Accurate 3D Locating and Tracking of Basketball Players from Multiple Videos

Anonymous Author(s)

ABSTRACT

With the development of pedestrian detection technologies, existing methods cannot simultaneously satisfy high-quality detection and fast calculation for practical applications, especially for accurate 3D locating and tracking of basketball players which needs effectively occlusions handling. We propose an algorithm which can robustly and automatically locate and track basketball players from multi-view videos. After extracting the foregrounds from the videos, the voxels in the basketball court space are reprojected back to the foreground images. Occupied voxels are accumulated and smoothed based on *integral space* for acceleration. Two Gaussian Mixture Models including Grouping Gaussian Mixture Model(GGMM) and Locating Gaussian Mixture Model(LGMM) are designed for continuous locating and grouping of individuals, and a simple blob detector is auxiliary to handle out-of-bound players. Our algorithm is insensitive to occlusions, shadows, lights and computation errors, which can be validated by our experiments.

CCS CONCEPTS

• **Computing methodologies** → *Image representations*;

KEYWORDS

tracking, integral space, Gaussian Mixture Model, grouping

ACM Reference format:
Anonymous Author(s). 2019. Accurate 3D Locating and Tracking of Basketball Players from Multiple Videos. In *Proceedings of ACM Conference, Tokyo, Japan, December 2018 (SA'18 Technical Briefs),* 4 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The 3D tracking of basketball players is complex because it is impacted by universal occlusions among players, the changing of lights, and the shadows. The tracking algorithm should be insensitive to noises and calculation errors. We propose a fully-automatical robust 3D locating, tracking, and grouping algorithm for basketball players from multi-view videos captured from low-cost and not high-speed video cameras. After building the video capturing environment and calibrating the cameras to obtain the intrinsic and extrinsic parameters, we first divide the basketball court space into fixed-size voxels as the size of the basketball court is invariant. Inspired by the idea of [3], we compute the occupancy value of each voxel and carve out unoccupied voxels. In order to efficiently dealing with the occlusions, we introduce *integral space* that is the 3D extension of integral images storing the sum of occupied values for smoothing acceleration to compute the visibility of the voxel on each camera for grouping. Two Gaussian Mixture Models(GMM) are designed for grouping and locating: Grouping Gaussian Mixture Model(GGMM) used for distinguishing the teams and Locating Gaussian Mixture Model(LGMM) used for distinguishing locations of players. As the enter or the exit of players will change the player number on the court, a simple blob detector is used to assist LGMM to determine the locations. As the binary foreground extraction of multi-view videos are independent respectively, CPU and GPU parallel calculating are adopted for the acceleration. One frame tracking result from a camera is shown in Figure 1. Experiments demonstrate that our system can robustly and automatically track the 3D locations and group the team category for basketball players, with practical commercialized prospects.

The contributions of the paper are:(i)**High Accuracy**. The 3D position of the player is obtained accurately based on an occupied voxels reprojection strategy from multi-view videos, which can effectively handle the occlusions of each other, shadows changing, and out-of-bound moving. (ii)**Full Automation**. The locating, grouping, out-of-bounds detecting for players are implemented based on our customized unsupervised tracking algorithm, which can achieve automatic locating and tracking. (iii)**Low Latency**. We use CPU and GPU parallel processing for foreground extraction, and apply *integral space* for smoothing to achieve a fast-tracking process (64% time length of the videos).

2 ALGORITHM

We use *n*(in our practice, n is 4) cameras to obtain *n* synchronized multi-view input videos, with intrinsic and extrinsic parameters of cameras respectively by calibration, in order to construct the mapping between the 3D basketball court space and 2D frames of videos. We use a chessboard for the camera parameters recovering. As the camera stay still during the whole game, the intrinsic and extrinsic parameters of cameras are constants for the videos. Other input data include the player numbers *p* and the court size, which makes our algorithm suitable for nonstandard basketball courts and games. Then, we deal with the frames from multi-view videos at the same moment in sequence as shown in the right of Figure 1.

Binary Foreground Extraction. For each frame at the same moment, the binary foreground needs to be extracted in order to identify players. Here we use Gaussian Mixture-based background/foreground segmentation algorithm to finish the task. The extracted binary foreground should be accurate for our tracking, and learning speed is set to 0.001. Our tracking algorithm is insensitive to the small amount errors and noises existed in the extracted foreground, as the affection of these errors and noises can be cut down in the following process of our algorithm. Besides, as the

SA'18 Technical Briefs, Tokyo, Japan

DOI: 10.1145/nnnnnnn.nnnnnnn

Figure 1: Tracking result and our processing for synchronized frames. Left: Our tracking result is marked on a frame from one of the four videos. The numbers upon rectangles indicate the identifiers of players. Two colors of rectangles represent two teams. The right-top figure shows locations of players, in which black dots and white dots represent different teams. The right-bottom shows the extracted foreground with color for tracking. Right: Our processing for synchronized frames.

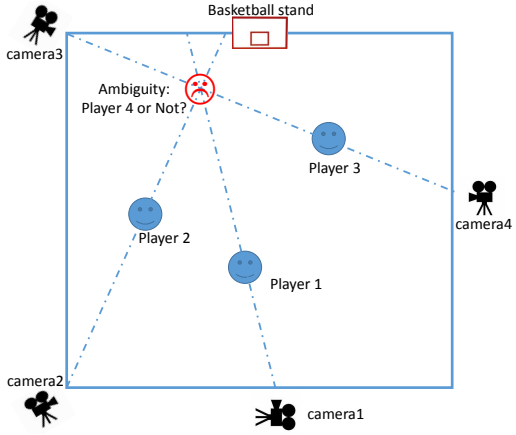

Figure 2: Improper configuration of cameras will lead to ambiguity in tracking.

extraction is independent of each video, parallel computation can be used for acceleration.

Occupancy values calculation. In order to track the 3D locations of players, we should determine where the players occupy in the basketball court space. The basic idea is to differential the court space and judge the occupancy state in each voxel, and then integrate occupancy values of neighborhood voxels for overall estimation. Possegger et al. [3] introduced the concept of occupancy volume based on the local mass densities of a coarse 3D reconstruction of the objects' visual hull. In contrast, we don't construct the visual hull of the player, but use the discrete space voxels for the computation of the occupancy value. This is natural, as the basketball court size is fixed.

The basketball field is divided into a set of cube voxels with the number of $n_x * n_y * n_z$, with the size is set to $10cm * 10cm * 10cm$ which can satisfy our needs. Here we set the height of the space to $2m$, which is enough for common people. For professional basketball players, the height can be set to $2.5m$ or even $3m$ because they are taller and jump taller than common people.

Each voxel is associated with some properties, including the 3D location of the voxel center, the distances to each camera optical center, the occupancy value by players, and the scores that belong to both teams with initial value 0. Both the 3D location of the voxel center and the distances from the voxel to each camera optical center are fixed when the capture environment is set up. Besides, as a player definitely occupy several voxels, we use a neighborhood cuboid of the voxel to simulate the occupancy space of the player. Then we can define the rectangles that the cuboid reproject back to the image by choosing the maximum and minimum x and y from the 8 reprojected vertices as the bounding box of the player in the frame of each camera. The reprojection process is based on the intrinsic and extrinsic parameters of each camera, and the computing can be done in advance as well as the voxelization.

For the occupancy values calculation, we firstly initial the occupancy value of each voxel by 1. Then we traverse the voxels one by one, reproject the center of the voxel back to the binary foreground of each camera and judge whether it falls in the background. If it is, the occupancy value of the voxel will be set to 0, meaning that the voxel is unoccupied and is carved out.

We use three strategies to conquer the adverse effects of occlusions, the errors or noises in the extracted binary foreground. First, the cameras are set to unable to see each other for removing ghost players that may be generated such as the Player 4 in Figure 2. Second, we discard the same number foregrounds from each camera in the beginning in order to obtain stable foregrounds with fewer noises for occupied value computation. Third, we smooth the occupied value of the voxel based on its neighborhood cuboid. The occupied value smooth equation of the voxel $S(x,y,z) = \sum(G(i,j,k))/(2*r_x+1)*(2*r_y+1)*(2*r_z+1)$ where $x-r_x <= i < x+r_x, y-r_y <= j < y+r_y$, and $z-r_z <= k < z+r_z$. The neighborhood cuboid size is $(2*r_x+1)*(2*r_y+1)*(2*r_z+1)$, making the voxel in the center.

As the smoothing process is time consuming with $O(n_x * n_y * n_z*(2*r_x+1)*(2*r_y+1)*(2*r_z+1))$ time complexity, we propose *integral space* which can be seen as the 3D extension of integral images to reduce the time complexity for $O(n_x*n_y*n_z)$. Specifically, Let I denotes the *integral space*, G denotes the initial voxelization

Figure 3: Position verification of our Algorithm. The blue rectangle has the same aspect ratio with the half basketball court, and the two circles in the rectangle denote the player's position.

space, and $G(x, y, z)$ denotes the occupancy value of the voxel on the location of (x, y, z), then $I(x, y, z) = sum(G(i, j, k))$ where $0 <= i <= x, 0 <= j <= y, 0 <= k <= z$. During the concrete calculation, $I(x, y, z) = G(x, y, z) + I(x-1, y, z) + I(x, y-1, z) + I(x, y, z-1) - I(x-1, y-1, z) - I(x-1, y, z-1) - I(x, y-1, z-1) + I(x-1, y-1, z-1)$, and the boundary voxels should be treated specially, i.e, if $x < 0$ or $y < 0$ or $z < 0$, the value of $I(x, y, z)$ will be zero. Now the smoothing process can be done by $S(x, y, z) = (I(x + r_x, y + r_y, z + r_z) - I(x - r_x - 1, y + r_y, z + r_z) - I(x + r_x, y - r_y - 1, z + r_z) - I(x + r_x, y + r_y, z - r_z - 1) + I(x - r_x - 1, y - r_y - 1, z + r_z) + I(x + r_x, y - r_y - 1, z - r_z - 1) + I(x - r_x - 1, y + r_y, z - r_z - 1) - I(x - r_x - 1, y - r_y - 1, z - r_z - 1))/(2 * r_x + 1) * (2 * r_y + 1) * (2 * r_z + 1)$, where $x - r_x <= i < x + r_x$, $y - r_y <= j < y + r_y$, and $z - r_z <= k < z + r_z$. For the boundary voxels, the denominator $(2 * r_x + 1) * (2 * r_y + 1) * (2 * r_z + 1)$ will be replaced by the corresponding cuboid size. Then the occupancy values for occupied voxels are more stable and no more than 1. In practice, the size of neighborhood cuboid is set as $5 * 5 * 7$ voxels.

Besides, we record the shortest distance from the foreground part to the optical centre of each camera by scanning all the occupied voxels, and compute the difference between the distance from the voxel center to the optical center and the shortest distance from the pixel of the rectangles of the neighborhood cuboid in the binary foreground to the optical centre. If the difference is larger than a user-specified threshold (such as $0.6m$), then the voxel will be set invisible for this camera, as this indicates that the voxel is occluded by other players, and this will be used for grouping.

For locating players, the coordinates of (x, y) are more important than z because players mainly move on the ground plane but not in the upright direction. So we investigate each z direction column voxels and make a voxel set M by the coordinates of voxels with the maximum occupancy value in that z direction, indicating that these voxels are most likely occupied by players. Based on M, we should distinguish each player by location, and group these players into two teams. We construct Grouping Gaussian Mixture Model (GGMM) and Locating Gaussian Mixture Model (LGMM) simultaneously to finish this task.

Grouping Gaussian Mixture Model Construction.

We choose a subset T from the voxels in M with those heights are taller than $60cm$ in the first frames of each video to construct GGMM, as the jersey color is only effective information for grouping, and the occupied voxels lower than this height are mainly legs with skin but no jersey. We extract all the foreground colors of the pixels in the image corresponding the visible voxels in T for the camera, and use the k-mean algorithm to divide them into two categories A and B, in order to initial two GGMMs respectively, with each GGMM has $p/2$ players classes. For the next frame, the occupied voxels will have scores for A and B based on the two GGMMs. Those invisible or lower than $60cm$ occupied voxels will have zero scores for both teams.

Locating Gaussian Mixture Model Construction. We choose a subset P from the voxels in M with the heights are taller than $20cm$ to construct LGMM, in order to remove the shadows interference in the extracted binary foreground. All the voxel coordinates (x, y) in P are clustered into p classes by k-means clustering to initiate the LGMM, where p is the player number. During the judicial process, we accumulate the team scores A and B respectively for the same class, and the higher score will determine the team belonging to that class. The p center locations calculated by LGMM are the final locations of the players. If the distance from the voxel to its class location is longer than 8 meters, the voxel's (x, y) coordinates will be added as a source of LGMM construction.

Continuous Tracking. In order to keep the consistency of tracking, we should discover players that go out of court instantly. We choose $60cm$ near the boundary lines as the boundary region for special consideration, and when the occupied voxels belong to this region, the coordinate (x, y) will be recorded. We judge the classification number in the current frames, and if there is no voxel in some classification, we will check the location of the class. If it isn't in the boundary region, this means the player may keep still or move very slowly, making him not be captured in the extracted binary foreground image. In this case, we just keep his location and set the occupied voxels with (x, y) as 1. At the same time, we count this failed times, and if 5 continuous frames happened, we will use the simple blob detector to detect again for acquiring the location of each class. If the location of the class is in the boundary region meaning a player maybe get out of the court, we use the simple blob detector to detect those coordinates recorded in the last frame, and the detected coordinates are used to update the class locations that have no voxels.

3 EXPERIMENTS

Our experiments are implemented on a PC with an Intel i7-4790 3.6GHz CPU, 16GB memory and a Nvidia GeForce GTX 970 display card. We use four DS-2CD4026FWD Hikvision cameras to capture the half basketball court videos with $1920 * 1080$ resolutions and 25 frames per second. For 1000 frames with 8 players of 4 synchronized multi-view videos, i.e., 40 seconds length for each video, the computation time is 102.831 seconds, which is shorter than the total 160 seconds(64%)), thanks to our integral space and the parallel acceleration of CPU and GPU.

In order to validate our tracking, we put range poles along the penalty line and record the videos with players walking along it as shown in Figure 3. The position distance error is less than $10cm$.

We also project the location back to the videos with two colors of rectangles to represent different teams, such as shown in Figure 1 and the accompanying video. When a player gets out of the boundary, the last tracked position will be preserved until he comes back to the court. From the video, we can see that the grouping and the location tracking are quite good, and out-of-bound players can also be smoothly transited during the tracking process. The shadows of the players or the slight moving of the people cannot cut down the tracking result, although they will be captured as the foregrounds.

We also have tested our algorithm on the APIDIS dataset (http://www.apidis.org/dataset/) as shown in Figure 4. All the videos we download from the internet are downsampling, making the calibration errors increases. Under that condition, we still can run out the location tracking results using our algorithm, based on only 4 cameras (1,2,4 and 7) of APIDIS for the half court testing. In this test, our algorithm only tries to divide the player into 2 teams, while the referees are also on the court in the game, making our grouping method inapplicable.

For comparison, the KSP [1] and POM method [2] cannot effectively to deal with the jumping case which often exists in the basketball court, while [3] need to construct the visual hull to obtain occupancy volume. All these methods cannot deal with the grouping problem. In contrast, our algorithm is more suitable for basketball players tracking. On the other hand, the ground truth of the 3D location for 3D players are difficult to obtain, so we can't compute the 3D position tracking accuracy directly. However, as shown in our accompanying video, as long as the 3D locations of players are obtained, the players can be tracked accurately and stable in the video.

Figure 4: The result that using our algorithm to track the data from APIDIS and the referees are also tracked correctly.

4 CONCLUSION

By carving out the unoccupied voxels in the basketball court space and employing our integral space for the occupancy value smoothing, the group and location of each player can be determined based on our Grouping Gaussian Mixture Model (GGMM) and Locating Gaussian Mixture Model(LGMM). The out-of-bound players are handled by a simple blob detector, and our algorithm can achieve automatical and continuous tracking for basketball players in the game. CPU and GPU parallel accelerations are utilized, leading to high-speed computation. Compared to state-of-the-art methods, our algorithm is more robust and practical for basketball players tracking.

REFERENCES

[1] J. Berclaz, F. Fleuret, E. Turetken, and P. Fua. 2011. Multiple Object Tracking Using K-Shortest Paths Optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 9 (Sept 2011), 1806–1819.

[2] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. 2008. Multicamera People Tracking with a Probabilistic Occupancy Map. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30, 2 (Feb 2008), 267–282.

[3] H. Possegger, S. Sternig, T. Mauthner, P. M. Roth, and H. Bischof. 2013. Robust Real-Time Tracking of Multiple Objects by Volumetric Mass Densities. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2395–2402.