Quaternion Representation of 3D Orientation and Rotation for Sensor Fusion Applications

By Cam Gordon, ECE '21

Introduction

A three-axis gyroscope is a sensor device that gathers data about how something rotates in three dimensions. They are ubiquitous in smartphones and found commonly in many systems that involve any sort of movement or rotation. Working with this sensor data can be tricky. Mathematically representing orientation and rotation in three-dimensional space is a complicated problem. Interpreting and performing computations with these representations is crucial in many systems that utilize a three-axis gyroscope. One of the most popular solutions is to use quaternions. A quaternion is a complex number that represents a single orientation or rotation in three-dimensional space. Quaternions can be difficult to understand, but make rotational computations simple to perform. They have distinct advantages over alternative representations, avoiding the mathematical edge cases that afflict other systems and offering superior computational efficiency. This technical report will demonstrate the mathematical foundation behind the quaternion and discuss the advantages of using quaternions to process sensor data.

What is a Quaternion?

Simply put, a quaternion is a four-dimensional complex number. A quaternion, q, takes the form:

$$q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$$

Where q_i are real numbers and i, j, and k are the three unit imaginary numbers. The q_0 term represents the real part of the quaternion and q_{1-3} are coefficients to the three imaginary parts. The three unit imaginary numbers arise from the statement:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$$

These four q_i values can be used to encode the information that represents a single orientation or rotation.

Mathematical Rules of Quaternions

Quaternions are governed by a few special mathematical rules that differ from those surrounding conventional complex numbers. Firstly, quaternions are not communicative. That is to say, for two quaternions, q and p:

$$qp \neq pq$$

This stems from the fact that the unit imaginary numbers themselves are not communicative. Multiplication between these numbers follow the following rules:

$$\mathbf{ij} = \mathbf{k} \qquad \mathbf{ji} = -\mathbf{k}$$
$$\mathbf{jk} = \mathbf{i} \qquad \mathbf{kj} = -\mathbf{i}$$
$$\mathbf{ki} = \mathbf{j} \qquad \mathbf{ik} = -\mathbf{j}$$

The conjugate of a quaternion q is notated q* and is defined as the following:

$$q = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$$
$$q^* = q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k}$$

The magnitude, or norm, of a quaternion is notated as |q| and is defined as:

$$|q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$$

Understanding Quaternions Through Examples

Quaternions are complex in more ways than one. The use of three imaginary numbers, rather than just the one that most people are used to, can be difficult to wrap one's head around. Understanding quaternions and their uses can be quite unintuitive, as it is impossible to visualize a quaternion, because it exists in four dimensions. For that reason, using an example in lower dimensionality can make it easier to conceptualize the quaternion.

A phasor is a two dimensional complex number that can be used to represent an angle or a rotation about a single axis. These 2D phasors are analogous to their 4D quaternion counterparts. Phasors are represented as **a + b**i, and a single phasor represents either an orientation or a rotation, just like quaternions. The difference being that phasors represent rotation about one axis, while quaternions represent rotation about 3 axes. Consider this 2D dimensional example:

Phasor Rotation Example

A phasor P represents an orientation. P has the value: P = 1 + 3i. Writing a phasor in this form gives a vector of two components that correspond to the two axes in two-dimensional space. This encodes a magnitude and angle. This phasor can be plotted as a vector in the complex plane:

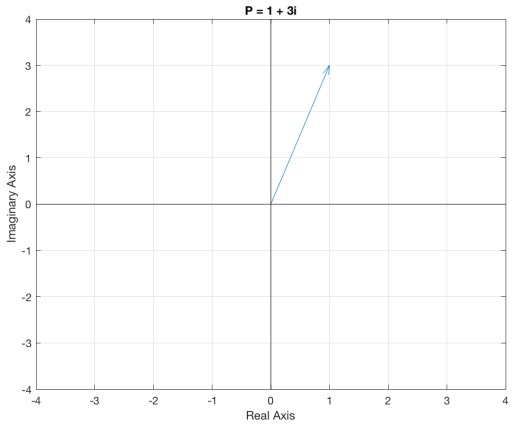

Figure 1: P = 1 + 3i plotted on the complex plane

We can rotate P by using another phasor. The phasor R represents a rotation operation. R should be of the form: R = cos(Θ) + i sin(Θ), where Θ is the angle of rotation. This will result in a unit phasor, or a phasor that has a magnitude of one. It is important that R be of unit magnitude, because this will ensure that we do not change the magnitude of P, only the angle.

Here let's use Θ = 45°. So R = cos(45°) + i sin(45°) $= \frac{\sqrt{2}}{2} + i\frac{\sqrt{2}}{2}$. R, too, can be plotted on the complex plane:

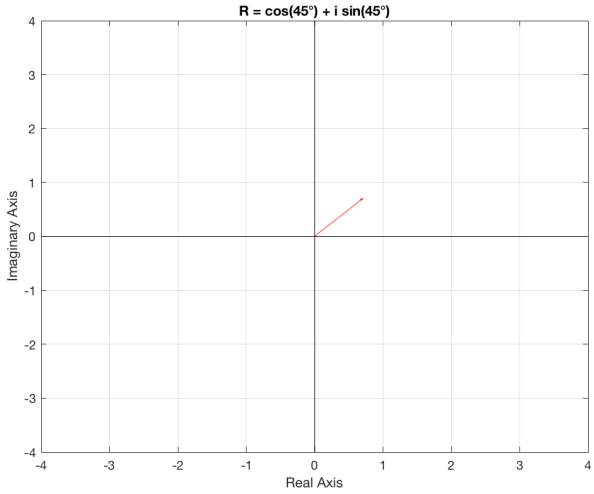

Figure 2: R = cos(45°) + i sin(45°)

To perform the rotation, we simply multiply P by R. The resulting phasor will be a 45 degree rotation of P:

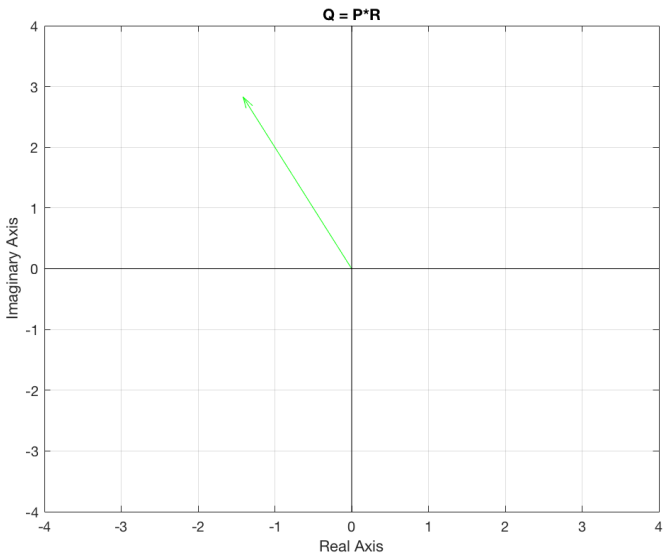

Figure 3: Q is the result of rotation R being performed on P

$$Q = PR = (1 + 3i)(\cos(45°) + i \sin(45°))$$
$$= -1.41 + 2.82i$$

This two-dimensional example is directly analogous to the rotation operation performed by a quaternion, with the helpful addition that phasor example can be visualized on a two-dimensional plot, while it would require four dimensions to plot quaternions.

Let's look at an example of the math behind a rotation in 3D space using quaternions:

Quaternion Rotation Example

An orientation in 3D space is represented by a quaternion P that has no real part. Let's give P an arbitrary value.

$$P = -2\mathbf{i} + 3\mathbf{k} - 4\mathbf{j}$$

This notation represents a vector of three elements that correspond to the three axes of three-dimensional space. Now we want to take this quaternion representing a magnitude and orientation and perform a rotation on it that will result in a new orientation but maintain the original magnitude.

To do this, we construct a quaternion, R, that represents a rotation operation. R should be of the form:

$$R = \cos(\Theta/2) + \sin(\Theta/2)\,(a\mathbf{i} + b\mathbf{j} + c\mathbf{k})$$

When we built the rotation phasor, all we needed was an angle, Θ, because there was a single axis of rotation. With a rotation quaternion, we must specify both Θ and the values of a, b, and c. This is because we need to specify what axis the rotation is occurring about. The vector v = [a, b, c] specifies the axis of rotation in three-dimensional space. Vector v should be of unit magnitude, which will result in quaternion R being of unit magnitude. This means that when the rotation operation is performed, the magnitude of P is not affected.

For example, let's choose an axis of rotation that exists in the plane of the **i** and **j** axes, bisecting these two axes. And let's rotate 60° around that axis. We would construct the following rotation quaternion:

$$[a, b, c] = \left[\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0\right] \qquad \Theta = 60°$$
$$R = \cos(30°) + \sin(30°)\left(\frac{\sqrt{2}}{2}\mathbf{i} + \frac{\sqrt{2}}{2}\mathbf{j} + 0\mathbf{k}\right)$$
$$= \frac{\sqrt{3}}{2} + \frac{\sqrt{2}}{4}\mathbf{i} + \frac{\sqrt{2}}{4}\mathbf{j} + 0\mathbf{k}$$

Now to perform the rotation operation, which will result in a rotated version of P with its original magnitude. In contrast to phasors, quaternions do not perform this operation by simply multiplying. To rotate P by R, one must left-side multiply P by R and then right-side multiply by the conjugate of R (remembering that quaternion multiplication is not communicative). That is, to produce Q, the quaternion result of this rotation, one should perform:

$$Q = RPR^*$$

This explains why the rotation quaternion uses $\Theta/2$ rather than simply Θ. One rotation operation corresponds to two multiplications, so as a result the rotation will be twice the angle contained within the sine and cosine.

Now let's perform the rotation of P by R:

$Q = RPR^*$

$= (\frac{\sqrt{3}}{2} + \frac{\sqrt{2}}{4}\mathbf{i} + \frac{\sqrt{2}}{4}\mathbf{j} + 0\mathbf{k})(-2\mathbf{i} + 3\mathbf{k} - 4\mathbf{j})(\frac{\sqrt{3}}{2} - \frac{\sqrt{2}}{4}\mathbf{i} - \frac{\sqrt{2}}{4}\mathbf{j} - 0\mathbf{k})$

$= 1.699\mathbf{i} - 0.699\mathbf{j} + 5.06\mathbf{k}$

Note that the resulting quaternion, Q, is three dimensional, corresponding to a point in 3D space, just as our initial quaternion P was. The magnitude of Q is the same as the magnitude of P, just as we intended.

Advantages of Quaternions in Sensor Fusion

Sensor fusion is the practice of combining data from multiple sensors to form more complete and accurate sets of data. An inertial measurement unit (IMU) will typically contain both a gyroscope and an accelerometer. These two sensors are capable of providing data on their own, but their data can be used in conjunction, and the result is more accurate. There exists a host of algorithms to perform sensor fusion on data from IMUs.

The most common sensor fusion algorithms, such as the Kalman filter, can be implemented using any representation of orientation. The most common alternative to quaternions for representing orientation mathematically is the use of Euler angles and rotational matrices. Using quaternions for sensor fusion has a few distinct advantages over this alternative representation.

The main advantage of using the Euler angle representation is that it is the most intuitive and easily visualized system for representing orientation and rotation. This representation system fixes three perpendicular axes in space, and describes three different angles of rotation about these three different axes. These three Euler angles, denoted as α, β, and γ, are sometimes called roll, pitch, and yaw. To rotate a three-dimensional vector based on these points, we multiply the vector by the following matrix, computed from the Euler angles:

$$\begin{bmatrix} cos\beta cos\gamma & -cos\beta sin\gamma & sin\beta \\ cos\alpha sin\gamma + cos\gamma sin\alpha sin\beta & cos\alpha cos\gamma - sin\gamma sin\alpha sin\beta & -cos\beta sin\alpha \\ sin\alpha sin\gamma - cos\gamma cos\alpha sin\beta & sin\alpha cos\gamma + sin\gamma cos\alpha sin\beta & cos\beta cos\alpha \end{bmatrix}$$

This matrix presents the possibility for a mathematical edge case known as gimbal lock. For example, if

$\beta = \pi/2$, then the matrix can be rewritten as:

$$\begin{bmatrix} 0 & 0 & 1 \\ sin(\alpha + \gamma) & cos(\alpha + \gamma) & 0 \\ -cos(\alpha + \gamma) & sin(\alpha + \gamma) & 0 \end{bmatrix}$$

Note that the angles α and γ now control all the same quantities, meaning that changing either of these angles will perform rotations along the same axis. This results in the loss of a degree of freedom from the system, and we will have an entire axis that we cannot perform a rotation around. This phenomenon is known as gimbal lock and it can be quite dangerous if it occurs in something like an aircraft. The Apollo 11 spacecraft famously had a gimbal lock problem and required specific operation to avoid this situation.

Special algorithms exist to smooth over this gimbal lock problem in Euler angle based sensor fusion, but using quaternions avoids this problem entirely. Additionally, the trigonometric nature of Euler angle computation makes sensor fusion computations nonlinear, while the use of quaternions in these algorithms becomes linear.

The other alternative to a quaternion-based representation of rotation is to use rotational matrices. With a rotational matrix, you construct a 3 by 3 matrix that performs a rotation upon a three-dimensional vector. The main disadvantage of this method compared to quaternions is the storage required for rotational matrices. The rotational matrix requires nine values to represent a single rotation, while quaternions require only four. This is a big reduction in storage space for no loss in information. This difference is especially important in small IMU systems that are short for memory. Sensor fusion is often performed on a microprocessor which can mean that storage space is expensive, and the savings that quaternions provide can be essential. Additionally, quaternions are arguably more intuitive and elegant than rotational matrices. A quaternion of rotation defines an axis and angle of rotation, while a rotational matrix has no such natural intuition for its rotation.

Conclusion

Understanding quaternions and their rules and methods of operation is an important skill for anybody using gyroscope or IMU data. Although quaternions can be unintuitive and confusing at first glance, they have a straightforward set of rules that govern their operation. Sensor fusion is an important step for improving the robustness of sensor data. Sensor fusion can be performed using any type of three dimensional representation of orientation and rotation, but quaternions have distinct advantages. Quaternion sensor fusion computation offers an elegant alternative to the matrix multiplication required by Euler angles and rotational matrices. Additionally, quaternions avoid the gimbal lock problem encountered by Euler angles and offer a superior memory footprint to rotational matrices. Sensor fusion algorithms are made nicely linear by using quaternions, while Euler angles systems become nonlinear due to their reliance on trigonometric functions. For all these reasons, quaternions are the representation of choice for modern IMU systems and sensor fusion algorithms.

References

Ben-Ari, Moti. *A Tutorial on Euler Angles and Quaternions*. Weizmann Institute of Science, 2014, https://www.weizmann.ac.il/sci-tea/benari/sites/sci-tea.benari/files/uploads/softwareAndLearningMaterials/quaternion-tutorial-2-0-1.pdf.

Deibel, James. *Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors*. Stanford University, 20 Oct. 2006, https://www.astro.rug.nl/software/kapteyn/_downloads/fa29752e4cd69adcfa2fc03b1c020f4e/attitude.pdf.

Funda, J., et al. "On Homogeneous Transforms, Quaternions, and Computational Efficiency." *IEEE Transactions on Robotics and Automation*, vol. 6, no. 3, 1990, pp. 382–88, doi:10.1109/70.56658.

Hoag, David. *Apollo Guidance and Navigation of Apollo IMU Gimbal Lock*. MIT Instrumentation Laboratory, Apr. 1963, https://www.hq.nasa.gov/alsj/e-1344.htm.

Kim, A., and M. F. Golnaraghi. "A Quaternion-Based Orientation Estimation Algorithm Using an Inertial Measurement Unit." *PLANS 2004. Position Location and Navigation Symposium (IEEE Cat. No.04CH37556)*, IEEE, 2004, pp. 268–72, doi:10.1109/PLANS.2004.1309003.

Sanderson, Grant, and Ben Eater. "Visualizing Quaternions." *Visualizing Quaternions: An Explorable Video Series*, https://eater.net/quaternions.

Valenti, Roberto G., et al. "Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs." *Sensors (Basel, Switzerland)*, vol. 15, no. 8, Aug. 2015, pp. 19302–30, doi:10.3390/s150819302.

Wyss-Gallifent, Justin. *Chapter 4: Gimbal Lock*. University of Maryland, http://www.math.umd.edu/~immortal/MATH431/lecturenotes/ch_gimballock.pdf.