

Introduction

This is a guide for those who have never done state estimation before. The idea behind state estimation is how to use sensor information to give the most accurate representation of position and orientation. In this example we will start with a simple example of a 6DOF(degree of freedom) IMU which would give us accelerometer and gyroscope data. In the future I hope to expand to 9DOF which would include a magnetometer and then adding GPS.

Sensors

Accelerometer: An accelerometer is a motion sensor which reports acceleration data in specific directions. They are small cantilever beams and are typically MEMS based. Typically a 3DOF accelerometer will report acceleration(units typically in a multiple of gravity, depends on the model) along the x y and z directions of the device (Directions usually specified on the IMU board). You can also purchase 1DOF accelerometers which will report acceleration in a single direction.

Gyroscope: A gyroscope is another motion sensor which reports angular velocity data(units usually rad/s but you will need to check the datasheet) about the x y and z directions of the device(Directions usually specified on the IMU board). Integrating the angular velocity data given by the gyroscope can give you the roll pitch and yaw of the device.

IMU: An IMU(Inertial Measurement Unit) is a device which packages various sensors onto one board. A 6DOF IMU typically incorporates a 3DOF accelerometer and a 3DOF gyroscope, while a 9DOF IMU will add on a 3DOF Magnetometer(Details to come)

State Estimation

As discussed above state estimation is the process of using sensor data to give an accurate representation of position and orientation of a device. To understand the math in the portion below you should have a basic understanding of physics, matrices, and calculus.

We will first represent position on the cartesian coordinate frame. Position will be represented as r , where $r = [x \ y \ z]$ and $[x \ y \ z]$ are your positions along the 3 cartesian axes. Orientation will be represented as ϕ where $\phi = [\alpha \ \beta \ \gamma]$. α is the roll (rotation about the x axis) β is the pitch (rotation about the y axis) γ is the yaw (rotation about the z axis). With this orientation standard the forward direction is typically along the x axis.

Coordinate Axes

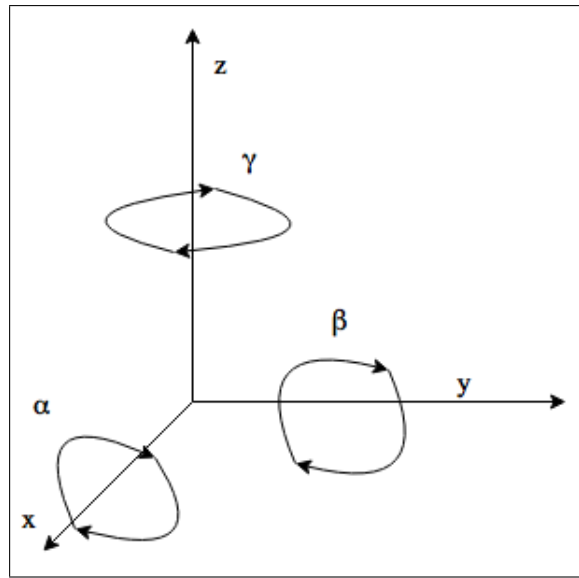


Figure 1: Example of coordinate axes and orientation labeling

State estimation is all about converting data between the global and local coordinate frames. The global frame is typically the data we want to output, it is the position and orientation of our device with respect to a global origin point, if we were talking about a rocket this could be the launch pad with which the rocket launches from. The local frame is the data with respect to a local origin point which is typically the center of our IMU.

Transformation

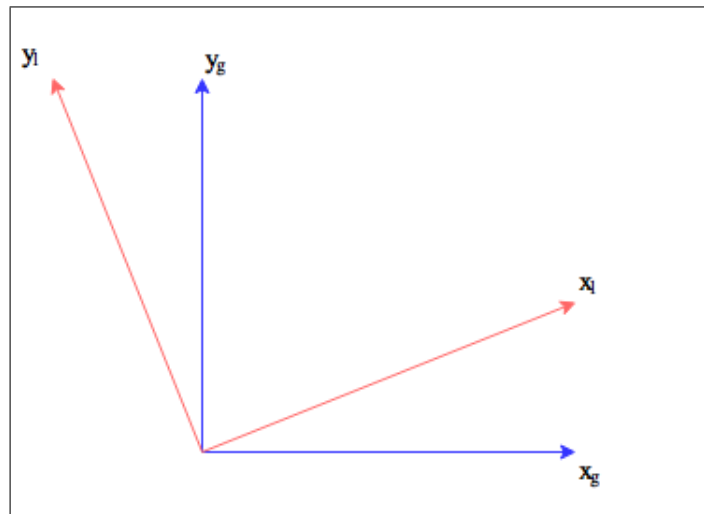


Figure 2: Transformation between local frame and global frame

When we receive data from the IMU it is always in reference to the local frame. For example if I have an IMU on my rocket the x axis always points in the direction of the nose cone, no matter which way the rocket rotates in space or accelerates the x axis will always

point along the nose cone. In this scenario the rocket's x axis points vertically (because the rocket would be pointing up on the launchpad), while the my x axis (or the global x axis) would be in some horizontal direction parallel to the ground. The question then becomes, how do I tell the rocket's position in the global x axis? If I am standing at a launch pad, the rocket launches and moves 2 feet away from me in the global x direction, how am I going to know that if the IMU's local frame is so different from my global frame? The answer is in transformations. Using the rotational data from the gyroscope and the acceleration data from the accelerometer we can create rotational transformation matrices to rotate that data from its local frame to the global frame.

Rotations

Lets start with a simple rotation on a 2D axis. We will use x and y. Lets call the x and y magnitudes in the global frame x_g and y_g , while the local frame magnitudes will be x_l and y_l . We will start with an example, lets say I have a point which is at (10,8) in the local frame, and I know the local frame has been rotated by 30 degree from the global frame. This example is shown in figure 3.

Transformation Example

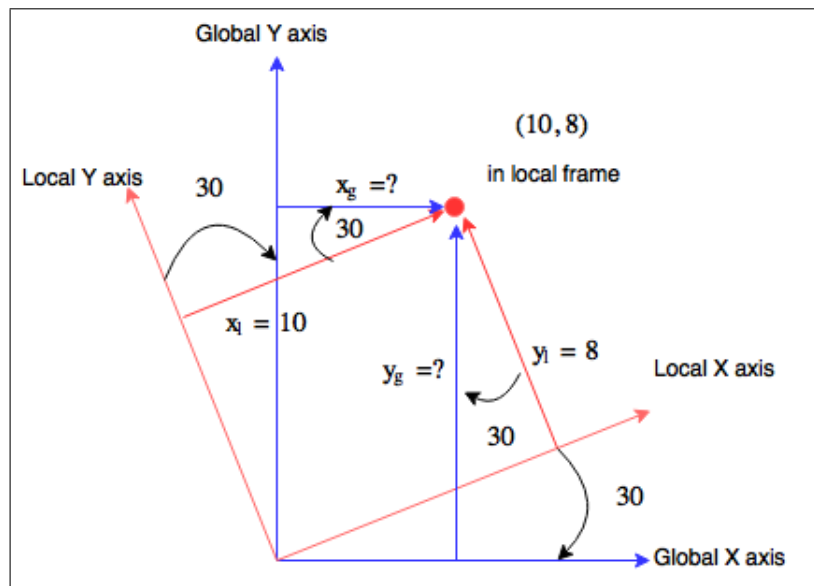


Figure 3: Example to transform a point from the local frame to the global frame

Visually you can see that I would have to do some trigonometry to obtain my position in the global frame. First we want to see what my x coordinate would be in the global frame.

Because of the rotation the x component in the local frame exists on both sides of the y axis. So to figure out our global x component we will have to consider the part on the right side of the y axis as well as the left side of the y axis. We can first find the whole component as if it were to the right of the y axis by taking the magnitude of x_l multiplied by $\cos(\theta)$ which would be $10 * \cos(30)$. Now we need to subtract out the component to

the left of the y axis. If you look carefully you can see that component is equivalent to the magnitude y_l multiplied by $\sin(\theta)$ which would be $8 * \sin(30)$.

We can use the same thought process to calculate the global y component. The global frame will be calculated in 2 parts, first we consider the contribution of the local y axis which is the magnitude of y_l multiplied by $\cos(\theta)$ or $8 * \cos(30)$. The second part is the contribution of the local x axis which is the magnitude of x_l multiplied by $\sin(\theta)$ or $10 * \sin(30)$. It might be difficult to understand from the explanation above, hopefully you can come to the same conclusion by looking at the figure. Finally you arrive at the equations below:

$$x_g = 10 * \cos(30) - 8 * \sin(30) \approx 4.66$$

$$y_g = 10 * \sin(30) + 8 * \cos(30) \approx 11.93$$

From this we infer that the location of the point in the global frame is actually (4.66,11.93)

Global Frame Coordinate

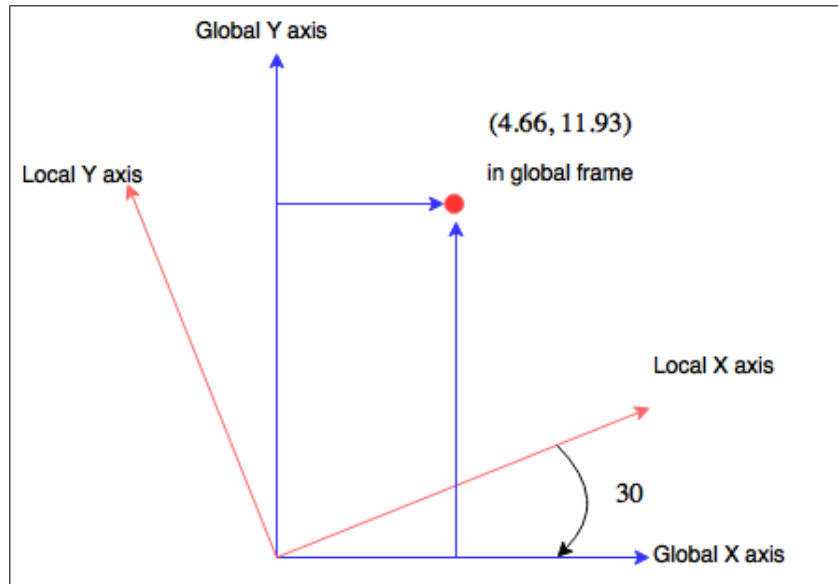


Figure 4: Final coordinate in the global frame

Now that we have completed this example lets try generalizing the equations we came up with. Our rotation angle is θ , local x and y magnitudes are x_l and y_l , and our global x and y magnitudes are x_g and y_g .

$$x_g = x_l \cos(\theta) - y_l \sin(\theta)$$

$$y_g = x_l \sin(\theta) + y_l \cos(\theta)$$

We can also put this into a matrix form: $\begin{bmatrix} x_g \\ y_g \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} * \begin{bmatrix} x_l \\ y_l \end{bmatrix}$

To generalize even further we can say: $r_g = R(\theta) * r_l$ where r_g is my global position, r_l

is my local position, and $R(\theta)$ is called a rotation matrix. The conclusion here is that any point in the local frame can be converted to a point in the global frame via a rotational matrix, and all you need to keep track of is how the local frame has rotated (θ) with respect to the global frame. In the next section we will cover 3 dimensional rotations.