Self-explanation

zhipengtang

July 2019

1 Coordinate Systems

1.1 NED Coordinate System

The NED coordinate system is the coordinate system where x-axis points at North, y-axis points at East, and z-axis points at the downside. We can think of the NED coordinate system as the world coordinate system. All data (roll, pitch, yaw, v_x, v_y, v_z) recorded in the metadata file are based on the NED coordinate system.

1.2 Drone Coordinate System

The drone coordinate system is the coordinate system where x-axis points at front, y-axis points at right, and z-axis points at downside. Figure 1 illustrates this coordinate system.

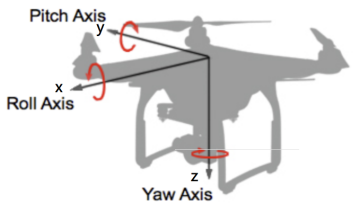

Figure 1: the Drone Coordinate System

1.3 Camera Coordinate System

The camera installed on the drone looks at the the ground with some angle. Due to the mechanism of Gimbal, the angle θ between the direction the camera looks at and the ground is fixed. The pitch of the drone doesn't interfere with this angle. The roll of the drone is cancelled by Gimbal, so the camera does

not roll with the drone. The only motion of the drone that can interfere the rotation of the camera is yaw. The camera yaws with the drone.

The camera coordiante system is the coordinate system where x-axis points at the direction where the camera looks, y-axis points to right, and $z$axis points to the downside.

2 Preliminary Knowledge

2.1 What is Rotation Matrix

2.1.1 Definition

The core of the conversion relies on Rotation Matrix, so we start from discussing this concept.

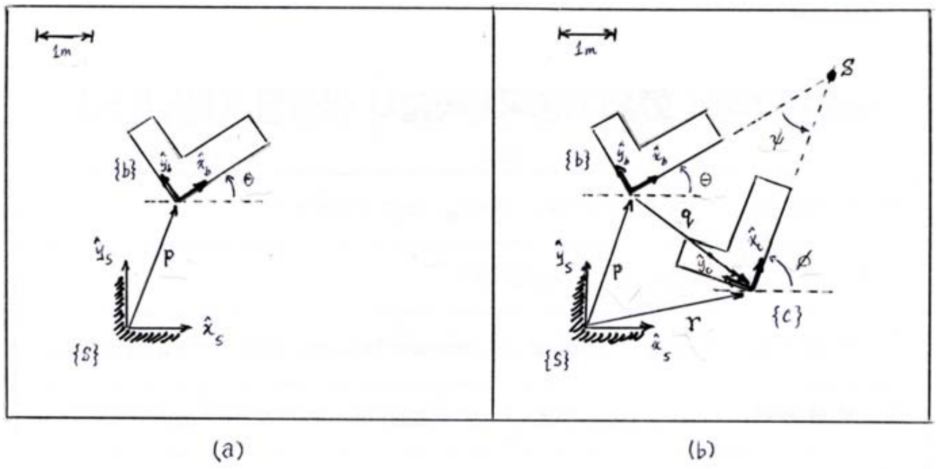

Figure 2: Rotation Matrix Example

First, we look at Fig 2 (a). $\{s\}$ is **the fixed frame** or **the world frame**. $\{b\}$ is **the body frame**, which is associated with a rigid body.

The origin and two axes of $\{b\}$ can be expressed in terms of the fixed frame as

$$\begin{aligned} p &= p_x\hat{x}_s + p_y\hat{y}_s \\ \hat{x}_b &= cos\theta\hat{x}_s + sin\theta\hat{y}_s \\ \hat{y}_b &= -sin\theta\hat{x}_s + cos\theta\hat{y}_s \end{aligned} \tag{1}$$

This can be represented more compactly as the vector and matrix form:

$$
\begin{aligned}
p &= \begin{bmatrix} p_x \\ p_y \end{bmatrix} \\
P &= \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix}
\end{aligned} \tag{2}
$$

P in the equation is **rotation matrix**. Notice that every column of the rotation matrix represents an axis of the frame after rotation expressed in terms of the linear combination of the original frame's axes. For example:

$$
\begin{aligned}
\hat{x}_b &= \begin{bmatrix} \hat{x}_s & \hat{y}_s \end{bmatrix} \begin{bmatrix} cos\theta \\ sin\theta \end{bmatrix} \\
\hat{y}_b &= \begin{bmatrix} \hat{x}_s & \hat{y}_s \end{bmatrix} \begin{bmatrix} -sin\theta \\ cos\theta \end{bmatrix}
\end{aligned} \tag{3}
$$

In other words, rotation matrix represents the orientation of the frame after rotation operation via two column vectors. When talking about a specific rotation matrix, we need to pay attention to the frame it is based on. For example, the rotation matrix P is indeed based on $\{s\}$. The two axes of $\{b\}$ is stored in the two columns of P. These two column vectors are expressed in terms of the axes of $\{s\}$.

In order to make things clear, for a rotation matrix R, we introduce the notation R_{ab}.

Definition 1 *Given two reference frames $\{a\}$ and $\{b\}$, the orientation of frame $\{b\}$ as seen from frame $\{a\}$ will be represented by the rotation matrix R_{ab}. That is, the three columns of R_{ab} are just vector representations of the $\hat{x}$, $\hat{y}$, and $\hat{z}$ of frame $\{b\}$ expressed in terms of coordinates for frame $\{a\}$.*

2.1.2 Properties

We now introduce three important properties of the rotation matrix.

Property 1 *Consider a free vector v in physical space. Given two reference frames $\{a\}$ and $\{b\}$ in the physical space, let v_a and v_b denote representations of v with respect to these two frames, we have*

$$R_{ba}v_a = v_b$$

Property 1 tells us how to convert a vector from one reference frame to another.

Property 2 *Consider a vector v_b in reference frame $\{b\}$ and v_b rotates with the rotation of $\{b\}$. After the axes of $\{b\}$ rotate to the orientation of frame $\{a\}$, v_b rotates to v_b', where v_b' is still expressed in terms of frame $\{b\}$*

$$R_{ba}v_b = v_b'$$

Property 2 tells us how to compute the coordinates of the rotated vector.

Property 3 $R_{ab}R_{bc} = R_{ac}$

Property 3 can be illustrated by Figure 2 (b). In this figure, we have $R_{sc} = R_{sb}R_{bc}$.

2.2 ZYX Euler Angles

Every rotation matrix can be expressed by a ZYX Euler angle, and vice versa. What is ZYX Euler angle? Assume we have the ZYX Euler angle (α, β, γ) and a reference frame $\{0\}$ with axes (x, y, z), we perform the following steps:

1. Rotate about the z-axis by an angle α, we get a new frame $\{1\}$ with new axes (x', y', z')
2. Rotate about the y'-axis by an angle β, we get another new frame $\{2\}$ with new axes (x'', y'', z'')
3. Rotate about the x''-axis by an angle γ, we get the final frame $\{3\}$ with axes (x''', y''', z''')

Notice that we choose to rotate the axis of the new (body) frame each time in ZYX Euler Angle.

The ZYX Euler angle is corresponding to the following rotation matrix:

$$R_{03} = \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix} \tag{4}$$

2.3 Roll-Pitch-Yaw Angle

Roll-Pitch-Yaw Angle is another representation for rotation matrices. It is very similar to ZYX Euler angle.

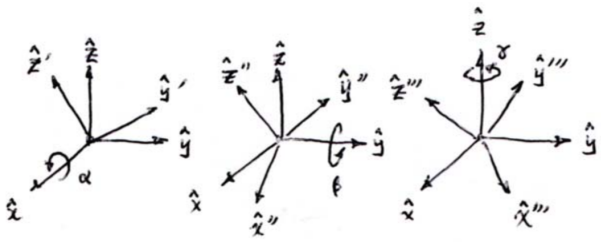

Figure 3: the Roll-pitch-yaw Angle

Assume we have the XYZ roll-pitch-yaw angle (γ, β, α) and a reference frame $\{0\}$ with axes (x, y, z), we perform the following steps:

1. Rotate about the x-axis by an angle γ, we get a new frame $\{1\}$ with new axes (x', y', z')
2. Rotate about the y-axis by an angle β, we get another new frame $\{2\}$ with new axes (x'', y'', z'')
3. Rotate about the z-axis by an angle α, we get the final frame $\{3\}$ with axes (x''', y''', z''')

Notice that we always rotate the axis of the original fixed frame in roll-pitch-yaw angle.

Interestingly, the rotation matrix corresponding to the XYZ roll-pitch-yaw angle is the same as the rotation matrix for ZYX Euler angle:

$$R_{(\gamma,\beta,\alpha)} = \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix} \tag{5}$$

3 Task

3.1 Sensor Data

The sensors on the drone record the status of the drone periodically. The format of each data point is like this:

$$(timestamp, latitude, longitude, altitude, pitch, roll, yaw, v_x, v_y, v_z)$$

Pitch, roll, and yaw are the angles which we apply to the y, x, and z axis of the NED frame, respectively. By rotating the NED frame according to pitch, roll, and yaw, we get the pose of the drone. $[v_x, v_y, v_z]^T$ is the velocity vector of the drone at a certain timestamp. This vector is based on the NED frame.

3.2 Problem Description

3.2.1 roll-pitch-yaw conversion

For any two adjacent data points from sensors, we have two drone frames and therefore two camera frames. We call these two camera frames CF_1 and CF_2, respectively.

In this problem, we want to know how we rotate the three axes of CF_1 to get CF_2. That is to say, we want the angle roll, pitch, and yaw. After applying these three angles to x, y, z-axis of CF_1, respectively, we get CF_2.

3.2.2 v_x-v_y-v_z conversion

For each data point, we have the velocity vector of the drone, i.e. $v_{NED} = [v_x, v_y, v_z]^T$, at that timestamp. This vector's coordinates are based on the NED frame.

In this problem, we try to express the velocity vector in the camera frame. That is to say, for each data point, we convert v_{NED} to v_{camera}. These two vector are essentially the same vector in the NED frame, but expressed in different coordinate systems.

4 Derivation

4.1 Axes of the Drone Frame

From the raw metadata of a data point, we have pitch, roll, and yaw. We can rotate the axes of the NED frame according to these three rotation angles to get the axes of the drone frame.

The standard convention for roll, pitch, and yaw is Roll-Pitch-Yaw Angle, as described in Section 2.3. Assume $yaw = \alpha$, $pitch = \beta$, $roll = \gamma$ at a certain data point. We have a rotation matrix $R_{NED,drone}$, which can convert the NED frame to the drone frame.

$$R_{NED,drone} = \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix} \tag{6}$$

Notice that the subscript means, the three columns of $R_{NED,drone}$ are the x, y, z-axis of the drone frame, respectively. The coordinates of these three axes are based on the NED frame.

Since we have the rotation matrix, we can get the axes of the drone frame. The x, y, z-axis of the NED frame are

$$x_{NED} = [1, 0, 0]^T \qquad y_{NED} = [0, 1, 0]^T \qquad z_{NED} = [0, 0, 1]^T$$

As mentioned in Property 1, the subscript means the coordinates of these three vectors are expressed in the NED frame. We sometimes omit the subscript because, in most cases, all vectors are expressed in the NED frame.

According to Property 2, we get the three axes x', y', z' of the drone frame by matrix multiplication:

$$\begin{aligned} x'_{NED} &= R_{NED,drone} x_{NED} \\ y'_{NED} &= R_{NED,drone} y_{NED} \\ z'_{NED} &= R_{NED,drone} z_{NED} \end{aligned}$$

Again, the coordinates of the three axes x', y', z' of the drone frame are based on the NED frame.

4.2 Axes of the Camera Frame

Now that we have the three axes x', y', z' of the drone frame, our next step is to get the the three axes x'', y'', and z'' of the camera frame. Notice that the coordinate x', y', z', x'', y'', and z'' are all based on the NED frame, and we omit the subscript to specify this.

4.2.1 x''-axis

Let's focus on x'' first. Remember that Gimbal keeps the angle between the x-axis of the camera frame and the ground fixed to θ. This means no matter how the vector x' points up and down (or more rigorously, no matter how the z-component of the vector x' changes), x'' points at a certain angle fixed to the ground (i.e. only x and y-component of the vector x' matters to x'').

To get x'', we use two steps: we first project x' to a vector level to the ground, and then project this vector down with angle θ.

Assume $x' = [x'_x, x'_y, x'_z]^T$. To project x' to the ground is easy. The projected vector x'_{ground} is simply the original vector with its z-component setting to 0. That is

$$x'_{ground} = [x'_x, x'_y, 0]^T$$

Then how do we further project x'_{ground} to x''? We can achieve it with trigonometric functions. As we see from Figure 4, after projecting down with angle θ, the new vector $x''_{unnormalized} = [x'_x, x'_y, h]^T$ where

$$h = |x'_{ground}| \tan\theta$$

The last step is to normalize $x''_{unnormalized}$ so that x'''s length is 1:

$$x'' = \frac{x''_{unnormalized}}{|x''_{unnormalized}|}$$

4.2.2 y''-axis and z''-axis

We can assume $y'' = [a, b, 0]^T$. The reason why the z-component of y'' is 0 is that the Gimbal cancelled the roll of the drone so that the camera is level to the ground. Furthermore, we assume $x'' = [i, j, k]^T$. We have the following equations

$$\begin{cases} ai + bj + 0k = 0 \\ a^2 + b^2 + 0^2 = 1 \end{cases} \tag{7}$$

The first equation is from the perpendicularity of x'' and y''. The second equation is from the length of y'' should be 1.

This equation system has two solutions:

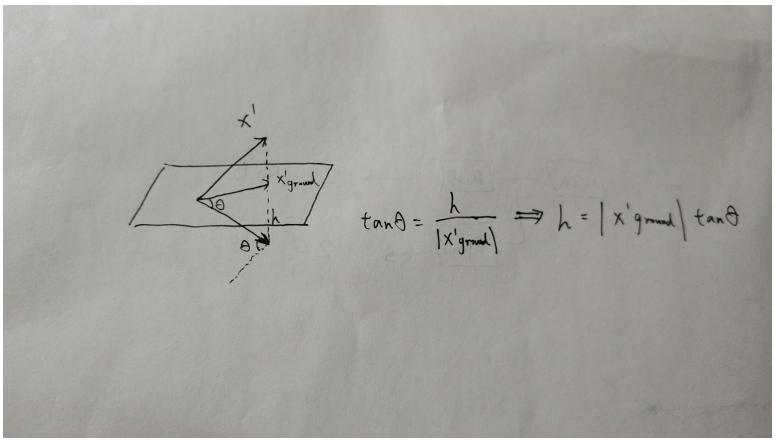

Figure 4: x-axis of the camera frame

$$\begin{cases} a = -\frac{j}{\sqrt{i^2+j^2}} \\ b = \frac{i}{\sqrt{i^2+j^2}} \end{cases} \quad or \quad \begin{cases} a = \frac{j}{\sqrt{i^2+j^2}} \\ b = -\frac{i}{\sqrt{i^2+j^2}} \end{cases} \tag{8}$$

However, only one solution is feasible because the infeasible solution has a z-axis pointing to the sky instead of the ground. To rule out the infeasible solution, we need to get the z-axis for both of the potential solutions for y''.

More specifically, for the two potential solutions of y'', we compute their corresponding z'' by

$$z'' = x'' \times y''$$

The z''-axis vector with negative z component should be ruled out since z''-axis points up to the sky. Only one solution remains.

4.3 Roll-pitch-yaw Extraction

Now we have the three axes of the camera frame, namely x'', y'', and z''. Their coordinates are all expressed in the NED frame. If we stack them together vertically, we get a rotation matrix $R_{NED,camera}$:

$$R_{NED,camera} = \begin{bmatrix} | & | & | \\ x'' & y'' & z'' \\ | & | & | \end{bmatrix} \tag{9}$$

For two adjacent data points in metadata, we can get two camera frames, i.e. the camera frame 1 (CF_1) and the camera frame 2 (CF_2), and therefore two corresponding rotation matrices $R_{NED,camera_1}$ and $R_{NED,camera_2}$. According to Property 3, we have

$$R_{camera_1,camera_2} = R^{-1}_{NED,camera_1} R_{NED,camera_2} \tag{10}$$

To get the corresponding values of roll, pitch, and yaw in $R_{camera_1,camera_2}$. We can use the following equations:

$$\begin{cases} \beta = \arcsin -R_{3,1} \\ \gamma = \arcsin \frac{R_{3,2}}{\cos \beta} \\ \alpha = \arcsin \frac{R_{2,1}}{\cos \beta} \end{cases} \tag{11}$$

where $R_{a,b}$ is the element of $R_{camera_1,camera_2}$ in the a-th row and the b-th column. (α, β, γ) is yaw, pitch, and roll, respectively. To rotate with these angles according to Section 2.3, we can transform CF_1 to CF_2.

4.4 v_x-v_y-v_z Conversion

From Section 4.3, we have $R_{NED,camera}$. It is not difficult to transform v_{NED} to v_{camera} with Property 1:

$$v_{camera} = R_{camera,NED} v_{NED} = R^{-1}_{NED,camera} v_{NED} \tag{12}$$