Basketball Tracking and Score Evaluation

January 7, 2016

Academic Supervisor: Professor Pascal Fua
Industrial Supervisor: Dr. Horesh Ben Shitrit
Student: Samuel Pierre Rey

1 Abstract

The goal of this semester project was to create a score detecting system to determine if a basket has been scored. The system processes the data generated by the PlayfulVision ball detection algorithm [1] [2].

In this semester project we created a 3D model of the basketball court present in the CSS. It was used to represent efficiently the data obtained with the PlayfulVision detection algorithm in the form of a vertices graph. The results of the tracking algorithms can be transformed into a video that can be displayed on the court. A score detection method algorithm has been developped. Severals algorithm were tested to track the ball. Their outputs were then used by the score detection algorithm to know if a score has occurred. We unfortunately did not test enough the algorithms to have enough information about their performances.

2 Introduction

This semester project took place in collaboration with the CSS of UNIL/EPFL. The CSS basketball court is surrounded by eight cameras used by the PlayfulVision detection software. (Figure 1)

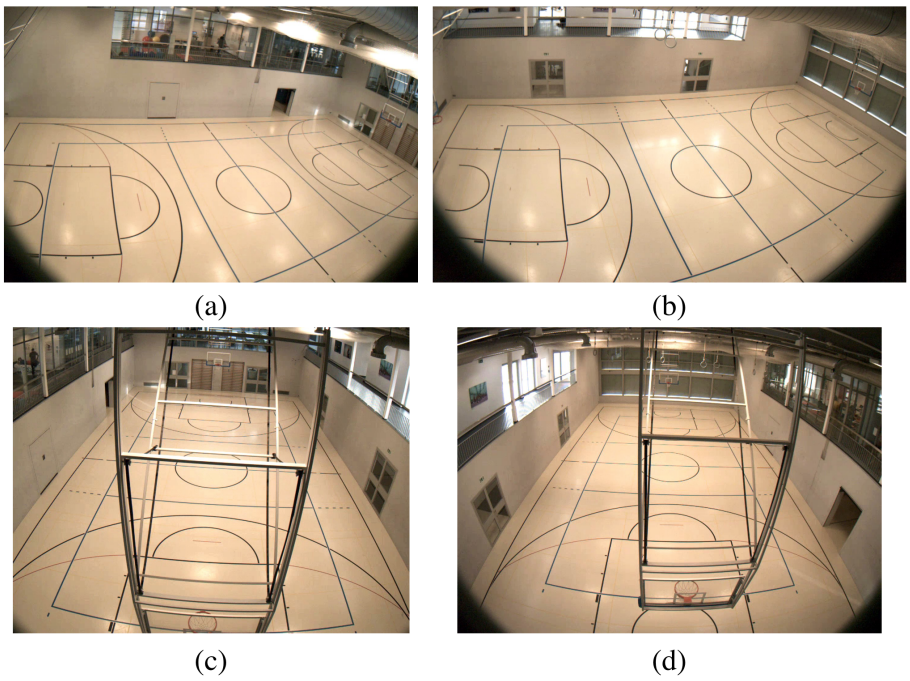


Figure 1: The CSS basketball court view from (a) Est (b) West (c) South (d) North

The cameras record the court and then the PlayfulVision detection algorithm gives a list all the ball detection that occurred during the recording. As the input were video recordings the time was in the form of frames. The cameras were recording at 30 fps. Finally, the list is saved as a text file which each line contains a frame and a position describes by the 3D coordinates in meters. Multiple detection can occur at the same frame.

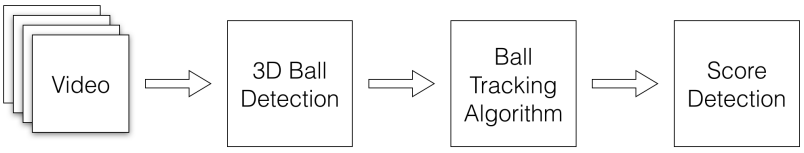
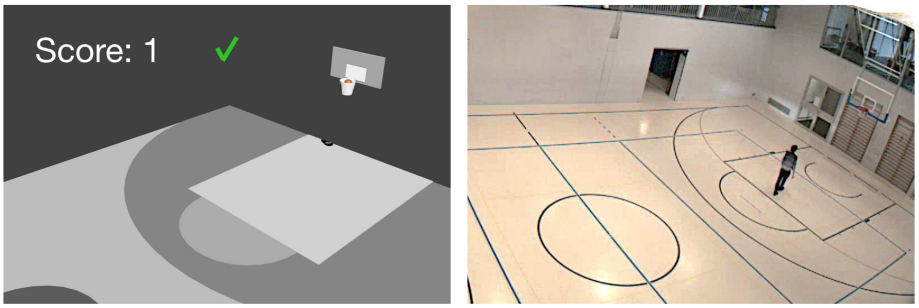


Figure 2: The pipeline of the score evaluation. First the videos are processed by the 3D ball detection algorithm. After that, the data are sent to the ball tracking algorithm. Finally, the ball tracking results are processd by the score detection algorithm to determine if a score occurred.

The pipeline process ball tracking and score detection is represented in figure 2. The 3D detection was already implemented in the PlayfulVision detection algorithm. Our task was therefore to create the tracking algorithm and score detection parts. Before working on them we first search a efficiant way of diplaying the data. A first attempt was done with a superposition of an image taken by one of the cameras and a data graph. As the quality of the image was too poor, we created a 3D model of the court. Then, a score detection method algorithm was developped using the coordinates of a small box placed inside the basket net of the model. After that, severals tracking algorithms were tested. We first create the Minimal Distance tracking algorithm. Then a more sophisticated one was developped using the velocity of the measurments to selects the next position of the ball. Afterwards the famous Kalman filter [7] was used to implement the Parallel Kalman Filter tracking algorithm. Unfortunately, its implementation did not work as hoped. Then a sequence of samples which their scores were known in advance was runned on the tracking / score algorithm and compared to the real results. (Figure 3)



(a) (b)

Figure 3: (a) The score detection algorithm detected a scored basket. (b) The video recorded by one of the eight cameras used by the ball detection algorithm of PlayfulVision to extract its position.

A video of the results of the tracking and score detection algorithms can be viewed at `https://vimeo.com/151012795`.
Even though the size of the samples was not enough and our kalman filter implementation was not successfull, we can however have an idea of their potential.

3 Data Representation and Display

This section treats about finding a efficient way of displaying the data recorded by the eight cameras installed all around the basketball court. A first attempt was to create a superposition of a graph of the data created with Matlab [4] over a picture from one of the camera of the sport center (Figure 4).

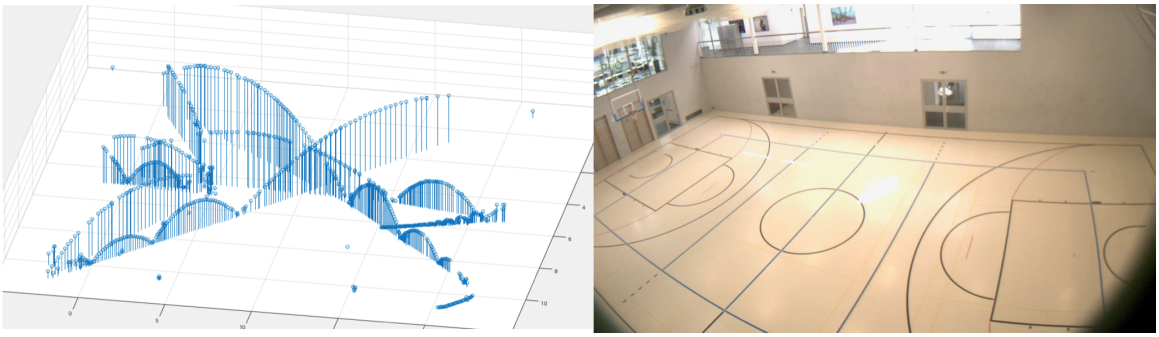


Figure 4: The data graph drawn with Matlab and a picture of the court taken by the camera 3

The problems encountered were the quality of the picture and the accuracy. The distorsion made difficult to superpose the data over the court. The accuracy was not enough to detect precisely where the ball was relative to the court. Moreover, the result of this process prevented any alterations of the view angle, which made difficult to precisely locate the ball relative to the court.

3.1 Creating A Virtual Basketball Court With Blender

The next choice was to model a 3D basketball court with Blender [5]. The 3D model allows a more accurate estimation of the ball position inside the virtual court. Two maps of the CSS of UNIL-EPFL were used to create the baskets and court.

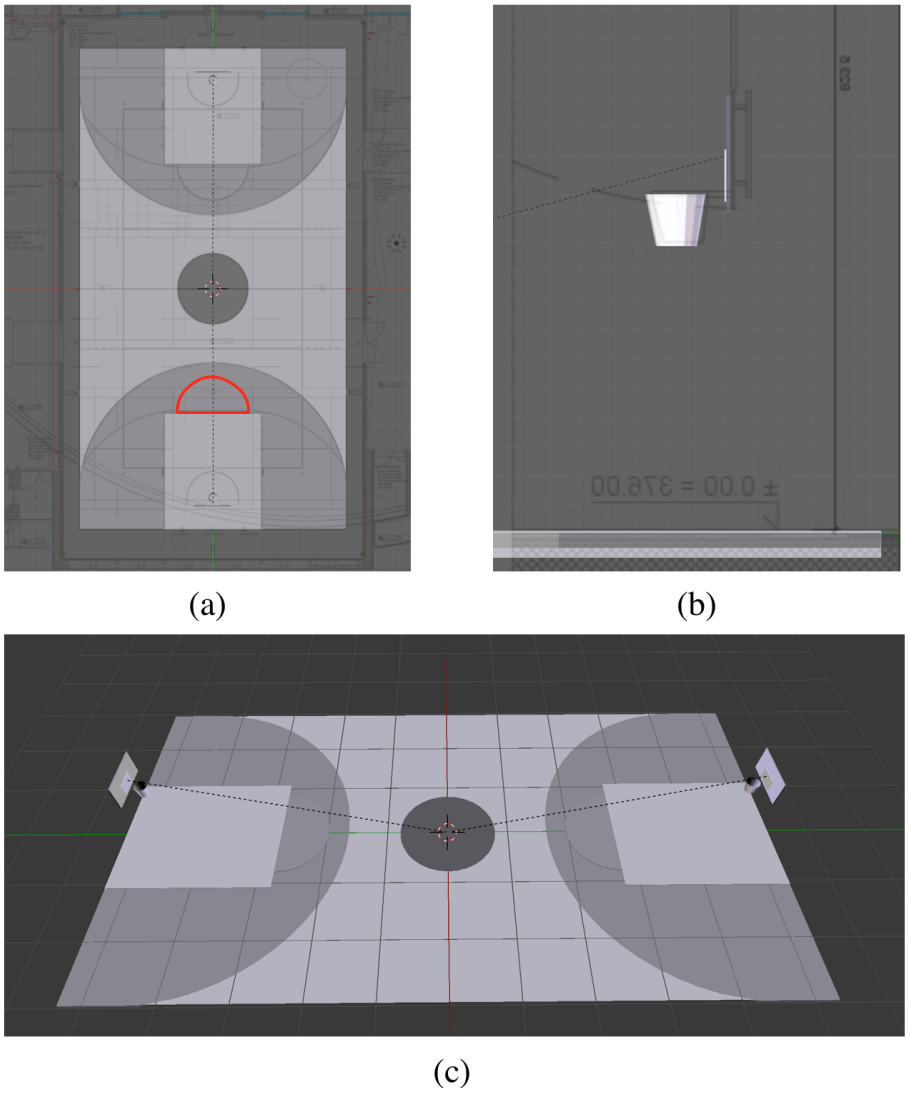


Figure 5: The 3D basketball court. (a) top view court with map with the restricted area in red (b) side view basket with map. (c) side view without map

As the court was not at the right scale, four reference points were added to the OBJ file. The PlayfulVision algorithm coordinate system uses the volleyball court as reference. Hence, the four reference point are added at the beginning of the data graph. Each of them is a corner of the volley ball court: $(0, 0, 0, 0)$; $(0, 0, 9, 0)$; $(0, 9, 18, 0)$; $(0, 0, 18, 0)$

The last part of the modelization is the walls. As they are used in the tracking algorithm, a precise position is required. To place more precisely the walls around the court, a sequence where the ball was thrown against them was used. (Figure 7)

3.2 Importing and Exporting Data

Two format are used to display the data. To display the graph of the data recorded we use the OBJ format [6] and to view the animation of the ball we use the COLLADA format.

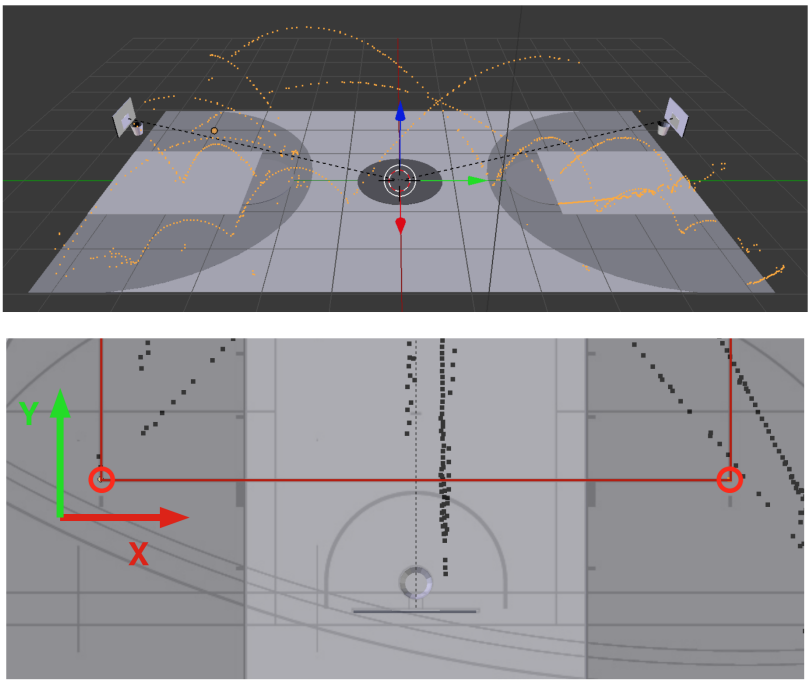


Figure 6: Two corners of the volleyball court that are used to adjust the scale of the court.

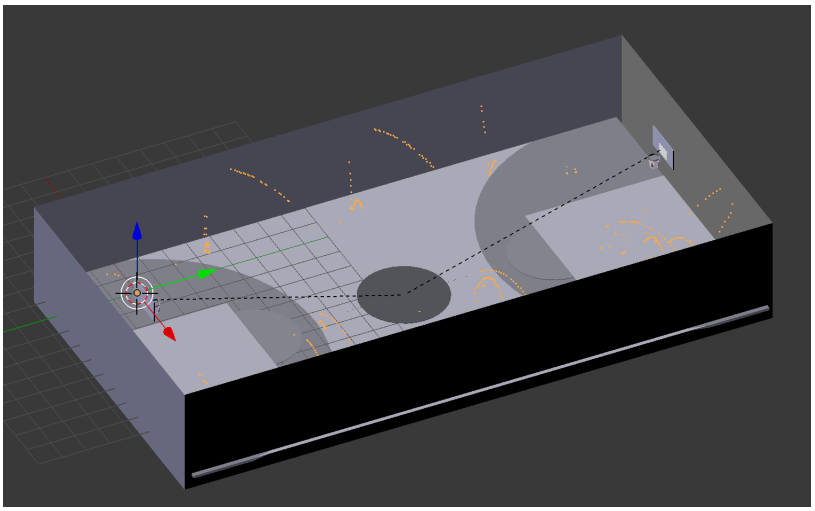


Figure 7

Raw Data	OBJ
Frame X Y Z	o Data v X Z -Y 0

Figure 8: Converstion between raw data and OBJ format

To import and exporting the data in Blender, the OBJ file format was chosen because of its simple semantic format. (Figure 8) Due to the Blender way of importing OBJ file, the Y and Z coordinates must be switched and Y must multilipied by -1.

To display the animation of the ball the COLLADA format is chosen. The COLLADA format allows to add the animation contrary to the OBJ format. The COLLADA file consists of the coordinates of the ball sphere and its location at each frame. To simplify the file a linear interpolation was chosen.

4 Detection of a basket

This section treats about the implementation of a basket detection algorithm. A simple and efficient way is to create a box inside the net in figure 9 of the basket and check if the ball go through it. If the ball was above it at time $t-1$ and is inside it at time t, then it we can make the approximation that the ball had to pass throug the upper side of the box. After that the ball is inside the box until time $t+n$. If the ball is bellow the upper side of the box at time $t+n$ a score is detected.

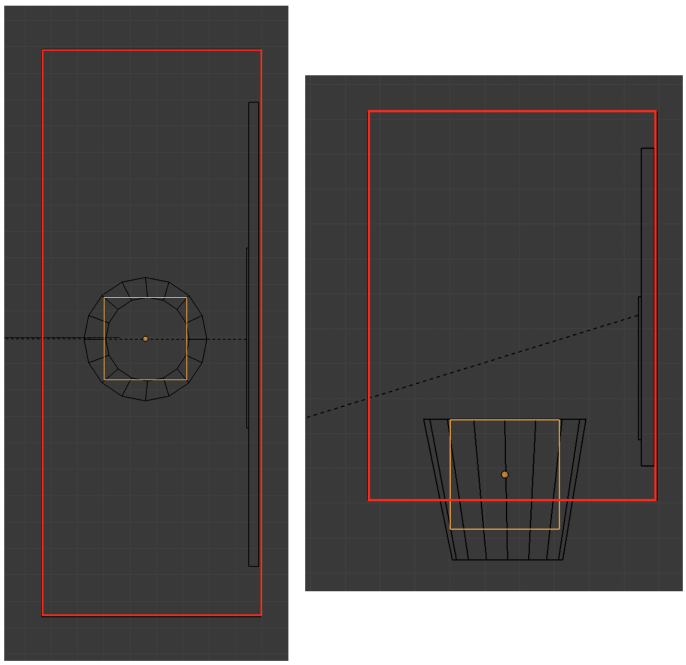


Figure 9: The bakset zone in red and the score zone in orange

5 Ball Tracking Algorithms

This chapter treats about the first two tracking algorithms that were used.

5.1 The Minimal Distance Algorithm

The MINIMAL DISTANCE algorithm. The input of the algorithm are a list of keyframes at time t and the previous keyframe. A keyframe refers to a set of a temporal coordinate (always given as a frame) and three spatial coordinates. $keyframe = (frame, x, y, z)$. The algorithm computes the euclidian distance between the position of the previous keyframe and the current ones. Then, it selects the one with the minimal distance compare to the previous keyframe and return it.

Algorithm 1 Minimal Distance Algorithm

```
1: function MINIMAL DISTANCE(k_{t-1}, L[k_{1_t}, k_{2_t}, ..., k_{n_t}])
2:     best ← 0
3:     dist_min ← ∞
4:     for i ← 0 to L.size do
5:         if dist(L[i], k_{t-1}) < dist_min then
6:             best ← i
7:             dist_min ← dist(L[i], k_{t-1})
8:         end if
9:     end for
10:    return L[i]
11: end function
```

The results of this algorithm can be found in the subsection 7.1

5.2 Multiple Steps Algorithm

The MULTIPLE STEPS algorithm. The input of the algorithm are the number of step wished s, the two previous keyframes and a list of the list of the measurements at time t up to $t + m$. Also, $s < L.size$ and $p > 1$ must be true. It computes a estimated keyframe with the ESTIMATED NEXT KEYFRAME function and runs the MINIMAL DISTANCE algorithm on the list of measurments at time i for s times. Then, it returns the best candidate. In the case of any results, then the estimated next keyframe at time t is returned.

The ESTIMATED NEXT KEYFRAME returns as estimated keyframe at a time $t+n$ based on the speed between the two keyframes passed as argument. The speed is computed by divided the distance of the two keyframes with their respective frames time.

Algorithm 2 Multiple Steps Algorithm

function MULTIPLE STEPS(s, k_{t-p}, k_{t-1}, $L\{[k_{1_t}, ..., k_{n_t}], ..., [k_{1_{t+m}}, ..., k_{n_{t+m}}]\}$)
  $keyframe_{best} \leftarrow \emptyset$
  $dist_{min} \leftarrow \infty$
  for $i \leftarrow 0$ to s **do**
    if $L[i] = \emptyset$ **then**
      $keyframe_{estimated} \leftarrow$ ESIMATED NEXT KEYFRAME(k_{t-p}, k_{t-1}, i)
      $keyframe_{tmp} \leftarrow$ MINIMAL DISTANCE$(keyframe_{estimated}, L[i])$
      if $dist(keyframe_{tmp}, k_{t-1}) < dist_{min}$ **then**
        $keyframe_{best} \leftarrow tmp$
        $dist_{min} \leftarrow dist(tmp, keyframe_{estimated})$
      end if
    end if
  end for
  if $keyframe_{best} = \emptyset$ **then**
    $keyframe_{best} \leftarrow$ ESIMATED NEXT KEYFRAME$(k_{t-p}, k_{t-1}, 1)$
  end if
  return $keyframe_{best}$
end function
function ESTIMATED NEXT KEYFRAME(k_{t-p}, k_{t-1}, n)
  $v_x \leftarrow \frac{k_{t-1}.x - k_{t-p}.x}{k_{t-1}.frame - k_{t-p}.frame}$
  $v_y \leftarrow \frac{k_{t-1}.y - k_{t-p}.y}{k_{t-1}.frame - k_{t-p}.frame}$
  $v_z \leftarrow \frac{k_{t-1}.z - k_{t-p}.z}{k_{t-1}.frame - k_{t-p}.frame}$
  $keyframe_{estimated} \leftarrow (t + n, k_{t-1}.x + v_x * n, k_{t-1}.y + v_y * n, k_{t-1}.z + v_z * n)$
  return $keyframe_{estimated}$
end function

The results of this algorithm can be found in the subsection 7.2

6 Kalman Filter To Track The Ball

In this chapter we will discuss about the implementation of the kalman filter to track the ball.

6.1 Introduction

The Kalman filter [7] [8] is a recursive estimator because it only needs the previous estimate and the current measurment to compute the current estimate. Also, it is an optimal estimator because if all noise is gaussian then the Kalman filter minimizes the mean square error of the estimated parameters. The kalman filter is an online process which means that it processes the new measurements as soon as they arrive . The state of the Kalman filter is represented by two variables:

- $\hat{x}_{k|k}$ the *a posteriori* state estimate at time k given observations up to and including at time k
- $P_{k|k}$ the *a posteriori* error covariance matrix

The Kalman filter is composed of seven equations written in the TRACKING KALMAN FILTER tracking algorithm.

6.2 Kalman Filter Tracking Algorithm

The inputs of the KALMAN FILTER algorithm are the list of the measurements at time t and the *a priori* estimate $\hat{x}_{k-1|k-1}$. Then, it computes $\hat{x}_{k|k-1}$, selects its nearest measurment in the list as z and computes the velocity between it and $\hat{x}_{k-1|k-1}$. If the list is empty, it will assigns the value of $\hat{x}_{k|k-1}$ to z. The value $\hat{x}_{k|k}$ is returned.

Algorithm 3 Kalman Filter

1: **function** KALMAN FILTER($L[k_{1_t,..,k_{1_t}}]$, $\hat{x}_{k-1|k-1}$)
2: $\hat{x}_{k|k-1} \leftarrow F * \hat{x}_{k-1|k-1} + B * u$ ▷ $\hat{x}_{k|k-1}$
3: $P \leftarrow F * P * F^T + Q$ ▷ $P_{k|k-1}$
4: $z \leftarrow \emptyset$
5: **if** $L.size \neq 0$ **then**
6: $z \leftarrow$ MINIMAL DISTANCE(L, x)
7: $z.velocity \leftarrow \frac{z.position - \hat{x}_{k-1|k-1}.position}{dt}$
8: **else**
9: $z \leftarrow \hat{x}_{k|k-1}$
10: **end if**
11: z
12: $y \leftarrow z - H * \hat{x}_{k|k-1}$ ▷ $\tilde{y}_k$
13: $S \leftarrow H * P * H^T + R$ ▷ S_k
14: $K \leftarrow P * H^T * S^{-1}$ ▷ K_k
15: $\hat{x}_{k|k} \leftarrow \hat{x}_{k|k-1} + K * y$ ▷ $\hat{x}_{k|k}$
16: $P \leftarrow (I - K * H) * P$ ▷ $P_{k|k}$
17: **return** $\hat{x}_{k|k}$
18: **end function**

This algorithm performs great for tracking a single parabola but is not mean to track bounces. If the measurements are very precise (which implies a high value of R), then it is able to track the bounces of the ball. In our case the measurements are not reliable enough. Hence, another algorithm for walls and floor bounce has be done.

The BOUNCE DETECTION algorithm detects if the ball bouces against a obstacle (floor or walls). It takes a vector x containing the position and the velocity of the ball. If a coordinate of the position of ball is outside its boudary, then its velocity is mutliply by $-e$, e as the coefficient of restitution of the ball (for a basketball ball $0.82 \leq e \leq 0.88$ [9]).

Algorithm 4 Bounce Detection

function BOUNCE DETECTION(x)
  if $x.p_x$ is outside the walls x coordinates **then**
    $x.v_x \leftarrow -e * x.v_x$
  end if
  if $x.p_y$ is outside the walls y coordinates **then**
    $x.v_y \leftarrow -e * x.v_y$
  end if
  if $x.p_z < 0$ **then**
    $x.v_z \leftarrow -e * x.v_z$
  end if
end function

The BOUNCE DETECTION algorithm is a good enough bounce approximation. To find the boundaries, the walls of the 3D model where exported as a OBJ file. The x coordinate boundaries of the walls are $[-2.01; 11.03]$ and the y coordinate ones are $[-3.35; 21.28]$. Several simulations of the implementation of the KALMAN FILTER tracking algorithm are attached in annex A.

6.3 Parallel Kalman Filter Tracking Algorithm

The PARALLEL KALMAN FILTER tracking algorithm works the same way as the KALMAN FILTER tracking algorithm. The only difference between the two is that the parallel one computes two kalman filter in parralel. If the measurement position is inside the basket zones, then it is more susceptible to be the ball measurement, because there are not the shadow problem in this place. Therefore the measurement is taken as more reliable and the value of the matrix R is greater inside than outside the basket zones. On the contrary, the kalman filter prediction is not reliable at all inside the basket zones. Hence, the value of the matrix Q is greater than outside the basket zones.

Algorithm 5 Parallel Kalman Filter

function PARALLEL KALMAN FILTER($L[k_{1_t,..,k_{1_t}}]$, x)
  $z \leftarrow$ MINIMAL DISTANCE(L, x)
  if $z \neq \emptyset$ **then**
    if z is inside a basket box **then**
      $x \leftarrow$ KALMAN FILTER($[z]$, x) with parameter Q_{in} and R_{in}
    else
      $x \leftarrow$ KALMAN FILTER($[z]$, x) with parameter Q_{out} and R_{out}
    end if
  else
    $x \leftarrow$ KALMAN FILTER($[]$, x)
  end if
  return x
end function

6.4 Application of the Kalman filter on the basketball model

To initialize the Kalman filter we must first define the transition matrix F, the control-input model B, the extraction matrix H, the noise covariance matrices Q and R.

The state vector x_k of the ball contains its position coordinate and its velocity values. Thus, $x_k = (p_x, p_y, p_z, v_x, v_y, v_z)$. The measurement only gives the position coordinates. A good approximation of the velocity is to compute the speed between the previous measurement and the current one, as the difference of time between the two is only $dt = \frac{1}{30}s$ (the videos are recorded at 30 fps). Thus $z_k = (m_x, m_y, m_z, \frac{m_x - p_{x,\hat{x}_{k-1|k-1}}}{dt}, \frac{m_y - p_{y,\hat{x}_{k-1|k-1}}}{dt}, \frac{m_z - p_{z,\hat{x}_{k-1|k-1}}}{dt})$ which will be written as $z_k = (m_x, m_y, m_z, \dot{m}_x, \dot{m}_y, \dot{m}_z)$ for more simplicity.

The tansistion matrix F is

$$F = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

The input-control B matrix is

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & dt^2/2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & dt \end{bmatrix} \tag{2}$$

The acceleration magnitude vector u is

$$u = \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \tag{3}$$

The sate esmite $x_{k|k}$ is thus given by the equation (1) is

$$x_{k|k} = \begin{bmatrix} p_x + v_x * dt \\ p_y + v_y * dt \\ p_z + v_z * dt - g * \frac{dt^2}{2} \\ v_x \\ v_y \\ v_z - g * dt \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & dt^2/2 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & dt \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \tag{4}$$

The measurement noise covariance matrix and the process noise covariance matrix are defined as:

$$Q = \begin{bmatrix} \sigma^2_{p_x} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{p_y} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{p_z} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{v_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{v_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2_{v_z} \end{bmatrix} \tag{5}$$

$$R = \begin{bmatrix} \sigma^2_{m_x} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{m_y} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{m_z} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{\dot{m}_x} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{\dot{m}_y} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2_{\dot{m}_z} \end{bmatrix} \tag{6}$$

Where σ^2 represents the variance of each quantity. The process noise covariance matrix represents the amount of error due to the model limitation and the computation precision. When the ball bounce against the basket ring, we cannot compute a good estimate with this model. Hence, the value of Q must be higher around the basket than somewhere else in the court.

Concerning the measurment noise covariance matrix R and the process noise covariance matrix Q their values are more difficult to find. The PlayfulVision tracking software gives different errors measurement in function of the height of the ball. For instance, when the ball is near the ground it is harder to differentiate a false measurement due to a shadow than the true ball measurement. Hence, the values of Q and R must change in function of the measurement position.

7 Results

This chapter treats about the results of the tracking algorithm discussed in this paper running on a sequence of samples of 47 attempts (15 scores and 32 misses). It is not sufficient to represent completely their ability of tracking. Indeed, these samples were made with only one person on the court. This limits the number of false measurements that can occurs in a real match due to the bigger numbers of round like objects (shadows for instance) and a lesser visibility of the ball. Also there were only from around the restricted area in Figure 5 which is not representative of all the throws that can occurs in a basket match.

7.1 Minimal Distance Trackingg Algorithm

	Number of sequences	Automatic detection
Scored baskets	15	10
Missed baskets	32	32
Total	47	42

Figure 10: The Minimal Distance tracking algorithm results

The Minimal Distance tracking algorithm performs well on these samples. This is due to the good quality of the detection results. Nonetheless, the few samples that were used to test the tracking algorithms are not representative enough to give us a good sense of their quality.

7.2 Multiple Steps Tracking Algorithm

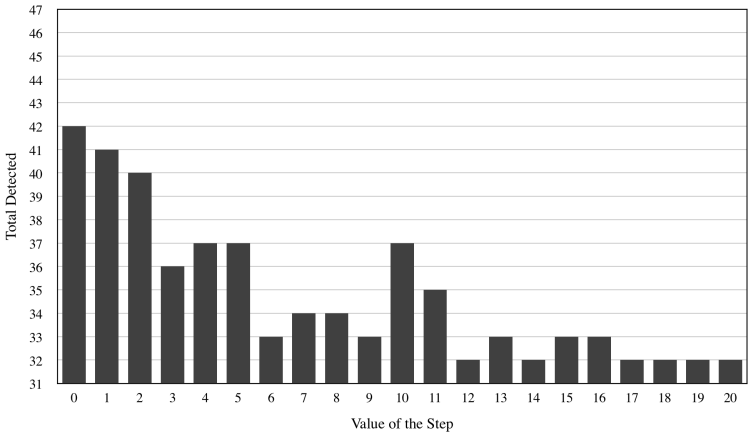


Figure 11: The results of the Multiple Setps tracking algorithm with various steps.

The samples were runned with different value for the step. The best result is when the step is equal to zero. We can see that the performance of this algorithm tends to decrease with bigger values.

7.3 Parallel Kalman Filter Tracking Algorithm

A video of all the samples and the Parallel Kalman Filter tracking algorithm running on them can be seen at `https://vimeo.com/151012795`.

	Number of sequences	Automatic detection
Scored baskets	15	11
Missed baskets	32	32
Total	47	43

Figure 12: The Parallel Kalman Filter tracking algorithm results

Multiple value had been tested for the matrix Q_{in}, Q_{out}, R_{in} and R_{out}. The ones with the more good score detected were ketpt. These values are the following:

$$Q_{in} = 1 * I$$
$$Q_{out} = 0.001 * I$$
$$R_{in} = 0.2 * I$$
$$R_{in} = 0.35 * I$$

Where I stand for a 6 by 6 identity matrix

The Parallel Kalman Filter tacking algorithm does not give much better result than the Minimal Distance tracking algorithm despite its process to distinct the reliability of a measurement in fonction of its position.

8 Conclusion

We created a 3D model of the CSS basketball court with Blender to display the data. An exporting and importing OBJ format file and a exporting COLLADA format file of the datat ways were designed. A score detection algorithm was creating with the help of the 3D court model. Three tracking algorithms were developped and tested. While two of them have showed good results (Minimal Distance and Parallel Kalman Filter), the little samples of data on which the algotrithm have been tested do not give us enough information concerning their quality. Also, our kalman filter tracking algorithm implementation did not give the results wished for simple curves. Therefore, the potential of the parallel kalman filter tracking algorithm has not been fully utilized.

The tracking algorithms can be improved by selecting the next position of the ball by attributing a weight coefficient to all the measurements that occurs in the same time. In addition, others techniques to detect a score could be developped. For instance, more than two keyframes could be used to determine if the ball have passed through the basket ring. If the ball trajectory is a straight line passing a few centimeters next to the score zone, then it could be evaluate as a score.

Finally, I am glad to have had the opportunity to work on this semester project. It allowed me to discover the challenging aspect of tracking and score detection algorithms. I learned the operation of the Kalman filter and its application on real tracking problems, even though the kalman filter algorithm did not work as hoped. Also, the implementation of the algorithms teached me a lot in terms of programming techniques.

Acknowledgements

This paper has been made possible with the provision of the baksetball court by the CSS of UNIL/EPFL [3]. The CSS team should be thanked for its ongoing commitment and support for this semester project. I would like to thank Professor Pascal Fua for supervising me during this semester project. I am also very grateful to my industrial supervisor, Dr. Horesh Ben Shitrit, for the help he provided me with his advices and feedback.

References

[1] PlayfulVision
`http://playfulvision.com`

[2] X Wang, V Ablavsky, HB Shitrit, P Fua *Take your eyes off the ball: Improving ball-tracking by focusing on team play* Computer Vision and Image Understanding, 2014

[3] CSS
`http://www2.unil.ch/css`

[4] Matlab
`http://ch.mathworks.com`

[5] Blender
`https://www.blender.org`

[6] Paul Bourk
`http://paulbourke.net/dataformats/obj/`

[7] R.E Kalman *"A new approach to linear filtering and prediction problems," Journal of basic Engineering, vol. 82, pp. 35 - 45,* 1991.

[8] Michalis Zervos *Real time multi-object tracking using multiple cameras* Semester project at CVLab, EPFL, Spring 2012.

[9] Dr Vassilios M Spathopoulos *"An Introduction to the Physics of Sports"*

Appendices

A Kalman Filter Performance Annex

In this section the results of the kalman filter tracking algorithm are presented. The value of the matrix Q and R are $0.001 * I$ and $0.2 * I$. First part is its application on a a linear curve, then its application on a quadratic curve with bounce. For the second part the negative acceleration of the curve correspond to the gravitational constant g.

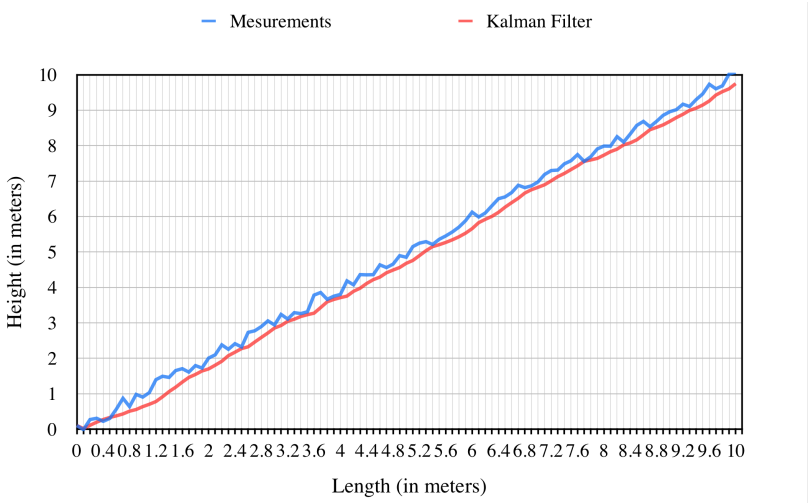


Figure 13: The kalman filter tracking algorithm results with a randomized linear curve taking all the measurments

In the figure 13 the linear curve has been randomized. Each points are randomly moved up to $\pm 0.2 meters$ in height and length.

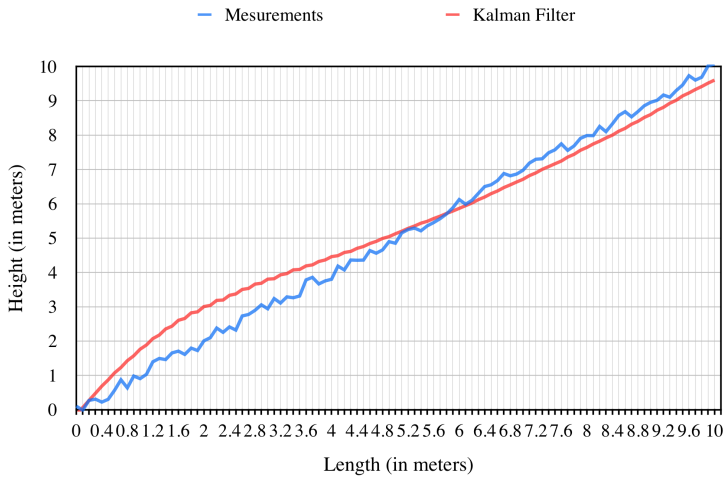


Figure 14: The kalman filter tracking algorithm results with a randomized linear curve taking only measurments at timw $2t$

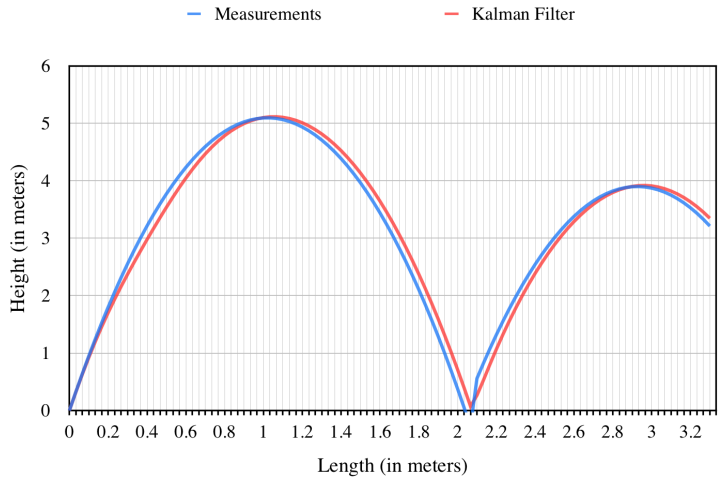


Figure 15: The kalman filter tracking algorithm results with a bouncing ball taking all the measurments

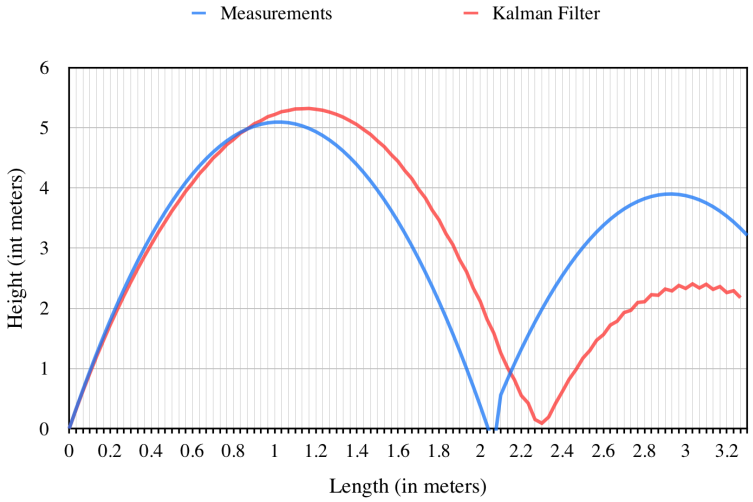


Figure 16: The kalman filter tracking algorithm results with a bouncing ball taking only measurements at time $2t$

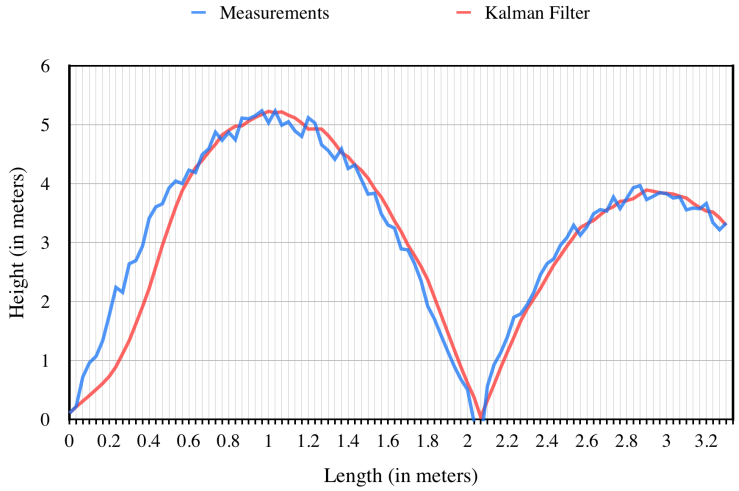


Figure 17: The kalman filter tracking algorithm results with a bouncing ball taking all the measurements

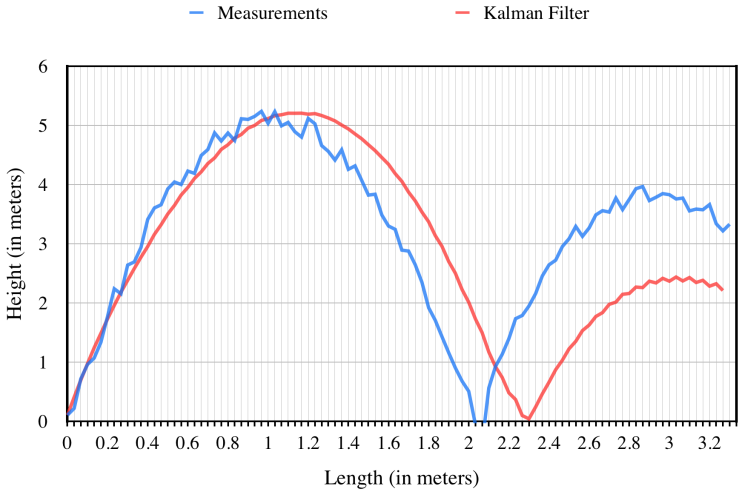


Figure 18: The kalman filter tracking algorithm results with a bouncing ball taking only measurements at time $2t$