République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

ÉCOLE NATIONALE POLYTECHNIQUE

DÉPARTEMENT D'ÉLECTRONIQUE

Mémoire de Master

Thème :

Position Estimation Using Motion Inertial Sensor

Encadré par :

Dr. Rabah SADOUN

Réalisé par :

Hacene BOUAROUA

Juin 2015

Je dédie ce travail à:

Mes chers parents

Mes deux chers frères Salah et Ahmed et chère sœur Menna

Mes chers oncles Kacem et Salah et leurs familles

Toute la grande famille

Tout mes amis Marouane. Brahim. Farouk. Tarek. Khaled. James et Sadeddine

B.Hacene

Remerciements

En premier lieu, je tiens à remercier Dieu, de m'avoir aidé et de m' avoir donné la patience et la foi de finaliser ce mémoire.

J' exprime toutes mes reconnaissances à Monsieur Rabah SADOUN, mon encadreur; pour sa patience, la qualité de ses conseils et sa serviabilité.

J'addresse mes sincères remerciements à tous les membres du Jury chargé d'examiner la soutenance de ce travail.

Je veux exprimer mon profond respect à tous les Enseignants de l'École Nationale Polytechnique pour la formation qu'ils m' ont donnée.

Finalement, je remercie tous ceux qui n'ont épargné aucun effort, de près ou de loin, pour me permettre d'accomplir ce modeste travail.

ملخص:

الهدف من هذا المذكرة هو تقدير لتحديد المواقع باستخدام 6-DOF IMU التي تحتوي على جيروسكوب3- محاور و حساس التسارع من 3محاور. وأيضا أوضحنا الخوارزمية خلال هذا البحث. كما تم اقتراح حلول لأخطاء الناتجة من الحساس.

كلمات مفتاحية: IMU – INS – GDOF – Drift

Abstract:

The aim of this work is the estimation of the positioning using a 6-DOF IMU which contain a 3-axis gyroscope and a 3-axis Accelerometer. The algorithm is explained through this project. Also the solutions to errors are proposed.

Keywords: IMU – INS – GDOF – Drift

Résumé:

Le but de ce travail est l'estimation du positionnement en utilisant un IMU 6-DOF, qui contient un gyroscope 3-axes et un accéléromètre 3-axes. L'algorithme est expliqué à travers ce projet. Aussi les solutions aux erreurs sont proposées.

Mots-clés : IMU – INS – GDOF – Drift

Contents

List of Figures

1 Introduction

In this project we intend to apply a method used to calculate the positioning and velocity of a body using a simple IMU (Inertial Measurement Unit). The sensor used is of MEMS (Microelectromechanical systems) type. Through this project we are going first to introduce Inertial Navigation Systems (INS). Second, explain how we obtain the position computation using an IMU. Third an implementation using an MPU6050 IMU sensor data and a MATLAB Code. Finally a general discussion about the algorithm and the results obtained. This project is a following of the work done in the graduation project since the real sensor data are taken from the Simulink MPU6050 bloc also the results of the orientation Filter will be used.

2 Inertial navigation:

Inertial navigation is a relative positioning technique in which an inertial measurement unit (IMU) is tracked relative to its initial starting point, orientation and velocity. An IMU that can be tracked in 3-dimensions usually consists of three orthogonal accelerometers and three gyroscopes that are aligned with the accelerometers, as shown in Figure 2.1.

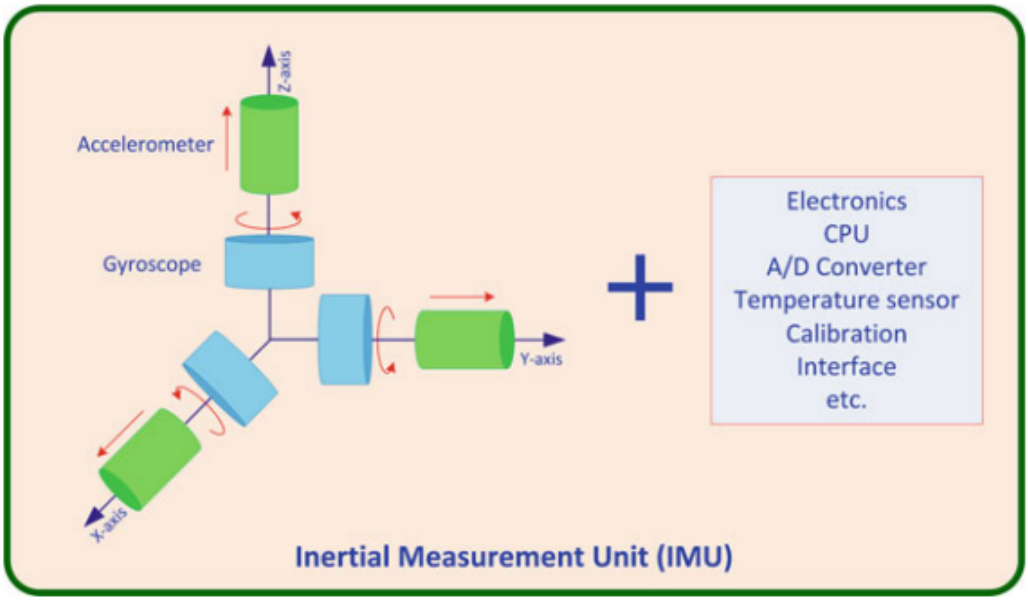

Figure 2.1: The components of a typical IMU.

An inertial navigation system (INS) Fig 2.2 consists of an IMU together with a navigation processor, which uses measurements from the IMU in order to track its orientation, position and velocity in some frame of reference in which they are desired (the reference frame). The processor uses angular rate measurements obtained from the gyroscopes in order to track the orientation of the IMU relative to this frame. The known orientation is then used to transform the specific force (acceleration due to all forces except for gravity) measured by the accelerometers into this frame. Acceleration due to gravity is then added to the specific force to obtain the acceleration of the device. Finally, this acceleration is integrated once to track velocity and once more to track the device's position in the reference frame.

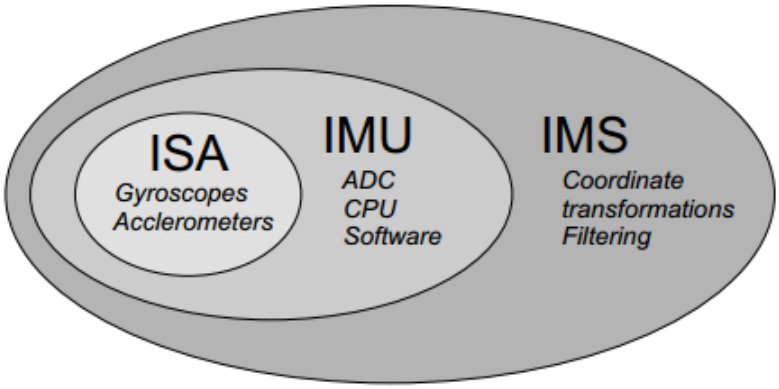

Figure 2.2: The component of an Inertial Navigation System.

Stable platform and strapdown INSs are both based on the principle of double integrating acceleration in the reference frame. Strapdown systems have reduced mechanical complexity and as a result tend to be physically smaller and lighter than stable platform systems. These benefits are achieved at the cost of increased computational complexity, however the requirements are now trivial relative to the computational power of modern processors. As a result strapdown systems have become the dominant type of INS.

3 The potential of inertial sensors:

MEMS IMUs contain three orthogonal gyroscopes and accelerometers. The gyroscopes measure the angular velocity of the IMU body with respect to inertial space. In other words they measure the IMU's angular velocity with respect to any inertial coordinate frame, which is any frame that does not accelerate or rotate with respect to the rest of the universe. The accelerometers measure the specific force acted upon the IMU, also with respect to inertial space. Since the inertial sensors are rigidly attached to the body of the IMU, all measurements are made in the IMU's own frame of reference. This is known as the body frame, as shown in 3.1.

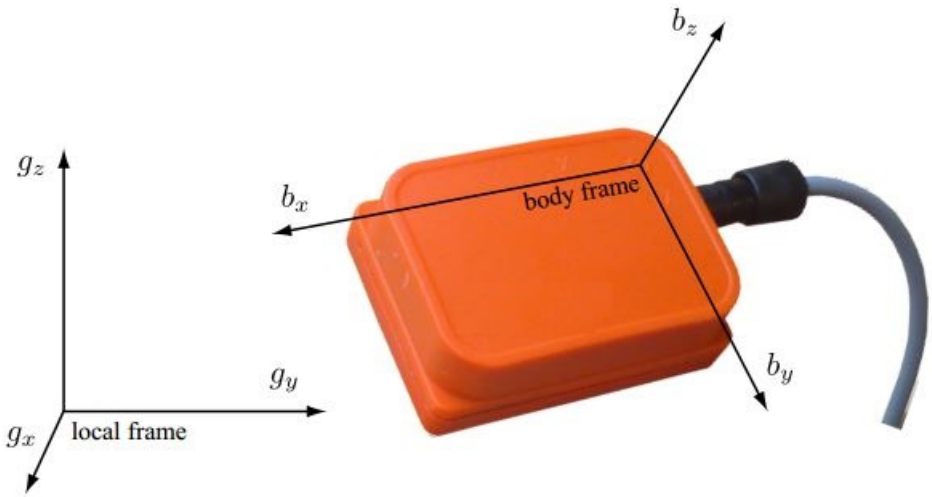

Figure 3.1: Illustration of body and local frames of reference [1].

The basic principle is to track changes in the device's orientation using the gyroscopes and Accelerometer measurements, which can be used to project the accelerations measured locally by the device into a global frame of reference. These accelerations can then be double integrated to track changes in the object's position Fig 3.2 . In practice however, errors rapidly accumulate in the tracked position due to the propagation of measurement errors through the projection and integration calculations. Such errors are collectively referred to as drift.

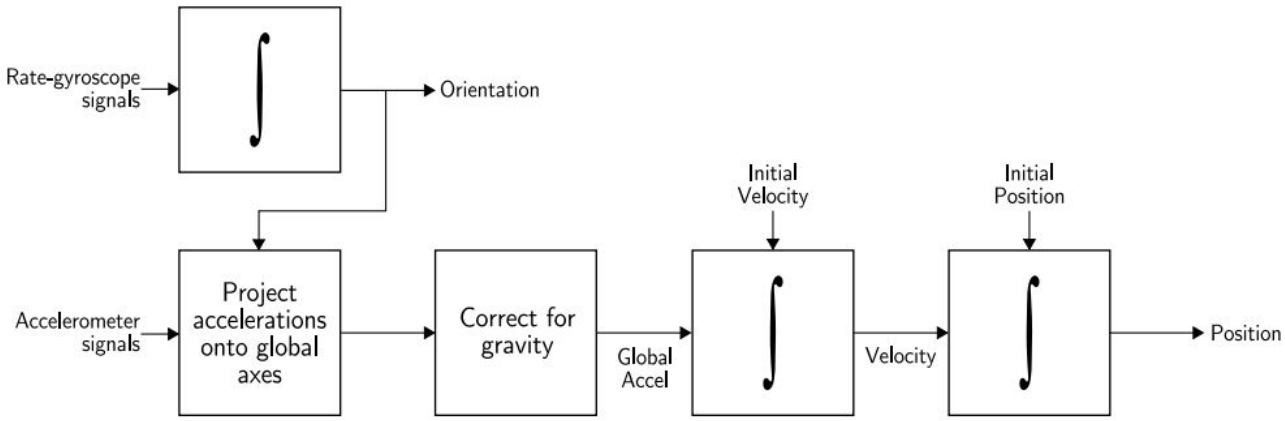

Figure 3.2: A typical inertial navigation algorithm

Let's explain briefly with mathematic equations how to obtain theoretically the position from a 6-DOF IMU sensor Fig 3.2. The orientation, or attitude, of an IMU relative to a local reference frame can be tracked by integrating its angular velocity [1].

$$\omega^b(t) = (\omega^{b_x}(t), \omega^{b_y}(t), \omega^{b_z}(t))^T. \tag{3.1}$$

The orientation of the device can be specified using several different representations, including Euler angles, quaternions and direction cosines. The direction cosines representation is used in the derivations below. In the direction cosines representation the orientation of the IMU's body frame relative to the local frame is specified by a 3×3 rotation matrix C , in which each column is a unit vector along one of the body axes specified in terms of the local axes Fig 3.1.

$$v^g = Cv^b \tag{3.2}$$

A vector quantity defined in the body frame is equivalent to the vector defined in the local frame. The inverse transformation is given by:

$$v^b = C^T v^g \tag{3.3}$$

Since the inverse of a rotation matrix is equivalent to its transpose. The orientation of the IMU at time t is given by $\mathbf{C}(t)$ calculated with each new data coming from the gyroscope data.

To track the position of an IMU the specific force given by the 3-axis accelerometer.

$$\boldsymbol{f}^b(t) = (f^{b_x}(t)\,, f^{b_y}(t)\,, f^{b_z}(t))^T \tag{3.4}$$

is first projected into the local frame of reference

$$\boldsymbol{f}^g(t) = \mathbf{C}(t)\boldsymbol{f}^b(t) \tag{3.5}$$

Acceleration due to gravity is then added to obtain the acceleration of the IMU, which is integrated once to obtain velocity and again to obtain displacement

$$\mathbf{a}^g(t) = \boldsymbol{f}^g(t) + \mathbf{g}^g. \tag{3.6}$$

$$\boldsymbol{v}^g(t) = \boldsymbol{v}^g(0) + \int_0^t \mathbf{a}^g(t)dt \tag{3.7}$$

$$\mathbf{s}^g(t) = \mathbf{s}^g(0) + \int_0^t \boldsymbol{v}^g(t)dt \tag{3.8}$$

Where $\boldsymbol{v}^g(0)$ is the initial velocity of the device, $\mathbf{s}^g(0)$is the initial displacement and is acceleration due to gravity specified in the local frame of reference.

4 Correcting orientation error using GDOF

The application of a gradient descent orientation filter can eliminate the errors from the gyro drift when calculating Orientation. We take for example the Madgwik AHRS algorithm; the algorithm use a fusion between the gyro data and the accelerometer data for better orientation estimation. But this method is efficient only if we have a Magnetometer in addition to the IMU (gyroscopes + accelerometer) 9-axis degree. With 6-axis degree IMU sensor the algorithm suffer from a clear drift in the yaw angle. This will make an error propagate in the next integration steps for the calculation of the position[2].

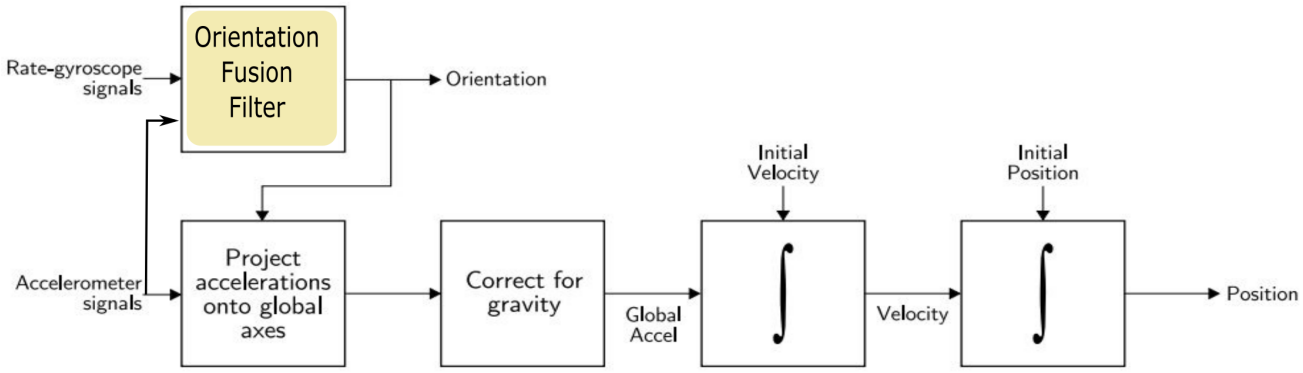

Figure 4.1: Correction of Orientation outputs using a Fusion Filter

4.1 Errors constraints:

With the existence of the Yaw angle drift, the user of the 6-axis degree IMU can calculate only the positioning orthogonal to the yaw variation angle plan for our case the z-axis since there is now error detected from the acceleration projection step. This type

of limited axis movement is applied as an example for the calculation of the frequency vibration of any vertical displacements. But this can be achieved only by the use of an accelerometer to compute a linear acceleration.

4.2 Drift Accelerometre Error correction:

The drift is due to the integration and it occurs even if the accelerometer bias is exactly zero. it is issued from the accumulating white noise in the sensor data reading, knowing that Noise are high frequency signals whereas drifts are low or very low frequency signals. To correct drift, we add a High pass filter to eliminate the drift error accumulated after each stage of integration. One when calculating the velocity and another one when integrating to velocity to find the position [3] as shown in Fig 4.2.

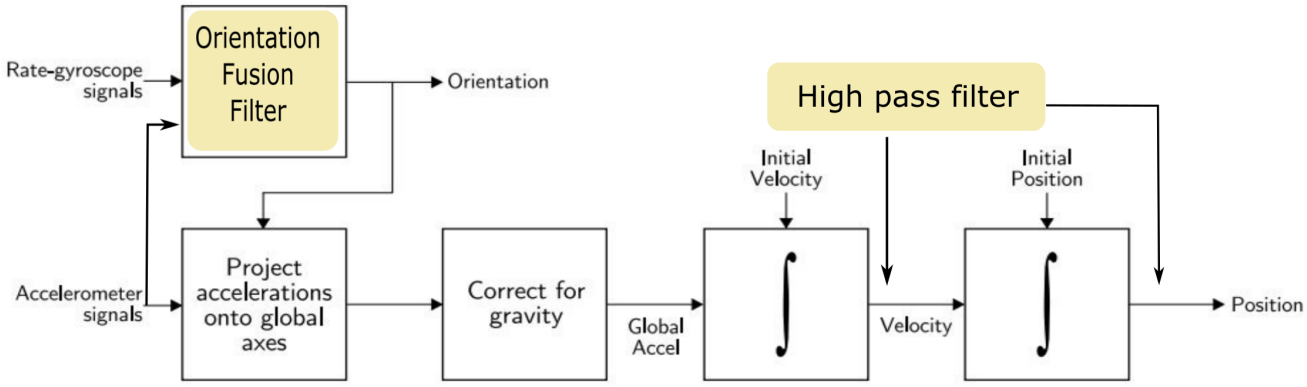

Figure 4.2: Addition of a High pass filter to eleminate the drift.

5 Implementation:

In the implementation we used as a sensor the MPU6050 IMU from Invensense,contained in the GY-521 evaluation board. the data acquisition is performed by the rapid prototyping platform build in the graduation project using Simulink and the Zedboard 5.1. Then the sensor data were saved to the Matlab workspace to be used in the localization algorithm explained above, so this is called offline simulation.

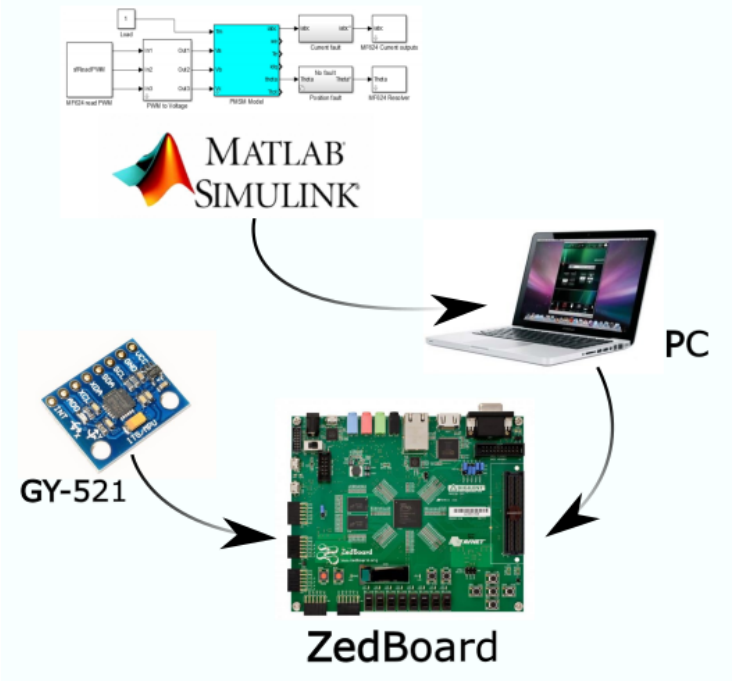

Figure 5.1: The design platform.

The 5.2 show the Simulink environment which contain a Sensor S-function bloc to feed a Simulink bloc named “To workspace”. The last one saves the sensor data to the MATLAB workspace.

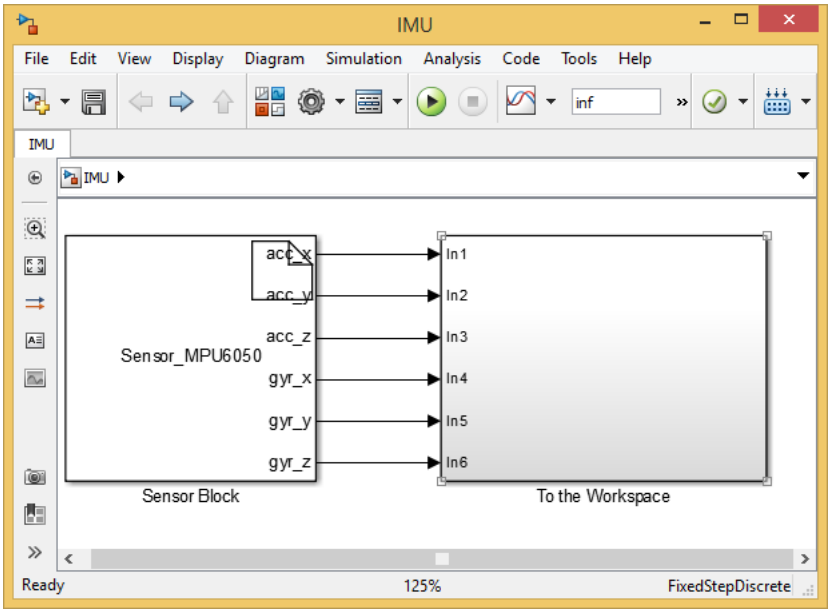

Figure 5.2: Mpu6050 Sensor Simumink Driver Bloc .

After a simulation was effected, six vectors representing the 3-axis gyroscopes and 3-axis Accelerometer containing a finite time running sensor data as shown in figure below fig 5.3.

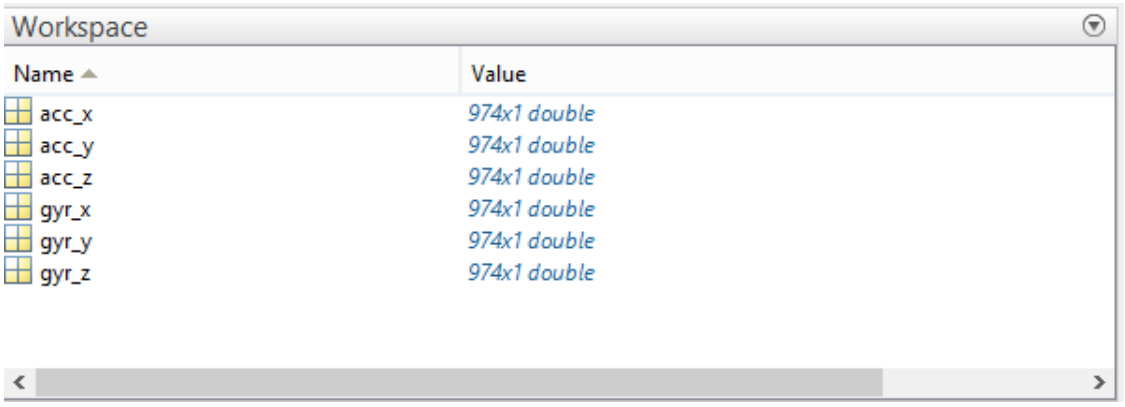

Figure 5.3: Obtained Data in the Matlab worckspace.

Next using data we are going to implement the localization algorithm using Matlab. The original code is available in GitHub [4]. Containing all the necessary libraries and functions to execute the code like "@MadgwickAHRS" responsible for the orientation calculation and the "quaternion_library" transform calculated quaternion to rotation matrix. "SixDOFanimation" function is used to animate the movement variation of the sensor reference as shown in the figure 5.5.

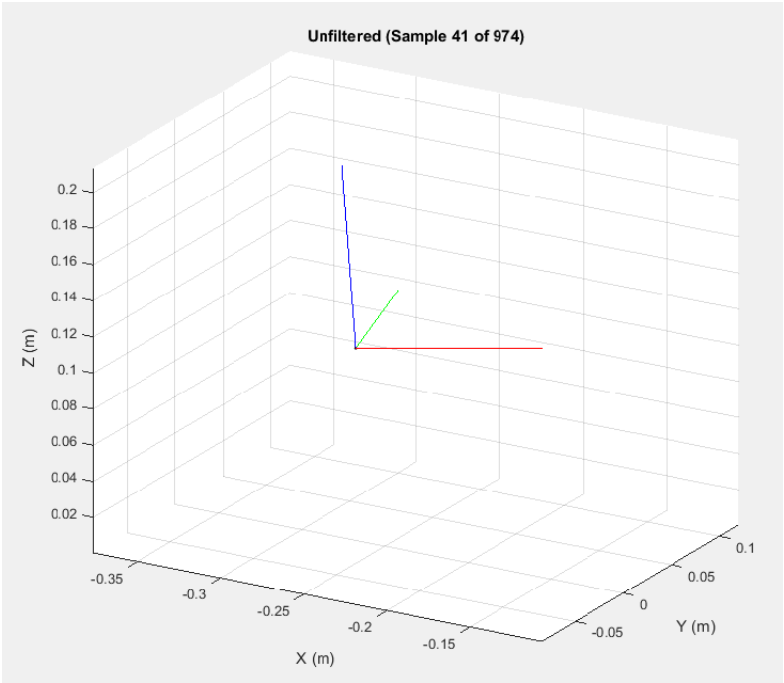

Figure 5.4: Classic attitude estimation

.

after the simulation run the "SixDOFanimation" function plot the total axis mouvements as shown in figure **??**.

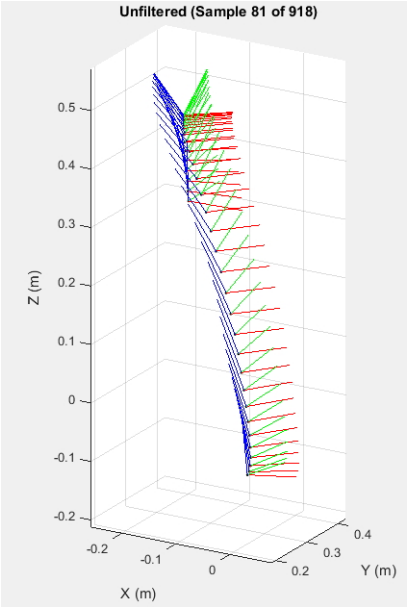

Figure 5.5: SixDOFanimation function animation

.

Results and Discussion:

After applying this method for the calculation of 3D positioning using only an IMU, some errors appears for example drift positioning due still exist as shown in figure 5.6.. The resulted displacement exceed the length of the wires used to attach the MPU6050 Sensor to the Zedboard which is 0.30 m. Also as discussed in section 4.1 if the calculated orientation are wrong the calculated positioning in the y-x plan will not fit the actual movement. In the next section a state-of-art is given, showing different techniques used to correct errors for the inertial navigation system.

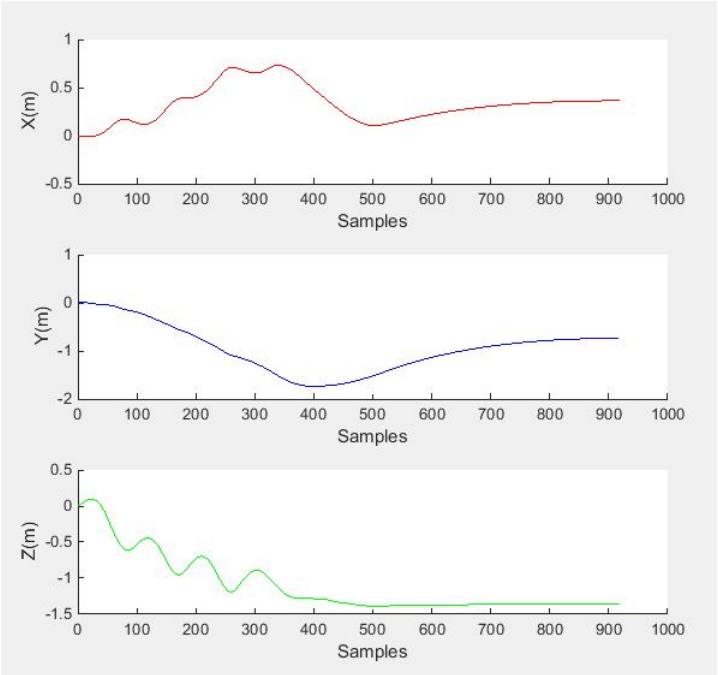

Figure 5.6: Resulted 3-axis positionig .

.

6 INS error correction:

It is now well-known that the use of numerical integration of acceleration/angular rate information from inertial sensors (accelerometers/gyroscopes) to obtain position/orientation information inherently causes position/orientation errors to grow with time, which is commonly known as "integration drift". For that reason, estimation of position/orientation using inertial sensors is performed with the help of externally-referenced aided sensors or sensing systems , or prior knowledge about the motion to correct for the drift. With the aided sensors or sensing systems, Kalman filters (KF) or extended-Kalman filters (EKF) are commonly used to fuse two sources of information: one coming from the inertial sensors, and the other from aided sensors or sensing systems in an attempt to correct for the drift. For example, correction of orientation drift using EKF and a magnetometer as an aided sensor is described in . Correction of position and orientation drift using EKF and ultrasonic sensors as aided sensors is presented in [1]. One of the drawbacks of having to rely on aided sensors to correct for the drift is that the accuracy depends on the update rate, availability, and reliability of the aided sensors. An example application of the use of inertial sensors with prior knowledge of motion is in human-walking studies. The use of prior knowledge of motion of human walking makes it possible to avoid the use of aided sensors or sensing systems for correction of the drift [3], allowing studies of natural walking outside the laboratory. Another application of the use of inertial sensors with prior knowledge of motion is physiological tremor sensing. In physiological tremor sensing for real-time compensation [5], zero-phase adaptive filtering algorithms based on truncated Fourier series such as weighted-frequency Fourier linear combiner (WFLC)[6] or band-limited multiple Fourier linear combiner (BMFLC)[7], which can detect periodic or quasi-periodic signals, are employed [8].

7 Conclusion

Through this project, an algorithm of position tracking was explained using only an IMU. Errors in orientation were corrected using GDOF algorithm and accelerometer drift was suppressed using a high pass filter. Finally, a state of art solutions were proposed to correct drift error in Inertial Navigation Systems.

References

[1] O. Woodman and R. Harle, "Pedestrian localisation for indoor environments", in *Proceedings of the 10th international conference on Ubiquitous computing*, ACM, 2008, pp. 114–123.

[2] S. O. Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays", *Report x-io and University of Bristol (UK)*, 2010.

[3] (). Oscillatory motion tracking with x-imu, [Online]. Available: http://www.x-io.co.uk/oscillatory-motion-tracking-with-x-imu/ (visited on 06/18/2015).

[4] (). Oscillatory motion tracking with x-imu, [Online]. Available: https://github.com/xioTechnologies/Oscillatory-Motion-Tracking-With-x-IMU (visited on 06/18/2015).

[5] C. N. Riviere, J. Gangloff, and M. De Mathelin, "Robotic compensation of biological motion to enhance surgical accuracy", *PROCEEDINGS-IEEE*, vol. 94, no. 9, p. 1705, 2006.

[6] C. N. Riviere, R. S. Rader, and N. V. Thakor, "Adaptive cancelling of physiological tremor for improved precision in microsurgery", *Biomedical Engineering, IEEE Transactions on*, vol. 45, no. 7, pp. 839–846, 1998.

[7] K. C. Veluvolu and W. T. Ang, "Estimation of physiological tremor from accelerometers for real-time applications", *Sensors*, vol. 11, no. 3, pp. 3020–3036, 2011.

[8] W. T. Latt, K. C. Veluvolu, and W. T. Ang, "Drift-free position estimation of periodic or quasi-periodic motion using inertial sensors", *Sensors*, vol. 11, no. 6, pp. 5931–5951, 2011.

Appendix

Main code :

```
%% Housekeeping
addpath('ximu_matlab_library'); % include x-IMU MATLAB library
addpath('quaternion_library');  % includerion  quatenlibrary
close all;                      % close all figures
clc;                            % clear the command terminal
%% Import data
samplePeriod = 0.01;
acc=[acc_x ,  acc_y , acc_z];
gyr=[gyr_x ,gyr_y ,gyr_z];

%% Process data through AHRS algorithm (calcualte orientation)
% See: http://www.x-io.co.uk/open-source-imu-and-ahrs-algorithms/

R = zeros(3,3,length(gyr));     % rotation matrix describing sensor
relative to Earth

ahrs = MadgwickAHRS('SamplePeriod', samplePeriod, 'Beta', 0.1);
% ahrrs = MahonyAHRS('SamplePeriod', samplePeriod, 'Kp', 0.5);

for i = 1:length(gyr)
    ahrs.UpdateIMU(gyr(i,:) * (pi/180), acc(i,:));  % gyroscope units
must be radians
    R(:,:,i) = quatern2rotMat(ahrs.Quaternion)';    % transpose because
ahrs provides Earth relative to sensor
end

%% Calculate 'tilt-compensated' accelerometer

tcAcc = zeros(size(acc));  % accelerometer in Earth frame

for i = 1:length(acc)
    tcAcc(i,:) = R(:,:,i) * acc(i,:)';
end

%% Calculate linear acceleration in Earth frame (subtracting gravity)

linAcc = tcAcc - [zeros(length(tcAcc), 1), zeros(length(tcAcc), 1),
ones(length(tcAcc), 1)];
linAcc = linAcc * 9.81;     % convert from 'g' to m/s/s


%% Calculate linear velocity (integrate acceleartion)

linVel = zeros(size(linAcc));

for i = 2:length(linAcc)
    linVel(i,:) = linVel(i-1,:) + linAcc(i,:) * samplePeriod;
end

%% High-pass filter linear velocity to remove drift

order = 1;
filtCutOff = 0.08;
[b, a] = butter(order, (2*filtCutOff)/(1/samplePeriod), 'high');
linVelHP = filtfilt(b, a, linVel);

%% Calculate linear position (integrate velocity)
```

```
linPos = zeros(size(linVelHP));

for i = 2:length(linVelHP)
    linPos(i,:) = linPos(i-1,:) + linVelHP(i,:) * samplePeriod;
end

%% High-pass filter linear position to remove drift

order = 1;
filtCutOff = 0.08;
[b, a] = butter(order, (2*filtCutOff)/(1/samplePeriod), 'high');
linPosHP = filtfilt(b, a, linPos);

figure('Name', 'Sensor Data');
axis(1) = subplot(3,1,1);
hold on;
plot(linPos(:,1), 'r');
xlabel('Samples');
ylabel('X(m)');
axis(2) = subplot(3,1,2);
hold on;
plot(linPos(:,2), 'b');
xlabel('Samples');
ylabel('Y(m)');

hold off;
axis(3) = subplot(3,1,3);
hold on;
plot(linPos(:,3), 'g');
xlabel('Samples');
ylabel('Z(m)');
hold off;
linkaxes(axis, 'x');

% plot(linPos(:,3))
% scatter3(linPos(:,1),linPos(:,2),linPos(:,3));

% scatter3(linPosHP(:,1),linPosHP(:,2),linPosHP(:,3),'r');

% %% Play animation
%
% SamplePlotFreq =2;
%
% SixDOFanimation(linPosHP, R, ...
%                 'SamplePlotFreq', SamplePlotFreq, 'Trail', 'Off', ...
%                 'Position', [9 39 1280 720], ...
%                 'AxisLength', 0.1, 'ShowArrowHead', false, ...
%                 'Xlabel', 'X (m)', 'Ylabel', 'Y (m)', 'Zlabel', 'Z
(m)', 'ShowLegend', false, 'Title', 'Unfiltered',...
%                 'CreateAVI', true, 'AVIfileNameEnum', false, 'AVIfps',
((1/samplePeriod) / SamplePlotFreq));

%% End of script
```