# Sensor Data Fusion for full Arm Tracking using Myo Armband and Leap Motion

Eider C. P. Silva, Esteban W. G. Clua, Anselmo A. Montenegro
Federal Fluminense University
Computing Institute - MediaLab
Niterói, Brazil
E-mail:{eidercarlos, esteban, anselmo}@ic.uff.br

***Abstract*—The Myo Armband (Myo) is a sensor which can be attached to the forearm, and allows to send to devices equipped with Bluetooth muscular patterns of arm movements triggered by some gestures made by the hand. Also, it produces rotation data based on a gyroscope, accelerometer and magnetometer embedded on it. The Leap Motion (LM) is a sensor-based cameras and infrared lights, which allows accurately to track the hands motion, including fingers. Therefore, this work describes the use of sensor data fusion techniques to combine the data from both sensors in order to create a 3D virtual simulation in real-time of an arm motion, including the forearm, hand and fingers. The purpose of this combination is also improve the performance, aiming to overcome the main sensors' limitations when they are used individually, as occurs with Myo, which is limited to produce only forearm motion data and the LM, which has a limited field of view and range tracking. The experiment results shows that the studied algorithm is able to combines the data from both sensors achieving a considerable gain in the precision and tracking of the arm.**

***Keywords—Sensor Data Fusion; Arm Tracking; Myo Armband; Leap Motion.***

## I. INTRODUCTION

Research involving the arm motion tracking from human body limbs have become important in recent years. The search for improvements of this kind of tracking techniques in real-time have been increasingly intensified, mainly due to their applications in several areas, including medical, biomedical, human-computer interfaces, virtual reality, robotics, and others. This kind of research also boosted the search for the improvement of techniques based on sensors, which using the data as input to some algorithms based on probabilistic methods, can estimate in real-time the human body limbs position and/or orientation [1].

In addition to prediction models, other research widely used in this context is the data fusion from different kinds of sensors (multisensor). It aims to improve the accuracy, detection, authenticity, reliability and range. As the main algorithms for sensor data fusion, can be highlighted the Kalman Filter [2, 3], Particle Filter [4], Iterative Closest Point (ICP) [5], Bayesian filter [6] and Sequential Monte Carlo [7] Hence, experiments performed in our work are based on Kalman Filter, which is described in more detail in section IV.

One of the motivations for this research's development is its use in various areas of application and the search for improvements due to several challenging problems found in promoting the data fusion using different sensors. In [1] is described about some challenges which can arise when it is decided to implement data fusion from multiple sensors, as follows:

*a) Data imperfection:* data collected by sensors is always affected by some level of inaccuracies as well as uncertainty in the measurements;

*b) Outliers and spurious data:* uncertainties in sensors can arise from the ambiguities and inconsistencies present in the environment where data is being collected by the sensor;

*c) Conflicting data:* fusion of some kinds of data can be problematic, especially when the fusion system is based on evidential data. To avoid producing counter-intuitive results, the fusion data algorithms must treat conflicting data with special care;

*d) Data modality:* different data obtained from measurement of a phenomenon such auditory, visual, and tactile must be handled by a data fusion system;

*e) Data alignment/registration:* sometimes different sensor data must be transformed into a common frame before a fusion occurs. This alignment problem produces a calibration error. For this reason, data registration is very important to the successful deployment of the fusion systems;

*f) Data association:* the data association problem may come in the measurement-to-track and track-to-track association forms. The first one refers to the problem of identifying from which target each measurement is originated. The latter, deals with distinction and tracking combination, estimating the state of the same real-world target.

*g) Operational timing:* a well-designed data fusion system, especially when it is implemented for real-time applications, should deal with timing variations in data. For example, in distributed fusion settings, different parts of the data may traverse different routes before reaching the fusion center, which may cause out-of-sequence arrival of data.

There are other challenges in implementing this kind of system, and still there is no algorithm able to overcome all

2159-6662/16 $31.00 © 2016 IEEE
DOI 10.1109/SBGames.2015.24

these challenges at once. The works in literature are focusing on solving just some of them. Thus, the work described in this paper seeks mainly to solve the problems of "Outliers and spurious data" and "Data Association", since the combination of Myo and LM generate ambiguous data, and it is necessary to deal with different data with different coordinate systems.

In the next section is described the related work that gives the theoretical basis for the development of this research.

## II. Related Work

The research about track the human body limbs are somewhat related to computer vision, robotics and smart systems. In addition to these areas, the work described in this paper is associated with graphic computing, because the data processed by the algorithm is a rotation in quaternion and we intend to use a 3D avatar model for Unity3D in order to show in a graphic way the results of the data fusion processing.

The work written in [8], which is the most similar existing in the literature compared to our research, implements the sensor data fusion for hand tracking using Kinect (the first version for windows) and LM. In this work, the fusion of sensors is applied to an existing augmented reality system, aiming the treatment of phantom limb pain (PLP) in upper limb amputees. With Kinect, is acquired 3D images from the patient in real-time. These images undergo a post-processing to apply a mirror effect along the body surface, being shown in 3D and giving the illusion that it has two arms. The patient uses the virtually reconstructed arm to perform certain tasks involving interactions with virtual objects. As the returned coordinates are expressed by the two sensors at two different reference frames, it was developed a calibration method based on Corresponding Registration Set Point algorithm (CPSR) which calculates the rigid transformation required to align the two reference frames. Thus, recording the position and orientation from the LM in the Kinect's frame of reference, the system becomes able to detect accurately the interactions of the hand and fingers, working even when the hand is not visible to the sensor, improving the user experience.

In [8], is shown satisfactory results from tracking the hand and fingers motions, considering that both sensors use cameras and high precision infrared sensors. However, the tracking field of view from both sensors is limited, because the Kinect have just 43° in vertical position by 57° in horizontal, whilst the LM can track an area of 150°, but with an effective range of 25 to 600 mm (millimeters) above the device. However, due these limitations, the user interaction environment is limited. Then, with the sensor combination is increased considerably the scope and the tracking area. What differentiates in our work is the equipped Bluetooth technology in Myo, allowing to continue sending data in an effective distance of 15 meters [9], and even when it loses the hands and fingers detection, it will continue capturing the forearm movements.

There are several other works using the sensor data fusion system for tracking and estimation of human body limbs position. The research developed in [4] consists in a system that implements the combination of inertial and visual sensors in a probabilistic manner to capture the motions from an arm in 3D. For this, is used the Particle Filter (PF) algorithm, also known as Sequential Monte Carlo method (SMC). Other works, such as those described in [7, 2, 10, 5, 11] are using similar systems. In [7] is used SMC methods for estimating the position of a human arm as well. Meanwhile, in [2, 10, 5, 11} also are used data fusion algorithms and different types of sensors for track and predict the positions from the whole human body. The biggest difference that can be highlighted between them is that in [2, 10] it is used Kalman Filter to process the data provided by the sensors, which usually are accelerometers, magnetometers and inertials.

Yun and Bachmann used the Iterative Closest Point (ICP) algorithm [10]. This algorithm aims to combine two linked sets of points (which are on different coordinate systems) in order to compute the translation and rotation to transform the first in the second coordinate system. Therefore, to track the human body, the first set of points corresponds to the points provided by the sensors, and the second corresponds to the points that are on the surface of a rigid body, which in this work is a 3D model of the human body simulating the outcome of the data processed. In a similar way, was made in the work described by this paper.

The work described in [11] explains a very interesting research and slightly different from the others. Therefore, the authors implement their own data fusion system and applies the combination of videos, which contains orientation data with inertial sensors extended to improve and stabilize the motion tracking of the human body. The main contributions of this work is the mitigation of the ambiguity problem acquired by the inertial sensors and the improvement of the application performance alleviating the problem of getting incorrect positions extracted from videos.

Another interesting and different work is described in [12]. This paper presents a recognition system of arm and hand gestures using the Kinect sensor (version 1 for Xbox 360) and high-definition 3D cameras. The reason to implement this combination is that using just one Kinect sensor is difficult to recognize hand gestures at a distance greater than 1 meter, due the resolution and image quality be very low. For this purpose, an additional sensor is required. Thus, is also used a web-cam in HD able to recognize gestures at a greater distance. As the two sensor's field of view is similar, they capture equivalents images. Then, the Kinect is responsible for retrieving the position and distance of the hand from the images processed, which is used for the gestures recognition. So, it is important to note that the sensor data fusion system is made by the own author.

## III. Leap Motion and Myo Armband

The Leap Motion (https://www.leapmotion.com/), is a small device, with 8 cm, which can be connected to a computer via USB. It contains two infrared cameras (each with a capacity of 1.3 million pixels) and three built-in infrared LED emitters, being able to capture and track the movements from 10 fingers and 2 hands at a rate of 200 to 300 frames per second and generate a display pattern and interaction of 3D data from the captured information. The device's software parse and process the objects in its field of view with a range of 25 to 600 mm above the device with a cover area of 150°.

Browsing a web site, manipulating a map with zoom in and zoom out, draw with high precision and handle 3D complex data views, is between the many possible tasks that can be executed using LM [13].

Among the limitations of using the LM, we can highlight the sensor tracking field of view, which have a short range and does not allow the user make movements demanding a large space. The lighting also affect the device's performance, i.e. high lighting environments causes noise in the motion tracking.

According to [14], another limitation occurs when the user's palm is in a perpendicular position to the device's surface. This causes the sensor be unable to detect the hand. However, despite the disadvantages mentioned here, it was explored the maximum device's capacity to conduct this research, and what its API can offer.

The Myo (https://www.thalmic.com/myo/), is a electromyographic (EMG) device like a bracelet, which should be positioned below the user's elbow. According to [15, 16], it uses 8 EMG sensors to detect the electric potential produced by the skeletal muscles of a forearm. Therefore, based on its sensors, it is able to determine which muscles are active and this information is used to link the activation patterns of the muscles with the gestures made by the user - which are more easily triggered by certain gestures made by the hand. In total, there are 5 different predefined gestures, and moreover, it is equipped with a 3D gyroscope sensor, a 3D accelerometer and a magnetometer. So, it becomes possible the combination of the predetermined gestures with the properties of three sensors to create a variety of gestures controls.

One of the great advantages of using Myo is the data communication via Bluetooth in its version 4.0, allowing any device equipped with this technology, in addition to digital games, smartphone apps, drones, and others be controlled. In addition, we highlight the immersion it provides allowing to be set in the user's arm, making true the concept of ubiquitous computing. In this context, we see as a limitation, the fact that sensors cannot identify the hand and fingers motions. However, it justifies the need for the implementation of the sensor fusion with LM in our work.

## IV. Sensor Data Fusion: Kalman Filter

Boström et al. describe on the definition of data fusion based on different research sources. In this work, the author proposes that information fusion is "the study of efficient methods for automatically or semi-automatically transform information from different sources and different points in time in a representation that provides effective support for human or automated decision-making" [17].

The sensor data fusion is a powerful research area in robotics and intelligent systems. It has been extremely important for the development of autonomous vehicles. Also, it has been used in several other areas, and as shown in the related work section of this paper, is a multidisciplinary research area.

Described in the paper written in [18], the Kalman Filter is the most popular estimate algorithm. In a technical form, as outlined by [19], it is a linear, discrete time, finite dimensional time-varying system that evaluates the state estimate that minimizes the mean-square error. The Kalman Filter employs the use of mathematical models to dealing with real-time measurements. It is a recursive algorithm and works by making a prediction of the future, getting a measurement from the present state, and after making a comparison between the prediction and the measurement, and adjusting its next estimate based on this resulting moderated value. Its prediction of the future depends on the state of the present (that can be position, velocity, acceleration, etc) as well as the information about any controllable parts trying to affect the current state situation.

The Kalman Filter algorithm can be represented by a math formula divided into two steps (Prediction and Update) and each step is composed by some equations that will be explained in more detail as following. In the first step, the estimation of the previous iteration state is used to obtain a prediction of the current state. This prediction is also known as an "*a priori*" estimate because it does not include any information obtained from the current state observations. In the update step, the *"a priori"* prediction is combined with the current observation to enhance the state estimate. It is important remember that this algorithm in most cases is used to compute more than just one value, as example, this work is dealing with *Quaternions* which is composed by **X**, **Y**, **Z** and **W** values. It is very common to find in literature works about sensor data fusion and/or prediction dealing with 3D space, speed, acceleration among others measurements. For this reason, keep in mind that when dealing with above one value, the variables shown in the equations below are represented in the form of matrices, and then the calculations mainly involve matrix operations like multiplication, addition and subtraction. Sometimes is need the use of transpose ($^T$) and the inverse ($^{-1}$) of a matrix.

The prediction step is composed by two equations (1), (2) as shown below.

$$\mathbf{x}^- = F\,x + Bu \quad (1)$$

$$\mathbf{P}^- = F\,P\,F^T + Q \quad (2)$$

As shown in equation (1), the variable ($\mathbf{x}^-$) is the current state estimation and stores the values we want to get from the measurements. Note that the superscript ($^-$) is used to determine that the value is a prediction not an estimate. The variable (F) is the state transition matrix. Multiplying the state value (x) with this matrix is obtained the state of the prediction for the next iteration of the algorithm. This calcule is responsible for applying time conditions to the state value obtained from each algorithm iteration, i.e., the time interval spent to run each iteration. Then, the algorithm is able to predict future values. (Bu) is respectively the input control matrix that apply the vector control inputs (u) parameters in the state vector (x). However, these variables (Bu) are not being used in this work because there is no control values

emitted by the sensors, just the orientation values related to the forearm.

The Equation (2) is responsible to determine the prediction covariance that infers the error quantity (noise). Hence, ($\mathbf{P}^-$) is the estimate of the average error value for each variable defined in the state matrix. In (Q), is defined the covariance matrix of the noise processes. In this work, that matrix is defined manually and is responsible for telling the algorithm the amount of errors that can be produced by the sensors. It is important note that for the first time executing the algorithm, all the values in the current state variable (x) are defined as zero, and then, the results of ($\mathbf{x}^-$) will have different values than expected. Moreover, the value obtained in ($\mathbf{P}^-$) is based just in static values. However, during the algorithm execution, these values are gradually adjusted.

Next, is shown the equations that composes the update step.

$$\mathbf{y} = z^- Hx^- \quad (3)$$

$$\mathbf{S} = HP^-H^T + R \quad (4)$$

$$\mathbf{K} = P^-H^T S^{-1} \quad (5)$$

$$\mathbf{x} = x^- + Ky \quad (6)$$

$$\mathbf{P} = (I - KH)P^- \quad (7)$$

The equation (3) compares the data collected by the sensors with the prediction made in the previous step. This way, the values stored in the matrix (z) are exactly the measured data collected by the sensors. ($Hx^-$), known as the measurement function, is responsible to put the values from (x) in the measurement space, i.e. in a unity and base equivalent to the measurements made by the sensors. This calculation is made to compare the observed value with the expected value, the result is then assigned to the (**y**) matrix, called in some works in the literature as measurement residual.

Following the algorithm execution, in the Equation (4), the variable (R) is the measurement noise, and ($P^-$) is the uncertainty covariance matrix from the prediction step. The linear equation ($HP^-H^T$) is taking the covariance ($P^-$) and putting it in the measurement (H) space like the way is made in equation (3) ($Hx^-$). Then, once in measurement space, the measurement noise (R) can be added to this equation.

In the equation (5), the matrix inverse ($S^{-1}$) is a linear algebra's way of computing 1/S.

The computation described in equation (6) just multiplies the measurement residual (**y**) by the *Kalman gain* and adds it to the state variable, that is, the computation of the new estimate.

Finally, in the equation (7), the variable (I) means an identity matrix, and it is the way which (1) is represented in multiple dimensions. The variable (H), as we have described in others equations above, is the constant measurement function. Then, (K) is the ratio of how much prediction versus the measurement is used. So, if (K) is larger, the result of (I-KH) is small, and (P) will be smaller than it was in the previous iteration. On the contrary, it will be larger. Therefore, the size of uncertainty is adjusted by some factor of the *Kalman gain* (**K**).

In summary, the equation (1) is responsible for the outcome of the current state prediction; and (2) is responsible for determine the prediction covariance which infers the number of errors (noise). Whilst in the update phase, the equation (3) compares the data collected by the sensors with the prediction made previously. In equation (4) it is made a comparison from the margin of error to the amount of expected errors. Then, the equation (5) updates the value provided according with the error value, given in equation (4), which is not only the prediction, but an amount equals the last sensor measuring. Thus, the status is updated in equation (6), and the (7) gives a new estimation error to be used in the next algorithm iteration. As can be noticed, these equations are executed in sequence and recursively.

## V. EXPERIMENTS

The experiments was conducted using the Unity3D engine for game development in the version 5.0, which is a very well known engine, and one of the most widely used in development of digital games. Also it was used C# language for implementing the data fusion algorithm.

Besides that, it was used in the experiments a 3D model consisting of an avatar produced by Ivan Bindoff, available for download in Unity3D Assets Store page at (https://www.assetstore.unity3d.com/en/). Also, it can be accessed by the author's website at (http://www.ivansapps.com/wordpress/ ?p=110).

The 3D model have all the human skeleton articulations. The arm and forearm are endowed of 3DoF (three degree of freedom) composed by pitch, yaw and roll rotations. Moreover, the hands are able to do all the possible hands motions. Hence, this model allows receive the data transmitted by the sensors and simulate the human movements. The complete body model is used because it promote more immersion and it is intended to use in future works. However, for the experiments made in this work are just being used the right arm, including the forearm, hand and fingers.

During the tests execution, the way that the LM device configuration was used, is different from the conventional - on a plane surface. As shown in Fig. 1 below, it was used in a head-mounted way, to easily allow the sensor data fusion and a "first person" interaction, mainly because we intend in future to apply it in a virtual reality application, viewing the results with the Oculus Rift.

Fig. 1. Myo Armband and Leap Motion integration settings for data fusion.

As we can see in the Fig. 1 above and knowing that the LM's field of view is limited, the application was configured to when the arm is out, use just the Myo rotation data. Then, when the arm is in the field of view, it is activated the Kalman Filter algorithm, fusing the (X) and (Y) rotation values from both sensors. Thereafter, the resulting values from the fusion are applied in the forearm. The (Z) value is not fused, because the Myo rotation around this axis is not suitable, mostly due the fact that the human forearm rotation in its own axis near the elbow is very short. However, the LM rotation around the own forearm axis is well performed.

The Kalman Filter algorithm, was written based in the the book [20]. The pseudocode is described in Code 1, as the following:

```
01 function init_kf(){
02  dt = 0.02;
03  X = new Matrix(8,1);
04  P = new Matrix.Identity(8);
05  Q = new Matrix.Identity(8)*(0.1);
06  F = new Matrix{
07  {1,0,0,0,dt,0,0,0},
08  {0,1,0,0,0,dt,0,0},
09  {0,0,1,0,0,0,dt,0},
10  {0,0,0,1,0,0,0,dt},
11  {0,0,0,0,1,0,0,0},
12  {0,0,0,0,0,1,0,0},
13  {0,0,0,0,0,0,1,0},
14  {0,0,0,0,0,0,0,1}
15  }
16
17  H = new Matrix{
18  {1,0,0,0,0,0,0,0},
19  {0,1,0,0,0,0,0,0},
20  {0,0,1,0,0,0,0,0},
21  {0,0,0,1,0,0,0,0},
22  {1,0,0,0,0,0,0,0},
23  {0,1,0,0,0,0,0,0},
24  {0,0,1,0,0,0,0,0},
25  {0,0,0,1,0,0,0,0}
26  }
27
28  R = new Matrix.Identity(8)*(0.3);
29  Z = new Matrix(8,1);
30  Y = new Matrix(8,1);
31  I = new Matrix.Identity(8);
32 }//Finish init_kf()
33
34 function predict(){
35  X_new = F * X;
36  P_new = (F * P * F.Transpose ()) + Q;
37 }
38
39 function update(Matrix Z){
40  Y = Z - (H * X_new);
41  S = (H * P_new * H.Transpose())+R;
42  K = P_new * H.Transpose() * S.Inverse();
43  X = X_new + (K * Y);
44  P = (I - K * H) * P_new;
45 }
46
47 function KalmanFilter(){
48  Z[1,1] = myo.transform.rotation.x;
49  Z[2,1] = myo.transform.rotation.y;
50  Z[3,1] = myo.transform.rotation.z;
51  Z[4,1] = myo.transform.rotation.w;
52  Z[5,1] = hand.GetArmRotation().x;
53  Z[6,1] = hand.GetArmRotation().y;
54  Z[7,1] = hand.GetArmRotation().z;
55  Z[8,1] = hand.GetArmRotation().w;
56
57  predict();
58  update(Z);
59
60  if(myoWorking && lmWorking) //Data Fusion
61  {
62      rightForeArm.rotation = new
Quaternion(X[1,1],X[2,1],Z[7,1],Z[8,1]);
63  }
64  else if(myoWorking)
65  {
66      rightForeArm.rotation = new
Quaternion(Z[1,1],Z[2,1],Z[3,1],Z[4,1]);
67  }
68 }
69
```

Code 1. Kalman Filter Pseudocode.

It is important to note in Code 1 above, that the matrix dimensions is generally (8x1) or (8x8). This dimension is to adjust all the computing to the matrix that represents the state, which stores 4 quaternion (x, y, z and w) values from the Myo and the same from LM.

In relation to its execution, the algorithm represented by the pseudocode in Code 1 must be initialized (calling the function *init_kf()* into a Unity3D function named by *Start()* or *Awake()*. Then, the *KalmanFilter()* function is put into the *LateFixedUpdate()* function. The last one, like all the initialization functions in Unity, have the execution schedule following a hierarchy. In this case, it is executed after all animations and in a constant time interval, running at each 0.02 seconds. This value is assigned to the **dt** variable in line

2. Hence, the execution in a regular execution time interval make possible to predict the future values. In the section IV, specifically when it is described about the equation (1) it is explained in more details about the effects of **dt**.

Following to the line 47, each time the *KalmanFilter()* function is executed, the rotation data is obtained from the sensors (the measurements). Then, is called the *predict()* function, which tries to predict the current state. After that (in line 58), the *update()* function is fed by the new measurements and the current state is updated based on the prediction and the measurements. With the estimate ready, it is stored in the variable (X). If the user's forearm is being covered by the LM's field of view, it is used the data fused values (X) to control the forearm (line 60). On the contrary, the data obtained from Myo takes the control (line 64).

## VI. Results and Discussion

In order to test the performance of the Kalman Filter algorithm, it was performed a test (video available in https://www.youtube.com/watch?v=EO4W-BJP888) with a user performing rotations around the x-axis of the arm. During the test execution, the rotation data from both sensors and the results from the data fusion was transformed from quaternion to Euler angles and saved in a text file. Then, we get some of data obtained at the moment that was running the fusion, in order to verify the performance of the algorithm, as shown in the chart (Fig. 2) and in the Table I below.

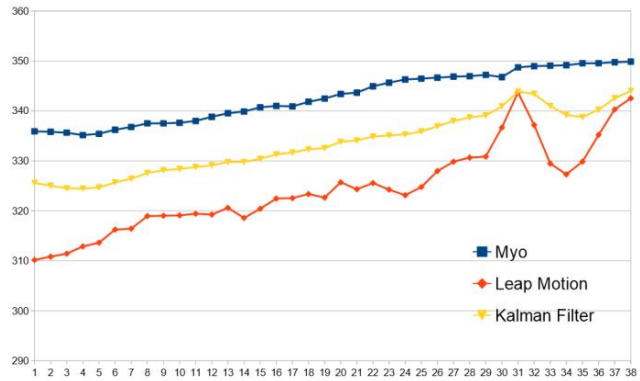


Fig. 2. Performance results of the data fusion using Kalman Filter.

TABLE I.

| Iterations[a] | Myo (E. a.) | LM (E. a.) | KF (E. a.) |
|---|---|---|---|
| 29 | 347,2204 | 330,8667 | 339,0986 |
| 30 | 346,7629 | 336,674 | 340,902 |
| 31 | 347,725 | 343,7002 | 343,8728 |
| 32 | 348,9605 | 337,165 | 343,3954 |
| 33 | 349,0514 | 329,4626 | 340,9944 |
| 34 | 349,1506 | 327,3166 | 339,1842 |
| 35 | 349,5265 | 329,8595 | 338,7618 |
| 36 | 349,5399 | 335,2146 | 340,2407 |
| 37 | 349,7448 | 340,3015 | 342,5309 |
| 38 | 349,8634 | 342,5358 | 344,0095 |

[a] Inputs (Myo and LM measurements in Euler angles) used to feed the Kalman Filter algorithm and the outputs (KF in Euler angles).

The Fig. 2 shows 38 iterations of the Kalman Filter execution. This iterations was saved during the tests where the user was performing the arm rotation around the x-axis, and when the forearm was being tracked by the LM, allowing the data fusion. The Table I is a chart's complement and shows in detail the saved values from the iteration 29th to 38th, where can be seen an interesting algorithm's behavior. From the 29th iteration, the LM rotation value get increased drastically, then after the 31th iteration, it get decreased and even with this sudden changing of values, as we can see clearly in the chart, the filter results are trying to alleviate and keep the average between the values emitted by both sensors.

## VII. Conclusions

### A. Overview

Based on the data displayed in the chart Fig. 2, confirming the values shown in Table I, we can clearly see that the Kalman Filter algorithm can establish an average of the values produced by Myo and LM. Based on these results we have verified the Kalman Filter's efficiency to be used as an algorithm to sensor data fusion, and it shows be very important to solve problems like arm tracking using Myo Armband and Leap Motion, being efficient and smoothing the motion when there is a noise produced by the sensors.

Even with the advantages described above, the algorithm is still in its basic version and we must to dealing properly and dynamically with the noise. As can be seen watching the test video (https://www.youtube.com/watch?v=EO4W-BJP888) the forearm motions have still a little salt when the LM starts the tracking.

### B. Future Works

Face of these results, our main goal is to improve the algorithm to deal with the noises properly and complete the full arm motion tracking. One interesting thing to do in the next step is to use other data fusion algorithms found in the literature, in order to compare the efficiency of each to solving this same problem. We also want to use this work in a virtual reality application, using Oculus Rift to improve the immersion and the visualization.

## VIII. Aknowledgments

The authors gratefully acknowledge CAPES for the financial support of this work.

## REFERENCES


[1] B. Khalegui, A. Khamis, F. Karray , and S.N. Razavi, 2013. Multisensor data fusion: A review of the state-of-the-art. *Information Fusion 14,* 1, 28-44.

[2] R. Zhu and Z. Zhou, 2004. "A real-time articulated human motion tracking using tri-axis inertial/magnetic sensors package". Neural Systems and Rehabilitation Engineering, IEEE Transactions on 12, 2, 295-302.

[3] G. Ligorio, and A. M. Sabatini, 2013. "Extended kalman filter-based methods for pose estimation using visual, inertial and magnetic sensors: Comparative analysis and performance evaluation". Sensors 13, 2, 1919–1941.

[4] Y. Tao and H. Hu, 2008. "A novel sensing and data fusion system for 3-d arm motion tracking in telerehabilitation". Instrumentation and Measurement, IEEE Transactions on 57, 5, 1029– 1040.

[5] S. Knoop, S. Vacek, and R. Dillmann 2006. "Sensor fusion for 3d human body tracking with an articulated 3d body model". In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, IEEE, 1686–1691.

[6] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello 2003. "Bayesian filtering for location estimation". IEEE pervasive computing, 3, 24–33.

[7] T. B. Moeslund, and E. Granum, 2003. "Sequential monte carlo tracking of body parameters in a sub-space". In Proceedings of the IEEE International Workshop on Analysis and Modeling of Faces and Gestures, IEEE Computer Society, Washington, DC, USA, AMFG '03, 84–91.

[8] B. Penelle, and O. Debeir, 2014. "Multi-sensor data fusion for hand tracking using kinect and leap motion". In Proceedings of the 2014 Virtual Reality International Conference, ACM, 22.

[9] Mark, 2015. "What is the bluetooth range"? https: //support.getmyo.com/hc/en-us/articles/202668603-What-is-the-Bluetooth-range-, July.

[10] X. Yun, and E. R. Bachmann, 2006. "Design, implementation, and experimental results of a quaternion-based kalman filter for human body motion tracking". Robotics, IEEE Transactions on 22, 6, 1216–1227.

[11] G. Pons-moll, A. Baak, T. Helten, M. Muller, H. P. Seidel, and B. Rosenhahn, 2010. "Multisensor-fusion for 3d full-body human motion capture". In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, IEEE, 663–670.

[12] M. Caputo, K. Denker, B. Dums, and G. Umlauf, 2012. "3d hand gesture recognition based on sensor fusion of commodity hardware". In mensch & Computer, vol. 2012, 293–302.

[13] L. M. D. Portal, 2015. "Leap motion developer portal - api overview". https://developer.leapmotion.com/documentation/cpp/devguide/Leap_Overview. html, July.

[14] L. E. Potter, J. Araullo, and L. Carter, 2013. "The leap motion controller: a view on sign language". In Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration, ACM, 175–178.

[15] J. Cheney, and D. Ancona, 2014. "Gesture controlled virtual reality desktop".

[16] K. Nymoen, M. R. Haugen, and A. R. Jensenius, 2015. Mumyo– evaluating and exploring the myo armband for musical interaction.

[17] H. Boström, S. F. Andler, M. Brohede, R. Johansson, A. Karlsson, J. Vanlaere, L. Niklasson, M. Nilsson, A. Persson, and T. Ziemke, 2007. "On the definition of information fusion as a field of research".

[18] R. E. Kalman, 1960. "A new approach to linear filtering and prediction problems". Journal of Fluids Engineering 82, 1, 35– 45.

[19] M. I. Ribeiro, 2004. "Kalman and extended kalman filters: Concept, derivation and properties". Institute for Systems and Robotics 43.

[20] R. R. Labbe, 2015. "Kalman and bayesian filters in python". http://github.com/rlabbe/ Kalman-and-Bayesian-Filters-in-Python, July.