

Distributed Wireless MEMS Networks

A software approach.

Tim Langens and Jeroen Doggen.

Abstract—Today many position aware devices are demanded. Micro-Electro-Mechanical Systems (MEMS) can detect their own acceleration. Out of these values the movement, and thus the new position, can be calculated. The problem of these systems is, they are very error-prone. In order to minimize the fault this paper introduces a new way of combining different MEMS measurements to achieve a more accurate relative positioning in regard of the object. This is done by combining measurements of different sensors. All data communication will be wireless over the IEEE 802.15.4 standard.

Index Terms—MEMS, WSN, wireless, sensor, network.

1 INTRODUCTION

MOVEMENT is difficult to observe with sensors. Micro-Electro-Mechanical Systems (MEMS) are sensors that can detect their acceleration. These values then can be converted into movement. The issue is these sensors are very error-prone and can drift easily. The source of the drift can be defined to their highly sensitivity grade to changes in the surrounding environmental conditions due the small size and the fabrication process [1].

MEMS sensors are very precise, some even to 1mG. 1mG acceleration can be translated into 0.06 degrees of tilt angle. This is very accurate but unfortunately it works only while not moving. When the MEMS accelerator measures a fault of 18mG this results in a tilt angle of 1 degree which means an error of 300 meter in one minute [2]. Combine this knowledge with the fact that some chips can drift 5 degrees per second [3] and thus result in errors with a margin of 1500 meter. This means that solutions to calibrate the MEMS chips regularly has to be adopted. Many different options exist to minimize this problem but all have their downside.

One solution is the use of an infrared light array. These lights are placed on a none-moving object and are used as a beacon. The sensor knows the place of this light array and calibrates itself upon it. This technique is used in the well-known Nintendo wii console[4]. The downside of this technique is that the light array can not be moved and therefore limits the usage of the technique itself.

Other techniques combine MEMS with a GPS[5] or with a gyroscope[2]. The biggest disadvantages of these techniques are the high-cost and the size of these systems.

Our solution will combine the macroscopical properties of the object with the MEMS data. By choosing for this solution we do not need any other hardware except the MEMS-sensors and the network hardware. Our test system will use the SunSPOT nodes. These wireless sensor network (WSN) nodes communicate over the 802.15.4 standard and send their data wireless to a base station. This gives the opportunity to easily integrate it in many applications. The combination of the data of the different MEMS-nodes and the macroscopical properties can happen on two different places. A first solution would be embedding the calculations in the nodes hardware. This path

has been researched in [6]. The other way is implementing the calculations in the software, where the data is gathered. This paper will handle the last concept.

2 FILTERING THE DATA

In the previous chapter we discussed the difficulties that were encountered with MEMS accelerators. The biggest problem of MEMS accelerators is they are very error-prone. Filtering the data can already diminish the error. In our research we tried to implement different filters to diminish this error.

The first filter is a Kalman filter. This adaptive filter is widely used to implement a Bayes filter. The Bayes process and thus the main process of the Kalman filter is showed in figure 1. The Kalman algorithm calculates a belief (x) represented by its mean and covariance at a certain time (t). This belief is calculated by updating the previous belief. For this update process the Kalman filter requires control (u) and measurement (z) values [7]. This results in two different equation groups for Kalman filters. The measurement update equations, who are used for the update, estimate an improved a posteriori belief. One can consider this value as a corrector for the previous a posteriori belief. The time update equations estimate a certain process in the future and can estimate the a priori states for the next time step. This equation can also be regarded as the predictor. By combining these two values the algorithm of the Kalman filter resembles a predictor-corrector algorithm. [8]

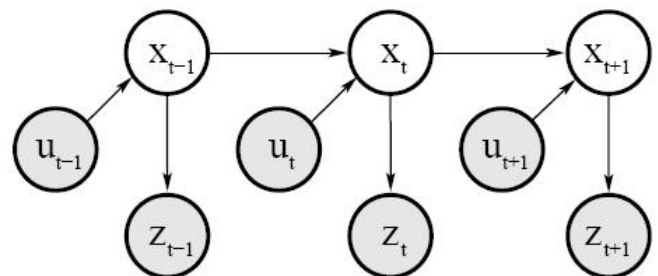


Fig. 1. Schematic representation of the Bayes process

Though the filter corrects itself after some time the problem with Kalman filters is that they are based upon beliefs. This can result in wrong conclusions and thus result in errors [9]. This is why the second filter is implemented. The second filter

• T. Langens and J. Doggen are with the University College of Antwerp, Paardenmarkt 92, 2000 Antwerp, Belgium
E-mail: tim@langens.eu, j.doggen@ha.be

will combine the data of the different MEMS modules with the information about the macroscopical properties, mainly the limitations, of the object where the MEMS accelerators are placed upon. The idea is that this second filter will filter out wrong beliefs and conclusions.

The big issue here is which data to combine. The accelerations filtered by the Kalman filter can be combined with the possible accelerations. Converting accelerations into movement and movement into accelerations is done with integrations which brings out an other problem. Because of the fault that appends with integrations, the integrations need to be limited to a minimum. Using the given method integrating has to be done twice. First the possible movements have to be converted in possible accelerations to use with our filter. After the accelerations are filtered all the accelerations still needs to be converted to movement. This means not only the filter uses wrong control data, also new faults are allowed into our data by integrating the data after our filters.

The logical solution is to implement the macroscopical filter after the conversion of the acceleration into movement. This solves both mentioned problems because we try to backfire the fault created by converting the accelerations, and we do not have to integrate our control data. The consequence is we can not combine the two filters into one filter and use the output of the second filter as input for the Kalman filter. Both filter processes are shown in figure 2.

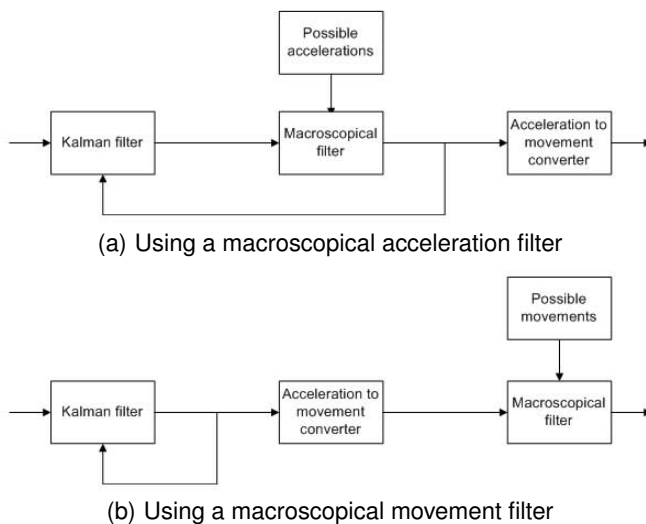


Fig. 2. The different filter processes

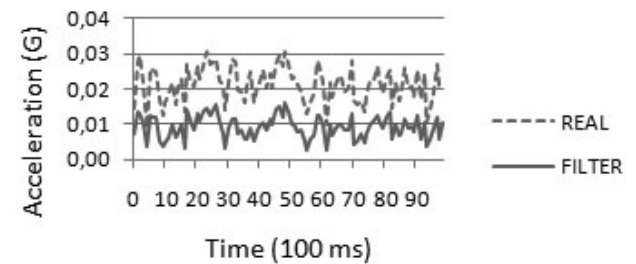
By comparing the measurements of the different MEMS accelerators, drifting MEMS accelerators can easily be detected and recalibrated. Different techniques can be chosen to detect the drifting accelerators. The most natural technique would be by excluding the drifting accelerator of the equation. This method which can be called the minority report method is easy to use but caution has to be taken. When more accelerators have a small drift in the same direction, the right measurements can cause the minority report and will be excluded of the equation. This results in wrongly recalibration of the right accelerators which will enlarge the error.

This problem solves itself by using more sensors. We can assume that the drifting error of the sensors is Gaussian defined with a mean of zero. This is just a general statistic theory which says that the more sensors are used the higher the probability

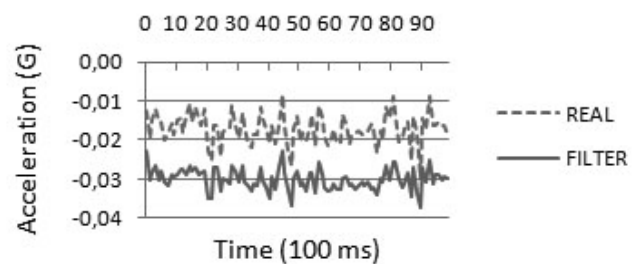
that the sensors have a different drift [7]. The only thing we still have to research is how many sensors are needed to accomplish this theory and to recalibrate automatically drifting sensors without enlarging the error.

2.1 Using Kalman Filters

Already mentioned is the adjusting work of a Kalman filter. We tried to implement this filter to decrease the driftproblem. Our results were quite remarkable. We discovered that by applying a Kalman filter the measurements sometimes improved but sometimes the problem enlarged. This can be seen in figure 3. Remember, when one wants to see how large the drift is, one has to look at how the drift fluctuates regarding to zero. The farther from zero, the bigger the drift.



(a) Spot A



(b) Spot B

Fig. 3. The graphs of the original and filtered data

We have to state that the Kalman filter we used was a first order Kalman filter. Probably the error would diminish by implementing a third or fourth order Kalman. Because we want to investigate the effect of combining data of MEMS sensors, we did the further research without a Kalmanfilter.

3 CALCULATING THE POSITION

A sunspot can detect 6 different values. The tiltangle in the three directions (x,y,z) and the acceleration in the same three directions. If we start reading this values we directly see that the drift is a big issue for the sensors. However, the biggest problem is the constant gravitation of the earth. This force of 1G can not be neglected that easily. By keeping the sensor horizontal there would not be any problem because then the force will be constant at the z-axis. The moment the sensor is tilted with regard to the ground the force will be divided over different axes. To avoid this kind of problem we decided to start our research in 2D.

3.1 Acceleration, Velocity and Position

To calculate position out of acceleration, the relation between acceleration, velocity and position has to be known. Acceleration can be defined as $a(t)$, velocity as $v(t)$ and position as $x(t)$. The function of acceleration is as follows:

$$a(t) = \frac{dv}{dt}$$

If the theorem of calculus is applied onto this function the function will look like:

$$\int_{T_1}^{T_2} a(t) dt = v(T_2) - v(T_1)$$

To calculate the new velocity this equation has to be written in function of the new velocity.

$$v(T_2) = v(T_1) + \int_{T_1}^{T_2} a(t) dt$$

Now the new position can be calculated out of velocity. The function of velocity is the following:

$$v(t) = \frac{dx}{dt}$$

Applying the theorem of calculus upon it.

$$\int_{T_1}^{T_2} v(t) dt = x(T_2) - x(T_1)$$

And the new position can thus be calculated with:

$$x(T_2) = x(T_1) + \int_{T_1}^{T_2} v(t) dt$$

or to calculate the position directly:

$$x(T_2) = x(T_1) + \int_{T_1}^{T_2} (v(T_1) + \int_{T_1}^{T_2} a(t) dt) dt$$

Luckily discrete numbers are used and thus the calculations can be done without integrations. Acceleration is represented in G or m/s^2 . The accelerations of the SUNspots are in G but converting G to m/s^2 is easily done:

$$1G = 9.81m/s^2$$

we also know that velocity is represented in m/s and position in m. This simplifies the calculations to:

$$x(t) = a(t) \times t \times t$$

The logical to go is by implementing this into our solution, and we calculate our new position. Unfortunately, the drift again causes big issues. If we stall the sensor the drift already gets converted into movement, and it looks like the sensor is trully slowly drifting away from the original position. We thought this could be solved by combining the position of different sensors but the main issue is that small, real movements and even big, slow movements dissolves in the drift.

4 COMBINING ACCELERATIONS

The original idea was to combine the calculated movement and adjust it with respect to the macroscopical properties of the object to solve the drift problem. Though when some movements already dissolves into the drift, this faults will be neglected and the preciseness of the data will be even less. We can conclude that we have to do something before we convert the acceleration to the new position. The new idea is

to combine the measured accelerations before converting them to movement and the new position. After we combining the accelerations, they can be converted to movement and position and the macroscopical properties to readjust the data some more can be used.

4.1 Constant Drift

The first question in combining the accelerations is about drift. We have to know how the behavior of the drift evolves. To know this we took several sunspots and put them still on a table. By sending the accelerometer data lying still we know how this drift evolved in time. To get a precise number of measurements we did this for 2 minutes. The results of this test can be found in figure 4.

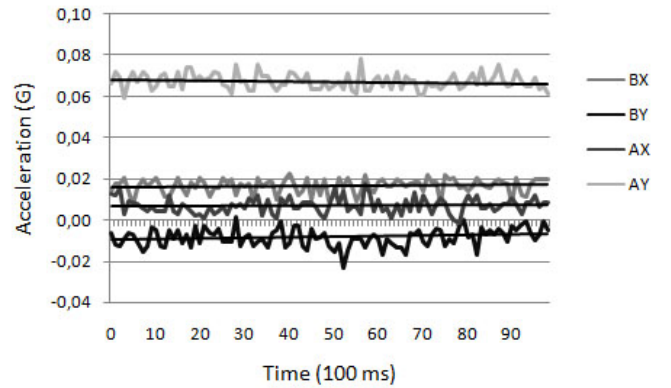


Fig. 4. Drift analysis

As can be seen, the drift is almost constant. Most sensors drift slightly decreases. Comparing the drifts with each other states that the drift is dependent to the spot and its axis. This conclusion opens the possibility to investigate a technique of taking the mean of some sensors to minimize the drift.

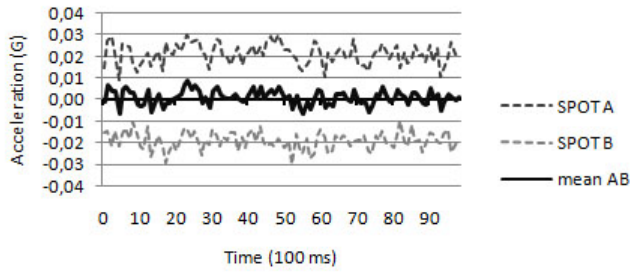
4.2 Mean Acceleration

In order to decrease the drift we are going to calculate the mean drift of different sensors. We already know that the drift is more or less constant per sensor and per axis. by combining the drift of different spots per axis and calculate their mean the drift error may decrease without filtering the real accelerations.

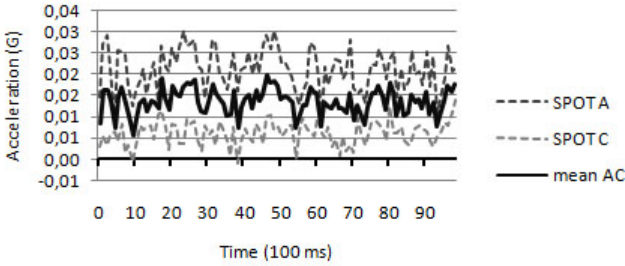
The main idea of this technique is based upon the fact that the drift of the various spots is not the same and not in the same direction. By using different sensors to measure the possible acceleration with an almost inverse drift, the drifts will lift up each other. A real acceleration will do the opposite, because the acceleration will be in the same direction for all spots. This theory limits the choice of sensors, since the used sensors need to have the same constant drift in opposite directions. Examples of this can be viewed in figure 5

To solve this issue more then two sensors can be used. We researched the evolution of the mean when using two to six sensors. By using more sensors the possibility that the drift is equally scattered in the two directions enlarges and the drift problem gets smaller.

The big problem rises with the use of measurement on more axes. The chance that one will find two sensors that have an opposite drift on two or even three axes is very small. And the more sensors one uses the more expansive the whole solution



(a) Good mean result



(b) Bad mean result

Fig. 5. Combining the drift with two sensors

becomes and the more non-practical the implementation will be. But also for this problem, we have come up with a solution.

4.3 Weighted Mean Acceleration

Not finding two sensors that are opposite identical in drift is a major problem in compensating the drift error. By giving a weight to the different measurements, this problem can be solved, and the identical opposite drifts can be created. The weight to apply onto these values depends on two factors.

One factor is the sign factor. The best solution would be a negative and a positive drift. To secure this factor on the different axes three sensors will be used. Then the possibility is bigger that one of the three is drifting in the opposite direction of the other on every axis. It could be possible that this is a different sensor each time.

The other weight factor is the share of the sensor in the total drift in function of the other sensors. To calculate this factor the way the drift of each separate sensor evolves needs to be analyzed. To do this first a certain amount of samples for each sensor is taken at the same time. Knowing that the drift is more or less constant, a constant per sensor for its drift can be calculated by taking the mean of the samples per sensor. This mean is a nice value to start comparing the different drifts.

the outcome is 6 constant, 2 per sensor. These constants can be represented as:

$$mA_x, mA_y, mB_x, mB_y, mC_x, mC_y$$

Since the calculation of the weights is identical for each axis, we are going to describe it only for the x-axis. After calculating the mean for each sensor, their share in the total drift is found by comparing them with each other. To do this their absolute value is used. For easy representation these absolute values are represented with A, B and C.

$$A = |mA_x|, B = |mB_x| \text{ and } C = |mC_x|$$

The next step is calculating their total drift (δ) 2 by 2.

$$\delta AB = A + B$$

$$\delta AC = A + C$$

$$\delta BC = B + C$$

To get a weight for each sensor and to adjust this weight so low drifting sensors have a higher weight then large drifting sensors, the relative portion each sensor has in the total drifts is needed. This share is, of course, zero when the sensor is not a part of the total.

for δAB :

$$\%A_{ab} = 1 - \frac{A}{\delta AB}$$

$$\%B_{ab} = 1 - \frac{B}{\delta AB}$$

$$\%C_{ab} = 0$$

The same process is done for the other total drifts.

Now there are 9 weights with a total of 300 percent. The total weight for each sensor can be the sum of its 3 weights or to calculate the mean easily, just the mean of its weights. for sensor A the weight will be:

$$\%A = \frac{\%A_{ab} + \%A_{ac} + \%A_{bc}}{3}$$

This weight for each sensor is only depending onto one factor, and can be called the relative weight. To implement the other factor in the weight, the ratio between the total positive and total negative drifts has to be known. These means are in absolute values and represented as:

$$\delta P \text{ and } \delta N$$

Calculating their ratio is easily done with following calculations.

$$\%\delta P = 1 - \frac{\delta P}{\delta P + \delta N}$$

$$\%\delta N = 1 - \frac{\delta N}{\delta P + \delta N}$$

The only problem with this calculation is that when all the sensors drift in the same direction the outcome will be 0. Therefore the corresponding ratio has to be adjusted to 1 when this happens.

The final weight (W) per sensor can be obtained by multiplying its relative weight with the ratio of its sign. So, if sensor A has a positive drift, the weight for sensor A will then be:

$$W_a = \%A \times \%\delta P$$

if sensor A would have a negative drift the weight for sensor A will be:

$$W_a = \%A \times \%\delta N$$

The result of this implementation can be found in figure 6. The most remarkable result was that using a weighted mean is mostly better then using the normal mean, but sometimes the normal mean is slightly better. The cause is that when the drift of the different sensors is already well spread in opposite directions the normal mean is almost the same as the weighted

mean as can be seen in 6(c) and 6(d). Therefore we can conclude that using our weighted mean is a better solution then using the normal mean.

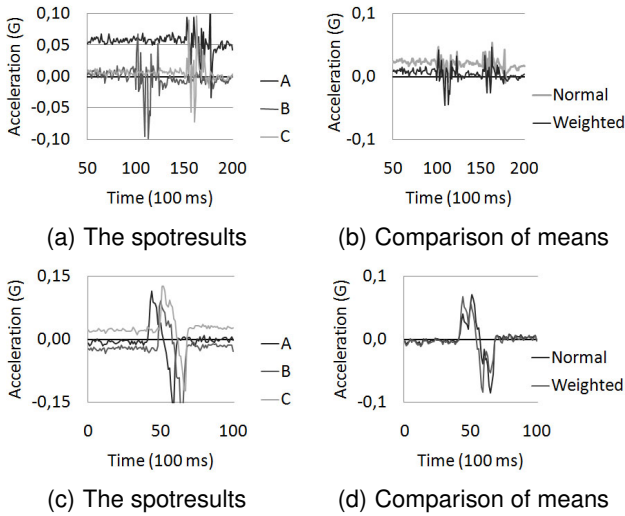


Fig. 6. Combining the drift with three sensors

4.4 Time Synchronization

Time is a very important factor in analyzing data. In figure 6 we see that the time is not well synchronized. Synchronizing the results of this test could give even better results, because now it sometimes looks like there are two small accelerations at different times in stead of one big acceleration. This can be seen in 6(a) and 6(b). Here spot C's results are used before the results of spot A and B are entered.

We had to start researching why synchronization was failing. We learned that the problem was made by lost packets in the network protocol. The transmitted message contains the time of sending, an ID to know which sensor is the owner and acceleration data measured during approximately 100 milliseconds. These 100 milliseconds do not only imply that a packet is send every 100 milliseconds, but it also means that the other sensors have to send their data in these 100 milliseconds.

We used this information to solve the synchronization problem. This is done by keeping track of the time stamp in the message. By using a time interval of 100 milliseconds, the time used to take samples, one can compare the data of the different sensors in this time interval. If a packet is lost the host neglects the owner of the packet and uses only the other received data. The results are shown in figure 7

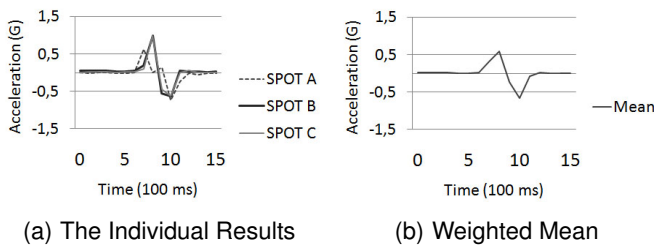


Fig. 7. Combining the drift with two sensors

4.5 Adjusting the Weights

We always started from the assumption that the drift was more or less constant. Figure 4 shows that the drift slightly changes in a good or bad way. Especially after an acceleration the drift can slightly change. This way using a constant weight is a short-term solution. To comply the method of weighted means to a long running implementation the weights have to be adjusted in time.

Drift always occurs even during accelerations. This means that after initializing the first weights we can continue recalculating the weights without regarding real accelerations. One have to keep in mind that the amount of samples used to calculate the weight has to be well-chosen. When using to many samples the weight will change very slowly and sudden changes in drift will be detected after a substantial amount of time, while choosing a scarce number of samples the weights will greatly differ and the mean will be closer to the non-weighted mean. The amount we chose for the weight was 50 samples. Remember that time intervals of 100 milliseconds are used and thus the weight is calculated on the drift of the last 5 seconds.

The result of the test can be viewed in figure 8. The graph shows how the drift changes after an acceleration and how the adaptable weight adjusts itself in time to filter the drift.

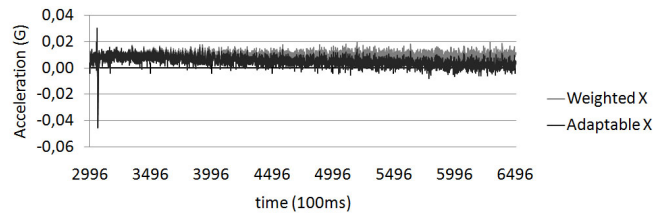


Fig. 8. Drift analysis

4.6 Determining the Position

The first intention of the master's thesis was to investigate a filter that uses macroscopical properties of an object to analyze data and correct it where needed. At the start of our research we soon discovered that developing such a filter was not possible due to the high drift and accelerations dissolving into this drift. We had to solve the drift problem before implementing a macroscopical filter, so we directed our research to a method to diminish the drift, the adjustable weighted mean method. The only remaining question still is if this filter can help determine the position with only MEMS sensors.

To answer this question the new position simply has to be computed out of the filtered acceleration. This is done by the method described in 3.1. The only thing not yet mentioned is that starting a movement is an acceleration, but also stopping the movement is an acceleration. In theory these two accelerations are the exact opposite of each other. So if we just used the incoming acceleration without regarding the previous accelerations we will never see a change in position. The actual acceleration (Δa) used to compute the position is:

$$\Delta a(t) = \Delta a(t - 1) + a(t)$$

The only problem left is the, already smaller, drift. To neglect this drift we did different tests and concluded that the drift of the two weighted mean methods was somewhere between 0

and 0.015 in either direction. Since we do not want to convert the drift into a new position, we implemented these values to neglect the drift, and also the minor accelerations.

To see the difference between the three discussed mean methods we calculated the position for all three. We moved the sensors for 40 centimeters after some time and moved them back to their original position. The results can be viewed in figure 9. Because of the short time of this test the adjusting weighted mean method and the weighted mean method were almost identical, figure 9(a). Figure 9(b) shows that the normal mean method has drifted more than 10 meter after only 20 seconds. In figure 9(c) the accelerations are compared with the new position.

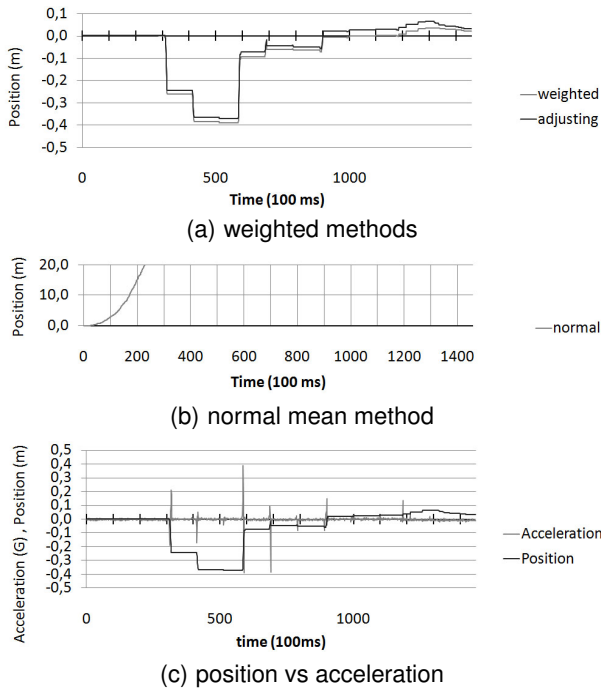


Fig. 9. Calculating the new positions

5 CONCLUSION AND FUTURE WORK

In this paper we started from the idea that macroscopical properties could help filtering the data from a MEMS accelerometer, without the use of other hardware like gyroscopes or GPS. The problem with the macroscopical information is that it is presented as positions and the incoming messages are accelerations. On these accelerations there is a lot of drift, some drift is even bigger than the real accelerations, resulting in large errors after converting the accelerations and its drift into new positions.

Converting the macroscopical properties into accelerations is also not an option, because the integrations in the conversion functions produce faults and then the integrations need to be used twice. First to calculate the accelerations of the macroscopical properties and second to get the new position.

A solution to minimize the drift with only MEMS devices was needed. Therefore we started combining data of different sensors. The outcome is a filter that uses a weighted mean of the different sensors and adapts the weight in time. This mean is not yet perfect but the drift is already decreased with 50 to 80 percent. Adapting the weight to more factors could decrease

the drift even more but at a certain point the complexity of the weight can start to have its downsides. A Kalman filter can offer a solution to keep the complexity of the weight low and still decrease the drift majorly. If a third or fourth order Kalman filter is implemented after the adaptable weighted mean filter drift could even diminish more.

Since the original idea still hasn't been researched, research to using the macroscopical properties to filter the converted data can still be done. To turn the disadvantage of the constant 1G force on the z-axis in an advantage, one can stack the sensors on each other, each single stack representing a single sensor module. Placing these modules on strategically chosen parts of an object could result in even measuring rotations of the object, while this normally can not be done with a single MEMS accelerometer.

REFERENCES

- [1] A.-H. Walid, "Accuracy enhancement of integrated mems-imu/gps systems for land vehicular navigation applications," Ph.D. dissertation, Department of Geomatics Engineering, University of Calgary, January 2005.
- [2] J. Collin and G. Lachapelle, "Mems-imu for personal positioning in a vehicle a gyro-free approach," Master's thesis, Department of Geomatics Engineering, University of Calgary, 2002.
- [3] D. X. N. Priyanka AGGARWAL, Zainab SYED and D. N. EL-SHEIMY, "Cost-effective testing and calibration of low cost mems sensors for integrated positioning, navigation and mapping systems," Ph.D. dissertation, Munich, Germany, October 8-13, 2006.
- [4] Nintendo, "Wii operations manual," Nintendo, Tech. Rep., 2006.
- [5] K. L. Stratton, "Investigation of gps/imu positioning system for mining equipment," *Final Technical Report*, p. 39, May 31, 2006.
- [6] K. Verstuyft and J. Doggen, "Distributed mems over wsn, an embedded approach," Master's thesis, Hogeschool Antwerpen, 2008.
- [7] W. B. Sebastian Thrun and D. Fox, *Probabilistic Robotics*. Massachusetts Institute of Technology, 2005.
- [8] G. Welch and G. Bishop, "An introduction to the kalman filter," *Department of Computer Science, University of North Carolina at Chapel Hill*, July 24, 2006.
- [9] S. Russel and P. Norvig, *Artificial Intelligence: A modern Approach*. Prentice Hall, 2003.



Student 1 Tim Langens received his B.S. in Applied Engineering: electronics-ict in 2007. Also in 2007 he started his M.S. at the University of Antwerp.



Promoter 1 Jeroen Doggen received his M.S. in Applied Engineering: electronics-ict in 2006.