# Player Tracking and Analysis of Basketball Plays

Evan Cheshire, Cibele Halasz, and Jose Krause Perin

***Abstract*— We developed an algorithm that tracks the movements of ten different players from a video of a basketball game. With their position tracked, we then proceed to map the position of these players onto an image of a basketball court. The purpose of tracking player is to provide the maximum amount of information to basketball coaches and organizations, so that they can better design mechanisms of defense and attack. Overall, our model has a high degree of identification and tracking of the players in the court.**

## I. INTRODUCTION

Automated player detection and tracking in team-sport games is of growing importance [1]. As the profits from sports are increasing substantially, teams are heavily invested more in gathering statistics on their athletes. Certain statistics, such as distance run during a match, can provide information on player's health. Moreover, real-time detection of players can be valuable in identifying the opponent's formation and strategy, and might give some insight on the likelihood of a certain play be successful. This can lead to better strategies.

Our project is relevant because it is able to gather information regarding the position of the players in the court, as well, as data related to the style of play of each team. Such information could be crucial to the winning of matches, if well analyzed by the team's coach. More generally, our project could be adapted to the tracking of players of any sports' match.

## II. METHODOLOGY

This project was developed entirely on MATLAB. In order to achieve our end goal of a two dimensional image with player positioning, we made use of a five step algorithm, each of which will be further expanded:

1) Court Detection - find lines of the court;
2) Individual Detection - detect individuals standing on the court;
3) Color Classification - Separate these individuals into two teams;
4) Player Tracking - Keep positions information frame by frame;
5) Mapping - translate onto a court

The data for the project consisted of multiple YouTube videos which were then cropped in order for us to do our analysis. We mainly selected videos in which we were able to see all major lines of the court in order to accurately perform the homography.

emails: {evanches, cibelemh, jkperin}@stanford.edu

## III. ALGORITHM

Fig. 1 illustrates the block diagram of algorithm developed. The subsequent subsections discuss each stage of the algorithm in detail.

### *A. Court Detection*

The video frames that we obtained from Youtube were initially converted from the BGR to the HSV (hue, saturation and value) color model. We then focused on the H-plane in order to create a binary model of the system. Then, we proceeded to perform erosion and dilation of the image in order to get rid of artifacts that were not related to the court. Subsequently, we made use of the Canny edge detector to detect the lines in our system. Finally, we performed the Hough transform in order to detect the straight lines in the system. This process is illustrated by Figure 2.

### *B. Pedestrian Detection by Histogram of Oriented Gradients*

The next stage is pedestrian detection through histogram of oriented gradients (HOG). HOG essentially builds histograms of the gradient orientations in localized portions of an image, which can be used to identify objects in a image. While it is difficult to establish definite characteristics for these histograms in order to detect a certain object, machine learning classifiers, such as support vector machine (SVM), can be used to identify a desired object in a image based on a training data set. For pedestrian detection no single feature has been shown to outperform HOG. However, the performance can be improved by using additional features to provide complementary information [2].

For this project we used the HOG detector from OpenCV. This was mainly motivated by the fact that OpenCV has already a default data set for pedestrian detections, and that the HOG feature calculation and SVM were already efficiently implemented. We used the "Daimler" dataset for pedestrian detection. This detector is trained using window size of 48 by 96 pixels. Thus, the HOG detector expects pedestrians to be of at least that size.

Fig. 3 illustrates an example of pedestrian detection of basketball players using HOG descriptors and SVM classifier. The inset in that figure illustrates the HOG descriptors of one of the players as well as an inverse, which corresponds to a reconstructed image from the HOG descriptors. This image shows that HOG descriptors carry significant information of the detection. In the sample frame all players were detected, but there were some false positives, and two players were detected by the same box because they were too close together.

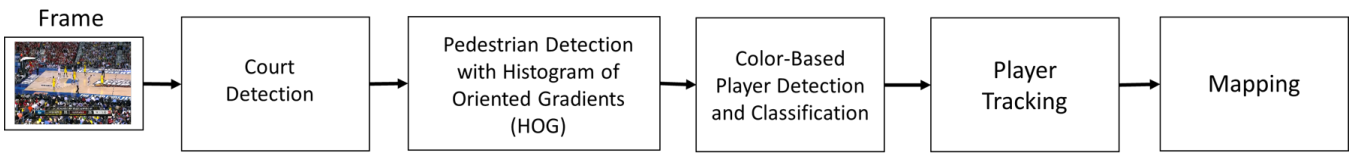



Fig. 1: Block diagram of algorithm for player detection and tracking.

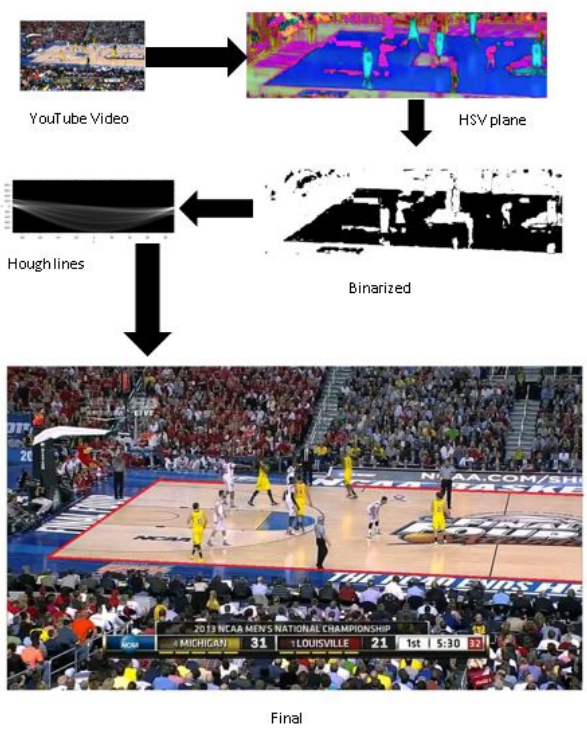



Fig. 2: This schematic represents the work performed in order to detect the edge lines of the system. The frame is initially converted to the HSV plane, then turned into a binary image. Later, we make use of the Hough transform in order to find the Hough Lines (straight lines) of the image

The HOG features with SVM have a miss rate of about 70% for pedestrian detection [2]. However for this particular application, the accuracy of the HOG detectors is expected to be smaller, since the players can fall, jump, or crunch to get the ball, and consequently will not be detected by the HOG detector. Thus, it is necessary a redundancy system to detect the players when the HOG detector fails. To this end, we built a color-based detector and classifier. This color-based detector detects the players based on their jersey's colors. The purpose of this detector is two-folded. Firstly, it is responsible for classifying the players according to their team as well as ruling out other "pedestrians" that might be detected by the HOG detector such as referees, coaches, audience members, for example. Secondly, the color detector should identify players within a HOG box. Quite often some players will be too close together or partially obstructed by other players. In these situations the HOG detector might

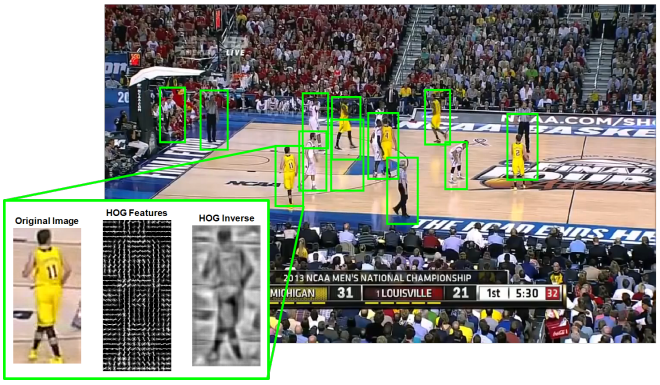


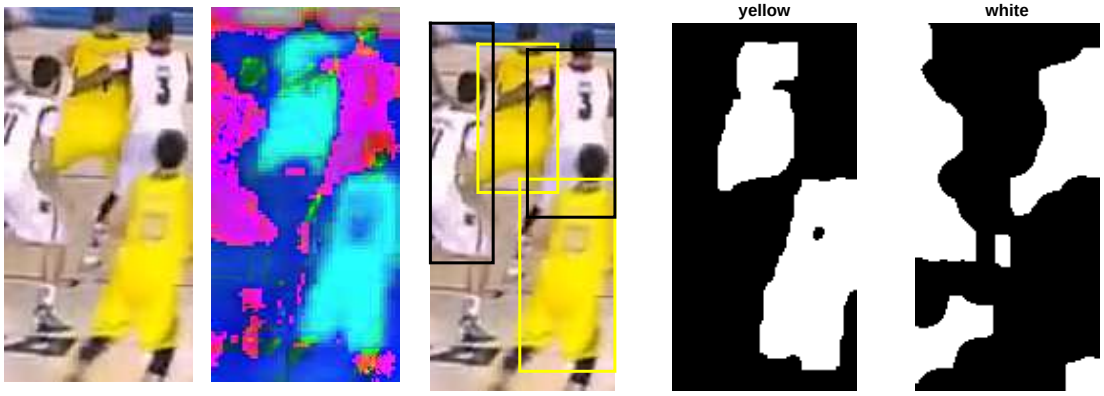

Fig. 3: Example of basket players detection using openCV pedestrian detection with HOG.

return a single detection (box) for all those players.

### C. Color-Based Detection and Classification

The color-based detector performs player's detection within a HOG box, which is a region of the original image classified as a pedestrian by the HOG detector. Color-based detection could also be used in a larger image (not necessarily in a HOG box only), for example detecting players in the entire court as done in previous projects [3], [4]. However, the HOG detector greatly improves the performance of this color detector, since the HOG boxes limit the scope to a couple of players. Thus other objects that might have the same color of the player's jerseys, such as details in the floor typically found in basketball courts, will not be detected as often as if an entire frame were used.

The color detector performs detection by using thresholds in the HSV space. The choice of the HSV space as oppose to RGB was motivated by the fact that the HSV enables higher discrimination between changes in color rather than saturation and brightness. For instance, a RGB-color-based initially implemented would constantly obtain false positives given by the reflections on floor. Given a set of images containing players from both teams, referees, members of the audience, etc. The histograms for all coordinates H, S, and V are calculated and thresholds are calculated using Otsu's method. Depending on the color of the two teams more than one threshold might be necessary to allow distinction between other elements that might appear in the image (e.g., yellow and floor). Once these thresholds are known we can derive logical expressions for color detection. For instance, yellow corresponds to (60º, 100%, 100%) in the HSV space, whereas white corresponds to (0º, 0%, 100%). Thus, we can distinguish between yellow and white by requiring the hue




(a) RGB (b) HSV (c) Detection by thresholding

Fig. 4: Illustration of color-based classifier. (a) represents the original image in RGB, (b) is the original image converted to HSV, (c) is the image after binarization through thresholding. Closing and dilation using circle of radius 10, and $15 \times 5$ rectangle is performed in the resulting binary image. The algorithm selects only boxes that meet certain criteria in size and extent. Thus, smaller boxes are ignored and false positives are avoided. As we can see in (c) all four players were classified correctly.

and saturation coordinates to be higher than their respective thresholds for yellow, and below a certain threshold for white.

Fig. 4 illustrates the color detection process for a white vs yellow detection. Once the thresholds are determined from a training data set, the color detection can be performed fairly fast since it basically comprises of comparisons and logical operations. To avoid some false positives, some additional criteria is enforced. For instance, boxes that correspond to more less than $5\%$ of the image area are disregarded. Moreover, valid boxes are expected to have height higher than width since basketball players are tall and they're standing.

The color detection also plays an important role as a backup of the HOG detector. As the game proceeds some players might become partially obstructed by other players, or they might simply not be detected by the HOG detector in a certain frame. In these different scenarios, the color-based detector will be called to find the missing player by performing detection in small neighborhood of the corresponding box from the previous frame. This works fairly well because we used videos of rate of 24 fps, thus the player is expected to be in the surroundings of where he was in the previous frame. These different conditions in adding players and dropping players in the detection is done by the tracking algorithm discussed in the next subsection.

Fig. 5 shows two sample frames after color detection. Note that Fig. 5-(a) corresponds to the same frame of Fig. 4, and the false positives were eliminated and the ambiguity of two players being detected by the same box was eliminated.

### D. Tracking

Once detected, the next goal is to establish a frame by frame positioning of the individual players in order to understand the play in total. Thus, a tracking algorithm kept track of the players' movements. This algorithm used the information from the previous frames for initial conditions on tracking. We dealt with the following scenarios.

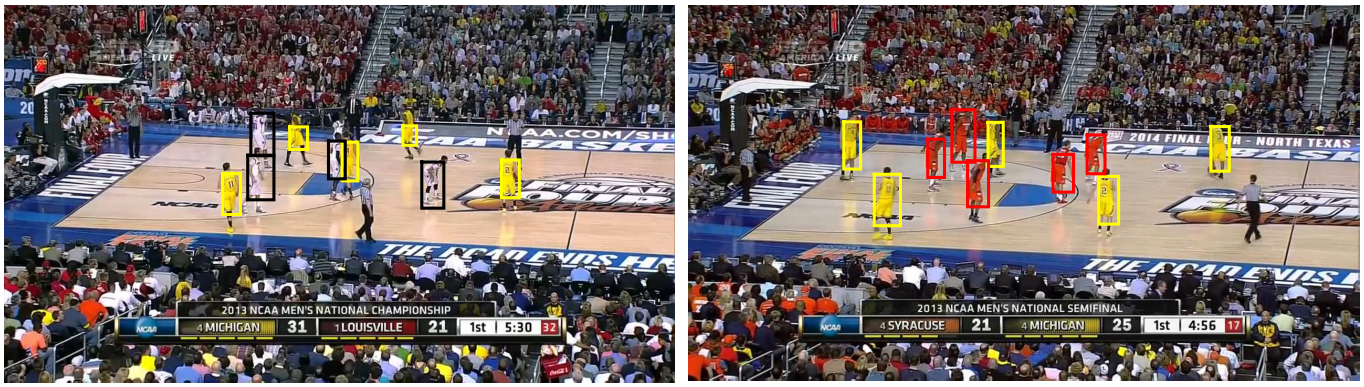
*1) Scenario 1: Player Detected in Consecutive Frames:* In this scenario, a player was detected by the HOG detector and color classifier in back to back frames. Because the frames are taken at a rate of 24 fps, the position of player from one frame to the next are highly related. Thus these boxes that are detected would have high overlap. In this case, the position of this player is updated with the new position, and the new position is saved.

*2) Scenario 2: Neighborhood Estimate:* Ins this scenario, a player was not detected by the HoG detector, but can be found by a color detection in the neighborhood. With this frame-by-frame scenario, the HOG does not detect every player in every frame. Players can be in motion and blurred, crouching, or in some way be undetectable to the HOG detector in a particular frame. A player detected in the previous frame would then have no correlation to a player in the current frame. We search for this missing player within a 20 pixel bound around the location of the player in the previous frame. If the color detector can identify a player with the same jersey within this box, it matches this player to this new position. If multiple players are found within this box, it matches this player to the closest previous position. Thus lose a HOG detection for a frame does not result in the loss of the player' positioning.

*3) Scenarios 3 & 4: Adding and Dropping:* The next two scenarios, though quite different, have a similar solution:

Scenario 3: A player was not originally detected in the first frame, but was found at a later time. In this scenario, a player was not detected in the 1st frame. The HoG detector was not able to originally find this player. This scenario will be referred to as an add. In the original frame, 9 players were detected on the court in Fig. 6a; however, in a later frame, this 10th player was added in Fig. 6b.

Scenario 4: A player who had been previously identified




(a) Michigan vs Louisvielle

(b) Michigan vs Syracuse

Fig. 5: Example of player's detection using HOG pedestrian detection and color-basedclassification in two different games.

(a) The first frame of the play

(b) A player was added

(c) 3 players merged

Fig. 6: Different tracking scenarios that might occur throughout the game.

by the HoG detector and color classifier was dropped. This will happen when players of the same team merge together, and the color classifier is unable to distinguish them as separate players. This scenario will be referred to as a drop. In the first frame for example, one is able to distinguish five separate players on the yellow team in Fig. 6a. However, because the players converge, only 3 yellow masses are distinguishable in Fig. 6c. Therefore, information is lost.

These scenarios have a combined solution, a minimum distance correlator. On every single frame, the boxes and positions are stored. If a player is dropped, after a certain period of time, the HoG detector will detect him again. Because no box from the previous frame correlates to him, he will have had no established previous position. The algorithm checks to see if a player who had been dropped is relatively close to his position. The distance allowed is fixed distance multiplied by the number of frames since this player was lost. If this player was within the distance of any previously dropped players, he is then correlated back to this original player, combating the drop issue. However, if no previous player was within range of this current player, a new player is added to the tracking data, combating the add issue.

This tracking algorithm solving scenarios 3 and 4 is far from perfect. Tracking itself is highly dependent on the detection information. False positives from the bench or the court can lead to bad minimum distance guesses if a player is lost. With players merging on the court, solving the drop problem is especially difficult. Players' motion change rapidly and do get convoluted, and using a players' original motion is not reliable to match missing players to previous frames. The algorithm is also sensitive to camera jitter. The best solution to this add and drop problem would be to have multiple stable camera viewpoints, which is not available to us from a broadcast, but would be available within a professional environment such as a NBA team.

### *E. Mapping via Homography*

The last step was related to the projection of each player's location in the top-down view of the court. By having the dimensions of the court, we are able to find a 3x3 homography matrix that is computed using an affine transform. Each player's position is then multiplied by the homography matrix that projects them into the model court, as shown in Fig. 7.

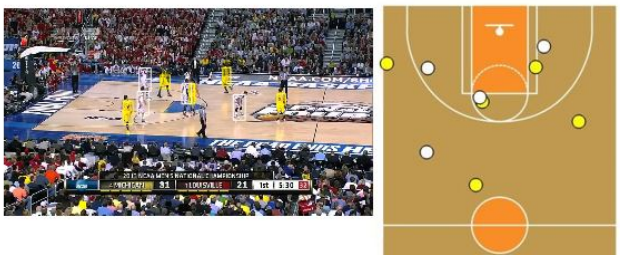


Fig. 7: Comparison between the players detected and the projected image in the court. As you can see the players match the position in the top-down court model.

## IV. ANALYSIS AND RESULTS

In this section we use the algorithm developed to analyze the positions of the players as the play progresses. As an example, Fig. 8 shows the heatmap of the players positioning as the play progresses. The position of the team is compatible with the video since we can see that the white team remains on the defense throughout the play and the yellow team only crosses the white team's defense line to score a basket.

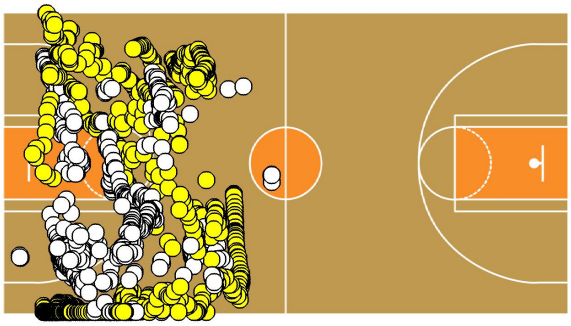


Fig. 8: Heatmap of the players location between teams yellow and white

A similar heat map can be created for the second game between Michigan (represented by yellow) and Syracuse (represented by red), Fig. 9. We can see the distribution of players in the field is compatible to that seen in the video, which indicates that our player detection and homography are working in conjunction in order to provide us with useful data related to the players location in the court.

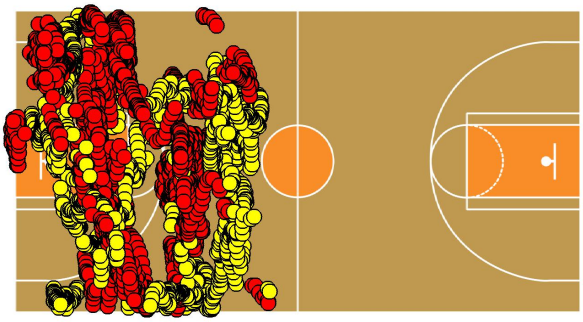


Fig. 9: Heatmap of the players location between teams yellow and red

In order to further validate the results of our tracking let's break down the Syracuse and Michigan game into the two teams. Then, if we analyze only the positions of the players from Michigan we get the image compatible with Fig. 10 for the first 60 frames. In those, we can see that we are correctly labeling players 1 through 5 in the Michigan game. Furthermore, there movements seem to be compatible to those represented in the video.

What is most impressive, however, is the ability of our player identification and tracking system to capture the

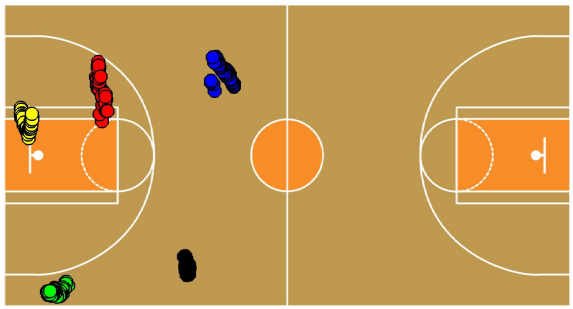


Fig. 10: Positions of players from Michigan team. Each player is represented by a different color. The movements of the players seem to be compatible to that of the video

movement of one single player that crosses the court, passing through many other players and is still correctly labeled and identified. This can be perceived by Fig. 11.

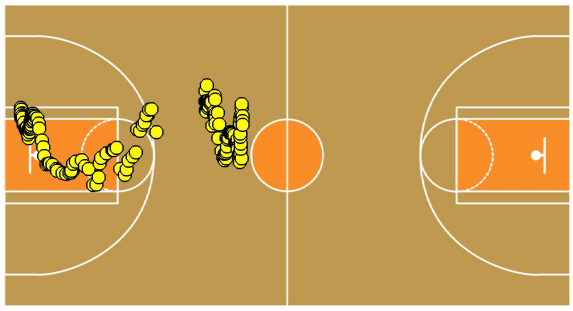


Fig. 11: Position of Michigan player identified as player 1. As we can see the color segmentation and the tracking appear as a viable way to track this player across the court

## V. CONCLUSION AND FUTURE WORK

We have developed an algorithm that accurately detects basketball players in a video and is able to accurately place them in a 2D top-view court. We have achieved such results making use of an association of the hog detector from OpenCV and color segmentation. Future work for this project would involve better adjusting the color segmentation thresholds to avoid artifact identification as well as improving the accuracy of our tracking system.

## VI. APPENDIX

### A. *Acknowledgements*

The authors would like to thank Andre Araujo for his mentorship throughout the quarter, and the rest of the EE368 teaching staff for putting together this incredible course.

### B. *Work distribution*

All students contributed to the making of the poster and writing of the paper.
Evan: Tracking of players

Jose: Color segmentation and player detection
Cibele: Court detection, homography and analysis

## REFERENCES


[1] Wei-Lwun Lu, Jo-Anne Ting, James J. Little, Kevin P. Murphy, "Learning to Track and Identify Players from Broadcast Sports Videos," IEEE transactions on pattern analysis and machine intelligence, 2011.

[2] Dollar, Piotr, et al. "Pedestrian detection: An evaluation of the state of the art." Pattern Analysis and Machine Intelligence, IEEE Transactions on 34.4 (2012): 743-761.

[3] Scott Parsons and Jason Rogers, "Basketball Player Tracking and Automated Analysis," EE368 final project, Spring 2013/2014.

[4] Matthew Wilson and Jerry Giese, "Basketball Localization and Location Prediction," EE368 final project, Winter 2013/2014.