IMU Data Recovery, Processing and Display

Decho Surangsrirat
NECTEC, Thailand

Vincent Maire
INSA Toulouse, France

July 2018

1 Abstract

This project aims to develop a system able to determine its own relative orientation in space. It uses an Inertial Measurement Unit (IMU): MPU9250. This IC is a 10 degrees of freedom sensor as it provides a 3D acceleration vector, an angular speed in both 3 coordinates, the intensity of the Earth Magnetic Field for each 3 axis, and the sensor's temperature.

The system consequently designed must be generic enough to be used as assistant for several and various medical tasks. However, to prove the accuracy and robustness of the system, it will be tested as an assistant for surgical screw implantation to improve the free hand technique, using the Jost et al paper [1] as reference. Thus, the software used in the present project has been rearranged to match with such a purpose.

This report will first describe the hardware used and its implementation. Then, it will cover the structure of the developed software and will detail the chosen solutions. Finally, it will explore the possible improvements that can be made and sum up this project.

2 Hardware architecture

As described in the figure 1, the system is divided into 3 main hardware components:

- The MPU9250 board which is the 10 DOF IMU
- An Arduino Nano, which communicates with the MPU9250 through an I2C protocol and recover the raw data
- A C# software, developed under Visual Studio, and yields the data from the Arduino to filter it, convert it into a quaternion representation and display the results to an user-friendly interface

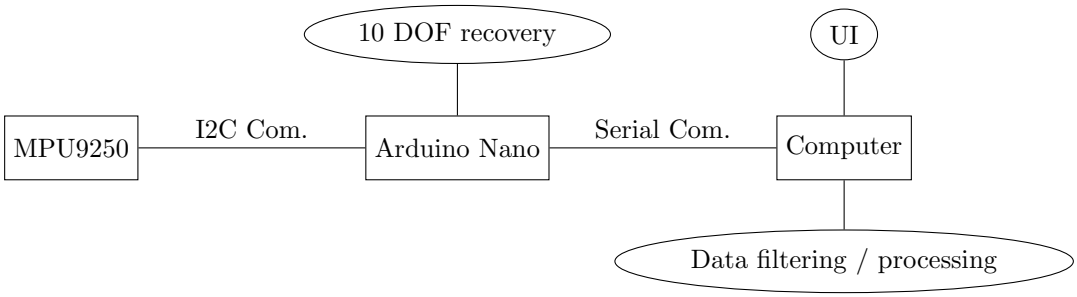

Figure 1: Basic architecture of the developed system

3 Arduino configuration

Only 4 wires are required to recover the data from the MPU9250:

- POWER: 3.3V/GND
- I2C Com.: SCL/SDA

The software downloaded on the Arduino [2] is open source, shared by FaBo and distributed under Apache license 2.0 (non-restrictive).

The Arduino only perform once the accelerometer and gyroscope calibration. However, the magnetometer data are still soft and hard iron biased (among others). Then, the raw values $\{ax, ay, az, gx, gy, gz, mx, my, mz, temp\}$ are transmitted through a high speed serial communication to the filtering and processing software.

4 Software's structure

Written in C#, developed with Visual Studio and implemented on a computer (Windows OS), this software has the capability to take 9 DOF raw data, process it into an understandable representation and display it to the user. An overview screenshot is displayed n figure 2.

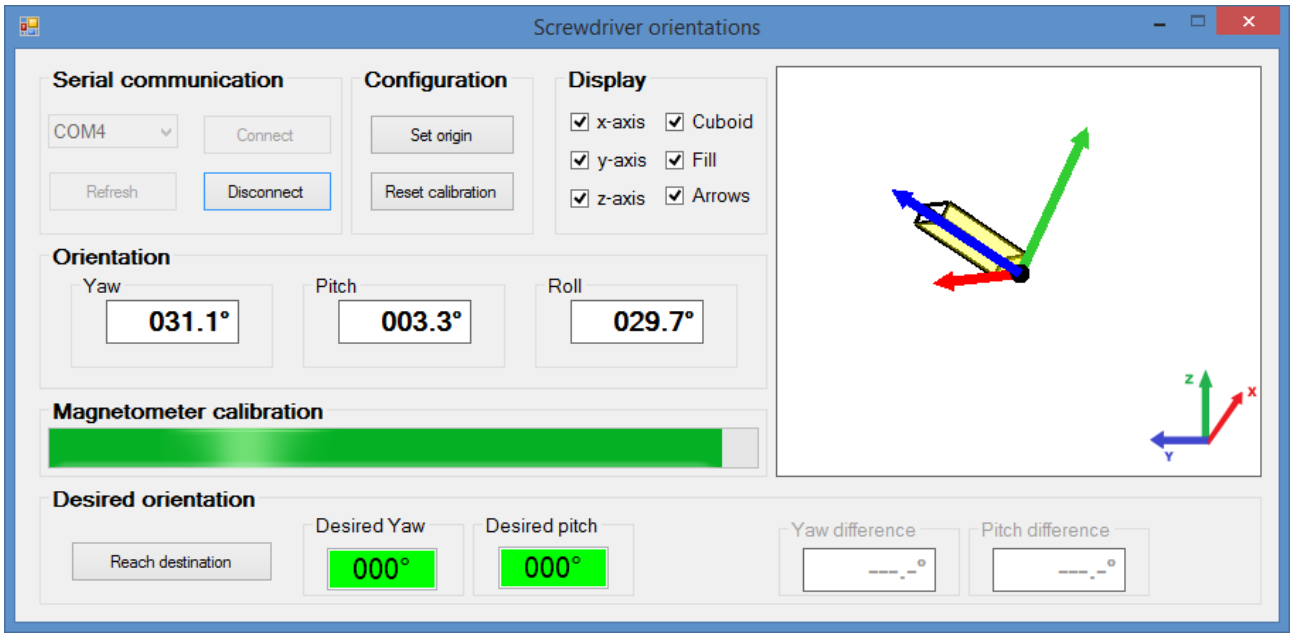

Figure 2: Software overview screenshot

Basically, the program is made of 3 threads described in the pseudo sequence diagram in figure 3:

- The serial thread that is triggered on data received. It parses the data from the serial and calibrate the received magnetometer values on-the-fly.
- The filter thread that implement a Mahony filter. It is triggered on a timer interrupt with a period of 2ms. It output quaternions and Euler angles from the raw data.
- The display thread is also triggered on a timer with a period of 20ms. It refreshes the display of the Euler angles and the 3D view.

5 Magnetometer calibration

Some corrections need to be done on the recovered magnetometer data. This section describes why this need to be calibrated and how we made it.

5.1 Origin of the magnetometer biases

Contrary to the gravity, the Earth magnetic field is not homogeneous everywhere on the planet. Thus, the magnetometer cannot use it as its reference to self calibrate. Moreover, the surrounding magnetic noise add some more uncertainty to the measurement.

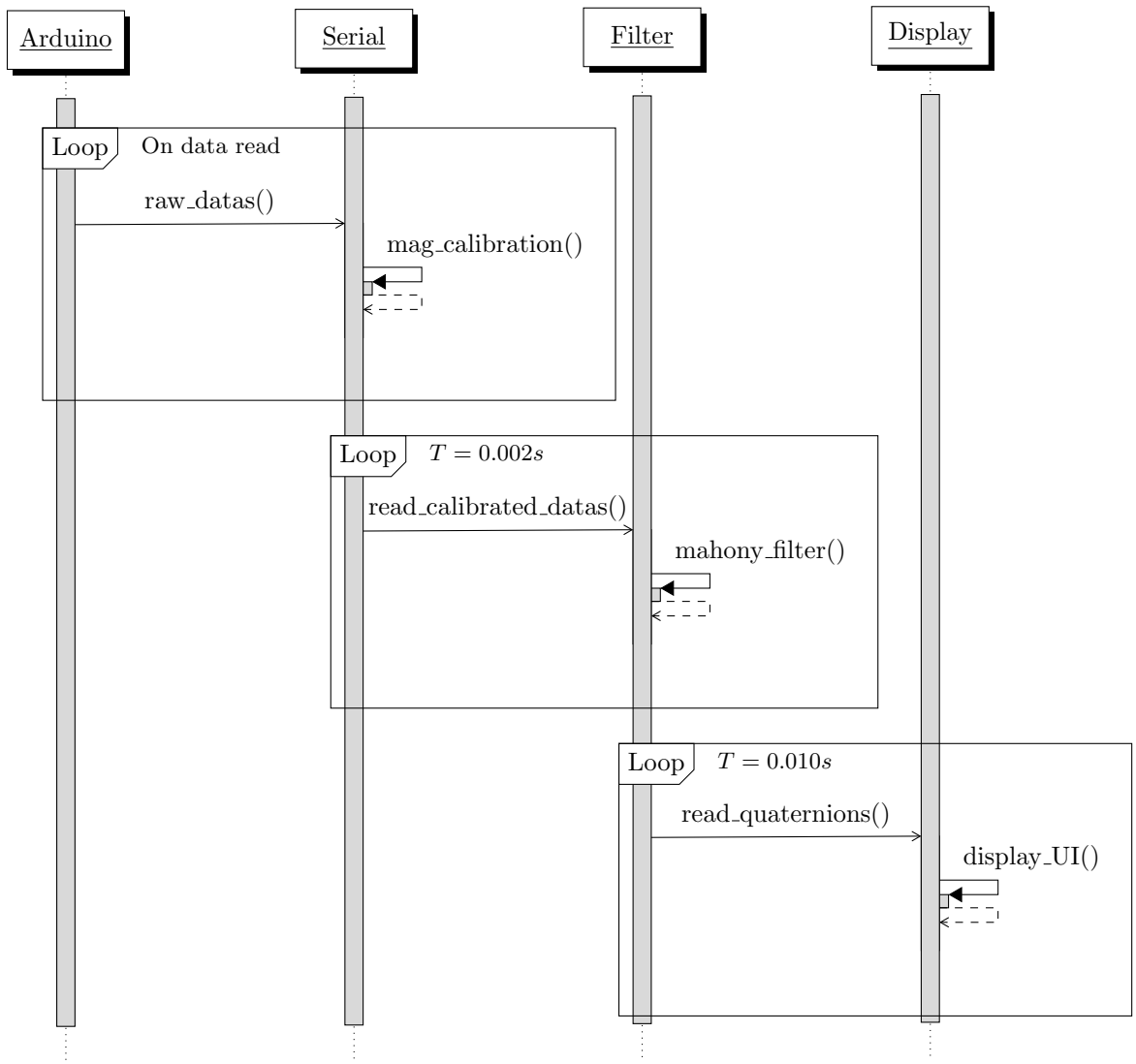

Figure 3: Sequence diagram

5.2 Proposed solution

A proposed solution described in the Renaudin et al paper [3] is to complete a full rotation of the IMU for each axis and to keep the minimum and maximum measured values. This solution has been implemented following the algorithm 1. It loops for each data measurement, which means that the calibration is performed on-the-fly. The more the user rotates the IMU, the better. However, this process partially solve only soft and hard iron distortion.

Algorithm 1 Pseudo code for magnetometer calibration

$min, max \in \mathbb{R}^3$
$min \leftarrow \{+\infty, +\infty, +\infty\}$
$max \leftarrow \{-\infty, -\infty, -\infty\}$
while $True$ **do**
  if $is\_raw\_data\_received$ **then**
    $mag \leftarrow read\_magnetometer()$
    for $i \leftarrow 1;\ i < 3;\ i++$ **do**
      if $mag[i] > max[i]$ **then**
        $max[i] \leftarrow mag[i]$
      else if $mag[i] < min[i]$ **then**
        $min[i] \leftarrow mag[i]$
      end if
      $offset \leftarrow \frac{max[i]+min[i]}{2}$
      $calibrated\_mag[i] \leftarrow \frac{mag[i]-offset}{max[i]-min[i]}$
    end for
  end if
end while

5.3 Results

This solution has been tested and seems to be powerful enough for our specific application. The figure 4 shows the difference between the raw magnetometer measurements and the calibrated ones. We notice that the calibrated measurements are quite well circularized and normalized as they lie inside the black circle in the 2D plot. In the 3D plot, we want the measurement to fit in the black sphere.

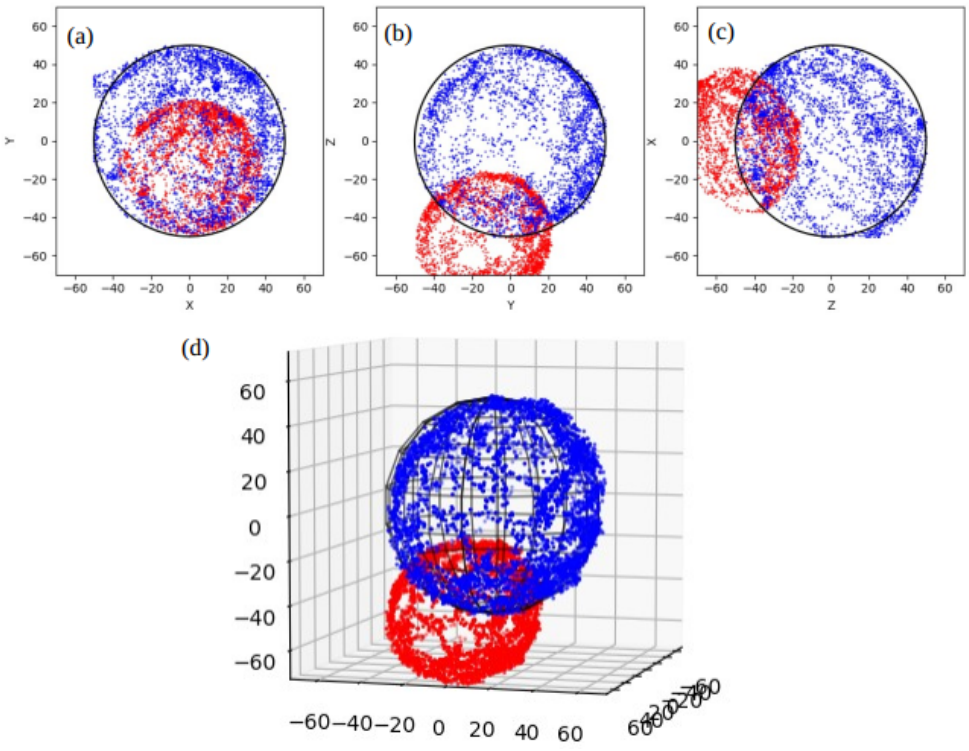

Figure 4: Comparison between raw data in red and calibrated data in blue. (a), (b) and (c) are 2D views of the magnetic field intensity of each x, y, z axis depending on another axis. (d) is a 3D plot of the magnetic field intensity depending on each axis.

However, this solution is the simplest to implement and is far to be perfect as it cannot compensate some of the sensor biases like sensor non-orthogonality. As the system is intended to be used in such places like hospitals, which contain many magnetic disturbances due to the heavy equipment and medical machines, finding a powerful calibration process should become an important feature to implement. Many research papers are dedicated to

3 axis magnetometer calibration. Vasconcelos et al propone in their study [4] a more powerful calibration using a geometrical approach, without needing the attitude of the IMU. Another research from Kok et al [5] uses the gyroscope and accelerometer to perform a magnetometer calibration.

6 Orientation processing

This section describes the computation steps required in order to get the IMU's orientation from the sensor's data.

6.1 Mahony filter

The system uses an open source implementation of a Mahony filter [6], a non-linear complementary filter, based on the work of Mahony et al [7]. It output the orientation in quaternion form. It is updated every $T = 5ms$ with a proportional gain $Kp = 15$ and integral gain $Ki = 0$.

Watching closely to the MPU9250's datasheet, we notice that the orientation of the axes of sensitivity of the compass and the accelerometer are different. Thus, the order of the parameters fed to the filter have to be chosen carefully. In our case and in this order: $(g_x,\ g_y,\ g_z,\ a_x,\ a_y,\ a_z,\ m_y,\ m_x,\ -m_z)$.

6.2 Using the quaternions

6.2.1 Quaternion rotation

Quaternions are a powerful representation of the attitude as, among other advantages, require less computation to perform operations on them than Euler angles.

Let $v = (v_1, v_2, v_3)$ be the 3D vector we want to rotate from the quaternion $q = (q_1, q_2, q_3, q_4)$.

$$v_R = \begin{bmatrix} 1 - 2q_2^2 - 2q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & 1 - 2q_1^2 - 2q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & 1 - 2q_1^2 - 2q_2^2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

In the same way, we can rotate a quaternion from another quaternion representation by multiplying them together.

6.2.2 Quaternion to Euler representation

Contrary to the Euler angles, quaternions lie into a 4 dimensional space, which make this representation gimbal lock issue free. However, it might be useful to display the Euler angles to the user are they are more understandable and readable.

Working with the proper Euler angles (not the Tait-Bryan ones), we should use the angles [$\Phi\ \Theta\ \Psi$] but for more convenience, let's use in the same order [*yaw pitch roll*]:

$$\begin{bmatrix} yaw \\ pitch \\ roll \end{bmatrix} = \begin{bmatrix} atan2(2(q_0q_1 + q_2q_3), 1 - 2(q_1^2 + q_2^2)) \\ asin(2(q_0q_2 - q_3q_1)) \\ atan2(2(q_0q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2)) \end{bmatrix}$$

7 Software features and usage

This section explores the software features and presents an user-oriented how-to-use description.

7.1 Features and their implementation

7.1.1 "Serial communication" box

Contains a drop-down list to choose the serial port COM, a connect and disconnect button, and a refresh button that scans the available COM ports.

7.1.2 "Configuration" box

It contains a "Set origin button" which defines the current orientation as the new local frame following this process:

Let q be the current measured quaternion from the Mahony filter, and $\tilde{q}$ the new quaternion in the new reference frame. This new reference frame is saved when the user clicks "Set origin" by setting $q_0 = q$. Then,

$$\tilde{q} = q.q_0^{-1}$$

The second button in this box is "Reset calibration" and it used to reset the magnetometer's calibration. By clicking it, the user resets the algorithm 1 (breaks the loop and starts again at the first line). This button is essential as the current way to calibrate the magnetometer is performed on-the-fly and if a too strong magnetic disturbances is measured once, the calibrated measurements will become biased.

7.1.3 "Display" box and picture box

The "Display" box manages what should be displayed in the picture box.

The picture box displays a cuboid and a compass in a 3D view. For now, this is performed with a 2D drawing library which rotates the vector using the quaternion rotation described in the section 6.2.1. Thus, it is sometime hard to distinguish the 3D system in such a 2D environment. To help the user, the compass is displayed in a correct order to be able to see which arrow is upfront.

When the user clicks the "Set origin" button described in the section 7.1.2, the angle of view will rotate following the new local frame. Thus, the system will be zeroed in the same orientation reference displayed in the bottom right corner.

7.1.4 "Orientation" box

It displays the yaw, pitch and roll angles in degrees in the local frame. This angle have to be interpreted carefully as they are subject to gimbal lock when the angle start to be bigger.

7.1.5 "Magnetometer calibration" progress bar

This progress bar uses the norm of the magnetic field of the calibrated magnetometer values and compare it to the expected one after calibration. This follows the equation below:

$$v_{\%} = \left(1 - \left|1 - \frac{n}{\hat{n}}\right|\right) * 100$$

With $v_{\%}$ the value of the progress bar, n the current norm of the calibrated values of the magnetic field and $\hat{n}$ the expected norm of the calibrated values.

The progress bar is filtered (moving average) to smooth its variations.

7.2 "Desired orientation" box

This box contains 2 fields that allow the user to choose the desired orientation for the screw implantation, in the local frame coordinate. Two others label display the difference in degrees between the current and desired orientation.

For convenience, the user enters the orientation in degrees as proper Euler angles. However, Euler angles, lying in only 3 dimensions are subject to gimbal lock. Thus, the orientation finding is accurate for small pitch angles only ($\sim < 45°$).

7.3 Using the screw implantation assistance

The following procedure describes how to use the screw implantation assistance:

1. After connecting the IMU, calibrating the magnetometer by orienting slowly the sensor in every 3D direction. The calibration progress bar should be almost full.

2. Aligning the sensor with the chosen axis (sagittal, axial, coronal). The axis orientation should be displayed on the sensor's box, as on figure 5 to help the user to find the axis.

3. Setting the origin by clicking the dedicated button.

4. Entering the desired angles in the "Desired orientation" box and launching the process by clicking the button "Reach destination".

5. Once the polar chart is displayed as shown in the figure 6, the goal is to match the cross and the circle by changing the sensor's orientation. They become green when they are close enough to each other (less than a 2° difference on both 2 axis).

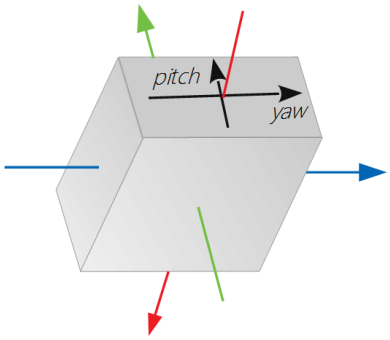

Figure 5: Example of the IMU's box on which the axes orientation is displayed

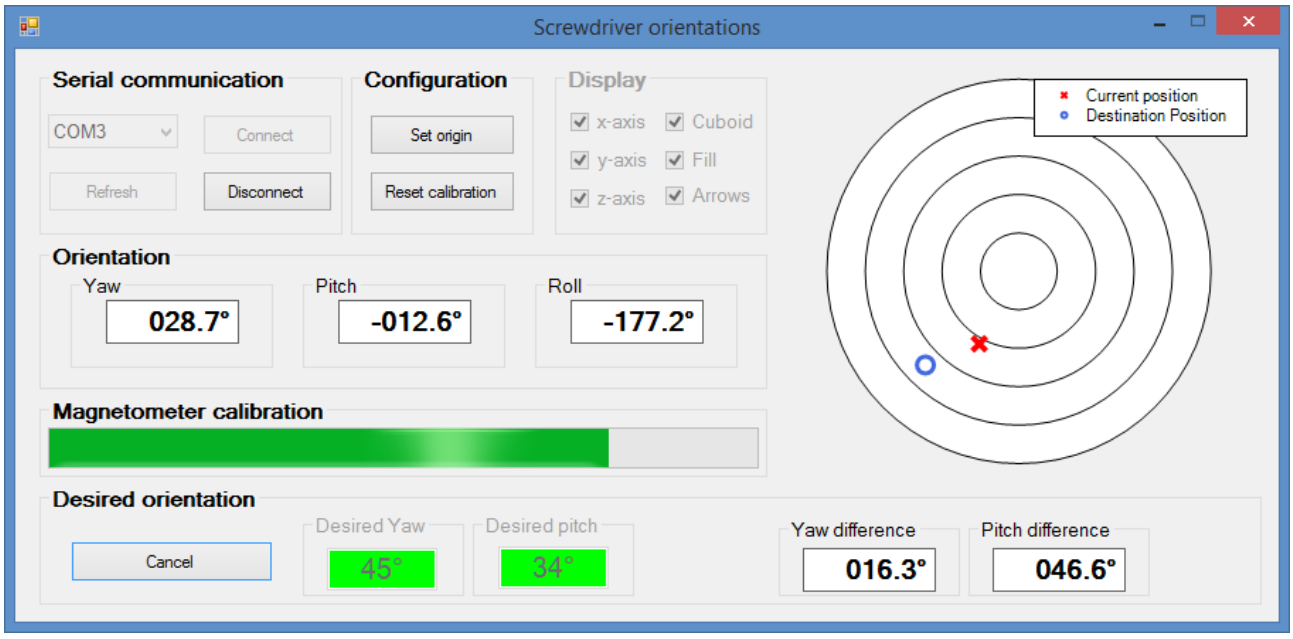

Figure 6: IMU's assistant for screw implantation orientation finding

8 Improvements

As this project can be considered as a demonstrator, it can be improved in many different way.

First, the UI can be improved by displaying a better 3D view of the system and by re-designing the interface.

Furthermore, the main issue with the computed orientation is its lack of accuracy in some positions. It is due mainly to the distortion of the magnetic field and could be sharply improved by using more powerful calibration techniques as discussed in the section 5.

Finally, the raw data filtering can be improved and some papers, as Valenti et al [8] 's one, describe a more accurate processing. Moreover, a comparison of different attitude algorithms can be found in the Cavallo et al 's [9] article.

9 Conclusion

To sum up this project, with a few hardware requirement and few computation steps, it is possible to recover the attitude of an IMU. Standalone IMUs are already available on the market but they are very expensive, contrary to our solution.

For now, we use this system as an screw implantation assistant but it can be used as a base solution for many and various applications.

10 Source code

The source code of the Arduino and the C# software can be found as a git repository on *Github* at the following URL: `https://github.com/Vincema/MPU9250Interface`.

References

[1] G. F. Jost, J. Walti, L. Mariani, S. Schaeren, and P. Cattin, "Inertial Measurement Unit-Assisted Implantation of Thoracic, Lumbar, and Sacral Pedicle Screws Improves Precision of a Freehand Technique," *World Neurosurgery*, vol. 103, pp. 11 – 18, 2017.

[2] FaBo, "Fabo 9 axis mpu9250 library." `https://github.com/FaBoPlatform/FaBo9AXIS-MPU9250-Library`.

[3] V. Renaudin, M. H. Afzal, and G. Lachapelle, "Complete triaxis magnetometer calibration in the magnetic domain," *Journal of Sensors*, vol. 2010, p. 10, 2010.

[4] J. Vasconcelos, G. Elkaim, C. Silvestre, P. Oliveira, and B. Cardeira, "A geometric approach to strapdown magnetometer calibration in sensor frame," *IFAC Proceedings Volumes*, vol. 41, no. 1, pp. 172 – 177, 2008. 2nd IFAC Workshop on Navigation, Guidance and Control of Underwater Vehicles.

[5] M. Kok and T. B. Schon, "Magnetometer calibration using inertial sensors," *IEEE Sensors Journal*, vol. 16, pp. 5679–5689, July 2016. arXiv: 1601.05257.

[6] x-io Technologies, "Open source imu and ahrs algorithms." `https://http://x-io.co.uk/open-source-imu-and-ahrs-algorithms`.

[7] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear Complementary Filters on the Special Orthogonal Group," *IEEE Transactions on Automatic Control*, vol. 53, pp. 1203–1217, June 2008.

[8] R. G. Valenti, I. Dryanovski, and J. Xiao, "Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs," *Sensors (Basel, Switzerland)*, vol. 15, pp. 19302–19330, Aug. 2015.

[9] A. Cavallo, A. Cirillo, P. Cirillo, G. D. Maria, P. Falco, C. Natale, and S. Pirozzi, "Experimental Comparison of Sensor Fusion Algorithms for Attitude Estimation," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 7585 – 7591, 2014.