DEVOLOPMENT OF AN ALGORITHM TO DETERMINATE THE ORIENTATION USING AN IMU

Author: Andoni Bisbal Castao
Facultat de Física, Universitat de Barcelona, Diagonal 645, 08028 Barcelona, Spain.

Advisor: José María Gómez Cama

Abstract: In this paper we will study an algorithm designed by Madgwick which is commonly used to determine the orientation of a quadcopter. The algorithm uses a group of accelerometers, gyroscopes and magnetometers integrated in what is called an IMU as input. Some differences have been found between the results obtained by the original paper and the implementation done by the author. Therefore, a thorough study has been made, finding a miscalculation in the equations. The results show a relative average error in the orientation of 1,44 ppm.

I. INTRODUCTION

Lately there is an increase interest in the determination of the orientation of a rigid body using sensors. The major reason is the stabilization of drones and other fliying devices. For this purpose, different filters have been used, like Kalman filter [3] or Madgwick [1].

These filters make an estimation of the orientation from the data coming from an Inertial Measurement Unit (IMU). The IMU is a device that contains three 3D sensors, one 3D sensor for magnetic field, another for accelerations and finally gyroscopes. With these sensor the orientation of a rigid body relative to the earth frame can be calculated.

The results of this algorithm are usually quaternions [4]. The main reason is that they are easily calculated with matrix algebra, and are more stable than the Euler angles (roll, pitch, yaw). This angles present some ambiguities commonly known as gimbal lock. Also, as the order of the Euler angles cannot be changed, it implies that a chain of steps has to be followed. As a result, errors are accumulated, introducing undesired effects on the yaw calculation.

The Kalman filter presents some disadvantages[2], specially when it has to be implemented in resource limited devices like the ones used in small drones. Hence, we will focus on the one presented by Madgwick. After its implementation, we have found differences between the results presented in the original paper and our. This has motivated a revision of the equations to find the origin of this differences.

One of the most important problems we found when we study the orientation of a rigid body, is the sensors response time, hence it is slow and it makes difficult the capture of information. The new algorithm [2] tries to minimize the response time using a hybrid method. For this reason is important to study the origin of the difference between the results. Therefore, in this paper we study the algorithm step by step and set aside the complete method, so it is easier to encounter the error. To find these error we study the part of the method which we calculate the orientation by a gravity and magnetic sensors, and we will use a simulation to study it. In such a way that, we do not have the problem of the response time, and the study step by step of the method will be correct.

Such we have said the new method used quaternions for solution the problem, so we will need a little introduction those objects, then we introduce them[4].

The quaternion is a four-dimensional complex number that we use for determine the orientation of a rigid body in three-dimensional space. The first component of it represents the rotated angle and the other three components represent the unitary vector which we apply the rotation. The quaternion ${}^A_B\hat{q}$ represents the orientation of the B frame relative to the A frame, with angle θ and unitary vector ${}^a\hat{r}$. The representation is shown in figure 1, followed by the equation 1 which defines the ${}^A_B\hat{q}$.

$$
{}^A_B\hat{q} = \{q1, q2, q3, q4\} = \\
\{cos(\frac{\theta}{2}), -r_x sin(\frac{\theta}{2}), -r_y sin(\frac{\theta}{2}), -r_z sin(\frac{\theta}{2})\}
$$

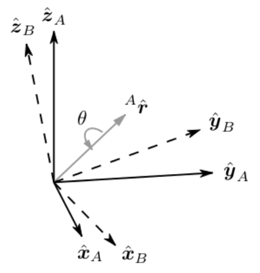

FIG. 1: The orientation of frame B relative to frame A with the rotation angle θ around the unitary vector ${}^A\hat{r}$.[1]

We can see the proprieties of these objects in the next book [4]. In this paper we will introduce some properties as we are using them.

II. ALGORITHM

To determine the orientation of our quadcopter, for example, respect to the earth, we will use three types of sensors: the gravity, the magnetic and the gyroscope sensors. The last mentioned, measures the angular velocity. Each sensors provide us information of the tri-axis so we can determine the direction of the field.

In our case, after implementation the algorithm of the following article[1], we have seen that it has an error. To search the error, we have focused on the part that studies the orientation, from the gravity and the magnetic fields sensors. To solve it, we will use a quaternion property that allows us to move a vector from a reference frame to another, using the equation 1.

$$ {}^S_E\hat{q}^* \otimes {}^E\hat{d} \otimes {}^S_E\hat{q} = {}^S\hat{s} \tag{1}$$

Where $\otimes$ is the quaternion product, we can realize it using the following equation 2.

$$ q \otimes p = [q_1, q_2, q_3, q_4] \otimes [p_1, p_2, p_3, p_4] = \begin{bmatrix} q_1p_1 - q_2p_2 - q_3p_3 - q_4p_4 \\ q_1p_2 + q_2p_1 + q_3p_4 - q_4p_3 \\ q_1p_3 - q_2p_4 + q_3p_1 + q_4p_2 \\ q_1p_4 + q_2p_3 - q_3P_2 + q_4p_1 \end{bmatrix}^T \tag{2}$$

Using the above property, we can create a function as the difference between equation 1 and the vector detected by the sensor, as shown in the equation 3.

$$ f({}^S_E\hat{q}, {}^S\hat{s}, {}^E\hat{d}) = {}^S_E\hat{q}^* \otimes {}^E\hat{d} \otimes {}^S_E\hat{q} - {}^S\hat{s} \tag{3}$$

The function can be considered an error. To minimize the error, we shall find the quaternion that minimizes it.

The desired quaternion will be encountered using a method called the descent gradient algorithm, which implements an iterative equation 4, where μ is the step-size and $\nabla f = J^T f$. The next equations 5 and 6 show the definition of the f function and the Jacobian, where $J_{ij} = \frac{\partial f_i}{\partial q_j}$.

$$ {}^S_E\hat{q}_{i+1} = {}^S_E\hat{q}_i - \mu \frac{\nabla f}{\|\nabla f\|} \tag{4}$$

$$ f({}^S_E\hat{q}, {}^S\hat{s}, {}^E\hat{d}) = \begin{pmatrix} d_x(q_1^2 + q_2^2 - q_3^2 - q_4^2) + 2d_y(q_1q_4 + q_2q_3) + 2d_z(q_2q_4 - q_1q_3) - s_x \\ 2d_x(q_2q_3 - q_4q_1) + d_y(q_1^2 - q_2^2 + q_3^2 - q_4^2) + 2d_z(q_1q_2 + q_3q_4) - s_y \\ 2d_x(q_1q_3 + q_2q_4) + 2d_y(q_4q_3 - q_1q_2) + d_z(q_1^2 - q_2^2 - q_3^2 + q_4^2) - s_z \end{pmatrix} \tag{5}$$

$$ J({}^S_E\hat{q}, {}^E\hat{d}) = \begin{pmatrix} 2d_xq_1 + 2d_yq4 - 2d_zq3 & 2d_xq2 + 2d_yq3 + 2d_zq4 & -2d_xq3 + 2d_yq2 - 2d_zq1 & -2d_xq4 + 2d_yq1 + 2d_zq2 \\ -2d_xq4 + 2d_yq1 + 2d_zq2 & 2d_xq3 - 2d_yq2 + 2d_zq1 & 2d_xq2 + 2d_yq3 + 2d_zq4 & -2d_xq_1 - 2d_yq4 + 2d_zq3 \\ 2d_xq3 - 2d_yq2 + 2d_zq1 & +2d_xq4 - 2d_yq1 - 2d_zq2 & 2d_xq_1 + 2d_yq4 - 2d_zq3 & 2d_xq2 + 2d_yq3 + 2d_zq4 \end{pmatrix} \tag{6}$$

If we compare the equations 5 and 6 with equations 7 and 8 that are the function and Jacobian of article [1],

$$ f({}^S_E\hat{q}, {}^S\hat{s}, {}^E\hat{d}) = \begin{pmatrix} 2d_x(\frac{1}{2} - q_3^2 - q_4^2) + 2d_y(q_1q_4 + q_2q_3) + 2d_z(q_2q_4 - q_1q_3) - s_x \\ 2d_x(q_2q_3 - q_4q_1) + 2d_y(\frac{1}{2} - q_2^2 - q_4^2) + 2d_z(q_1q_2 + q_3q_4) - s_y \\ 2d_x(q_1q_3 + q_2q_4) + 2d_y(q_4q_3 - q_1q_2) + 2d_z(\frac{1}{2} - q_2^2 - q_3^2) - s_z \end{pmatrix} \tag{7}$$

$$ J({}^S_E\hat{q}, {}^E\hat{d}) = \begin{pmatrix} 2d_yq4 - 2d_zq3 & 2d_yq3 + 2d_zq4 & -2d_xq3 + 2d_yq2 - 2d_zq1 & -2d_xq4 + 2d_yq1 + 2d_zq2 \\ -2d_xq4 + 2d_zq2 & 2d_xq3 - 2d_yq2 + 2d_zq1 & 2d_xq2 + 2d_zq4 & -2d_xq_1 - 2d_yq4 + 2d_zq3 \\ 2d_xq3 - 2d_yq2 & +2d_xq4 - 2d_yq1 - 2d_zq2 & 2d_xq_1 + 2d_yq4 - 2d_zq3 & 2d_xq2 + 2d_yq3 \end{pmatrix} \tag{8}$$

we can observe the differences between them. The first difference is into the function, where [1] uses the successive property of quaternions, $q_1^2 + q_2^2 + q_3^2 + q_4^2 = 1$. With this property, in the article changes the quadratic components of the function to reduce the variables. The change is correct, but it has not been done homogeneously, therefore the function still depends on all variables. Hence when we derivates the function and obtain the jacobian, it is not correct. The reason of the error is because the article does not contemplate the implicit derives respect to the reduced variables.

In the following part, we will focus on the simulation of the previous content or, in other words, how the problem will be developed. In our case due to the properties and features of the platform, which allow us to run the program into the quadcopter, the C programming language will be preferred.

In the first steps of the simulation the earth frame is configured in a way that the gravitation field is allocated in the Z-axis, the magnetic field in the X-axis and the initial quaternion with the value of (1,0,0,0).

Throughout the development of it, we will add a rotational axis and the desired angle of rotation of the sensor. From the initial point, we rotate this sensor in a constant angular velocity, until we reach the previous defined angle. For every time step, we determine the quaternion of the sensor. Using it we calculate the gravity and magnetic vector that will be measured by the sensor. This is done using the equation: ${}^S_E\hat{q}^* \otimes {}^E\hat{d} \otimes {}^S_E\hat{q} = {}^S\hat{s}$. Afterwards, we calculate the position of the sensor with the equation 4.

III. RESULTS

To study the algorithm we take different results and analyze them. In the first part we show the quaternion that represents the orientation of the sensor frame relative to the earth frame with different directions of the axis that has been rotated. All rotations have been of $\frac{\pi}{2}$ rad with a velocity of $\frac{\pi}{180}$ rad/s.

In table I we see clearly how the calculate quaternion represent correctly the axis of rotation and the angle, because the error is very small.

	q_1	q_2	q_3	q_4	v^r
error	$\pm 10^{-6}$	$\pm 10^{-6}$	0	0	$(1,0,0)$
error	$\pm 10^{-6}$	$\pm 10^{-6}$	0	$\pm 10^{-6}$	$(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0)$
error	$\pm 5 \cdot 10^{-6}$	$\pm 6 \cdot 10^{-6}$	0	$\pm 10^{-6}$	$(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$
error	$\pm 3 \cdot 10^{-6}$	$\pm 4 \cdot 10^{-6}$	0	$\pm 10^{-6}$	$(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{3}})$

TABLE I: table that show the error between real and simulate quaternion for different directions of unity vector v^r.

Another form to see that, is showing the error of rotated angle at every time step. In this case, we give $v^r = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{3}})$ and we rotated this axis $\frac{\pi}{2}$ rad at the velocity of $\frac{\pi}{90}$ rad/s. In figure 2, we can see the rotation and in figure 3 we can see the error of each simulated angle.

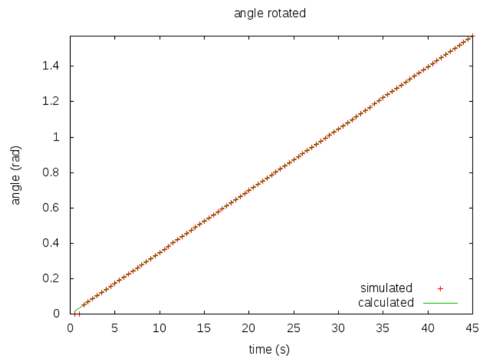

FIG. 2: The angle rotated at every time step of each angle calculated and simulated.

Also it is interesting to see how modify the error just as velocity of rotation, this error show the difference of first component of the calculated and simulated quaternion at different velocities.

In figure 4 we can see how the error is small. In this part we can determinate that the algorithm is self-consistent at low velocities. But, when we will put

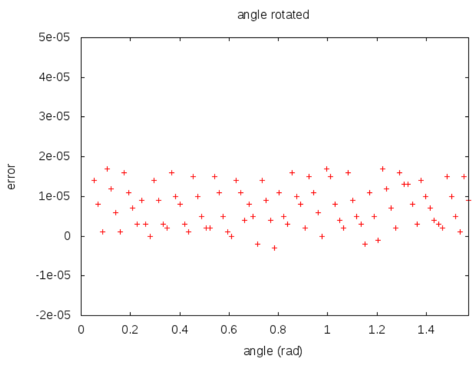

FIG. 3: The error at every time step of each angle.

high velocities in the program, the algorithm is not self-consistent. The velocity at which the algorithm begins to be non consistents is $\frac{pi}{6}$ rad/s, which can be considered a high velocity for quadcopters. These problem exist because the algorithm compares the quaternion result with the previous quaternion encountered with equation 4, and the change is too big.

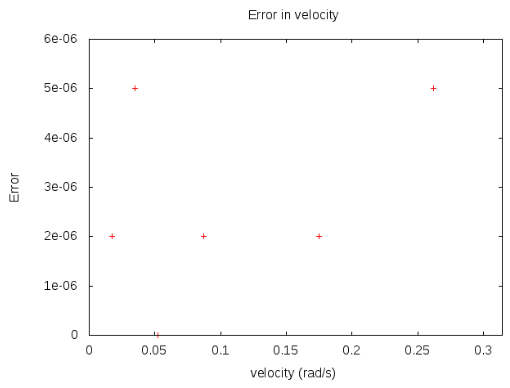

FIG. 4: The error absolut of first component of quaternion at different velocities.

Finally, we show the average of error of angle to velocity, position and angle in table II.

	average error
velocity	$2.67 \cdot 10^{-6}$
position	$1.44 \cdot 10^{-6}$
angle	0.0338

TABLE II: the average error of angle to velocity, position and rotation angle.

IV. CONCLUSIONS

The error in the new algorithm has been solved. As we have seen the error appears while creating the Jacobian of the function.

We can observe that the algorithm is useful to calculate the orientation of a rigid body, for example a quadcopter. As the results show, the method provides us a complete solution of the orientation once the error has been solved. If we compare the results with the results of [2], we can observe that are correct and consistent. Also, we can see that the method is not auto-consistent at high velocities, but in our case it is not a major problem.

V. FUTURE WORK

The new method has some pending studies, for example, we would like implement the gyroscopes into the program and obtain the hybrid method, similar to [1]. Also, with the complete method, we could study the implementation the algorithm in a real case and study the behavior of it. The studies of a real case implies the study of response time in the sensors.

Finally, also it will be interesting to find the way that the program will be auto-consistent at high velocities. And thus bring the method in other fields of study.

VI. ACKNOWLEDGMENTS

I would like to thank my advisor, José María Gómez Cama, for the long hours of debate and be considerate with my stubbornness in some moments of the debate. Also, I would like to thank my family and my girlfriend for the patience and confidence that have had in me. And finally, I would like to thank my friends for the help to write this paper in english.

[1] Sebastian O.H. Madwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays", *April 30, 2010*

[2] Sebastian O.H. Madgwick, Andrew J.L. Harrison, Ravi Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm", *IEEE International Conference on Rehabilitation Robotics, Rehab Week Zurich, ETH Zurich Science City, Switzerland, June 29-July 1, 2011*

[3] R. E. Kalman, "A new approach to linear filtering and prediction problems" *Journal of Basic Engineering, vol. 82, pp. 35-45. 1960*

[4] John Vince, "Quaternions for Computer Graphics".chapter 5,6 and 7.