} \mathbf{P}_1 \\ \mathbf{1} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1 & \mathbf{t}_1 \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \mathbf{P} \\ \mathbf{1} \end{bmatrix} \tag{19}$$

Thus, it can represent the transformations from $\mathbf{P}$ to $\mathbf{P}_1$, then to $\mathbf{P}_2$ in a uniform way.

$$\begin{bmatrix} \mathbf{P}_2 \\ 1 \end{bmatrix} = \mathbf{T}_2\mathbf{T}_1 \begin{bmatrix} \mathbf{P} \\ 1 \end{bmatrix} \tag{20}$$

Original coordinate of 3D point **P** and **P** in homogeneous coordinates,

$$\mathbf{P} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \tilde{\mathbf{P}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{21}$$

Homogeneous coordinate is defined only up to scale, all of them represents the same 3D coordinate

$$s\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x/w \\ y/w \\ z/w \\ 1 \end{bmatrix} \tag{22}$$

where s is a scalar.

4 Illustration of the inverse of the pose

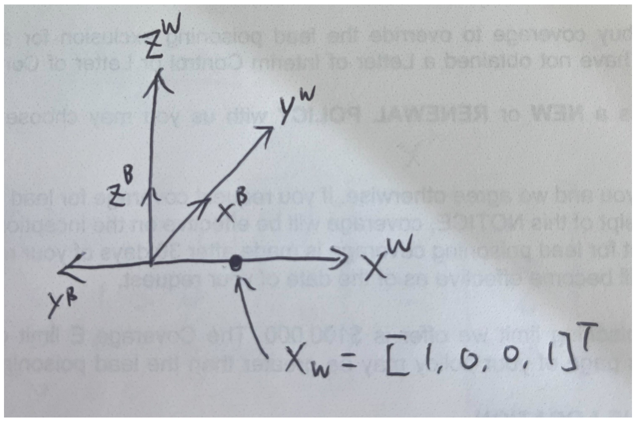


Figure 2: Illustration of the inverse of the pose.

Let's take a look at this example, the body frame equals to apply a 90 degree counterclockwise rotation around the z-axis. The pose of the body here is simply

$$\mathbf{T}_{WB} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{23}$$

Given a point $\mathbf{X}_W$ in the world frame,

$$\mathbf{X}_W = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{24}$$

The point in body frame is

$$\mathbf{X}_B = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix} = \mathbf{T}_{WB}^{-1}\mathbf{X}_W = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{25}$$

5 Conversion between Coordinate Conventions

The coordinate convention usually consists of left-hand and right-hand system. To tell if a coordinate system is left-hand or right-hand, we just thumbs up for both hands. The direction of the thumb is the direction of the z-axis, and the direction of the fingers is the direction of the x-axis to y-axis. Right hand system is more common, but some softwares like Unity and UnrealEngine do uses left hand system.

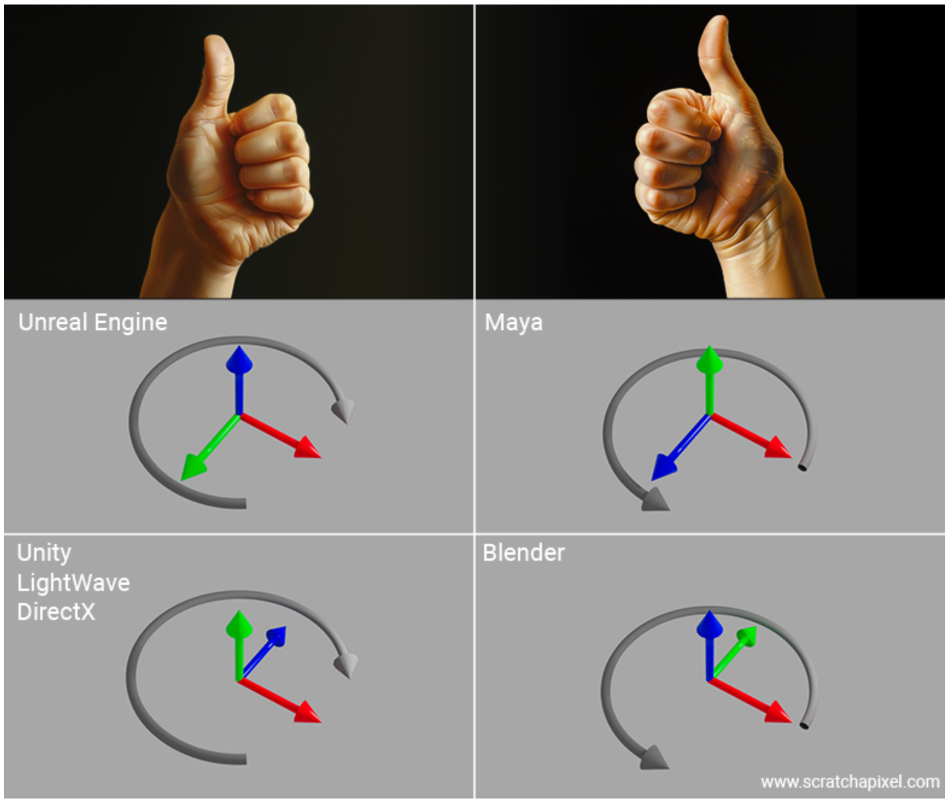


Figure 3: Illustration of the left and right hand systems.

The confusing part usually not just from the left and right hand system, but also from the difination of X, Y, Z axis. For example, in ROS, X axis is forward, Y is left and Z is up. Different softwares also uses different coordinate conventions. OpenCV, COLMAP, Open3D usually follow the same convention, ROS, OpenGL all use different conventions. The following figures the illustration of the different conventions.
The problem then is how to convert the coordinate convention between different softwares. The answer is to use the a rotation matrix. This is also called *change of basis* in linear algebra.
We first start with converting a reference 3D landmark point $\mathbf{X}$ (in homogeneous coordinates), from the OpenCV convention to ROS convention. The same point will have different coordinates in different conven-

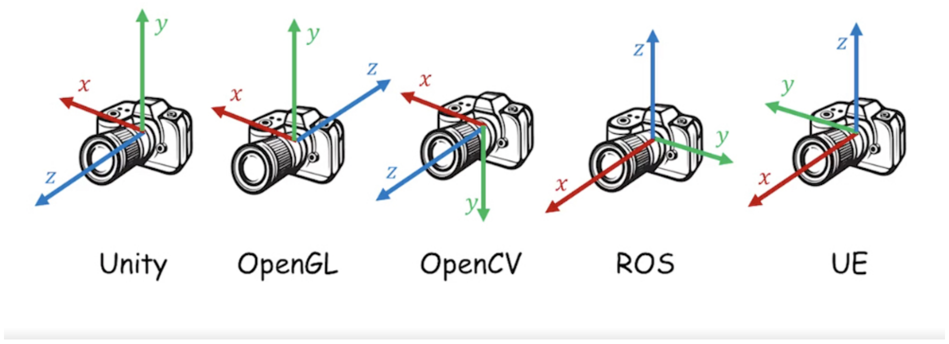


Figure 4: Illustration of the different coordinate conventions.

tions, obviously.

$$\mathbf{X}_{\text{ROS}} = \mathbf{T}_{\text{OpenCV\_to\_ROS}}\mathbf{X}_{\text{OpenCV}} \tag{26}$$

$$= \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{X}_{\text{OpenCV}} \tag{27}$$

So what are we doing here? It's actually quite intuitive, the resultant X-axis in ROS equals to the Z-axis in OpenCV, the Y-axis in ROS is the flip of the X-axis in OpenCV, and the Z-axis in ROS is the flip of the Y-axis in OpenCV. The camera pose in world frame is,
The tricky part comes from converting a trasnformation, aka a pose, from one convention to another. We will have to do a little dirivation here. In Visual odomtry or geometric vision, assume we compute the relative pose $\mathbf{T}_{C_0C}$ between the current frame C and the first frame C_0, and also reconstruct a 3D landmark point $\mathbf{X}$ in the first frame C_0.

$$\mathbf{T}_{WC} = \mathbf{T}_{\text{WC0}}\mathbf{T}_{C_0C}\mathbf{T}_{CC} \tag{28}$$

$$= \mathbf{T}_{\text{OpenCV\_to\_ROS}}\mathbf{T}_{C_0C}\mathbf{I} \tag{29}$$

And in general, the conversion between two conventions is,

$$\mathbf{T}^{\text{tgt}}_{WC} = \mathbf{T}_{\text{worldconversion}}\mathbf{T}^{\text{src}}_{WC}\mathbf{T}^{-1}_{\text{cameraconversion}} \tag{30}$$

This equation can easily be proved, assume we have a pose $\mathbf{T}_{AB}$, indicating the pose of frame B in frame A. We want to get the pose of frame D in frame C. And we know the conversion A $\rightarrow$ C, meaning transform a point from A to C, denoted as $\mathbf{T}_{CA}$, B $\rightarrow$ D, meaning transform of a point from B to D, denoted as $\mathbf{T}_{DB}$.

Here B and D are camera frames, and A and C are world frames.

$$\mathbf{T}_{CD} = \mathbf{T}_{\mathrm{CA}} \mathbf{T}_{AB} \mathbf{T}_{\mathrm{BD}} \tag{31}$$

$$= \mathbf{T}_{\mathrm{CA}} \mathbf{T}_{\mathrm{AB}} \mathbf{T}_{\mathrm{DB}}^{-1} \tag{32}$$