Basketball computer vision system for automatic detection and tracking of players and ball

Serban Cristian Tudosie, Francesco Vannoni

Abstract– Tactics and statistics in professional basketball teams are widespread. This operation can be optimized and speed up by an automatic computer vision system. We aim at developing a system capable of action tracking and understanding in basketball games using computer vision approaches and deep learning models such as Detectron2.
Our system tracks player trajectories from videos and rectifies them to a standard basketball court, showing also the player who owns the ball.

1 INTRODUCTION

The approach we have chosen deals at first with an analysis of the court of the game. In order to achieve this we need to find a panorama of the court obtained through mosaicing which helps us in extracting the bird's eye view and in warping each video frame into the court.
The panorama gets enhanced by averaging over it multiple frames. This allows us to have cleaner lines of the court and to remove occlusions.
After the court has been pre-processed, we detect the ball with template matching on detected circles [1]. Once the ball is detected, a tracking procedure is activated(using *'CSRT Tracker'*).
Finally we detect the players through Detectron2 [2], we split them into two teams by using team color and we isolate the referee. Then we plot the players into a 2D map of the court in their corresponding position.

2 COURT RECTIFICATION

To rectify the court, our framework uses an action of the given game where the camera only moves in one direction and the corners are included in at least one frame.

2.1 Panorama

Firstly, we have generated a panorama from video frames. In order to achieve this we have use the SIFT [3] algorithm iteratively to compute correspondences between one frame and the panorama obtained up to this point. This allows us to find an homography and to stitch the images together.
Some problems had to be faced:

1. The first frame had to start from the center of the court, to maintain the symmetry between the left and the right side of the court.

2. Some of the lines delimiting the court were occluded due to the presence of players and the referee.

The first one is solved by applying the panorama framework two times, one for the LHS and one for the RHS. For code simplicity we compute the LHD by working with flipped frames. Then we reapply the stitching between the LHS and the RHS obtaining the panorama.
The second one is solved by applying a weighted average of warped frames into the previously computed panorama:

$$AvgP(i,j) = \begin{cases} P(i,j) & \text{R(i, j)} < 100 \\ \dfrac{w_1 P(i,j) + w_2 R(i,j)}{w_1 + w_2} & \text{otherwise} \end{cases}$$

where: $P(i,j)$ is the panorama matrix, $R(i,j)$ is the matrix of the new warped frame into the panorama, and w_1, w_2 are the weights (in our case we choose as values respectively 1 and 0.7). The improvement can be seen in Figure (1).

2.2 Binarization and Opening

Observing that in the resulting panorama with averages the court can be seen as the biggest "object" with an uniform and bright color we proceed by binarazing the panorama with Otsu's algorithm. Then, in order to improve our analysis on the court we apply erosion and dilation of the same structuring element on the binarized image as we can see in Figure 1. We apply the "Opening" approach (erosion followed by dilation) to delete impurities in the horizontal lines by using the following structuring element:

$$StructuringElement = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

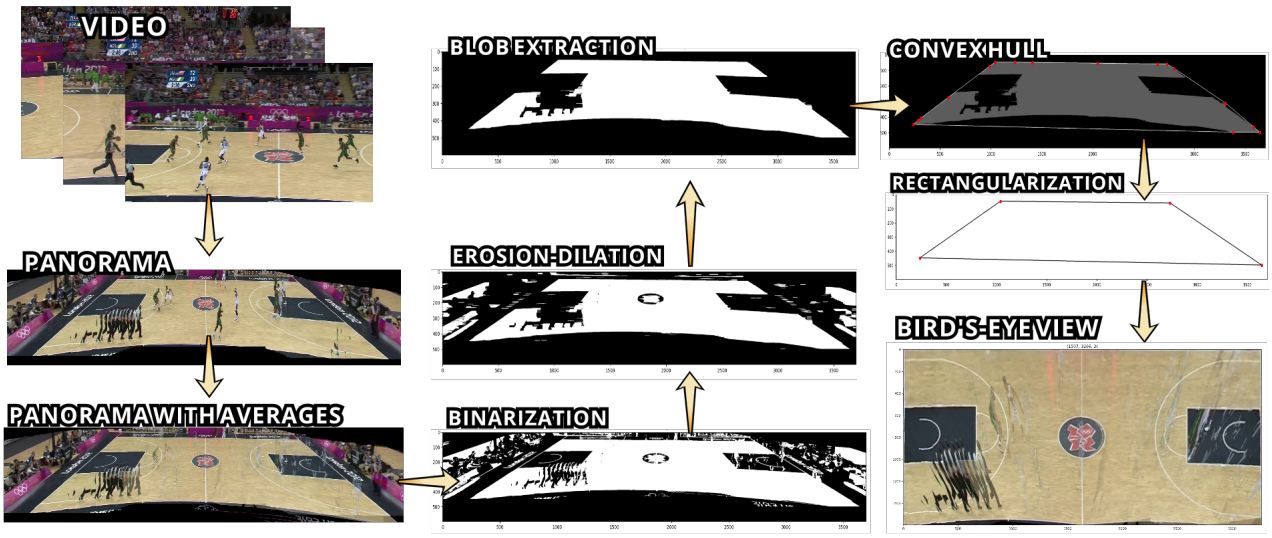

Figure 1: Overview

2.3 Blob Extraction and Rectangularization

The improved image now allows us to have the court as a blob separated from the others. To obtain it we find the biggest blob by thresholding on the area of all the blobs detected and we extract it. From this point on, in order to get the final homography we have to find the four corners. This is done by computing the convex hull on the blob, and subsequently the rectangular polygon fitting on the convex hull.

2.4 Bird's eye view and 2D map

At this point we define the corners of the bird's eye preserving the real proportions of a basketball court. Then we compute an homography between the corresponding panorama with averages corners found in the rectangularization and the bird's eye view ones. The bird's eye view is associated with a stylized 2D map having the same ratio between width and height (a correspondence between 2D map and bird's eye view coordinates is setted).
The scope of the 2D map (Figure 2) is to show the position of the players on the court. In order to do it is necessary to detect players (as shown in next sections) and to compute their position in the map. The position is obtained by multiplying the detected player coordinates by two homographies: the first one between frame and pano with averages, the second one between pano with averages and bird's eye view.

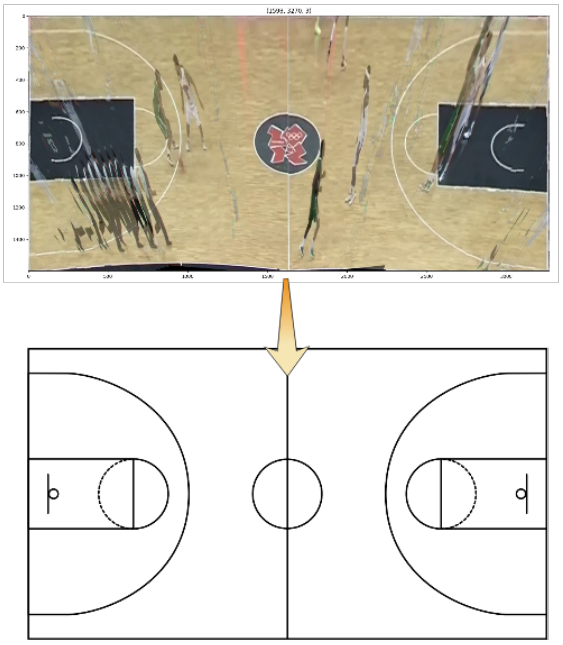

Figure 2: homography - 2D map correspondence

3 BALL DETECTION AND TRACKING

In order to better understand the flow of the gameplay and the action we decided to include include the detection and tracking of the ball in our system. This is showed in the final 2D map, plotting with a special marker (red contour) the ball owner.

3.1 Detection

As shown in [1] to detect the ball we start detecting circles in the picture, filtering them by radius to reduce the search space. In order to achieve it we use Hough Tranform. The resulting circles are used by the detection algorithm chosen, which in this case is "Template Matching" [4] with Normalized Cross Correlation similarity function.

$$NCC(i,j) = \frac{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} I(i+m,j+n)T(m,n)}{\sqrt{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} I(i+m,j+n)^2}\sqrt{\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} T(m,n)^2}}$$

In particular from the circles' centers and the radius we compute the bounding boxes. The previous are used to extract from the frame the cropped section that is used for template matching.
Template matching returns a list of values which are the similarity function between the cropped section and the slided template ball. Consequently we consider the cropped section as bounding box for the ball if the maximum value between the ones obtained with template matching is higher than a threshold.
More template balls have been used as a template to improve the result. Better results could be provided by working on a higher resolution video. Indeed sometimes the ball is mismatched with the head of the players.
Lastly the performances of the detection could be improved by using deep learning.

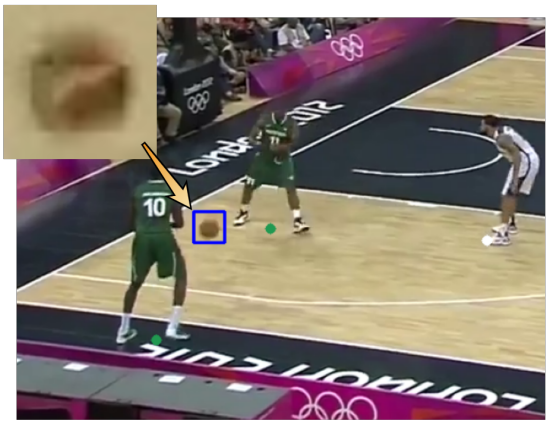

Figure 3: Ball Template and detection

3.2 Tracking

Rather than detecting the ball at each frame, we start tracking the ball after one detection. This makes the framework faster and allows to have a bounding box of the ball also in cases of occlusion. We use CSRT [5] to track the ball.
In order to improve the system we perform a detection every five frames. In particular we check if the ball detection computed on the current bounding box returns a score higher than a threshold. If so, we keep the same bounding box, otherwise we start detecting a new one.

3.3 Ball Assignment

The system has to be able to assign the ball to the one who owns it. This is crucial to understand the scheme of an action and analyze how the players on the court move with respect to the ball owner.
We also prefer to assign the ball instead of plotting the corresponding position of it at each frame, because usually the ball is not co-planar to the court.
In order to assign the ball to a player we use the intersection over union (IoU) between the bounding box of the ball and all the bounding boxes of the players (discussed in the next section).

$$IoU(A, B) = \frac{A \bigcap B}{A \bigcup B}$$

where A and B are respectively the areas of the two input pictures, or bounding boxes. After computing all the IoUs we assign the ball to the player who gave the maximum result in the computation.

4 PLAYER DETECTION

Our system requires a positioning of the players on the court. The usage of a standard computer vision approach is not sufficient due to several factors such as the different possible poses of a player. This is why we decided to use the pre-trained deep learning model Detectron2 [2] for person detection and image segmentation.

4.1 Detectron2

We use an instance segmentation model trained on the COCO dataset to detect the players and to compute the segmented part of the them. We only extract the classes referring to persons (class 0) and the masks corresponding to the segmented parts. The usage of the segmented part is crucial in detecting the teams (4.4).

Figure 4: Segmented Frame

4.2 Position Detection

We define the coordinates of the player position from the segmented part computed as in the pseudo code below:

```
input: PlayerPixels

output: pos_x, pos_y

begin

    top ← argmin_y(PlayerPixels)
    bottom ← argmax_y(PlayerPixels)
    pos_x ←PlayerPixels[top][1]
    pos_y ← PlayerPixels[bottom][0]
    return pos_x, pos_y

end
```

where $PlayerPixels$ is the bidimentional array of the pixels of the segmented player, $argmin_y$ and $argmax_y$

are two functions returning respectively the minimum and the maximum of the list along the y axis.
The minimum along the y axis represent the head of the player while the maximum if one of the foot. Consequently we use as x coordinate the x coordinate of the head and as y coordinate the y of the foot. The resulting coordinates (x,y) is the position of the player in the 2D map. This is an approximate result which might be not coplanar to the court but it gives good results.
A more precise approach would be computing the mean between the coordinates of the right and the left foot.

4.3 Tracking by Detection

To add the functionality of identifying each player over a period of time, we included a tracking procedure with the usage of the detected bounding boxes. To do so we used IoU as in (3.3) between the bounding boxes of all the players in consecutive frames. We associate a bounding box to an identified player if the IoU between his previous bounding box and one of the current ones is higher than a threshold.

4.4 Team Detection

The major difference among teams and referee is jersey color, one team is white, the other is green and the referee is mostly black. We want to determine the player's team based on the jersey color.
We follow the approach proposed by [6]. Firstly we identify the ranges of the three colors in HSV due to its superior performance compared to RGB color-space varying illumination levels. Then we create a mask for the three color and we use bitwise and to apply the masks to the bounding box of the player.
In this way we extract three colors for each player and we choose the one that has the biggest number of non black pixels.

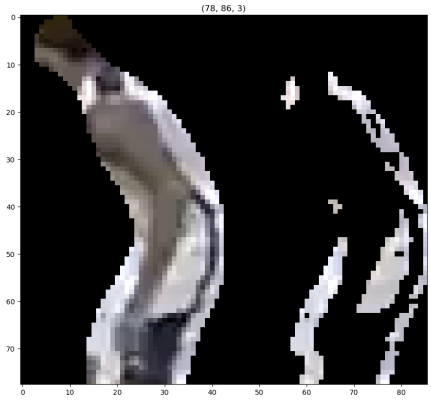

Figure 5: Example of white color detection

5 CONCLUSIONS

We have developed a system that shows in a clear plot the movement of the players allowing a better understanding of games scheme. The results obtained are good, especially considered that they are obtained with an arbitrary camera.

5.1 Results

The results are shown in Figure 6. In the upper image we can see the current frame with the detected players and their relative colors corresponding to the teams (color point near player's feet).
In the bottom one, we have the bird's eye representation with the players and the ball detected as seen in the previous sections.
Regarding the players:

- They are represented through circles of the same color of the team.
- Inside the circle we can see the number that represents the player's ID.
- The bigger circles are the current positions, while the smaller ones represent the history of the positions.
- The player who owns the ball is circled with red.
- Only the players inside the court are plotted in the bottom map.
- The referee is not drawn in the 2D map.

The system allows the access to past positions that an ID had.

5.2 Future Works

In order to improve our system, several approaches could be implemented alongside it.
We are listing some of them:

- In addition to the image segmentation model we could use another Detectron2 model which returns also the players' skeleton. This would allow to get better results on the planar position of the player by using the feet (the mean between left and right foot coordinates).
- As done for the players we could use deep learning also for ball detection (instead of template matching).
- In order to speed up the process we could run the code on GPU using CUDA and we could use SURF instead of SIFT (in court rectification).
- Using more than a camera would avoid problem of occlusion which happen mainly on the ball but sometimes also on the players.
- It would be a great improvement for the homographies between the real court and the 2D map to have a calibrated camera.

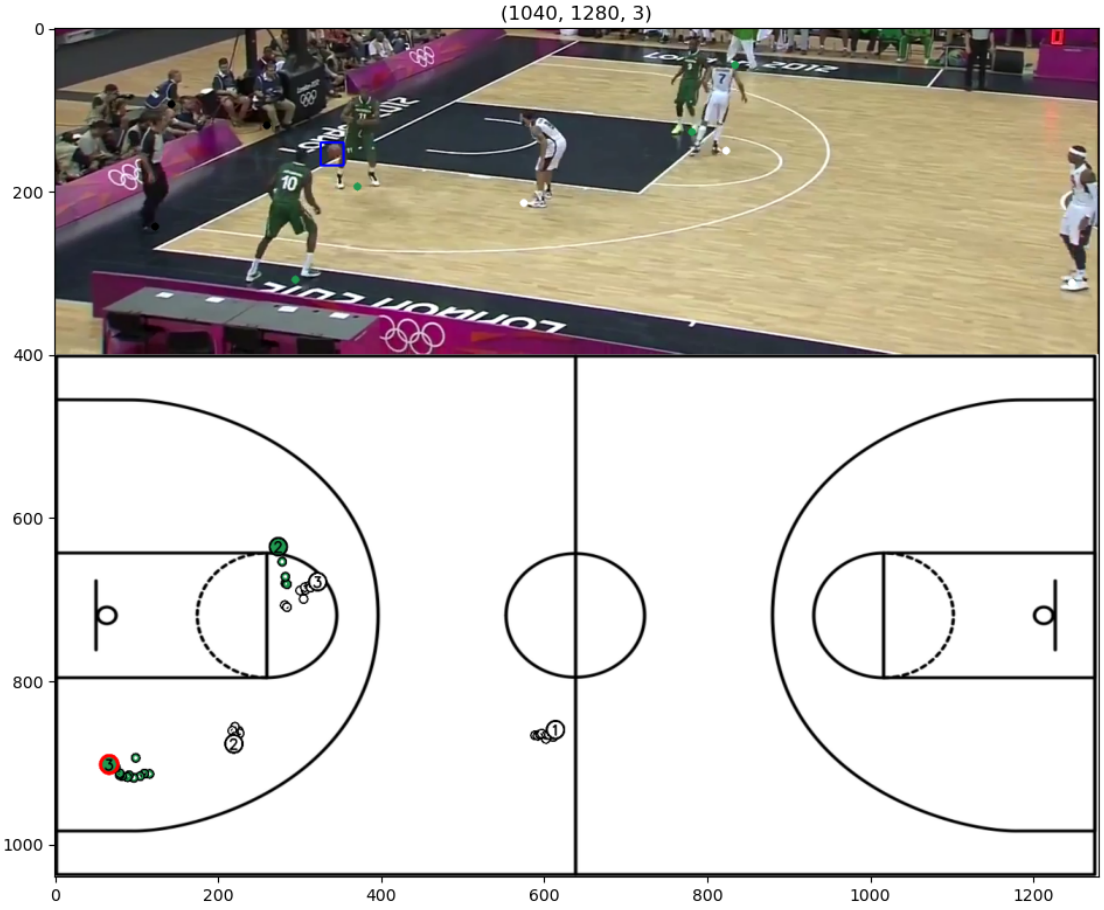

Figure 6: Demo Frame

References

[1] M. Leo, T. D'Orazio, N. Mosca, and A. Distante, "Sift approach for ball recognition in soccer images.," pp. 207–212, 01 2008.

[2] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2." `/https://github.com/facebookresearch/detectron2`, 2019.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[4] A. Hofhauser, C. Steger, and N. Navab, "Edge-based template matching and tracking for perspectively distorted planar objects," in *Advances in Visual Computing, 4th International Symposium, ISVC 2008, Las Vegas, NV, USA, December 1-3, 2008. Proceedings, Part I* (G. Bebis, R. D. Boyle, B. Parvin, D. Koracin, P. Remagnino, F. M. Porikli, J. Peters, J. T. Klosowski, L. L. Arns, Y. K. Chun, T. Rhyne, and L. Monroe, eds.), vol. 5358 of *Lecture Notes in Computer Science*, pp. 35–44, Springer, 2008.

[5] "CSRT Tracking, OpenCV." `/https://docs.opencv.org/master/d2/da2/classcv_1_1TrackerCSRT.html`.

[6] C. Kuan, "Football players tracking — identifying players' team based on their jersey colors using opencv." `/https://towardsdatascience.com/football-players-tracking-identifying-players-team-based-on-their-jersey-colors-using-opencv-7eed1b8a1095`.