Automatic Video Editing: Original Tracking Method Applied to Basketball Players in Video Sequences

Colin Le Nost[1], Florent Lefevre[2,3](✉), Vincent Bombardier[2], Patrick Charpentier[2], Nicolas Krommenacker[2], and Bertrand Petat[3]

[1] Ecole Nationale Supérieure des Mines de Nancy, Campus Artem - CS 14 234, 54042 Nancy, France
[2] Université de Lorraine, CNRS, CRAN, 54000 Nancy, France
florent.lefevre@univ-lorraine.fr
[3] CitizenCam, 132 rue André Bisiaux, 54320 Maxéville, France

Abstract. The main task here is to track several basketball players during a game and to be able to retrieve their whole trajectories at the end. The final application is to get some statistics about each players and to identify some special events like free throw or to determine when a counterattack is going to happen. The originality of the solution states in the way the tracking is performed: instead of studying the close environment of each player, all the players are detected on each frame then we are using specific informations like background, speed vector, color or distance between players to link player's positions and create the whole trajectories. We will compare our results with a benchmark of algorithms to see that our solution is quite efficient in term of tracking and speed.

Keywords: Automatic editing · Tracking · Sports analysis

1 Introduction

Automatic video editing allows small events to be available to a much larger audience. Indeed, many events cannot be broadcast because of the fixed cost of production (crew and equipment). By automatic video editing, i.e. automatic selection of the best viewing angle in a multi-camera system, the live video stream where the action takes place can be provided to the spectator. CitizenCam[1], a French company which offers multi-camera automatic recording solutions, wants to retransmit on the web every type of event to the greatest number of people. To achieve this goal, CitizenCam choose to reduce costs by automating recording and broadcasting while using IP cameras. The gathering of statistical knowledge on the scene is required to understand the action and perform camera selection. The specific context of this study is the case of indoor sport broadcast, especially

[1] This work results from a collaboration between CitizenCam and CRAN.

A. Mansouri et al. (Eds.): ICISP 2018, LNCS 10884, pp. 117–126, 2018.
https://doi.org/10.1007/978-3-319-94211-7_14

Basketball games. For this article we are interested in tracking players in order to determine key events such as free throws or counter-attacks, and also to obtain statistics on each player. The dataset is available in [1] and includes different views of a game of basketball; from the side and above. While watching this footage, we can detect two types of challenges which influence the precision of the tracking.

First, some of them are due to the nature of basketball:

- *Occlusion:* During tracking, players can be hidden during a certain time and it can be hard to recover from it. Two sub-cases can be identified: first, if two players are crossing each other. Second, when a player is hiding another during a static phase.
- *Rotation:* It implies that appearance models are complex to use because the looks of the players are changing function of how they rotate and where they are located; seing a player from a side is not the same than above.
- *Acceleration:* Some tracking algorithms use difference between two frames to determine the next position, but if there is a brutal and unexpected change of direction, it can be difficult to perform a good tracking.
- *Groups:* Based on the structure of basketball, most of uncertain situations imply just two players, excepting some categories: beginning, injury, celebration of a goal and ending. Then a lot a similar players stand next to each other and it is hard to follow them.

Second, some issues are directly related to the video caption. The footage is actually recorded with different fish-eye cameras (wide-angle). If we focus on the view from above that we are using, which is recorded with a 180° security camera, we can observe two issues in our analysis.

- *Cropped Image:* Some correction has ever been applied to make the video watchable but it cropped the image, so we need to determine when a player is going out of the window and when he is back.
- *Distortion:* The distortion is not completely corrected so it implies that the size of a player is changing function of his position. For instance a player in the center is way bigger than in a corner, so we need to correct this.

After having exposed this different challenges, it appears clear that a lot of information is contained into the nature of the game. Because of this major constraints, it makes sense to develop a specific solution instead of using generic algorithms in order to make the tracking smarter, i.e. better and faster.

2 Available Techniques

In order to evaluate the results of our solution, it is necessary to compare it to different algorithms. Because we are working with OpenCV, we can observe that some tracking algorithms are ever implemented. The algorithms available are Boosting, KCF, MedianFlow, Multiple Instance Learning and Tracking-learning-detection trackers. Since MedianFlow tracker [5] and MIL tracker [6] are not adapted to our application (random displacement, quick rotation), we will focus on the other trackers. For a more exhaustive comparison, please see the work of Janku et al. [8].

- *Boosting Tracker:* based on the AdaBoost algorithm, which uses the surrounding background as negative examples to find the most discriminative features of the tracked object. Because it is based on the appearance, changes of the player like rotations or light changes are normally well handled [2].
- *Kernelized Correlation Filter (KCF) Tracker:* The main goal of a tracker is to distinguish the target from the environment. This algorithm translates and scales different patches in order to find the best one. To improve computation power, some improvements have been done by seeing that the studying matrix is circulant [3] and that a correlation filter can be applied [4].
- *Tracking-learning-detection (TLD) Tracker:* The main approach here is to detect an object at one frame, then detect if the object is there in the following frames. Function of that, the tracker is updated differently [7]. This algorithm is supposed to be able to handle rapid motions and partial occlusions.

After this review, we can say that KCF, Boosting and TLD are suitable for our study. We will compare the results of our solution to these algorithms.

3 Implementation

Our solution is implemented in Python and includes few steps of processing.

On each frame, we are performing several actions:

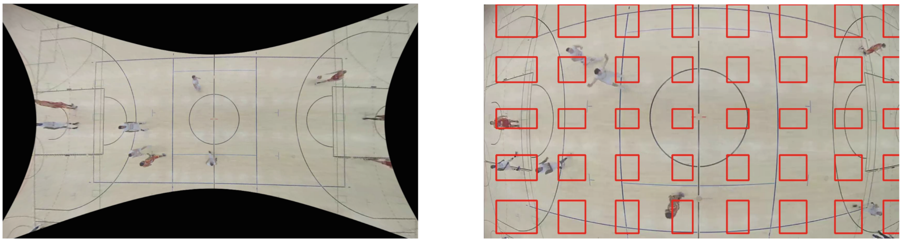
1. A background subtraction model is used to remove the background. Because the camera is fixed, it is pretty efficient.
2. The subtraction is cleaned with closing and opening transformations in order to remove blur and to retrieve clean players.
3. The Suzuki algorithm [9] is used to find the different players.

Then, the different objects are linked on each frame based on criteria like surface, center distance, speed vector direction and main color. But in order to get good results, the distortion should be first corrected (at least limited), otherwise the algorithm would need to link two players way to different between the center and a corner of the image.

3.1 Distortion

The proprietary software associated with the camera has ever been used to retrieve the actual view. Using the raw footage is not an option because the camera uses a panamorphic lens; the distortion is not radial and there is no easy way to correct it without having access to the main characteristics of the camera. Then we need to work on the corrected image. The first try was to estimate the distortion model while asserting this one was radial: after solving the equations, we can get a good result at the center of the picture, but the borders areas were completely deformed (Fig. 1a) and unusable.

Because of the bad results of the previous approach and because undistorting the video was slowing too much the algorithm, we decided to use a pixel/mm ratio created manually which is changing across the frame function of the position. The main advantage to have this correction is that it becomes possible to set parameters in millimeters and no more in pixels. The projection of the function we found can be seen in the Fig. 1b.


(a) Radial Correction

(b) Estimation of the distortion

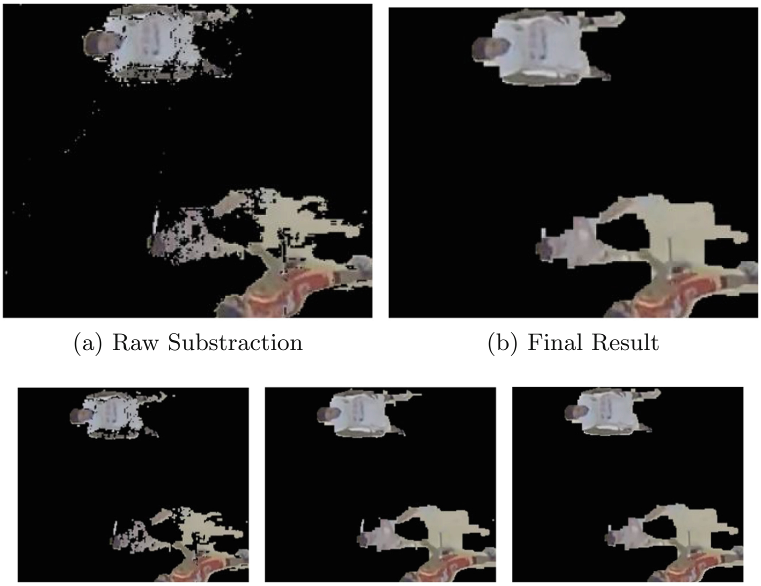
Fig. 1. Distortion

3.2 Background Subtraction and Contours Extraction

To be able to detect players, a background subtraction [14] is used. Because the camera is fixed, the history used to calculate this subtraction needs to be the longest. When applying it, some blur can still been seen (see Fig. 2a), particularly the lines of the field. Moreover, some players are divided in chunks, so some cleaning is necessary. Three morphological transformations are applied to the image: first, a small opening (2 × 2 pixels, see Fig. 2c) to remove the blur, then a closing to unify the chunks (9 × 9 pixels, see Fig. 2d) then a final opening to clean the contours of the objects (3 × 3, see Fig. 2e). The final result can be seen in Fig. 2b. The last step of processing is detecting players. Based on the subtraction, a contour detection algorithm is used. We took the Suzuki Algorithm [9], which follows the contours in order to determine the whole shape of the tracked object. Then the bounding boxes of the contours are filtered to avoid the nested one. Finally the too small players (based on a surface parameter) are removed.

3.3 Score Estimation

To sum up the previous steps, now are available on each frame different boxes which are normally players. The main challenge here is to link the different boxes across the frames in order to retrieve the whole trajectories. *Terminology:* the *target* names the player tracked at the previous frame, the *players* are all the detected boxes on the current frame. On each frame, we are calculating an association score between the target and the players. Under certain conditions, the player with the best score is associated with the target. The whole decision tree can be found in Fig. 3.



(a) Raw Substraction

(b) Final Result

(c) First Opening

(d) Closing

(e) Last Opening

Fig. 2. Succeeding steps of the subtraction

Distance Evaluation. The distance is here critical because there is no way that the target can move faster than a certain limit. Based on the information available, we can create for each player a distance score to the target defined in formula 1, where *distance* is the euclidean distance between the player and the target, and max is the maximum value to link a player with a target. This value has been set at 3 meters for experimentations and we keep only the players 1.2 meters ($score > 0.6$) apart for the association of the players to the targets

$$score = \begin{cases} 1 & if\ distance > min, \\ 0 & if\ distance <= max, \\ \frac{max-distance}{max-min} & else \end{cases} \tag{1}$$

Speed Vector Projection. Based on the previous data, occlusion can be solved under some conditions. For instance, the speed vector can help to determine when two players are crossing each other, but this information is relevant only if the two vectors are not co-linear and if at least one of the modules of the speed vectors is not too low. Currently the score is calculated function of the angle between the two vectors (see Fig. 4).

Color Comparison. When two players are too close, it is not possible to determine exactly where they stand, but determining their positions when they are separating from each other is possible. To increase our accuracy, and because

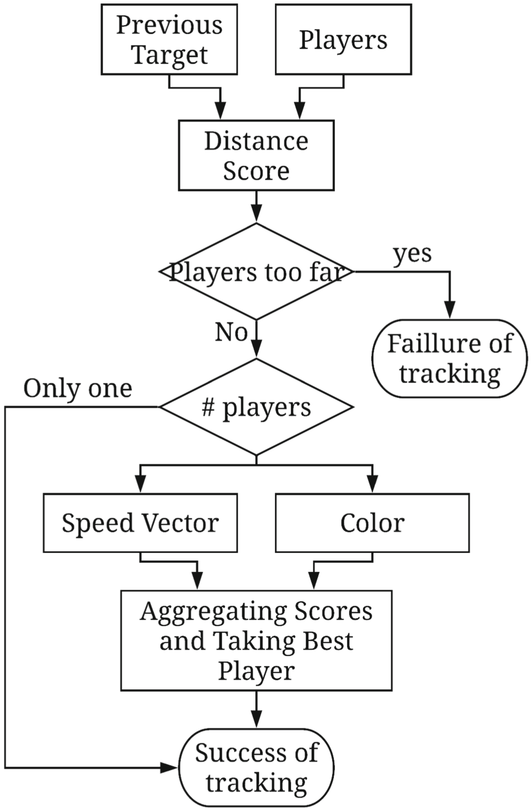
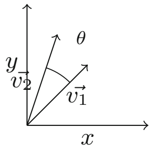


Fig. 3. Structure of the score



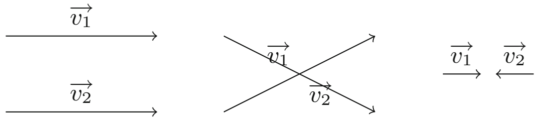
$$\theta_{degrees} = \left| \arctan \frac{\vec{v_1} \cdot \vec{x}}{\vec{v_1} \cdot \vec{y}} - \arctan \frac{\vec{v_2} \cdot \vec{x}}{\vec{v_2} \cdot \vec{y}} \right| * \frac{180}{\pi}$$

Fig. 4. Projection of the speed vectors

there is two teams with different wearings, the color gives us precious information. The first guess was to use conditional statements on the RGB (Red Green Blue) color space, but the main issue here is that this space is not linear and pretty dependent of the luminary exposition. That is why we change the color space. The LAB color space seems pretty efficient: L states for lightness, a for green-red scale and b for blue-yellow [10]. Because one team is wearing red, it is fine to calculate a ratio between red pixels (=a higher than a certain limit which depends of the LAB implementation) and the total amount of pixels. To determine who is who, we are comparing the ratios before losing the players with the ratios after the separation. One issue with this implementation is that it is asking a lot of resources to compute the color information; indeed the moment we will need this information is unknown. In order to make the program faster, calculating it only during few frames is enough.

4 Results

For all the considered algorithms, if the tracking fails, there is no recovery system. Because all of them fail at some point, it was not efficient to use a metric to show how efficient they were. It makes more sense to determine if they are able to solve different issues as exposed in the first part of this article. We manually annotated the position of each player in different footages corresponding to specific situations (between 5 and 10 extracts for each case, see Fig. 5). Thanks to the ground truth, we calculated an accuracy score [15] based on the amount of situation solved. The results can be found in the Table 1. The processing time per image (P. T. I.) expresses the average time (in milliseconds) needed to detect and track a player in an image.



(a) Parallelism (b) Crossing (c) Static

Fig. 5. Problematic situations

To sum up the results, we can see that two algorithms are well performing in our study: the KCF tracker and proposed method. Both of them are able to follow most of the time the players, excepting in the corners where the distortion is too important. Concerning the algorithms that are not performing well, bad results of Boosting can be explained by seeing that the tracker is often blocked on lines; because this descriptive feature is not moving too much, it is possible to suppose that the algorithm get fooled. Concerning TLD, the main issue we observe is that the player is sometimes lost during few frames and the focus is placed on another player randomly chosen in the image, but it often goes back to the tracked player after few frames. Smoothing the trajectory to remove this jumps could improve the tracking. Anyway, these two algorithms are way slower than KCF.

The results begins to be interesting when we are speaking about scalability. The main issue with the KCF tracker is that we need to set a tracking object for each player we want to follow, so it slows down the speed of the process. On the contrary, most of the processing time of our solution is taken by the background subtraction. It means that tracking new players just asks new resources to link boxes, which is fast. So our solution is more suitable for tracking several players. Finally we can observe the result window of the algorithm in Fig. 6.

Table 1. Accuracy of the different algorithms

Situation	Criteria	KCF	Boosting	TLD*	Proposed method
Unique player	Regular tracking	++	++	−	++
	Acceleration	++	+	+	++
	Low speed	++	++	+	++
Different players	Static	++	−	−	++
	Crossing	++	+	+	++
	Parallel	+	−	− −	+
Same team (red)	Static	−	− −	−	−
	Crossing	+	−	−	++
	Parallel	+	− −	− −	−
Same team (white)	Static	−	− −	+	− −
	Crossing	+	−	−	++
	Parallel	−	− −	−	−
Processing time per image (PTI)		0.017	0.049	0.157	0.012

$++ : acc \geq 75\%, + : acc \geq 50, - : acc \leq 50\%, -- : acc \leq 25\%$

*Some errors can cause the algorithm to jump on other players during few frames

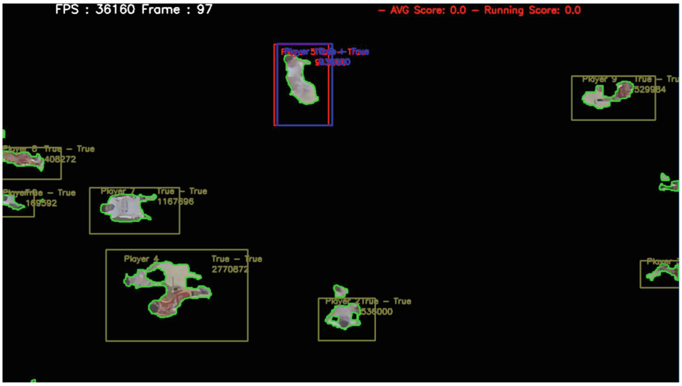
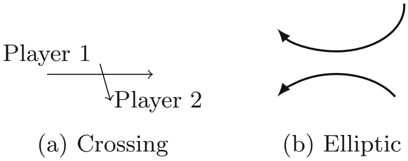


Fig. 6. Detection process

5 Conclusion and Perspectives

During this study we expose challenges caused by automatic video editing applied to basketball. We see that most of this issues fool the generic algorithms but they can be solved using specific informations like color, positions or speed of the players. Implementing a dedicated solution makes the algorithm way better in term of accuracy but mainly in term of speed. If a long time of calibration of the algorithm is needed, we can explain it by different factors: quality of the video, same color of the wearing and the background or diffraction. This promising results let us expect that improving the model while implementing the multi-camera system or modifying the subtraction step will makes the solution still better. With the final trajectories of all the players, it should be easier to determine specific actions like counterattacks and free throws to finally perform the automatic video edition.

Even if we see promising results, the implemented algorithm is still facing some issues, like the other algorithms. Most of them are related to the similarity between the background color and the outfit of one team. For instance, if two players of the same color are crossing each other at low speed, the player can be lost. To solve this problems, we thought to few improvement that can be done to improve the accuracy.



(a) Crossing (b) Elliptic

Fig. 7. General issues

First the tracking is one-target only. It means that it is not possible to use informations of previous tracking. For instance, we can see in Fig. 7a the trajectories of 2 players. If we try to follow the player 2, and that the color can't help us, the algorithm will face issues to find the relevant player when they will cross each other because $|\overrightarrow{v_2}|$ is too low to give information (the player can go backward). But if at the same time, you are tracking the player 1, because $|\overrightarrow{v_1}|$ is significant, you will be able to him. Combining these informations with a system of rules could help to solve this issue. Moreover a criteria on the distance can lower the multi-tracking task complexity.

The algorithm can be fooled if the players are in the configuration of Fig. 7b. Speed is big enough to consider the angle of speed vector and the tracking will probably fail. One answer to that could be to improve the background subtraction step as shown in [11].

As seen in the dataset, more videos than just the above view are available. A multi-camera model as implemented in [12] or in [13] could help to fix most of the issues listed above.

References

1. Games recorded in 2016 from BCMess, a Luxembourg Basketball club. https://www.citizencam.tv/v/ywWDBMZXHg
2. Grabner, H., Grabner, M., Bischof, H.: Real-time tracking via on-line boosting. In: BMVC, vol. 6 (2006)
3. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: Exploiting the circulant structure of tracking-by-detection with kernels. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012. LNCS, vol. 7575, pp. 702–715. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33765-9_50

4. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. TPAMI **37**(3), 583–596 (2015)
5. Kalal, Z., Mikolajczyk, K., Matas, J.: Forward-backward error: automatic detection of tracking failures. In: 2010 20th International Conference on Pattern Recognition (ICPR), pp. 2756–2759. IEEE (2010)
6. Babenko, B., Yang, M.-H., Belongie, S.: Visual tracking with online multiple instance learning. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009, pp. 983–990 (2009)
7. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning-detection. IEEE Pattern Anal. Mach. Intell. **34**(7), 1409–1422 (2012)
8. Janku, P., Koplik, K., Dulik, T., Szabo, I.: Comparison of tracking algorithms implemented in OpenCV. In: MATEC Web of Conferences, vol. 76, p. 04031 (2016)
9. Suzuki, S., Abe, K.: Topological structural analysis of digitized binary images by border following. Comput. Vis. Graph. Image Process. **30**(1), 32–46 (1985)
10. ISO 11664–4: 1976 L* A* B* Colour Space. Joint ISO/CIE Standard, ISO. ISO 11664–4 (2008)
11. Zeng, Z., Jia, J., Yu, D., Chen, Y., Zhu, Z.: Pixel modeling using histograms based on fuzzy partitions for dynamic background subtraction. IEEE Trans. Fuzzy Syst. **25**, 584–593 (2017)
12. Hayet, J.B., Mathes, T., Czyz, J., Piater, J., Verly, J., Macq, B.: A modular multi-camera framework for team sports tracking. In: IEEE Conference on Advanced Video and Signal Based Surveillance (2005)
13. Du, W., Hayet, J.B., Piater, J., Verly, J.: Collaborative multi-camera tracking of athletes in team sports. In: Workshop on Computer Vision Based Analysis in Sport, Environments, pp. 2–13 (2006)
14. KaewTraKulPong, P., Bowden, R.: An improved adaptive background mixture model for real-time tracking with shadow detection. In: Remagnino, P., Jones, G.A., Paragios, N., Regazzoni, C.S. (eds.) Video-Based Surveillance Systems, pp. 135–144. Springer, Boston (2002). https://doi.org/10.1007/978-1-4615-0913-4_11
15. Davis, J., Goadrich, M.: The relationship between precision-recall and ROC curves. In: 23rd International Conference on Machine Learning, vol. 6, pp. 233–240 (2006)