BASKETBALL PLAYER TRACKING

Simon Xie, Clive Unger, Kevan Patel

ABSTRACT

The purpose of our project was to develop an algorithm that can automatically track player location and movement in sports from game clips. If successful, we postulate that athletes can be tracked in real-time by TV Cameras operating on an obviously more complex version of our algorithm during a live broadcast. For our study, we decided to focus on tracking basketball players due to the smaller player count and well-defined court. We developed a Python algorithm that can track basketball players on each team from a clip of a basketball game. Tracking players can be beneficial for sports organizations to monitor the health and workload of their players as well as analyzing and developing play strategy. Fundamentals principles and methodologies of our algorithm are transferable to tracking players in any sport, which is the ultimate goal.

INTRODUCTION

The global sports industry is valued at an estimated $500 billion and is projected to reach $615 billion by 2022. Sports viewership pervades race, ethnicity, religion, and socioeconomic class and brings people together, making it a core component of our modern society. Due to its importance and continuous rapid worldwide growth, there has been a significant interest in applying image processing and computer vision techniques to sports to optimize both play itself and viewership experience in recent years[1]. In particular, the ability to track athletes while they perform has intriguing potential. Ownership and coaching staff of a sport's club are very interested in monitoring the performance and health of their athletes. With sports contracts reaching exorbitant amounts, and injuries continuing to rob all sports of some of its most popular and electrifying athletes, being able to track and monitor workload during a game and after can help minimize injury and save professional and amateur clubs millions of dollars. In addition, tracking also has potential in allowing coaching staff to evaluate strategy and its effectiveness for both the opposition and their own teams.

In 2005, Harvard vision scientists utilized image segmentation techniques to successfully detect an athlete's jersey number in sports video clips[2].Since then many other studies have been conducted on developing a system to automatically track players during a live broadcast video. Common documented challenges include developing the algorithm so that it can deal with player's who are overlapping or running by each other (as in football) and having it be effective when the camera angle constantly changes and varies in magnification during the broadcast[3]. For our project, we decided to focus on creating an algorithm that can track players over the course of a few seconds from a non-live broadcast video of a sporting event that has already been completed. We aspire to provide a proof of concept of athlete tracking, and hope that fundamental elements of our algorithm can be applied to live-tracking and analyzing player movements in live broadcasts. We specifically focused on tracking basketball players of both teams from a basketball game clip as the lower player count and well-defined court make it ideal for developing a proof-of concept algorithm. The end goal of our algorithm is to have players separated by team and their

positions distinctly marked on a 2D map of a standard basketball court. The positions will then change accordingly as the respective players move in the input video.

METHODS

The project was written in Python coding language. We developed and followed a structured approach that can be transferable to sporting events aside from basketball. We tried to automate as much of the process as possible so none of the processing relied on manual intervention. The video used in this project is a Texas vs Baylor game downloaded from YouTube [4]. We selected this video because it is our school's team, but we also acknowledge that the Baylor player jerseys are much easier to detect.

We break down the basketball player tracking into five key steps:

1. Court Extraction
2. Detect Players
3. Identify Player Jersey Colors
4. Find Player Coordinates
5. Map Points to 2D Plane

COURT EXTRACTION

We first extract the court from the image so we can focus only on the actors within the court and ignore the background audience and other noise. Identifying the average court color in the RGB or BGR color spaces is difficult because the values are heavily affected by shadows and lighting. Therefore, the frame is changed from the BGR color mapping to HSV. The HSV color mapping stands for Hue, Saturation, and Value. The Hue represents the wavelength of the colors, so it is much more reliable for color extractions because the shading of the color does not affect the Hue value. We plot a histogram of all the Hue values in the first 10 frames and identify the most frequent value as the court color since it takes up the majority of the frame. This value is then used to binary threshold each frame with a tolerance of +/- 7. Various morphology filters are applied to smooth out the image. The result image is then used as a mask for processing the rest of the image.

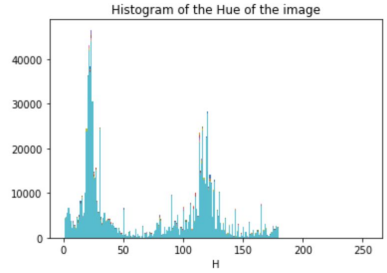

Figure 1: Histogram of Hue Values for one frame

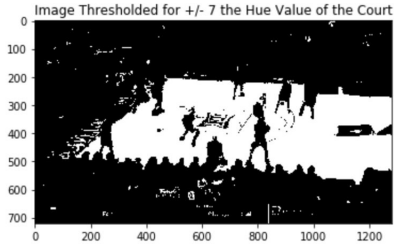

Figure 2: Applied binary threshold with a +/- 7 tolerance

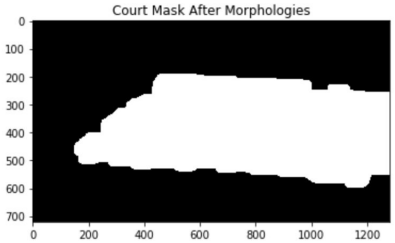

Figure 3: Erode and dilate filters applied to remove artifacts

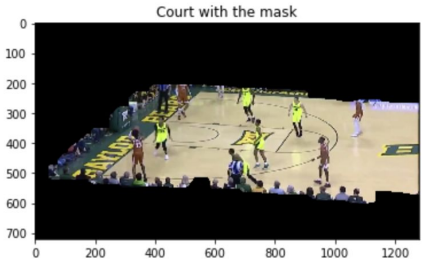

Figure 4: Overlaid RGB image on court mask

DETECT PLAYERS

Once the court mask is applied to the frame, we are left with just the players on the court. We utilize the HOG (Histogram of Oriented

Gradients) and Linear SVM method to try our best to detect players. Thankfully, OpenCV already has a pretrained HOG + Linear SVM model that can be used to detect pedestrians, which we will use to detect the players[5]. The model will take an input of bounding box size and will pan over the whole image to detect which bounding boxes in the image contains a pedestrian. We decided to use the RGB color space to find these bounding boxes as the HOG detector performed better in this color space. The model also returns the weights or confidence the model has for each box. The higher the weight is, the higher the confidence the model has that it detected a person. We tested a threshold confidence that worked the best, and we determined that a threshold of around .6 worked the best. One problem with the output of the model is that there can be redundant bounding boxes that capture a single player. Therefore, we must use the non_max_suppresion function from the imutils library. A quick summary of what this function does is that it takes bounding boxes that overlap each other by a certain threshold, and merges the boxes into one box. After running this function, all of the bounding boxes discovered will correspond to a unique player.

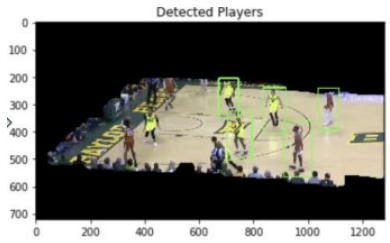

Figure 5: Players detected using Hog pedestrian detection

IDENTIFY PLAYER JERSEYS

Once the bounding boxes for the players are computed, we find the peak Hue value for each box with the court values masked out. In one of the histograms shown below, we see that one of the peak Hue values is 38. We then added all the peak hue values to a list. The histogram of this list should be bimodal since there are two primary jersey colors. We threw out the outliers in the list as we may have detected the referee or crowd members. Finally, we averaged the 2 clusters over Hue values, and those values are used to represent the average jersey colors of each respective team.

The image with the court mask is then thresholded on these hue values +/- 7 and two masks are created for each team color. Morphological filters are applied to eliminate noise and smooth out the player blobs.

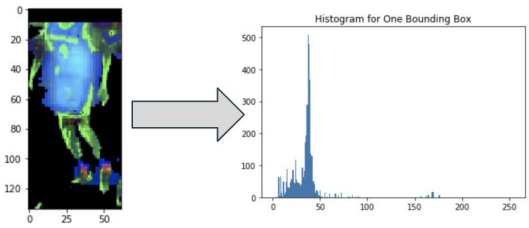

Figure 6: Peak Hue value identified for Hog bounding box

FIND PLAYER COORDINATES

From the player masks there should be roughly 5 large blobs that represent players. Sometimes there is more or less depending on how complex the frame is. We then identify the 5 largest connected components in the image using OpenCV's connected component algorithm. The centroids of each connected component is used to represent the player coordinates. The y-coordinate is shifted down slightly so the coordinate is closer to the player's feet.

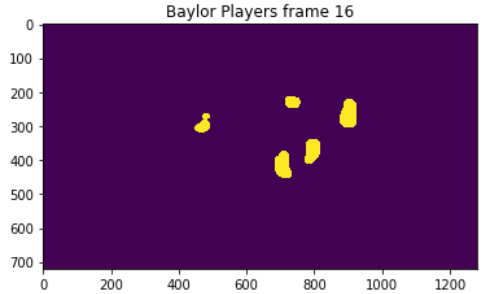

Figure 7: 5 resulting blobs after application of Baylor player mask

MAP PLAYER POINTS 2D PLANE

We manually identify 6 points in the 3D space that correspond to points on the 2D representation of a basketball court. These points are used to calculate a homography matrix using OpenCV's findHomography function. This returns a matrix which can map points in the original space to the source space. The player coordinates are then multiplied by this matrix to get their respective coordinates on the 2D-map.

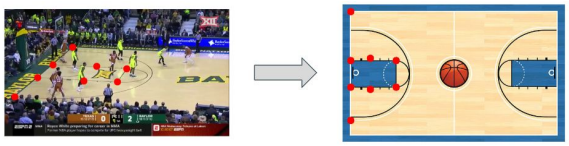

Figure 8: Demonstrates how the manually labeled reference points are mapped to 2D court map

We found many inaccuracies resulting from the camera movement. To compensate, for each frame we find the top left corner of the court mask which roughly corresponds to the top left corner of the court in the image. The offset of this coordinate from the original manually labeled top-left corner is used to adjust all points. The homography matrix is recomputed each frame to adjust for the camera movement. We found this greatly improved performance, but were unable to handle camera zoom.

RESULTS

After mapping the blobs to the 2D court map, we can see all of the results of a single frame in the figure below. The Baylor players are shown in the blue circles and the Texas players are shown in the green rectangles. All of the Baylor players are accurately mapped since they have a very bright and distinct color. However, the Texas players were not detected as accurately since their colors are not as bright and distinct. Also, one of the Texas jerseys is outside of the court mask, so that player was not easily detected. We were not able to dilate the court mask anymore since a lot of the writings on the floor and the bench/audience were detected, so we had to settle with a court mask that is only the size of the court.

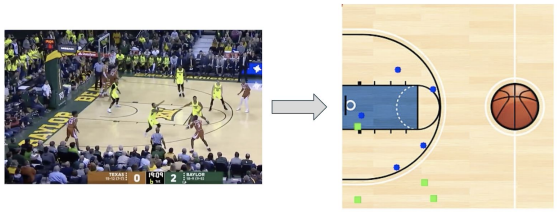

Figure 9: Texas and Baylor player locations mapped to 2D court

The work for the project was evenly distributed between team members. The original team was Simon and Clive, however Kevan joined later because his original partner dropped the course. Simon primarily focused on the player detection aspect using the HOG identifier and extracting the jersey colors. Clive started the original court mask concept and developed the connected components function to find the player coordinates. Kevan found the mappings from the 3D court image to the 2D image to create the homography matrix. He then worked with Clive to adjust for camera movement. Everyone worked on the paper and presentation.

FUTURE APPLICATIONS

The methods we developed in the project could easily be applied to any sporting event. For example, in a soccer game, the color of the field would be separated from the color of the players and the identification and tracking would be performed the same way.

Given more time, we would have spent more time testing out methods on multiple videos, so that it could be a general tool to track players from any basketball game. One of the challenges with creating a general tracking is identify the key points of the court to calculate the homography from. In our method we manually labeled points, so we could use something like SIFT or SURF to identify the court lines.

WHAT WE LEARNED

This project was so fun to work on and we ended up learning a lot about image processing. First we understood the trade-offs between different color spaces. We tested alternatives to RGB such as YCbBr before sticking with the HSV colorspace. We learned how HSV is the best way to extract colors from an image that has multiple different lighting scenarios. Beyond just color extract we saw how different color spaces are advantageous for certain algorithms. For example the HOG pedestrian detection performed slightly better in the RGB space. We also realize that if we were to apply neural networks to the application we would need to consider which color spaces to use and how they might affect performance.

Further, we learned how the basic concepts we learned at the beginning of class can be taken a long way to achieve solid results. We did not use machine learning in this project, simply because we saw no obvious applications that we could leverage immediately. Therefore, we relied on simple morphological filters to denoise our player extraction. Simple intuitions could guide the shapes of the structuring elements, but experimentation ultimately decided the final filter design. It was surprising how much time was spent tweaking numbers and observing the results.

Lastly we learned that tracking multiple agents in a frame is a difficult problem! We ran into multiple issues that hurt our accuracy. For example, the camera movement was hard to compensate for, especially when the camera zooms in to a shot. Another issue occurs when players overlap each other and are detected as one entity or stand on the edge of the court and are cut out by the mask. By the end of the project, we had so many more ideas for how to improve the project, we wish we would have had time to experiment and implement them all!

Special thanks to Professor Bovik for the great course.

REFERENCES

[1] Li, Guangjing, and Cuiping Zhang. "Automatic Detection Technology of Sports Athletes Based on Image Recognition Technology." *SpringerLink*, Springer International Publishing, 18 Jan. 2019, link.springer.com/article/10.1186/s13640-019-0415-x.

[2] Ye, Qixiang, et al. "Jersey Number Detection in Sports Video for Athlete Identification." *NASA/ADS*, July 2005, ui.adsabs.harvard.edu/abs/2005SPIE.5960.1599Y/abstract.

[3]Lu, Wei-Lwun, et al. *Learning to Track and Identify Players from Broadcast Sport Videos*. IEEE Transactions on Pattern Analysis and Machine Intelligence, www.cs.ubc.ca/~murphyk/Papers/weilwun-pami12.pdf.

[4]"Texas vs Baylor Men's Basketball Highlights." *Youtube*, 27 Feb. 2019, https://www.youtube.com/watch?v=-jddnr32hfU.

[5] Rosebrock, Adrian. "Pedestrian Detection OpenCV." *PyImageSearch*, 2 Aug. 2018, www.pyimagesearch.com/2015/11/09/pedestrian-detection-opencv/.