Optimisation of an embedded sensor fusion for 3D positioning

ASGHARI Seyed Amir
Sorbonne University, Engineering faculty
Paris, France
amirsinasghari@gmail.com

PONCELET Renaud
Sorbonne University, Engineering faculty
Paris, France
renaud.poncelet@gmail.com

BALDE Hamidou
Sorbonne University, Engineering faculty
Paris, France
hamidou2balde@yahoo.fr

CAGLAYAN Ibrahim
Sorbonne University, Engineering faculty
Paris, France
ibrahimcag94@gmail.com

ABSTRACT

Nowadays, we can find different sensors which enable to position a real object in a virtual word. Almost all of them are big, expensive and not always efficient (occlusion or luminosity issues). In many cases, sensors have qualities and defaults. The most representative example is the the fast but imprecise sensor or the slow but precise one. However by combining many of them, we can get rid of the defaults and keep only the qualities. The objective is to create a sensor fusion algorithm to enable compact, cheap and efficient 3d positioning.

1. INTRODUCTION

This article is the continuation of [4], where the whole project has been initiate and developed. This article provide the simulation of the presented project that will also be presented here.
There are many techniques for 3D positioning. Firstly, some technique uses multiple camera setup like in [6], where they detect and track an object by using the geometry between the target object and cameras to finally provide a 3D positioning of the object. The main contribution was the refinement stage that correct detection and tracking error by imposing geometric constrain. However, the setup is long and cannot be made in any room quickly.
Then, some are using this visual information with inertial information in order to fusion the data like in [1] using the principle of data fusion as described in [3], where they each decision is weighted according to the reliability i.e the precision of the detectors. In this configuration, they rely on a 3D model of the scene that enables to predict the appearances of the features by rendering the model using the prediction data of the sensor fusion filter. Their paper presented a markerless vision-inertial tracking system that works in small and large scale rooms, with good or bad luminosity. The final system is efficient and robust but one of the main issue of this kind of application is the cost of the cameras.
This paper is focus on another principle that is inspired from the HTC Vive. This technique uses light house scanning to track the object as in [5]. This article will firstly explain the lighthouse operating principle in section 2. Then, it will explain the hive track operating principle in section 3. Thereafter, it will present the modelling and the simulation made to test the sensor fusion for 3D positioning in section 4 and finally it will present the results in section 5. This paper is related to the project : https://hivetracker.github.io, and it also have his own project github at
https://github.com/VR3Dtracking.

2. OPERATING PRINCIPLE

The lighthouse tracking system uses the principle of triangulation to position an object. This technique can be used to approximate the distance between two point that cannot be measured like it's shown in figure [2], where the two person can measure the distance of the rock by knowing the distance between each others and their angle when they look at the stone. The lighthouse system is the same, but it's done in 3 dimension. So its main objective is to obtain four angles to be able to compute the position of an object. There are some requirement for positioning an object by using the light House. First, a room needs to be defined where the object can be tracked only in this space. Then, the object needs to have photo-diodes on him, because the lighthouse will emit infra red light (that can be presented by a laser) and get the value of the angle from the data of the photo-diode. Finally, the two lighthouses needs to be synchronized in order to have workable values.

This is how the system works: two (or more) lighthouses will be positioned on the opposite corner of the room. The object, that have a photo-diode on it, will be in this room. One of the lighthouse (Lighthouse A) will emit a flash that the photo-diode will receive. Then it will sweep a laser plan left to right (vertically). After it scanned half a period, it will emit another flash and then sweep another laser plan down to top (horizontally) for another half a period. The other lighthouse (lighthouse B) will do exactly the same right after the horizontal sweeping of the first lighthouse. After this,

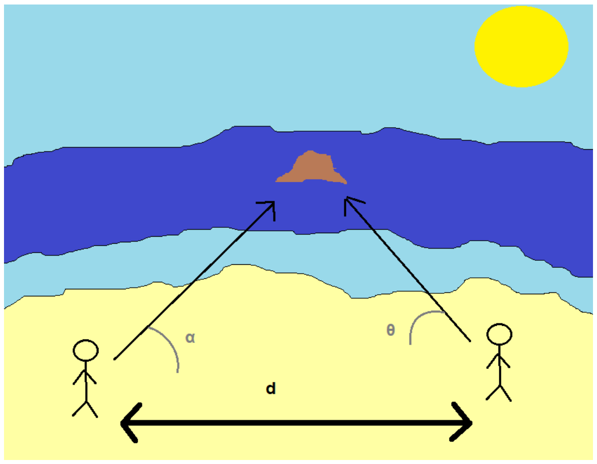


Figure 1: Illustration of the use of the triangulation method, two person look at a rock and obtain an approximate angle between their gaze and the other person α and β. Then, knowing the distance d between each others, the distance between the beach and the rock can get measured.

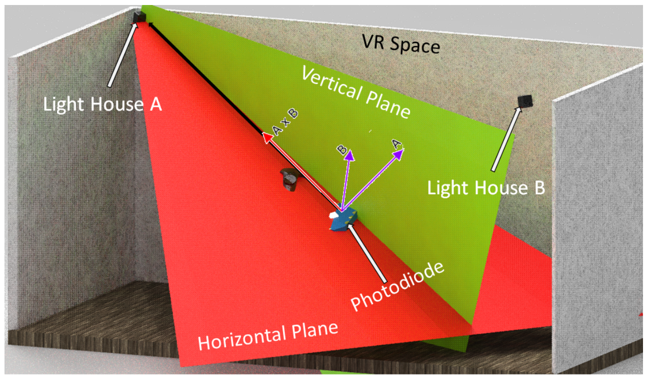


Figure 2: Demonstration of a lighthouse positioning system, where the two scanning plan (horizontal and vertical), in red and green can be seen [4]

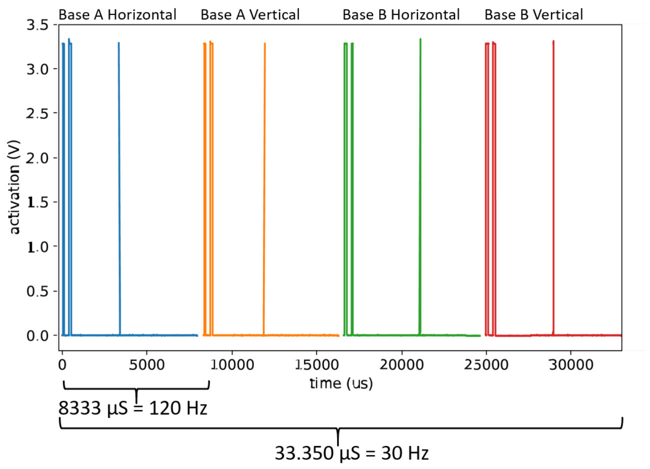


Figure 3: Light signal sequences (4 are shown): wide pulses are base station flashes and short pulses are laser plane scans. One lighthouse takes 8.3333 μseconds to sweep a semi-period, therefor, a complete positioning takes about 35 μseconds [4]

the value of the photo-diode throw time will be like figure [3]. The lighthouse requires $8.333\mu s$ to sweep a semi-period, a total of $33.333\mu s$ is therefor required for a full positioning. Time is therefor a critical information to get the angles that can be get by doing a simple proportional ratio. However, to avoid any confusion for the following calculus, it is mandatory to define a clear and precise landmark for the room and the lighthouses.

Figure [4] shows the landmark of the word as it will be used for the following parts, and figure [5] shows the one of a lighthouse. As it can be seen, the vertical sweep will start from z positive, pass by y positive and ends in z negative. For the horizontal sweep, it will start at x negative, pass by y positive too and ends in x positive. For example, if the object is in the y axis, both sweeping will detect the object at the middle of their sweeping.

If this method can detect the object everywhere one the box, there is one line where the object cannot be positioned and this line is the one between the two lighthouses. to be able to eliminate the ambiguity, a third lighthouse will be needed. However, since this line is one the roof of the room, this issue can be ignored (figure [6])

3. HIVE TRACK OPERATING PRINCIPLE

The objective is to use the position given by the light house system that it is precise but with a slow rate and the data from the MEMS (Microelectromechanical systems) sensors from the IMU (Inertial measurement unit) which are less precise but with a fast rate and process data fusion in order to have a faster rate and a good precision. Kalman filter is using to realize this data fusion. First, let's define the state system for a 6 DOF (degrees of freedom) object. The following explanations are adapted from [2], but it is

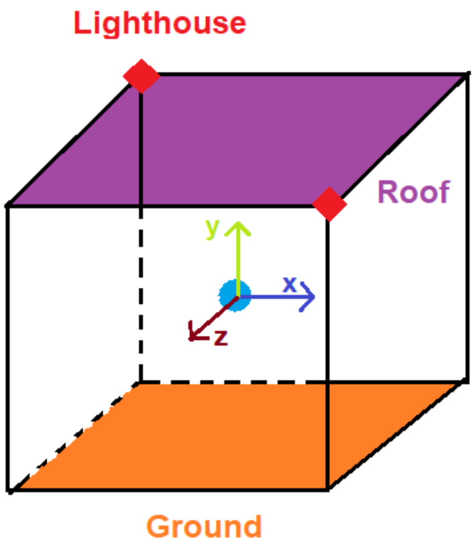


Figure 4: A modeling of the room and it's landmark. The box represent the Room, the roof is represented in purple, and the two lighthouses are positioned in the opposite corner of the room. The ground is represented in orange and the landmark positioned in the middle of the box is the Landmark of the world. In this case, if the box cube with a dimension of 1, then the two lighthouses are positioned in (-0.5,0.5,-0.5) and (0.5,0.5,0.5)

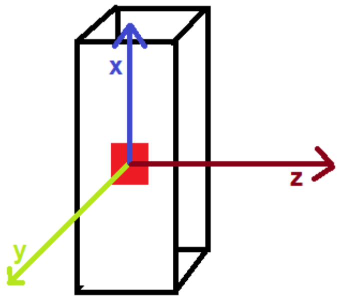


Figure 5: A modeling of the lighthouse and its landmark, the landmark is positioned where the laser is emitted. The direction vector of the sweeping plan when the lighthouse does the horizontal sweep starts at $+z$ and ends at $-z$ passing by y. The direction vector of the sweeping plan when the lighthouse does the horizontal sweep starts at $-x$ and ends at $+x$ passing also by y

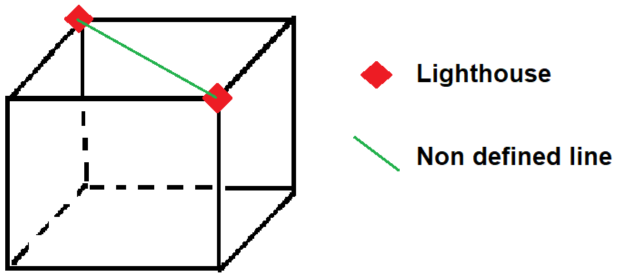


Figure 6: The line where the object cannot be positioned. The black box represents the room and the lighthouses with the red squares. Since the triangulation uses the intersection of two lines, it can't find a point of intersection when the two lines are mixed. The line on witch the position of the object cannot be detected is the green one

actually classical kalman filter application. The state is :

$$\mathbf{X} = \begin{pmatrix} \begin{pmatrix} x & \frac{dx}{dt} & \frac{d^2x}{dt^2} & \alpha & \frac{d\alpha}{dt} \end{pmatrix}^T \\ \begin{pmatrix} y & \frac{dy}{dt} & \frac{d^2y}{dt^2} & \beta & \frac{d\beta}{dt} \end{pmatrix}^T \\ \begin{pmatrix} z & \frac{dz}{dt} & \frac{d^2z}{dt^2} & \gamma & \frac{d\gamma}{dt} \end{pmatrix}^T \end{pmatrix} \tag{1}$$

note dt the time elapsed since the last estimate. Then for an instant k the estimate at $k+1$ is :

$$\mathbf{X}(k+1) = A\mathbf{X}(k) + \mathbf{w}(k) \tag{2}$$

with

$$\mathbf{A} = \begin{pmatrix} \mathbf{F} & \mathbf{0_{5\times5}} & \mathbf{0_{5\times5}} \\ \mathbf{0_{5\times5}} & \mathbf{F} & \mathbf{0_{5\times5}} \\ \mathbf{0_{5\times5}} & \mathbf{0_{5\times5}} & \mathbf{F} \end{pmatrix}$$

and $\mathbf{w}$ a zero mean white gaussian noise of assumed known covariance matrix

$$\mathbf{Q}(k) = E[\mathbf{w}(k)\mathbf{w}(k)^T].$$

For this 6 DOF object let's assume that

$$\mathbf{F} = \begin{pmatrix} 1 & dt & \frac{dt^2}{2} & 0 & 0 \\ 0 & 1 & dt & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

. This model allows to estimate the next state from the actual state. Actually, that's obviously not enough to have a good estimation if dt becames to big or if the model can't rectify his state from a measurement. The kalman filter can rectify the state from a measurement

$$\mathbf{y}(k) = \mathbf{C}\mathbf{X}(k) + \mathbf{b}(k) \tag{3}$$

where $\mathbf{C}$ is the measurement matrix and $\mathbf{b}$ is a white gaussian observation noise with zero mean and with assumed known covariance matrix

$$\mathbf{R}(k) = E[\mathbf{b}(k)\mathbf{b}(k)^T] \tag{4}$$

The matrix $\mathbf{C}$ depends on the sensor which feeding the kalman filter. For no measurement $\mathbf{C} = \mathbf{0_{12\times12}}$, for a measurement the matrix $\mathbf{C}$ have ones corresponding to each measurements. With this feeding from measurements the state can be evaluated as :

$$\begin{aligned}\hat{\mathbf{x}}(k|k) = \hat{\mathbf{x}}(k|k-1)+ \\ \mathbf{K}(k)[\mathbf{y}(k) - \mathbf{C}\hat{\mathbf{x}}(k|k-1)]\end{aligned} \tag{5}$$

where **K** is the kalman gain. There is many integration of kalman filter on c++ language. In our case, dt is not a constant. It depends on many factors like the final rate we want for kalaman estimation, also it depends on the measurement rate which is chaotic because of the using of several sensors and also because of the occlusions which destroy measurements. All this constraints can easily be manage in c++ by using an object which contains all the kalman filter information. Then, the algorithm can update the filter depends on the case :
- update without measurement
- update with positions measurements from the LEDs and light house system
- update with measurements from the IMU

4. MODELLING & SIMULATION

For the simulation the model presented in section 3 has been simplified. In order to have concrete results in a short time the measurements has been reduced to the positions from the LEDs/light house system and to the accelerations from the IMU. All this measurements are actually computed measurements in simulation. Let's explain how this computed measurements have been generated.

4.1 Graphical User Interface

The objective of the simulation is to first create a 3D model for a better visualization, and then to provide pre-measurements of what can be expected from real data. The GUI for Graphical User Interface can do both. The room is represented by a box, the LightHouses by two sphere on the two top corners and the object by a sphere initialy at the center. The scanning is represented by colored spheres turning around the LightHouses. These spheres actually represent the normal vector of the plan of the scanning as explained in section 2.

To obtain the time between the initial flash and the scanning, the algorithm will start a timer after the flash. And it will acquire the time when the scalar product between the LightHouse-Object vector (that is only known by the algorithm) and the normal vector is the smallest, since the normal vector changes with time. A zero scalar product means that the object is in the scanning plan, since that exact moment cannot be detected, taking the smallest value is a good approximation of that moment. After that step, we can easily get the angles with a simple cross product, since we know the time of a full scanning.

$$\theta = \frac{\tau\ \pi}{T} \tag{6}$$

where θ is the angle as it can be seen in figure [7], τ the acquired time and T the time for a full scanning (that is eqaul to $8.333\mu s$ as seen before). After the obtaining of the four angles, by using the spherical coordinate obtained with [7], the vector LightHouse-object can be obtained. Then, since the position and the orientation of the lighthouses is known, the algorithm can compute the localization of the object in the room.

$$x = sin(\phi) cos(\theta)$$

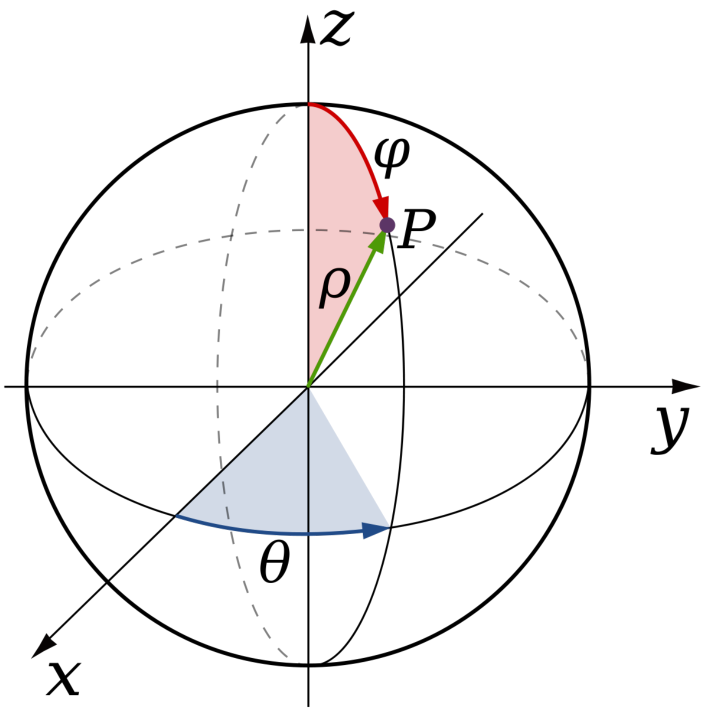


Figure 7: Spherical coordinate representation.

$$y = sin(\phi)\ sin(\theta) \tag{7}$$

$$z = sin((\phi)$$

The vector LightHouse-object is first defined in the Lighthouse landmark, but is then returned in the world landmark by doing the same rotation and translation the lighthouse had for it to be in its actual position. Doing this transformation ensure that every value given is in the world landmark, witch will eliminate any source of confusion about the position. Technically, this had to be done with the calibration of the lighthouses between them, but in the simulation, the rotation is known from the beginning (same thing for the translation).

4.2 Object localization from photo-diode

Localizing the object in 3D space with this method ends up with the calculus of the point of intersection of two lines. From the previous data, the problem can be seen as two points (Position of the LightHouses) and two vectors (the Lighthouse-object vector obtained thanks to the angles by using [6] and [7]). From two points and vector: two lines. From two lines: one point that will be the position of the object.
However, if in the theory this point exist exactly, it is really rare that two lines intersect in a 3D space. And since the scanning moment is approximated, the intersection point will be approximated too. The trick to compute this non-existent point is to find a third line that is perpendicular to the two lines. The intersection point will then be the middle of this line (see figure[8]). Let A and B be the lighthouse coordinate and u and v their respectiv LightHouse-object vector. The two lines that come from these values (the red nand the blue in [8]) are represented by:

$$A + \lambda u$$

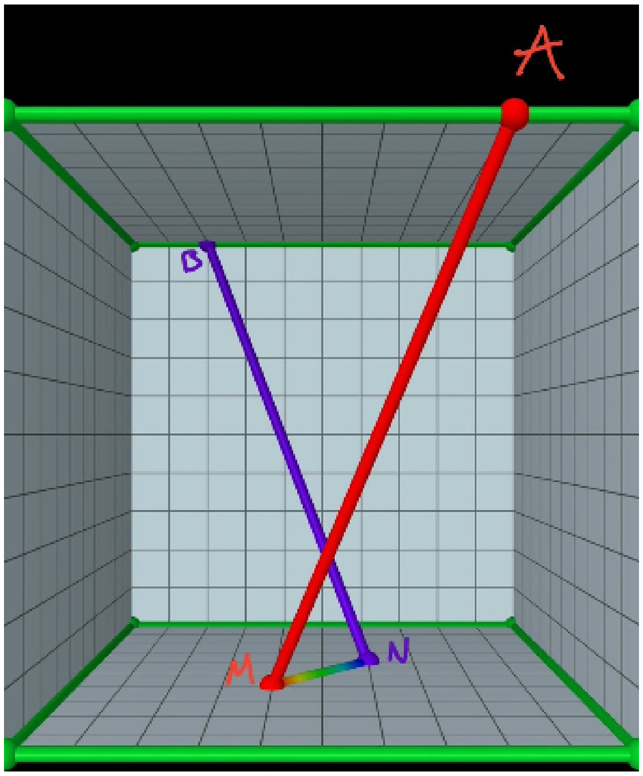


Figure 8: Non-existent intersection point trick where A and B are the two lightHouses and MN the perpendicular line. [4]

and

$$B + \nu v$$

with $\lambda, \nu \in R$. The points M and N are characterized by the fact that the vector MN is orthogonal to the two starting lines, or, equivalently, to the vectors u and v. To calculate them, we can write $M = A + \ u$ and $N = B + \nu\ v$ so that $MN = AB + \nu v - \lambda u$. The two orthogonality conditions with the scalar product are:

$$< MN, u >= 0$$

$$< MN, v >= 0$$

There is then two equation with two unknown that are λ and ν. Since the middle of the MN is the most representative point for this probleme, the coordinate I that represents the estimated position of the object will be:

$$I = [(a1 + b1 + \lambda u1 + \nu v1)/2\ ,\ (a2 + b2 + \lambda u2 + \nu v2)/2\ ,$$

$$(a3 + b3 + \lambda u3 + \nu v3)/2]$$

4.3 Object localization from the acceleration

The results obtained from the photo-diode is precised but takes too much time, a total of $35\mu s$ to get the complete position of the object. To be able to estimate the position of the object in this time-lapse, the acceleration can be obtained by deriving twice the position obtained thanks to the photo-diodes and estimate the position of the object with

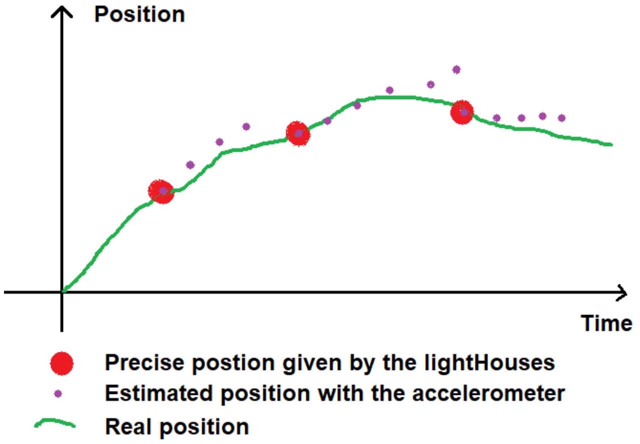
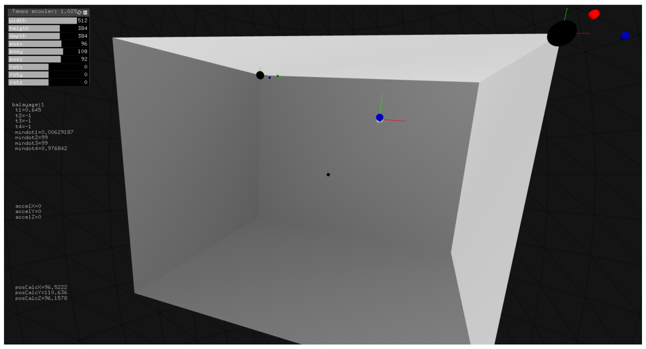


Figure 9: Estimated position with the accelerometer and the photo-diode throw time. While the algorithm waits for the precise position of the object, it compute it with the values of the accelerometer



```
Temps ecouler: 1.029
width    512
heigth   384
depth    384
posx      96
posy     108
posz      92
rotx       0
roty       0
rotz       0

balayage:1
 t1=0.645
 t2=-1
 t3=-1
 t4=-1
 mindot1=0.00629187
 mindot2=99
 mindot3=99
 mindot4=0.976842

 accelX=0
 accelY=0
 accelZ=0

 posCalcX=96.5222
 posCalcY=110.636
 posCalcZ=96.1578
```

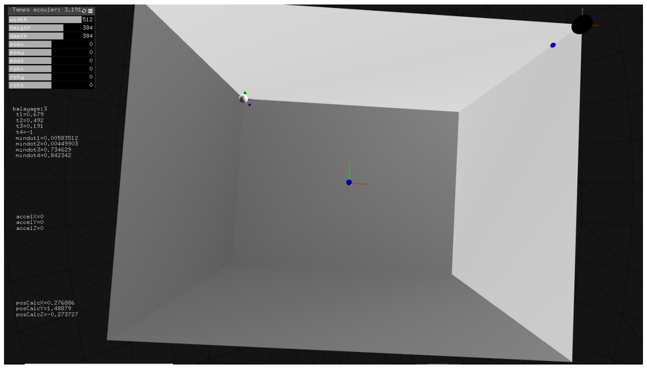
Figure 10: Simulation of the room, where the white ball represents the exact position of the object and the blue one its estimated position with the lighthouse.

it while it get the precise value of the position given again by the photo-diode. This position is obtained at each time by integrating it twice. Figure [9] shows how this method works.

5. RESULTS

In this section we will show you the results obtained with our algorithm. We used OpenFramework for the GUI. Since the simulation of $35\mu s$ won't allow us to see anything, a full positioning is done in 10 seconds. Figure [10] and [11] shows the simulated world, on the top corner we can see the two lighthouses, the colored balls around it are the normal vector of the plan that we mentioned in section 4.1 and the two balls in the box are the true position (white ball) and the estimated position (blue ball). There are also other values of the dimension of the box and the exact position of the two balls. We can see that estimated position is really close to the real position. However, the more far we go from the lighthouses, the less precise it became.

As said before, time is a critical value for this method. The function to obtain the time is *getTime* which gets the



```
Temps ecouler: 3.191
width 512
heigth 384
depth 384
posx 0
posy 0
posz 0
rotx 0
roty 0
rotz 0

balayage:3
 t1=0.679
 t2=0.492
 t3=0.191
 t4=-1
 mindot1=0.00583512
 mindot2=0.00449903
 mindot3=0.734629
 mindot4=0.842342

 accelX=0
 accelY=0
 accelZ=0

 posCalcX=0.276886
 posCalcY=1.48879
 posCalcZ=-0.273727
```

Figure 11: Simulation of the room, where the white ball represents the exact position of the object and the blue one its estimated position with the lighthouse.

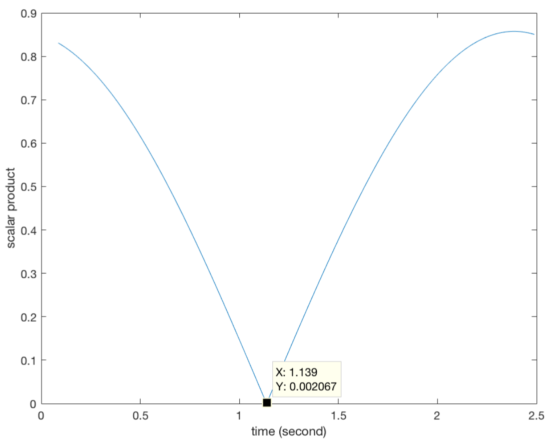
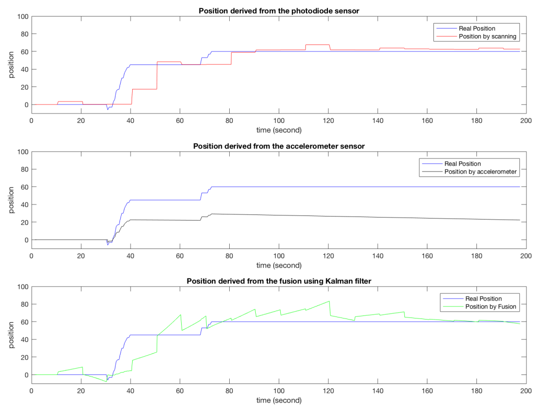


Figure 12: Evolution of the scalar product between the LightHouse-object vector and the normal to the plan vector throw time, in a case where the object is around the middle of the room

time from the computer. Thus, the value obtained is never the same for the exact same position. It means that the precision of the simulation depends directly on the power of calculus of the computer. If we want to simulate the whole position computing in 1 second, the results will be really bad.
This imprecision leads to another imprecision that we mentioned before. Indeed, we cannot get exactly the moment that the scanning plan pass throw the object and the scalar product between the LightHouse-object vector and the normal to the plan vector will most likely never be obtained at its true values (see figure [12]).

At this step, we have a precise but slow 3D positioning. We now wants to use the Kalman Filter introduced to you in section 3. Figure below show a typical response of each sensors and of the kalman filter :



We can see that the kalman filter response worse than the scan response. That is due to a bad tuning of the Kalman filter because of the lack of time. However, we can see as expected that the position derived from the photo-diode sensor is precise but slow. We can expect good prospects for improvement with a better tunning of the filter. Above all, the Kalman filter will be easier to tune as the GUI is now easy to use.

6. REFERENCES

[1] G. Bleser and D. Stricker. Advanced tracking through efficient image processing and visual–inertial sensor fusion. *Computers & Graphics*, 33(1):59–72, 2009.

[2] F. Caron, E. Duflos, D. Pomorski, and P. Vanheeghe. Gps/imu data fusion using multisensor kalman filtering: introduction of contextual aspects. *Information fusion*, 7(2):221–230, 2006.

[3] Z. Chair and P. Varshney. Optimal data fusion in multiple sensor detection systems. *IEEE Transactions on Aerospace and Electronic Systems*, (1):98–101, 1986.

[4] D. K. C. H. D. M. A. K. Daro R. Quiones, Gonalo Lopes. Hive tracker: a tiny, low-cost, and scalable device for sub-millimetric 3d positioning. february 2018.

[5] O. Kreylos. Lighthouse tracking examined. *Internet-Blog" Doc-Ok. org". Online verfügbar unter http://doc-ok. org*, 2016.

[6] Y. Lee and A. Yilmaz. Real-time object detection, tracking, and 3d positioning in a multiple camera setup. In *The ISPRS Workshop on Image Sequence Analysis*, volume 55, page 56, 2013.