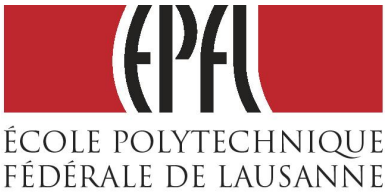


École Polytechnique Fédérale de Lausanne

Universitat Politècnica de Catalunya

Master Thesis

Tracking of a Basketball Using Multiple Cameras

Student:
Neus Agelet Ruiz

Supervisor at EPFL:
Jérôme Berclaz

February 22 - July 2, 2010

I would like to express my thanks to my supervisor at EPFL, Dr. Jérôme Berclaz, for the possibility to participate in his project, guiding me and inspiring me throughout the project with his valuable suggestions and knowledge on the field. I truly appreciate his constant feedback on the project which always lead me into the right direction.

Finally I want to give my most sincerely gratitude to the people whose contribution made this thesis possible: my colleagues from EPFL, UPC and the CVlab department. Especially to Ben Shitrit Horesh Beni for his help and support.

Abstract

This master thesis presents a method for tracking a basketball during a basketball match recorded with a multi-camera system. We first developed methods to detect a ball in images based on its appearance. Color was used through a color histogram of the ball, manually initialized with ball samples. Then the shape of the ball was used in two different ways: by analyzing the circularity of the ball contour and by using the Hough transform to find circles in the image.

In a second step, we attempted to track the ball in three dimensions using the cameras calibration, as well as the image methods previously developed. Using a recursive tracking procedure, we define a 3-dimensional search volume around the previously known position of the ball and evaluate the presence of a ball in all candidate positions inside this volume. This is performed by projecting the candidate positions in all camera views and checking the ball presence using color and shape cues. Extrapolating the future position of the ball based on its movements in the past frames was also tested to make our method more robust to motion blur and occlusions.

Evaluation of the proposed algorithm has been done on a set of synchronized multi-camera sequences. The results have shown that the algorithm can track the ball and find its 3D position during several consecutive frames.

Contents

Chapter 1

Introduction

Recent advances in computer vision makes it possible to develop automatic sports analysis without the need of special devices. Using only video sequences of a game, computer vision allows to automatically extract important information about players' performance.

The analysis of a team sport match can be very valuable to coaches and referees to study the development of a match and corroborate their decisions on the court.

In any ball game, such as soccer or basketball, the ball is undoubtedly the focus of attention. The ball presents several characteristics that makes it very specific:

- **Color**. The color of the ball is designed to be visible for players and spectators. In basketball, the color most widely used is orange with black stripes, but it can have also single or multiple colors.
- **Size**. The ball always has a defined standard size. In basketball, it measures 23 or 24 cm of diameter.
- **Shape**. The ball has a spherical shape that can help us distinguish it from other objects with any other similar characteristics.

Despite those remarkable attributes, following a ball from a video is not an easy task, sometimes even for the human eye. In a typical video of a match, the players are usually close to each other and in close contact with the ball, creating frequent occlusions. Moreover, the ball's trajectory is far from linear, because players constantly throw it from one end of the court to the other, they bounce it fast, they dribble... For those reasons, the localization of a ball in an image with vision algorithms is a complex problem. Among the commonly faced difficulties are:

- The small size of the ball and its fast movements and frequent changes of direction make the ball difficult to track. A high speed can additionally create motion blur.
- Complete or partial occlusions make detection of the ball difficult.
- Several objects, such as court lines or parts of the players' bodies, present some similar characteristics to the ball and may mislead our tracking.

From just one view point, the ball is not always visible. The players are constantly passing in front of the ball, grabbing it and, consequently occluding it completely or just partially.

In this work, we attempt to track a basket ball from images coming from multiple fixed cameras. The multi-camera system helps us to maintain a constant view of the ball in spite of occlusions and enables us to determine its 3D position.

By taking advantage of the characteristics of a ball, we track it throughout a match while trying to overcome all the mentioned difficulties. We used a multi-camera system to do it and used the correspondence across views to reject false positives and to determine the 3D position of the ball.

1.1 Objective

The main goal of this thesis is to develop an algorithm which allows a reliable tracking of a ball through a recorded basketball match. This main objective could be reached by following the steps below:

- Detection of the ball in individual camera views using image features.
- Tracking-based approach that recursively update the 3D ball position using view correspondences and image features.
- Extrapolation of the future position of the ball.

In Chapter 2, we talk about our dataset and the properties of the cameras. We also talk about the tools we use to implement all the methods described above.

In Chapter 3, the detection of the ball in individual camera views is explained. When working in individual camera views, we use the color and shape properties of the ball to detect it in an image.

To identify the color of the ball we train the system providing it with samples of the ball for the camera that we use. When the training is done, the image can be scanned for regions of the expected ball color.

When using the shape to detect the ball, we faced the problem of partial occlusions (when the ball is held by players). The shape was only used as a relevant characteristic when it was far from the players. Then we used background subtraction to make the detection stage robust to motion blur.

Chapter 4, deals with the use of the multi-camera views and the search, based on color, size and shape properties, of the 3D position of the ball. We also explain the extrapolation of the ball position in future frames.

The use of the multi-camera view enables us to find the 3D position of the ball based on a tracking approach. We use the previous known position of the ball to define a 3D search window around it and backproject every possible position into all the individual views. Finally, we use the color, size and shape properties together with the correspondence across views to analyze each possible position and evaluate the ball presence at this location. We select the most fitting position as the current ball position.

When tracking the ball across a few frames, we can extrapolate its position in the future frame. We combine the color, shape and position extrapolation to detect the ball.

After following all these steps we are able to track the ball and locate it in the 3D space during several consecutive frames.

Chapter 5 demonstrates the results obtained by the techniques developed during this project. Finally, Chapter 6 concludes this report and gives an outlook to forthcoming future work.

1.2 Motivation

Tracking a basketball in a recorded match can be useful in many ways.

The basketball coach of a team can use it as a reference for analyzing the performance of his team during a match. He might be able to focus on different strategies and to know the weak points of his team.

It might be useful as well for referees to corroborate their decisions in the court.

Even TV shows specialized in basketball could use it to analyze and comment the match with more precision. The presenter could show the players strength or flaws during the whole match. They could also show plots of the ball trajectory to enhance their images.

Finally, it can also be used in combination with a method to track players

to recreate a 3D version of a given basketball match and analyze it from the best point of view.

1.3 Related work

Several game analysis systems are known for soccer games such as [1] , [7], [8], [9] , [10] and [11]. There are also game analysis for other games like tennis table, [3].

A model of a soccer video sequence in 3D is presented in [1]. They calibrate the cameras using the lines of the playground. Then the players are tracked using their 3D coordinates and a correlation-based template matching algorithm. However, they only use one camera to recreate the 3D scene, resulting in an approximation of the three dimensions. Furthermore, since they only use information from one camera view point they only track the players and not the ball which moves in three dimensions.

An algorithm for determining an optimized camera viewpoint for a soccer scene by tracking the ball is described in [7]. They use a multi-camera system to overcome occlusions and find the 3D location of the ball by epipolar geometry. The ball is detected and tracked by template matching and the correspondence across views and frames. For determining the optimized camera viewpoint for each frame, they evaluate the camera selection based on the trajectory of the ball. They select the camera that gives the maximum evaluation.

An algorithm for tracking a soccer ball in a multi-camera system based on particle filtering is demonstraded in [8]. They use the centroid, size and motion-blur of the ball to model the imaging processes and the ballistic motion, bounce and rolling to the model the dynamics. They apply the particle filter in each camera view and then merge them across views to find the 3D position of the ball.

In [9] a multi-camera system to track players and a ball and estimate their positions from video images of a soccer match is presented. The players are tracked by extracting shirt and trousers regions. They resolve occlusions by considering the colors, positions, and velocities of the players in all the views. The ball candidates are estimated using color and motion information, and then the ball position is determined among them based on motion continuity. The ball position is estimated by fitting a physical model of movement in the 3D space of the observed ball trajectory. They use the knowledge of the players position to search for ball candidates (motion regions) in a radial direction from them in the subsequent frames. They are able to track both players and the ball in a sequence of images.

In [10] an algorithm for ball recognition using circle Hough transform and neural classifier is described. They use first the circle Hough transform to detect the region of the image that is the best candidate to contain the ball. Then the neural classifier is applied on the selected region to confirm if the ball has been properly detected or a false positive has been found. They limit the search of the area to only a region of the image. Although they manage to detect the ball in most of the cases, there are also times when they detect false positives. They do not track the ball across frames to select ball candidates. They apply their method to isolated images instead of using a multi-camera view system and corroborate the ball candidates across frames.

The demonstration of a technique for estimating the 3D trajectory of a soccer ball from multiple fixed cameras is presented in [11]. They use several features of the ball to track it, such as: ground plane velocity, longevity, normalized size and color. The selection of ball candidates is performed on the output of the Kalman filters which makes use of velocity information to classify the ball. They corroborate their decisions with a tracking of the ball and matching candidates across views to solve occlusions. Finally, they model the appearance of a moving ball to improve the ball classification problem. They are able to estimate the the ball motion, but they require two 3D ball positions.

Finally, in [3] an algorithm for automatically ranking the interesting segments of a video sequence of a table tennis game is described. They rank accordingly to the table position, the players actions and the ball trajectory. The players are tracked by the changing mask and a trajectory analysis. The ball is first detected using color, shape, size and position limits on the table. Then the motion of the ball and its appearance are used to track the ball by a Bayesian decision framework. This parameters are updated across frames by a Kalman filter and an incremental Bayesian algorithm. They manage to highlight interesting sequences of the video records but they still need to improve the evaluation of these highlights.

Chapter 2

System Overview

In this chapter we present an overview of the arrangement of the system of cameras used to analyze the basketball match. The video sequences and tools used during this project are described as well.

2.1 System environment

A basketball court measures 28×15m. Only half of the court is analyzed due to the fact that the dataset consists of a training match that takes place in one half of the court only. The basketball match is filmed by five fixed cameras installed around one half of the court, at the positions shown on Figure 2.1. All cameras are located four meters above the ground and aiming at the basket, where most of the action takes place.

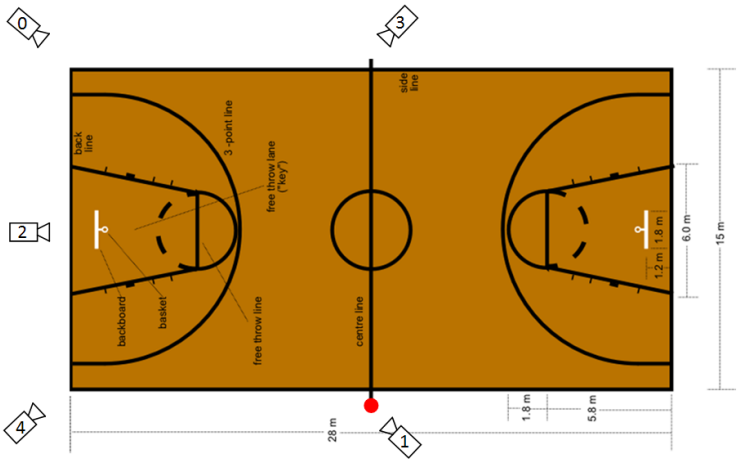


Figure 2.1: Basketball court.

In Figure 2.1, each camera position is indicated with the camera symbol and its respective number. The red point, indicates the origin of the coordinates system, with z=0 at the ground level.

The orientation of the axes used for the location of the cameras and the 3D coordinates of the ball is shown in Figure 2.2.

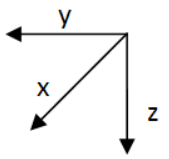


Figure 2.2: Orientation of the axes in the dataset

We use multiple cameras to have several points of view. This way it is possible to retrieve the 3D position of the ball and we handle better the occlusions.

This cameras are fixed so their calibration is known and it exists a correspondence between the 3D real world and the 2D images.

There were some problems with the calibration between cameras. It looks like the cameras (0,2,4) where slightly shifted with respect to cameras (1,3). A small offset has been added to axis y and z to their world coordinates to compensate for the deviation.

2.2 Video sequences

The five cameras in different positions give us five different video sequences with different points of view.

In this project, we work with a compressed version of the original video sequences. All the five video sequences have a length of 2000 frames. They were taken at a frame rate of 25 frames per second with a resolution of 720×576 pixels.

2.3 Tools

The tracking algorithm was implemented in C++. We took advantage of the C++ OpenCV library specialized for Computer Vision due to useful algorithms which it provides, like edge detection, Hough transform for finding circles and 3D coordinates treatment, among many more.

Other implementations done by the CVlab in EPFL were used as well. We used their color histogram data structure that we needed for training the

(a) camera 0; (18,20,-4)

(b) camera 1; (-1,-1,-4)

(c) camera 2; (20,8,-4)

(d) camera 3; (-1,13,-4)

(e) camera 4; (22,-4,-4)

Figure 2.3: The five cameras views and their respective approximated locations in meters. As explained in Figure 2.2, the z-axis is inverted.

algorithm with the color of the ball. The camera's calibration parameters were also provided by them as it was the method for back-projecting 3D world coordinates to the 2D coordinates of each camera view. Finally, we also used their background subtraction video sequence which is synchronized with the video sequences we used as dataset.

Chapter 3

Ball Candidate Detection in isolated images

In this chapter we explain several approaches to localize the ball in single images. The potential ball positions are identified by means of color and shape.

3.1 Ball candidate detection

Before working with the multi-camera views, we focus on testing some approaches in just one camera. We try to detect the ball in a single image of only one camera.

We try different approaches to find the ball in each image. First, we try determining the ball position by its color. But when the ball moves fast, it becomes blurred and part of the background is mixed in the color of the ball. So we also try to find the ball in the image using its shape.

When testing the color and the shape in only one view, we apply the methods to the whole image. In the color method, we analyze one by one all the pixels of the image and compute their likelihood of belonging to the ball. Conversely, on the shape method, we analyze only the contours of the image.

3.2 Color

A basketball is usually orange with some black stripes. Although it can be of other colors, that is the most widely used.

In order to be able to check the color of the ball in a single image, we take some samples of the ball from each camera and create a histogram of

the color of the ball specific for that match. With it we can test if a certain position of the image is likely to be part of the ball or not.

Figure 3.1: Basketball

3.2.1 Initial samples

The color of the ball is subject to the court illumination during the match. Depending on the lighting of the court and the white balance of the cameras, the color of the ball may vary slightly or considerably from one part of the court to the other.

(a) Bad white balance

(b) Good white balance

Figure 3.2: Different lightings of the court

In Figure 3.2, we can observe the differences of light between matches. Figure 3.2a shows a bad white balance. Whereas, in Figure 3.2b, the light is more homogeneous and that everything receives an equal amount of light. Thus, the color of the ball is closer to the orange that you expect from a basketball ball.

Since the light in the court may differ substantially from one match to another, we need a way to know which is or are the colors of the ball for a particular match. We created a color histogram of the ball that could

include all the variations of colors of the ball.

In order to do so, we took manually some initial samples of the ball. Images of just the ball, without any other colors. Then we generated a histogram based on these samples. The histogram is generated analyzing each pixel of all the samples and counting the density of the colors. We consider the density of a color to be the normalized number of times this color was added to the histogram.

The initial samples were gathered separately for every camera, since we cannot assume that all the cameras have exactly the same configuration. The white balance might vary from one camera to the other. Therefore, if different samples are used for different cameras, a different histogram is used for each set of samples.

3.2.2 Determining ball color probability

To identify the regions in the image that have a similar appearance to the ball, we evaluated every pixel of the image and computed its probability of belonging to the color histogram of its respective camera by checking the density of the color of the pixel in the histogram.

We obtain, for every pixel, a probability that its color matches the one of the ball. This approach detected the ball, but also some noise like court lines and player's body parts. We did not try to find the exact position of the ball in the image since there were too many candidates. Instead, we used this approach to know that the color histogram was able to find the ball.

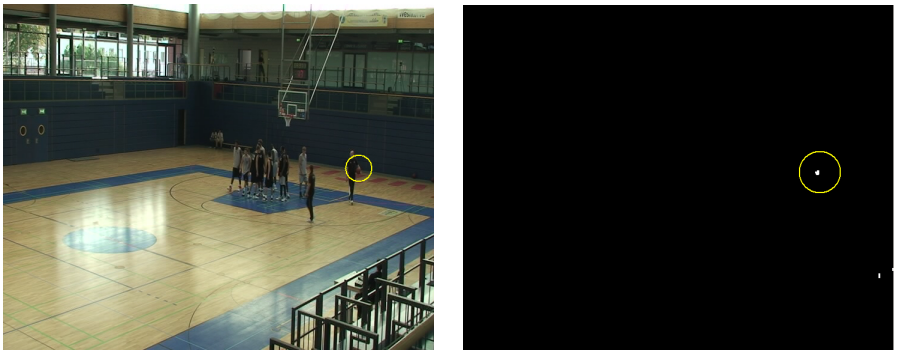
Applying a binary threshold, to eliminate the pixels with low probability, and an open morphological operator, to eliminate the unconnected pixels, we could eliminate most of the noise and have a rough idea of the position of the ball, which would be more precise with the integration of the multi-camera views and their correspondence.

The color approach could find the ball when this one was not blurred. However, when we applied the method to the whole sequence the results were not always as precise as in Figure 3.3.

3.3 Shape

Another approach that we experimented with, was to take advantage of the spherical shape of the ball. Articles such as [3] and [10] have also used this characteristic in order to find a ball in an image.

We used several methods to detect the ball:



(a) Original image

(b) Image after applying the threshold + Open operator

Figure 3.3: In the thresholded image, all the pixels with a probability lower than 20% that are isolated are discarded. This gives us an idea of where is the ball (indicated with a yellow circle), with almost none false positives.

- **Region circularity**: this method was inspired by [3] and consisted of using the variance of the radius of a labeled region to measure its circularity. This approach, though, was not always successful since sometimes the edge detection algorithm would return open contours or would return the ball as more than one contour.

- **Circles detection**: we also tried to apply the Hough Transform for detecting circles, as in [10], to see if the results improved. Although with this approach we could detect the ball in some cases, in other we were completely wrong. The hits percentage was only around 50% of the times.

3.3.1 Detecting edges

To have a rough overview of the shape approach performance we tried to detect the edges in a sample image. We used the Canny [2] edge detector algorithm to detect the edges. We were able to see that the ball could be detected by its shape. An example of ball edges is shown in Figure 3.4.

Unfortunately, when analyzing the whole sequence we realized that the ball contours in the image have not always a circular shape. Sometimes, it has a fuzzy shape that is far from a circle and therefore, the edge detection algorithm has problems detecting it. When a player is holding the ball, it is hard for the algorithm to separate the ball from the player, returning an undefined shape. As we can see in Figure 3.5, the ball might also be partially occluded, losing its round shape.

Figure 3.4: Good shape of the ball. Elliptical shape, due to image aspect ratio, but still close to circular

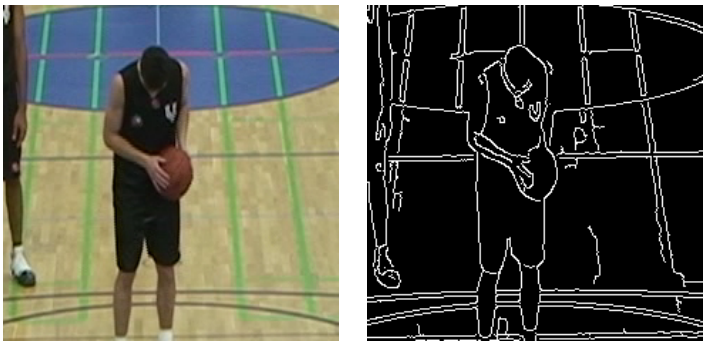


Figure 3.5: Partially occluded ball. The ball loses its circular shape

The worst cases are when the ball is moving fast and has a blurred elliptical shape. It is then, practically invisible to the edge detection algorithm. For those cases we used background subtraction to facilitate the detection of the ball to the Canny edge detection algorithm.

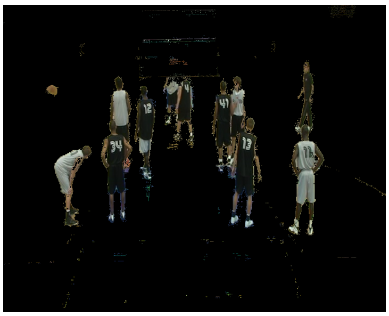
We used a binary video sequence where the background is black and the foreground white, illustrated in Figure 3.6b, in order to know which pixels belong to the background and which foreground. To avoid losing the shape of the ball when it is in contact with a player, we modified the original image and subtracted the background. In the original image we just put to black all the pixels that where also black in the binary video sequence. With that, we had the images without the background, but with the foreground in color, as shown in Figure 3.6c.

The combination of the background subtraction with edge detection worked very well in the cases where the ball was not close to a player. In Figures 3.4 and 3.5, no background subtraction is used.

Despite the above mentioned problems with the shape, we thought that it could still complement the color detection method and be helpful in the

(a) Original image

(b) Image from the binary video sequence



(c) Original image with background subtraction

Figure 3.6: Background subtraction sequences

cases where the color algorithm was not returning any satisfactory response. The shape detector was meant to help when the ball was not held by a player and it was moving fast, that is when the ball was thrown or was in the process of being bounced as in Figure 3.7

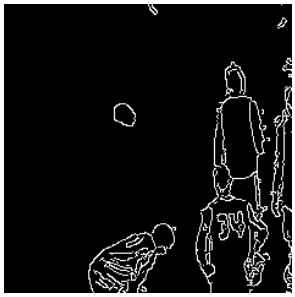
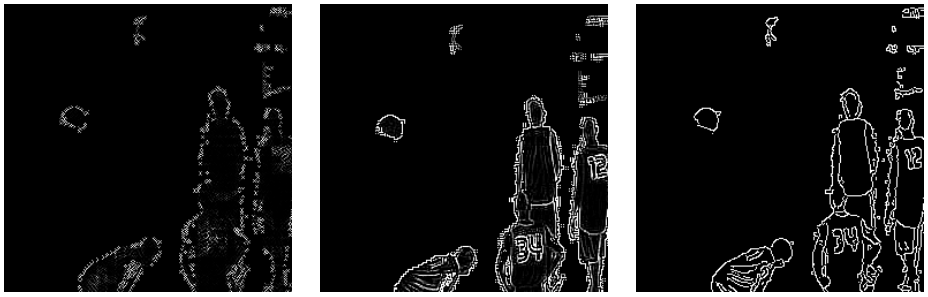


Figure 3.7: Shape detection when the ball is not close of any player. Background subtraction is used for the edge detection.

Canny edge detection

Several edge detection methods were compared to decide which one was the best for our situation. We compared the results given by the Sobel [12], Laplacian and Canny [2] edge detection algorithms provided by OpenCV. After trying all of them with different parameters, the Canny edge detection was the most accurate one. Shown in Figure 3.8, each edge detector method with the background subtraction.



(a) Sobel (b) Laplace (c) Canny

Figure 3.8: Edge detection algorithms with background subtraction

The Canny edge detector was developed by John F. Canny in 1986 and uses a multi-stage algorithm to detect edges in images.

The algorithm first smooths the image convolving it with a Gaussian filter to eliminate noise and avoid meaningless edges. The result is a slightly blurred version of the original image without the noisy pixels.

An edge in an image may point in a variety of directions, so the Canny algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image. An edge detector operator (Sobel in OpenCV) is used to compute the first derivatives in x and y, then they are combined into four directional derivatives. The algorithm then tracks along these derivatives and suppresses any pixel that is not at the maximal (non-maximum suppression). The points where these directional derivatives are local maximal are then edge candidates.

Intensity gradients which are large are more likely to correspond to edges than if they are small. It is in most cases impossible to specify a threshold at which a given intensity gradient switches from corresponding to an edge into not doing so. Therefore Canny uses thresholding with hysteresis. This means that there are two thresholds, an upper and a lower. If a pixel has a gradient larger than the upper threshold, then it is accepted as an edge pixel; if a pixel is below the lower threshold, it is rejected. If the pixel's gradient is between the thresholds, then it will be accepted only if it is connected to

a pixel that is above the upper threshold. Canny recommended a ratio of up:low threshold between 2:1 and 3:1.

Once this process is complete we have a binary image, where each pixel is marked as either an edge pixel or a non-edge pixel.

3.3.2 Region circularity

The first approach that we implemented to detect the shape of the ball in an image of contours is based on the circularity of the ball. As in [3], we tested the circularity of a labeled region to quantify its similarity with a circle. We obtained the contours image by applying Canny edge detection, as explained earlier. OpenCV has a method to segment all disconnected shapes of the canny edge detection into labeled regions.

$$\mu_j = \frac{1}{N_j} \sum_{i=0}^{N_j-1} \|(x_i, y_i)_j - (c_x, c_y)_j\| \tag{3.1}$$

$$\sigma_j = \frac{1}{N_j} \sum_{i=0}^{N_j-1} \Big[\|(x_i, y_i)_j - (c_x, c_y)_j\| - \mu_j\Big]^2 \tag{3.2}$$

In (3.1) we compute the mean μ_j of all the radius of the contour j from its mass center, where $(x_i, y_i)_j$ is the position of the pixel i of the contour and $(c_x, c_y)_j$ its mass center. In (3.2) we compute the variance of the radius σ_j of the ball candidate j. In both equations N_j is the total number of pixels of the contour j. In this case we consider the mass center of the labeled shapes return by OpenCV as a ball candidate.

We applied this method to the whole edge image and filtered which labeled blobs from the image were ball candidates by their radius variance and area. With this, only blobs with a small radius variance and an area close to the area of the ball in the image would be considered ball candidates. We made a manual estimation of the radius of the ball by the size of the labeled blobs that were selected as ball candidates. In images like Figure 3.7 we can see that it detects the ball.

We first tried this method in a short sequence where the ball was far from the players, like in Figure 3.7. We managed to detect the ball on that sequence. We used the algorithm with a sequence where the ball was clear and without players in the middle. However in some cases the ball is formed by several blobs and the algorithm cannot find it.

3.3.3 Circle detection

There was still another approach that we wanted to test: the Hough transform to detect circles. We tried to find the ball by applying this algorithm to see if we could obtain better results.

The Hough transform [5] [4] is an algorithm used to find straight lines in an image. There is a variant of it [6] that enables the user to detect circles instead. We used the OpenCV implementation of this latter algorithm to search for ball candidates in the image.

When the Hough transform has finished processing the image, it returns the center position and radius of all the circles that it found. We could use this information to find ball candidates in the image

The Hough transform algorithm provided by OpenCV can be customized to one's needs through several parameters:

- The edge's image you want to analyze;
- Resolution of the accumulator used to detect centers of the circles;
- Minimal distance between centers of the circles;
- Upper threshold used in the Canny edge detector;
- Accumulator threshold at the center detection stage. The smaller it is, the more false circles may be detected;
- Minimum radius of the circles;
- Maximum radius of the circles;

We define an estimation of the size of the circles we wanted to find and at which distance we would find one from the other. This filters a lot of false positive in the algorithm.

Despite our efforts, when applied to a whole image, the algorithm returned too many circles. We needed to somehow restrict the search area and to classify the results in order to use this approach.

Chapter 4

Ball Tracking in Multiple Cameras

In this chapter we present several approaches to localize and track the 3D position of the ball throughout the court.

A tracking approach that updates the 3D ball position from frame to frame is used. We start with the previous known position of the ball and search into a 3D search volume. Inside this 3D search volume, we project every 3D position candidate into the camera views and evaluate its likelihood in all views using color, size and shape cues. We also tried to anticipate the future position of the ball based on its last known positions, by assuming that the ball is moving in a straight line at a constant speed.

4.1 Estimating 3D position of the ball

Using our multi-camera system, we want to find the position of the ball in three dimensions. The ball might be perfectly visible in all cameras at once, only in some of them or it might be partially occluded in others. We have considered two possible approaches for finding the 3D position of the ball from the camera views:

- **Detection approach:** This is the approach that we used in the previous Chapter 3. It consists in identify ball candidates in every view independently and try to match them across views.

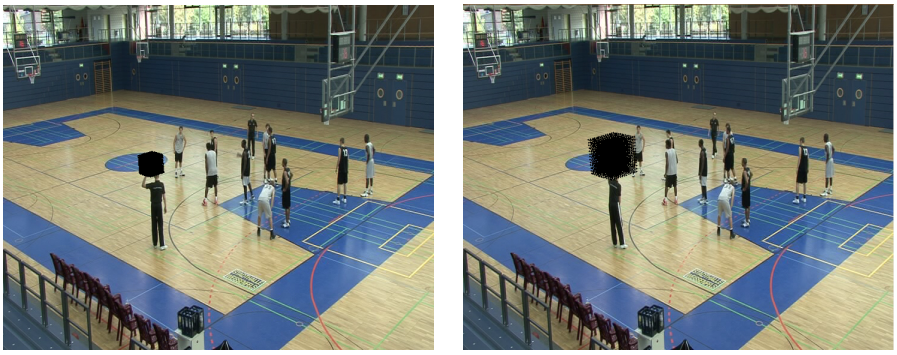
  This approach did not work well because there were too many ball candidates to find their respective match in the other views, and most of the time the ball was partially or completely occluded in some of the views.

- **Tracking-based approach:** In this approach, we consider potential 3D ball positions inside a search volume, back-project them in every view and evaluate the presence of a ball at these positions in the images.

  For this approach, we rely on temporal consistency: the starting point is the ball position in the previous frame. We follow several steps:

  1. We manually initialize at the beginning of the sequence the starting 3D position of the ball.
  2. From that position we create a 3D search volume window, that has a cubic shape.
  3. We test one by one different positions inside this cube. For every position, we project its representation in each camera view.
  4. Finally, we analyze the likelihood of the projected position in all views using color and shape descriptors.

  This way we do not lose the correspondence between the real world and the image. Furthermore, we reduce the volume of candidates across views to only one set per view. This leads to a higher precision in our results than with the detection approach.



(a) Small window, $(60 \times 60 \times 60cm)$ (b) Big window, $(120 \times 120 \times 120cm)$

Figure 4.1: Different search windows with different step sizes

As shown in Figure 4.1, we defined two search window sizes: a small and a big one. The small search window is the one by default. Its size is just big enough to contain the ball movement, even when it moves fast. In fact, the window will be three times the size of the ball to be able to find it even when it moves fast. However, when the ball is lost, with this size of search window it was very difficult to find the ball again. So, when the reliability

of the algorithm decreases below a confidence threshold, and the probability that we lose the ball is high, we increased the search window. This way if the ball moves fast and is lost, we might be able to re-find it.

The search window has been restricted to be inside a region slightly bigger than the court, so if the ball goes off limits we will not lose it. We also assume that the ball will neither go higher than eight meters nor below the ground. Also, we remove from computation all the cameras in which the ball is not in sight.

4.2 Combining different approaches

We first implemented each approach – color, shape and motion prediction – separately. But only when used in combination they should operate best, since each algorithm can compensate for the drawbacks of the others. So we combined them through a Bayesian decision framework and choose the ball candidate with the maximum likelihood

First we evaluated the likelihood of the appearance methods -color and shape-, before combining them with the extrapolation of the next position of the ball.

$$P(B_i = 1|\theta) = \frac{P(\theta|B_i = 1)P(B_i = 1)}{P(\theta)} \tag{4.1}$$

Here θ represents the image information (pixels) of the ball at location i and B_i is the random variable standing for the presence of the ball in the position i. We can assume that $P(B_i = 1)$ is constant since the probability of a region to contain the ball is uniform in the whole image and $P(\theta)$ is also constant since the ball is always the same. So we have:

$$P(B_i = 1|\theta) \propto P(\theta|B_i = 1) \tag{4.2}$$

We combined the likelihood of a ball candidate B_i of having the shape of the ball with the likelihood of a ball candidate B_i of having the color of the ball, Equation (4.3).

$$P(B_i = 1|\theta) \propto P(ballColor|B_i = 1)P(ballShape|B_i = 1) \tag{4.3}$$

Then we combined the appearance methods $P(\theta|B_i = 1)$ with the extrapolation of the next position of the ball $P(B_i = 1|B_{ext})$.

$$P(B_i = 1|\theta, B_{ext}) = \frac{P(\theta|B_i = 1)P(B_i = 1|B_{ext})}{P(\theta|B_{ext})} \tag{4.4}$$

In Equation (4.4), B_{ext} represents the extrapolated position of the ball regarding its last position and velocity. B_{ext} does not depend on i, the actual position of the ball, and θ is independent from the previous ball location.

$$P(B_i = 1|\theta, B_{ext}) \propto P(\theta|B_i = 1)P(B_i = 1|B_{ext}) \tag{4.5}$$

We selected the ball candidate with the combined maximum likelihood. Where i represent all the ball candidates and we choose the maximum.

$$\arg\max_i P(B_i = 1|\theta, B_{ext}) = \arg\max_i (P(\theta|B_i = 1)P(B_i = 1|B_{ext})) \tag{4.6}$$

This way, we wanted to overcome the cases when the color and the shape methods were losing track of the ball. The extrapolation method would serve as guidance to the appearance methods to know which is likely to be the position of the ball.

4.3 Color

In the multi-camera tracking-based approach, we adapt the color method mentioned in Section 3.2 by improving the color histogram of the ball and the algorithm itself. During official matches the ball has a known standard size, usually 23 or 24 cm of diameter. We take advantage of the color and size characteristics to find the ball in the images.

We use the previous method of using some initial samples of the ball for each camera to create a color histogram. But we apply the method only to one ball candidate at a time. We evaluate its likelihood of belonging to the ball, and choose the ball candidate with the maximum likelihood.

First, we improve the previous algorithm by only selecting candidates that would have enough surrounding orange pixels within the radius of the ball.

Second, we adapt the color histogram to the ball throughout the video sequence, and not only to a set of initial samples. So, once we are able to find the ball in the images, we take new dynamic samples for every ball that we find, only from those images where the confidence in the ball is high. Our color histogram evolves through time and becomes more accurate as it absorbs all the previous pixels that formed the ball.

4.3.1 Ball radius in each camera

As mentioned previously, the diameter of the official basketball ball is 23 or 24 cm. We can use the camera calibration along with the real size of the ball to compute the size of the ball projection in the image. We will then evaluate the color of those pixels within the radius. That way, noise like court lines or players shoes can be filtered.

$$dist_i = \|(pos_x, pos_y, pos_z) - (cam_x, cam_y, cam_z)_i\| \tag{4.7}$$

$$radius_i = ballRadius \left(\frac{focalLenght_i}{dist_i} \right) \tag{4.8}$$

First, in (4.7), the distance between the ball (pos_x, pos_y, pos_z) and the $camera_i = (cam_x, cam_y, cam_z)_i$, is computed. This distance allows us to determine the radius of the ball (4.8) in the image. $ballRadius$ is the real radius of the ball and $focalLenght_i$ is the focal length of the camera i. We use a radius of $ballRadius = 12cm$.

4.3.2 Selection of a ball candidate

Once we have the color histogram trained for the ball in each camera, We determine the likelihood of every ball candidate to belong to the ball. We select the ball candidate with the maximum likelihood. We use the multi-camera tracking-based approach explained in section 4.1.

The likelihood is computed by accumulating all the density probability of every pixel in the radius of the ball, Equation (4.9). The density probability of a pixel is the normalized number of times that the color of the pixel has been introduced in the color histogram.

$$L(a_1 \dots a_n|\theta) = \sum_{j \in n}^{n} \log P(a_j|\theta) \tag{4.9}$$

In Equation (4.9), θ represents the ball, $a_1 \dots a_n$ are all the pixels that we take into account for evaluating their likelihood, with $n = \pi r_i^2$. r_i is the radius of the ball in camera i, $P(a_j|\theta)$ returns the density probability that the pixel(a_j) belongs to the ball.

This density is then normalized, Equation (4.10), so a ball closer to one camera has no bigger weight in the final decision of the position of the ball. The n represents again all the pixels that we take into account, $n = \pi r_i^2$.

$$\bar{L}(a_1 \dots a_n|\theta) = \frac{L(a_1 \dots a_n|\theta)}{n} \tag{4.10}$$

In Equation (4.11), we sum the normalized color density of every camera, where the ball might be in sight and normalize it. N indicates the number of cameras where the ball might be in sight. This does not mean that the ball is not occluded, it just means that the ball candidate is in the camera view.

$$\bar{l}(ballColor|B_i = 1) = \frac{\sum_{i=0}^{N} \bar{L}(a_1 \dots a_n|\theta)}{N} \tag{4.11}$$

Finally, the values of the probability of the ball candidate B_i to belong to the the color of ball $ballColor$ range from 0 to $-\infty$. We translate them to a range from 0 to 1, the latter one being the maximum. This way, we can combine the color method with others and all will be in the same range of values, Equation 4.12.

$$P(ballColor|B_i = 1) = \exp(\bar{l}(ballColor|B_i = 1)) \tag{4.12}$$

4.3.3 Dynamic samples

We use the initial samples to choose the first ball candidate. But, once this candidate position is chosen we can add new samples of that ball to the histogram.

We have to check two things before deciding to add or not the new samples. First, we just add samples of the ball candidates whose likelihood score are above a confidence threshold. Only then, we can analyze every pixel of the ball candidate inside the radius of the ball.

Second, we need to know in which cameras the ball is visible and in which it is not. Otherwise, we would start adding color samples to the histogram that might not form part of the ball. Then, if the probability of belonging to the ball of each considered pixel is very high (with a confidence of 80%) we consider that pixel as part of the ball. Which means that the pixel is probably not occluded in that camera. We can add that pixel to the color histogram of the ball, of the respective camera, without worrying that it will tint the color histogram of that specific camera.

That way, the color histogram of the ball is more accurate as time goes, resulting in a good representation of the color of the ball throughout the game.

4.3.4 Color problems

The tracking based on color works well when the ball is moving rather slow. It detects the ball with a precision of ±10cm in the 70% of the tracked

frames. But when the game begins, the ball starts moving very fast and gets blurry. When this happens the color of the background gets mixed with the color of the ball and the histogram sometimes stops recognizing those pixels as part of the ball.

(a) Good color of the ball

(b) Color of the ball mixed with the background

Figure 4.2: Normal Color of the ball and the ball when it is blurred and its color is mixed with the background

This problem would be minimized if the shutter speed of the cameras was faster. It is also possible that it does not exist in another dataset.

4.4 Shape

We adapted the shape method explained in Section 3.3 in the multi-camera tracking-based approach by introducing the ball candidates positions and the correspondence across views.

We improved the method mentioned in Section 3.3.2, that measured the circularity of a contour based on the variance of its radius, by selecting as center of the contour the candidate position of the ball, instead of its mass center, and see if there would be enough contours with low variance radius to recreate the shape of the ball

4.4.1 Region circularity for broken contours

The method described in Section 3.3.2, namely using the circularity of a labeled contour image to detect ball candidates, had a low reliability. The labeled blobs that the Canny edge detection algorithm returned were sometimes just fragments of the ball. In other words, the ball would be represented by several blobs. This had an impact on the variance and the area. The ball detection was no longer accurate.

(a) Ball mixed with the hand of the player

(b) Deformed shape

(c) Ball formed by several blobs

Figure 4.3: Deformed ball

To improve this method when the contours of the ball were broken into several blobs, we decided to change the center of the blob. Instead of considering the mass center of the blob to compute the variance, we took the ball candidate position as the center and looked in a certain radius range if there where some blobs. We considered the variance of the radius in Equation (3.2) according to the center of the ball candidate.

The area is not needed in this case, because we know in advance what the radius of the ball must be. So, as opposed to Section 3.3.2, we are just checking if, there is a circle of a given radius at a given position.

The radius range from the ball candidate (the supposed center) to the blobs was determined by the mean of the radius from the ball candidate to the blob. This mean had to be bigger than half of the real radius of the ball in the image and smaller than the double of it, as shown in Equation (4.13). μ_i is the mean of the radius from the ball candidate to the blob and r_i the radius of the ball in the image i.

$$\frac{r_i}{2} < \mu_i < 2r_i \tag{4.13}$$

4.4.2 Selection of a ball candidate

To evaluate the likelihood of each ball candidate, we rated the results depending on how similar to a circle and how close to the actual radius of the ball the labeled blobs were.

$$L(\theta|B_i = 1) = log(\mathcal{N}(\mu_i, ballRadius_i, \sigma_1)\mathcal{N}(v_i, 0, \sigma_2)) \tag{4.14}$$

In Equation 4.14, the likelihood of a ball candidate B_i belonging to the ball θ is maximum if the encountered blobs around the ball candidate have a very low radius variance v_i and its mean radius μ_i is close to the

real radius of the ball $ballRadius_i$ in the image i. $\mathcal{N}(\mu_i, ballRadius_i, \sigma_1)$ is a Gaussian function centered at the radius of the ball in the image with $\sigma_1 = ballRadius_i/2$ that evaluates how close the mean radius of the blobs μ_i is from the actual radius of the ball in the image. $\mathcal{N}(v_i, 0, \sigma_2)$ is another Gaussian function that evaluates how close to 0 is the radius variance of the labeled blobs selected as part of the ball. We choose a $\sigma_2 = 2$ since only want those ball candidates with a very low variance.

$$P(ballShape|B_i = 1) = \exp\left(\frac{\sum_{i=0}^{N} L(\theta|B_i = 1)}{N}\right) \tag{4.15}$$

The probability that the ball candidate B_i belongs to the shape of the ball *ballShape* for each camera is added and normalized, where N is the number of cameras where the ball is in view.

4.4.3 Shape problems

We tried to evaluate the likelihood of the ball candidates by measuring its circularity and size. But the results where not satisfactory enough. It was hard to say when all the encountered blobs where in fact a ball or something else (like a head). In the end, most of the time the algorithm considered the heads of the players more circular than the ball itself.

4.5 Motion

The movement of the ball in a basketball match is difficult to predict. Its direction and speed are constantly changing. Nevertheless, we consider the past trajectory of the ball and its velocity to try to extrapolate which would be its next position.

4.5.1 Extrapolation of the next ball candidate

We extrapolate the ball candidate by analyzing the latest positions of the ball and calculate a weighted mean of them. In Equation (4.16), K is the number of past frames that we consider. We set $K = 10$ in order not to look too far into the past. The weight is $w_K = 5$ at the closest past frame and every frame is decreased by a factor of 0.7, so the latest position of the ball have more weight.

$$V_{t-1} = \frac{\sum_{i=K}^{0} w_i(posball_i - posball_{i-1})}{\sum_{i=K}^{0} w_i} \tag{4.16}$$

The extrapolation of the ball position at frame t is computed by adding the velocity V_{t-1} to the position to the ball B_{t-1}.

$$B_{ext} = B_{t-1} + V_{t-1} \tag{4.17}$$

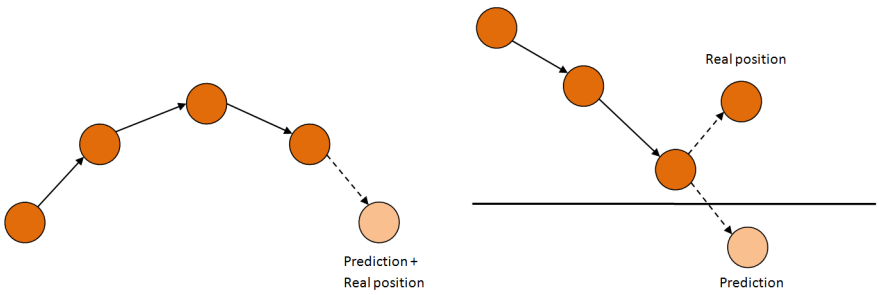
4.5.2 Evaluation of the candidate's space

In order to evaluate the likelihood of the space of the ball candidates, we weight the space of potential candidates with a Gaussian centered around the extrapolated position of the ball. If the candidate is the same as the extrapolation then the likelihood is maximum and the likelihood decreases as the candidate's position gets away from the extrapolated one.

$$P(B_i = 1|B_{ext}) = \mathcal{N}(B_i, B_{ext}, \sigma) \tag{4.18}$$

In Equation (4.18), $P(B_i = 1|B_{ext})$ is the probability that ball candidate B_i is in the extrapolated next position of the ball. B_{ext} represents the extrapolated position of the ball knowing the last position of the ball B_{t-1} and its velocity V_{t-1}. $\mathcal{N}(B_i, B_{ext}, \sigma)$ is the Gaussian centered in the extrapolated position of the ball B_{ext}, B_i is the ball candidate that we evaluate and σ is the standard deviation of the Gauss function.

4.5.3 Motion problems



(a) Good prediction (b) Wrong prediction

Figure 4.4: Schematic picture of different movements and predictions of the ball

In Figure 4.4a, we can see the ideal prediction of the next position of the ball. But it is not always as easy to predict. In Figure 4.4b, we can see that our extrapolation would be the opposite of the real movement of the bouncing ball. This latter one could be more or less easy to redirect, since we know that the ball cannot go below the ground.

But there are some cases where the ball is not so easy to predict. For instance, when a player has the ball and suddenly changes its direction. In those cases this method based on the previous direction and velocity of the ball will never predict it right.

Although the hypothesis was that it would predict more or less when the ball was in the same direction and constant velocity, the reality is that the ball is bouncing most of the time. This means that the prediction is wrong too often to use this method even as a support to the others.

Chapter 5

Results

In this chapter we show the results achieved in Chapter 4, by the color model alone. In a first part, we explain the procedure that we used to obtain the results. Then we present and analyze those results.

5.1 Tracking procedure

In Section 4.3, we explained how the color method works and decides which ball candidate to choose. Here we show how the algorithm operates and how we managed to achieve our results.

The algorithm is initialized manually with the position of the ball in the first frame. From that position it searches in all three dimensions in a window of $60 \times 60 \times 60cm$ with a resolution of $5cm$. We project the 3D world coordinates of the ball candidate to the image coordinates of each camera. With this, we obtain the ball center in each camera that we want to analyze.

5.2 Tracking performance

The best performance was obtained by the color method alone. We are able to track the ball during most of the frames of the video sequence from our data set. In Figure 5.1, the tracked movements of the ball until frame 1200 are plotted. The ball starts at the initial position, mentioned previously, and moves around the court. We can clearly see its route. After the frame 1200 the ball is still tracked with a reinitialization of the tracker, but it bounces and moves, making it difficult to read a plot if we draw all the movements.

To track the ball, we gave a numerical score for every candidate position and selected the maximum of them as the ball position. This score computation is explained in Section 4.3.2 in detail and shown in Figure 5.2. The

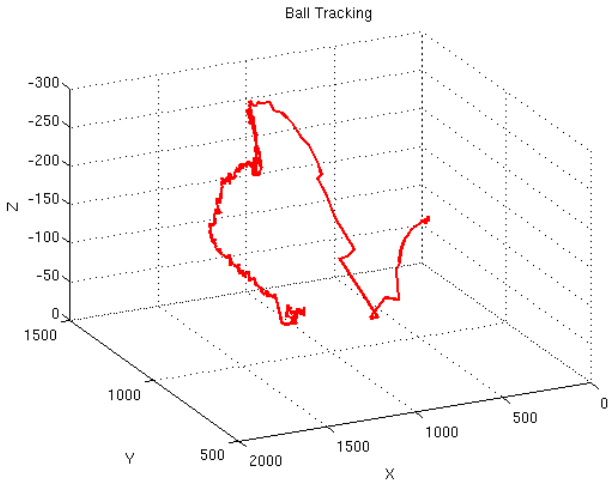


Figure 5.1: Ball tracked position in centimeters from frame 1 until frame 1200.

score is normalized by all the cameras. In some of the cameras the ball is occluded, or partially occluded and in some frames the color of the ball is mixed with the background, resulting in a very low score value. We can observe that after frame 760 (nearly at frame 800), the values of the score decrease and the reliability of the algorithm is diminished as well. However, from time to time the confidence in finding the ball increases again, shown as the spikes from frame 1200 and 1400 for instance. Despite we track the ball the whole sequence, sometimes, the confidence of finding the ball is so low that there is numerically no difference between tracking the ball and not tracking it. We consider that we track the ball when the likelihood is above 0.01.

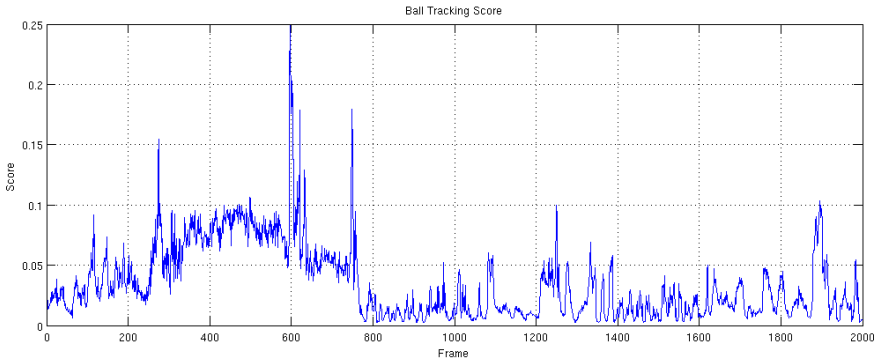
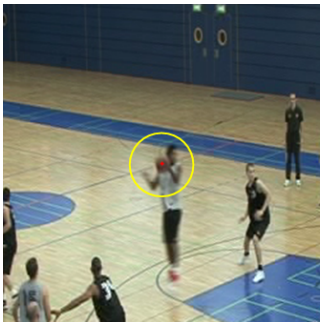
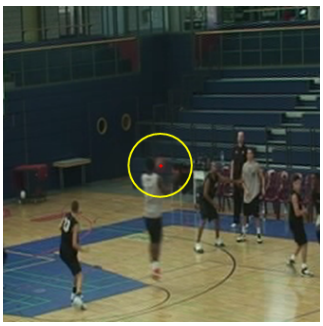


Figure 5.2: Ball tracking score of all the sequence with the reinitialization of the tracker.

In Figure 5.3, we can observe that the ball is being tracked in all five cameras, even if in camera 4 the ball is mostly occluded.



(a) camera 0



(b) camera 1

(c) camera 2

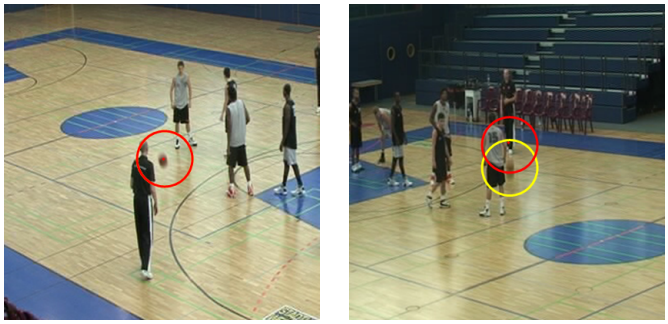
(d) camera 3

(e) camera 4

Figure 5.3: Frame 960 of the sequence, the red point indicates the best ball candidate that the color algorithm found. Position of the ball: $(890, 1063, -245)cm$ with a score of 0.0202. A yellow circle surrounds the ball.

We lose confidence in finding the ball when the game starts, since until then the ball is being carried gently and slowly by just one player. At frame 760 that player puts the ball in motion and we have a lower confidence on the position of the ball.

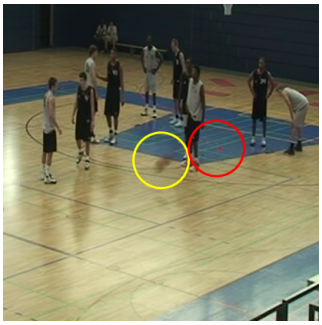
One of the cases when we lose the ball is in Figure 5.4. We can observe



(a) camera 0

(b) camera 1

(c) camera 2



(d) camera 3

(e) camera 4

Figure 5.4: Frame 779 of the sequence, the red point indicates the best ball candidate that the color algorithm found. Position of the ball: $(765, 1208, -85)$ with a score of 0.0034. The tracked ball is surrounded by a red circle whereas the real position of the ball is surrounded by a yellow one.

that from camera 0 the ball is clear enough for the algorithm to find it. But unfortunately, the ball is moving in the same axis as the focus of the camera, resulting in the optical effect that the ball is not moving. Only in camera 2, 3 and 4 the ball is moving in the axis perpendicular to the camera, but the ball is not visible in camera 2. For camera 1, 3 and 4 the ball is completely blurry and it is hard to see even for human eyes, this makes the tracker

mistake the the background and the foreground of the image, losing track of the ball. This blurriness makes the tracker have a low confidence in the selected ball candidates, losing them from time to time, but still keeping track of them most of the sequence.

Once the ball is lost the algorithm starts selecting candidate positions of the ball without enough reliability, and it looks like is moving randomly throughout the court. Sometimes, the lost selected candidate would get close enough to the ball in some of the views and start following it again. For analyzing the performance of the color method we manually reinitialize the tracked every time that the ball is lost. We reinitialized the tracker a total of 12 times during all the video sequences. Every time that we reset the tracker the ball is found again during several more frames.

We analyzed the precision of the tracker by comparing the distance of the tracked position of the ball with its real position. The real position of the ball has been manually determined. A sample every 25 frames has been taken to estimate this distance. A total of 80 samples have been taken throughout all the video sequence, all different than the manually reset of the tracker.

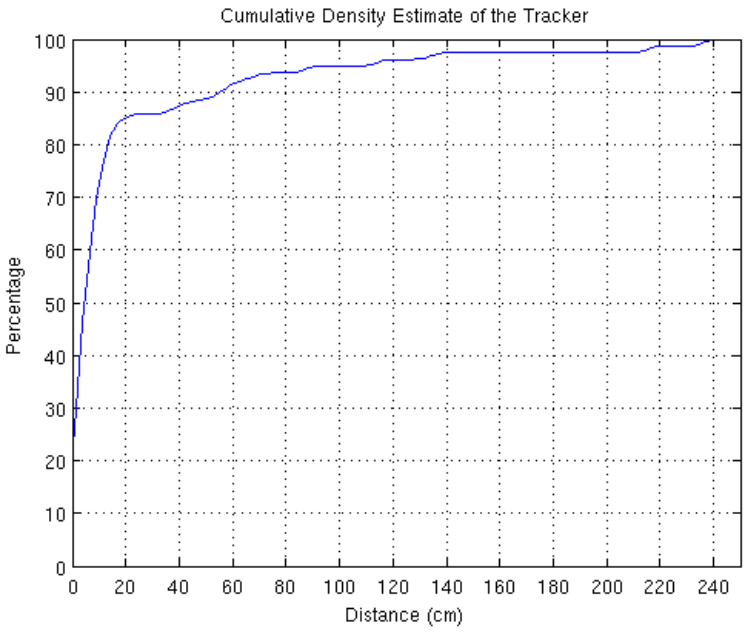


Figure 5.5: Cumulative density estimator.

The Figure 5.5 illustrates the cumulative density estimate of the precision of the tracker. We obtained it through the Matlab method Ksdensity, which

is a non-parametric method used to estimate the smooth cumulative density probability of a histogram. This cumulative density estimator shows that in the 70% of the samples the distance between the tracked position of the ball and its real position is below 10cm.

The color algorithm can track the ball during several frames, but is affected by several factors, such as blurriness of the ball. This factors can be addressed by changing the shutter speed of the camera. Nevertheless we tried to suppress them by complementing the algorithm with the shape and motion methods. But the performance did not improve. In the end we only considered the performance of the color method that was able to track the ball the whole sequence, with some manual reinitialization of the tracker.

Chapter 6

Conclusions

6.1 Conclusions

In this report, we have developed an algorithm for tracking a basketball during a recorded match. We have developed various techniques to track the basketball based on color, size, shape and motion prediction.

We first tried these techniques on isolated images obtained from a single camera to evaluate their robustness. Once they returned satisfactory results we extended them to a multi-camera environment.

From all the experiments that we performed, we observed that color-based search exhibits a good performance. However, it is affected by factors such as lighting, occlusion and also at times blurriness.

The combination of the color method with the size of the ball, can track the ball most of the frames of the video sequence. Its main drawback is that it is affected by the blurriness of the motion of ball, which can be addressed by changing the shutter speed of the camera. This blurriness causes the algorithm to lose track of the ball and also makes it difficult to track its position accurately.

In order to overcome the drawbacks such as blurriness and fuzziness faced while using the color-based descriptor and to obtain additional information, we used the other descriptors i.e. the shape and motion prediction descriptors. However, the color-based method alone obtained the best performance.

6.2 Future work

Among the future work that can still be done about our basketball tracking algorithm, one interesting feature would be to implement a detection method

for the case when the ball is lost. It would be useful to have a method looking at the whole image for the ball, to try to reinitialize the tracker.

It could also be useful to use the sound of the ball to know when it touches the ground. In fact, the ball makes a very characteristic sound that could be easily recognized. This way we could reduce the z dimension in the search space of our tracker to look for the ball since the ball would be on the ground.

Other ball trackers like [3] and [11] use the Kalman filter to have a correspondence across frames. Although the Kalman filter is intended for tracking algorithms, it might not work well due to the unpredictable movement of the ball. So, it could be also improved by the particle filter, just like in [8], which might work better with the sudden changes of direction of the ball.

Finally, another way of improving our algorithm would be to use an optical flow algorithm to detect the movement on the video sequence. This, combined with an algorithm to track the players, might enable us to differentiate the ball from the players and obtain a better estimation of the ball position.

Bibliography

[1] T. Bebie and H. Bieri. **SoccerMan - Reconstructing Soccer Games from Video Sequences**. pages 898-902, 1998.

[2] J. Canny, **A Computational Approach To Edge Detection**. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679714, 1986.

[3] W. Chen and Y. Zhang. **Tracking Ball and Players with Applications to Highlight Ranking of Broadcasting Table Tennis Video**. In *IMACS Multiconference on "Computational Engineering in Systems Applications"(CESA)*, pages 1896-1903, October 2006.

[4] R.O. Duda and P. E. Hart. **Use of the Hough Transformation to Detect Lines and Curves in Pictures**. In *Comm. ACM*, Vol. 15, pp. 1115, January, 1972.

[5] P.V.C. Hough. **Machine Analysis of Bubble Chamber Pictures**. In *Proc. Int. Conf. High Energy Accelerators and Instrumentation*, 1959.

[6] C. Kimme, D. H. Ballard, and J. Sklansky. **Finding circles by an array of accumulators**. In *Communications of the Association for Computing Machinery*, 18: 120122. 1975.

[7] K. Matsumoto, S. Sudo, H. Saito and S. Ozawa. **Optimized Camera Viewpoint Determination System for Soccer Game Broadcasting**. In *IAPR Workshop on Machine Vision Applications*, 3(22):115-118, November 2000.

[8] T. Misu, A. Matsui, M. Naemura, M. Fuji and N. Yagi. **Distributed Particle Filtering for Multiocular Soccer-Ball Tracking**. In *ICASSP*, 3:937-940, 2007.

[9] Y. Ohno, J. Miura and Y. Shirai. **Tracking Players and Estimation of the 3D Position of a Ball in Soccer Games**. pages 145-148, 2000.

[10] T. D'Orazio, C. Guaragnella, M. Leo and A. Distante. **A New Algorithm for Ball Recognition Using Circle Hough Transform and Neural Classifier**. In *Pattern Recognition*, 37:393-408, 2003.

[11] J. Ren, J. Orwell, G.A. Jones and M. Xu. **Tracking the Soccer Ball Using Multiple Fixed Cameras**. In *Computer vision and Image Understanding*, 113:633-642, 2008.

[12] I. Sobel and G. Feldman. **A 3x3 Isotropic Gradient Operator for Image Processing**. In *R. Duda and P. Hart (Eds.), Pattern Classification and Scene Analysis*, pp. 271272, New York: Wiley, 1973.