Analysis of Tennis Rallies from Single-View Broadcast Videos

John Boccio
Stanford University
Department of Electrical Engineering
jboccio@stanford.edu

Abstract

Being able to track the shot path of a tennis ball from a match's video broadcast feed would enable us to automate the gathering of rich metrics for professional players. Using the ball position information gathered from such a method would allow us to analyze various hitting patterns which could be very useful for sports analytics. Today's tennis datasets do not provide such fine-grained details but instead give high-level statistics on metrics such as percentage of aces, 1st serves won, break points won, and games won on a specific surface type for a single professional across their entire career. In this work, I will create an automated method for automatically calibrating the broadcast video camera from a single view of the point. Additionally, I will be using a deep learning approach to track the 2D position of the tennis ball throughout the frames. Once you have a calibrated camera and a 2D position of the tennis ball, additional algorithms can be applied on top of this project to extract additional information. Although this is outside the scope of my project, you could do things such as extract the 3D position using prior physics knowledge alongside the calibrated camera and tennis ball tracking. To automatically calibrate the camera and track the 2D position of the tennis ball in frames, I have split the problem into a few different steps: tennis court detection, camera calibration using tennis court key points and prior world knowledge using tennis court dimensions, and 2D tennis ball position extraction using deep learning.

1. Introduction

With the rise of machine learning and data analytics, the demand for large and very detailed datasets has increased. One area where this has been seen is within the field of sports analytics. Many individuals, and even corporations, are willing to pay for datasets that they can use to try to predict who will win a match. The datasets that are available today are severely lacking in finer details. They will typically provide more marco level statistics such as the percentage of points they win, percentage of unforced errors, percentage of points won at net, and average match time for each player across their entire career. While these are all still useful in some scenarios, it lacks detailed information about that individuals play style. If you had a calibrated camera and information on the tennis ball path, new algorithms could be built up upon those methods to do things such as detect where the ball bounced and the 3D shot path. Using this, you could begin creating a profile on that player which describes the kind of hit patterns a player uses and which ones they are weak against. As an example, if one tennis player tends to hit very consistent deep baseline shots and another player does not return deep baseline shots well, then a dataset built off of these algorithms would be able to provide these metrics to you. Gathering that sort of data manually would not only be very tedious but also time consuming and costly.

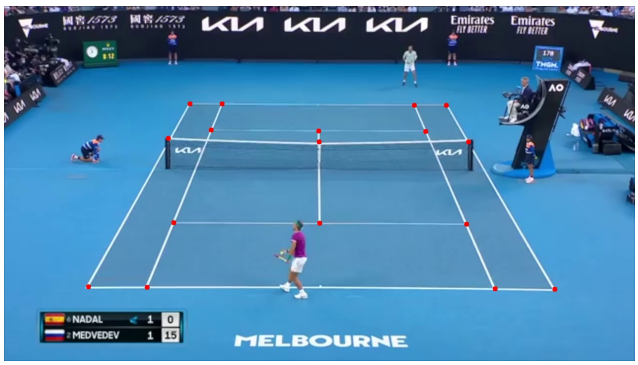

Figure 1: Tennis court with keypoints in red. These keypoints include where all of the tennis court lines intersect as well as the keypoints at the end of the net line. We know the dimensions of the tennis court so we can gather the world coordinates of these keypoints which can be used to calibrate the camera.

What I am going to show in this paper is that we can automatically calibrate a camera and get the 2D position of the ball throughout the frames fairly accurately just from a

single view broadcast video of a tennis match point. Unfortunately, there are no readily available datasets I which had the tennis court labeled to help with determining the accuracy of an automatic camera calibration algorithm, so I will be creating a small one myself to help me gather some basic quantitative measurements. As for tracking the 2D position of the tennis ball through the frames of the broadcast video, I will be using an open sourced CNN called TrackNet [3, 4].

In order to approach this problem, I have split it up into a view separate subproblems: tennis court pixels detection, data fitting using the detected court pixels, tennis court keypoint extraction, automatic camera calibration using the keypoints, and tennis ball tracking throughout the frames using TrackNet. As mentioned earlier, doing things such as detecting where the ball is landing and the 3D path of the tennis ball throughout the video is outside of the scope of the paper. However, the methods outlined in this paper can be used as a starting point for such algorithms.

2. Background & Related Work

In computer vision, homogeneous coordinates are often used so that way we can leverage the common notation used within linear algebra. Homogenous coordinates describes the position of a point in 2D as p and its associated 3D position as P, which can be seen in equation 1. The last coordinate is used as an inverted scaling factor for the other coordinates.

$$p = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, P = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

$$p = M \cdot P \tag{2}$$

Lines can also be represent compactly using vector notation. The equation of the line in 2D takes the form of $ax + by + c = 0$ and its vector representation is given in equation 3.

$$l = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{3}$$

Using this representation, the point of intersection of two lines can also be formulated by using the cross product as shown in equation 4.

$$p = l_1 \times l_2 \tag{4}$$

A line can be fit to a set of 2D points by setting up the equation $ax + by + c = 0$ in matrix notation. Then the SVD of matrix A can be used to find the line l which fits the data points. The formulation for this is shown in equation 5.

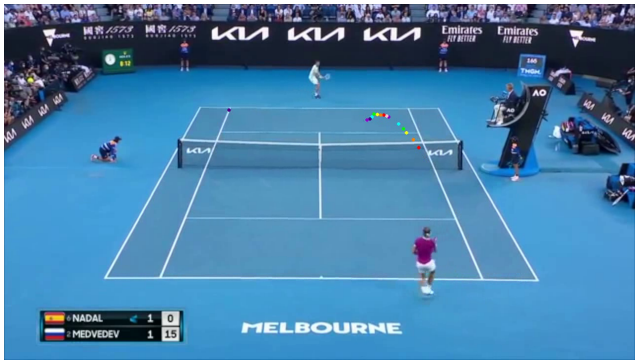

Figure 2: Frame from broadcast video with TrackNet applied. The colored circles show the past 15 positions of the ball that TrackNet predicted.

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ & \vdots & \\ x_n & y_n & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \boldsymbol{A} \cdot l = 0 \tag{5}$$

Camera calibration refers to finding a matrix M (as seen in equation 2) which can map 3D positions to 2D positions. The method in which we find the values of the matrix M is outlined in equation 6 and comes from [2] which explains its derivation in much more detail.

$$\begin{bmatrix} P_1^T & 0^T & -u_1 P_1^T \\ 0^T & P_1^T & -v_1 P_1^T \\ & \vdots & \\ P_n^T & 0^T & -u_n P_n^T \\ 0^T & P_n^T & -v_n P_n^T \end{bmatrix} \cdot \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix} = \boldsymbol{P} m = 0 \tag{6}$$

In order to perform automatic camera calibration in a frame of the tennis match, the system of linear equations in 6 will be solved. In this equation, the points P_i represent the world coordinate positions of the keypoints of the tennis court. These are known beforehand since we have the dimensions of a standard tennis court and is the reason why we are able to calibrate the camera from the single view broadcast video frames. Figure 1 shows the points p_i in red, these are the points in the 2D image that are associated to the world coordinate points P_i. The points p_i will need to be detected through a computer vision algorithm which will be outlined in detail in section 3.1. The methods in which this was done had a lot of inspiration from J. Han, D. Farin, and P. H. de With's paper on "Generic 3-D Modeling for Content Analysis of Court-Net Sports Sequences" [1]. Combining all this information together and using equation 6 allows us to perform the automatic camera calibration.

Figure 3: Result of masking out all the non-white pixels on a frame from a broastcast video. No additional algorithms are applied except for masking in this image.

In addition to calibrating the camera, this paper will also use TrackNet [3, 4] to track the 2D position of the tennis ball throughout the frames of the broadcast video. TrackNet utilizes a convolutional neural network (CNN) to track the tennis ball and a visualization of it's output can be seen in figure 2. Once you have a calibrated camera and the 2D position of the tennis ball in the image, further algorithms can be created to do things such as extract the 3D tennis ball path. Two papers which do this are "Estimating 3D Positions and Velocities of Projectiles from Monocular Views" [5] and "3D reconstruction of ball trajectory from a single camera in the ball game" [6]. I will defer the reader to those papers if they are interested.

3. Implementation

3.1. Automatic Camera Calibration

In order for the system of linear equations described in equation 6 to not be underdetermined, we need to have at least 6 keypoints identified in the 2D image. These points must correspond to points in the world coordinates that we know. Additionally, those 6 points cannot lie in the same plane as that would also cause the system of linear equations to become underdetermined. So in order to achieve camera calibration, the keypoints that will be used are the four corners of the tennis court and two endpoints of the net line.

The way in which these keypoints are extracted begins with first finding all the pixels that correspond to the tennis court lines and the net line. With these pixels, we can then use a line fitting algorithm to find each of the lines of the tennis court and the net line. Finally, the intersection of the lines will give us the keypoints that we need to perform automatic camera calibration. The finer details of each step in this process will be outlined in the next few sections.

3.1.1 Detecting Tennis Court Pixels

The first task step that we take to extract all the pixels of the tennis court lines is to apply a mask which will remove all non-white pixels. The reason we are looking for white pixels is because the tennis court lines and the net line are white. Since the pixels aren't extirely white, a range of RGB values must be provided to mask colors that are not close to a white color. The mask will keep all pixels which have RGB values between $[180, 180, 100]$ and $[255, 255, 255]$. The pixels that do not fit within this range will be set to black: $[0, 0, 0]$. Then the image in converted into a binary image to make things simplier to deal with. Since we are dealing with broadcast videos, this still leaves a lot of noise in the image as white pixels can be seen in areas other than the tennis court. The goal now is to remove the white pixels that come from objects other than the court lines. An example of all the white pixels detected in a frame from a broadcast video is shown in figure 3.

A safe assumption that can be made is that the tennis court is very often in the center of the image. Due to this, we can safely set all pixels around the border of the image to black. We want to make this border as large as we can without risking accidently cutting off some of the actual tennis court. Cutting off 10% on each side of the frame seemed to be a safe assumption that helped with reducing the noise in the image.

A final algorithm is applied to the binary image to try to remove the white pixels that do not appear within a line of other white pixels. The method in which this is done is also outlined in [1]. Let τ_h and τ_v represent the width of the horizontal and vertical tennis court lines in pixels. For each white pixel, check if the pixels τ_h to the left and right of the given pixel is white. Similarly, check if the pixels τ_v above and below the given pixel is white. If the pixels τ_h to the left and right or the pixels τ_v above and below then we should keep the current white pixel. However, if the pixels to the left, right, above, and below are all white, then we should not keep the current white pixel as it is occurring within a block of other white pixels and we are trying to find the tennis court lines. A few different scenarios are visually shown in 4 for further understanding.

3.1.2 Finding Tennis Court Lines from White Pixels

Now that the white pixels that mostly come from the tennis court lines and the net line have been identified, the next step is to fit lines to each of them. In order to fit a line that each of the white pixels belongs to, the RANSAC algorithm will be used. RANSAC allows you to specify a residual in which the data points are considered inliers. In order to detect the court lines, the residual can be set to approximately the width of the tennis court line in pixels. With this configuration, a single run of RANSAC with the data points being

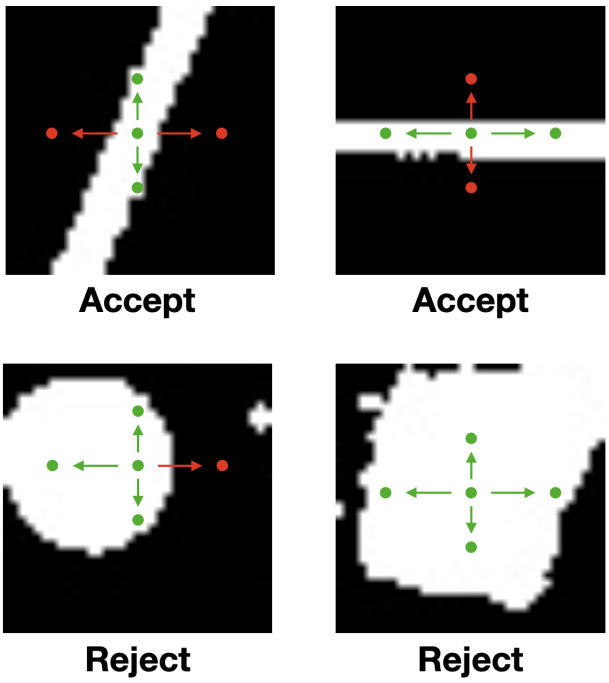

Figure 4: Examples of filtering out white pixels using τ_v and τ_h

the locations of the white pixels in the image will result in a set of white pixels that occur within a line and has the maximum number of inliers within the residual offset. A line can be fit to the inliers of this particular run of RANSAC to obtain line which corresponds to a tennis court line or the net line in the frame. The method in which this can be done is shown in equation 5. With that tennis court line or net line identified, the white pixels that were the inliers from that run of RANSAC should be removed from the data points. Then RANSAC can be ran again to find the next set of inliers which corresponds to a different tennis court line. There are a total of 9 tennis court lines and 1 net line that need to be identified so this process should be repeated 10 times to extract all the lines.

3.1.3 Finding Tennis Court Keypoints from Tennis Court Lines

With the 9 tennis court lines and 1 net line identified, the keypoints can be identified by systematically finding the intersection of lines corresponding to a particular keypoint. The point of intersection of two lines can be found by using equation 4. For example, to find the keypoint at the bottom left of the tennis court, the intersection point of the left doubles line and the close baseline should be used. To be able to do this, each of the lines must be identified with a line on the tennis court so that the intersection points can be computed with the correct lines.

In order to identify what each line corresponds to on the tennis court, a simple approach was taken. First, identify which of the lines are horizontal by looking at the angle of that line. If the angle of the line is below some threshold θ_h, then it will be classified as a horizontal line. Once the horizontal lines have been identified, find the midpoint of each of the lines and sort them by their y-coordinates. The order of the sorted horizontal lines corresponds to the order of the lines in a tennis court: far baseline, far service line, net line, close service line, close baseline. The same approach can be used for the vertical lines, which are the lines that were not labeled as horizontal. Once the lines are sorted by the x-coordinate of their midpoints, then you can identify them as the left doubles line, left singles line, center service line, right singles line, and right doubles line. One check that can be done to verify that the court detection was done correctly is to check that there were 5 horizontal lines detected and 5 vertical lines detected.

Now that each of the lines are labeled with the line that they correspond to on the tennis court, it is possible to correctly intersect the lines to find the keypoints. As stated before, the point of intersection of the left doubles line and the close baseline will return the keypoint in the bottom left, the point of intersection of the right doubles line and the far baseline will return the keypoint in the top right, and so on. Doing this for all of the pairs of lines that intersect will find all of the keypoints on the ground plane (the tennis court). The final step is to find the keypoints that correspond to the top of the poles that hold up the net. The line that corresponds to the net line is known so the endpoints can be found by looking at the furthest left and furthest right inlier that occurs between the left doubles line and the right doubles line. These two points are the endpoints of the net line and thus gives us the final keypoints that are needed to calibrate the camera.

3.1.4 Calibration Using Keypoints

Now that the keypoints have been located, calibrating the camera is quite straightforward. All that is required is to solve the system of linear equations to find the matrix M which maps 3D coordinates to the 2D pixel locations as shown in equation 6. To map out the corresponding 3D world coordinate points, the bottom left keypoint of the tennis court can be used as the origin, $[0, 0, 0]$. A standard professional tennis court is 23.77 meters long and 10.97 meters wide. The world coordinates of the top left, top right, and bottom right keypoints are therefore $[23.77, 0, 0]$, $[23.77, 10.97, 0]$, and $[0, 10.97, 0]$, respectively. Lastly, the net line splits the court in half and has poles that have a height of 1.07 meters are placed 0.91 meters away from

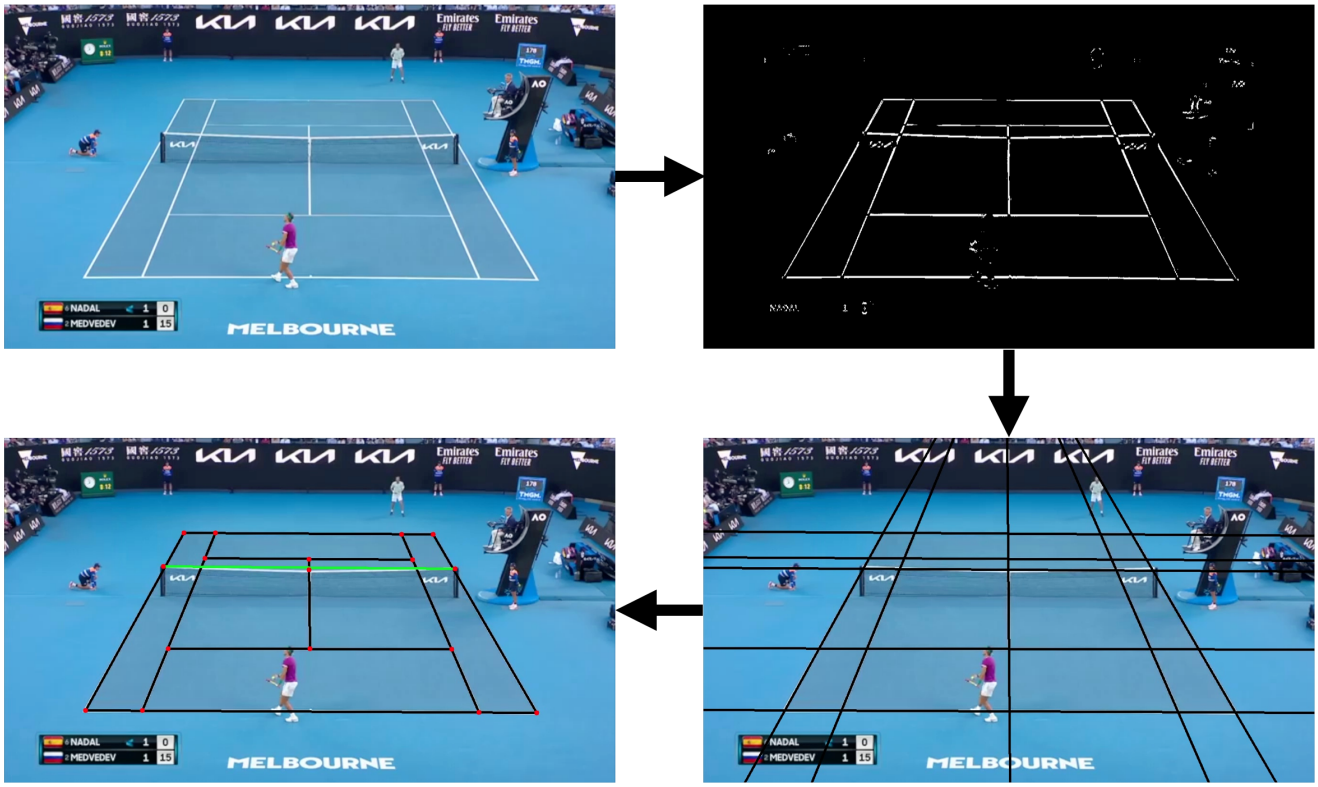

Figure 5: Top left shows input frame from broadcast video. Top right shows white masked image with the τ filtering algorithm to detect white pixels within court lines. Bottom right shows lines fitted to RANSAC inliers. Bottom left shows detected keypoints by intersecting court lines and finding the net line.

their doubles lines. This results in the top of the left pole and the top of the right pole having world coordinates of $[5.485, 0.91, 1.07]$ and $[5.485, 10.51, 1.07]$. Since these we have all 6 of the corresponding 2D points (as shown in figure 1) for these 3D world coordinates and they are not within the same plane, we can successfully solve the system of linear equations.

3.2. Extracting 2D Tennis Ball Position

As an added bonus to this paper, the 2D tennis ball position will also be tracked in the 2D image. This isn't such an easy task as the tennis ball often moves quite fast, it is tiny, occlusions occur by the net, lines, and players, and the publically available tennis match videos are typically around 30 fps. This means that the ball can appear blurred and enlongated in the direction it is moving. These are all challenges that TrackNet [3, 4] has overcome. TrackNet utilizes a convolutional neural network (CNN) to predict the position of the ball throughout the video.

TrackNet takes in 3 consecutive frames of size `640 x 360` and outputs a heatmap of where it is predicting the ball to be. It does so by utilizing an encoder-decoder network which outputs the final heatmap image. Once the heatmap is generated, a hough transform is performed to detect circles. The circle that is detected is where the tennis ball is. In order to train this CNN, they created a dataset from tennis matches that were pulled off of YouTube and labeled where the ball was using a combination of manual labeling and some auto-labeling techniques to speed up the process.

The authors of TrackNet were kind enough to open source their work along with the weights of their trained CNN which can be seen in figure 6. TrackNet allows anyone to predict the position of the ball in the frames of the video with up to 99.7% accuracy. A visualization of the tennis ball position predictions can be seen in figure 2.

4. Experiments & Results

To run some experiments on the tennis court detector for automatic camera calibration, a few videos were pulled off of YouTube. Using the frames from these broadcast videos of the tennis matches, some experimentation was able to be done with the algorithms described throughout this paper. Some of these experiments can be seen in figures 5, 1, and 1.

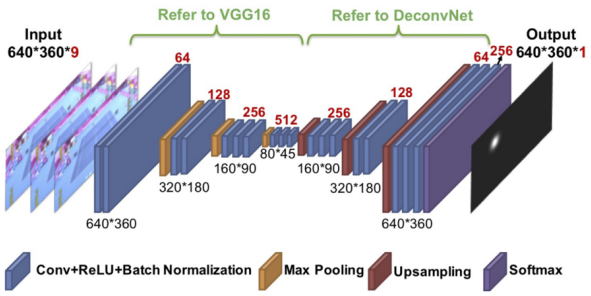

Figure 6: Convolutional neural network architecture for TrackNet [3, 4]

Qualitatively speaking, the method which was described in this paper and uses RANSAC to detect the court lines after the white pixels are extracted is quite robust and appears to track the keypoints very well throughout a rally from a broadcast video. It is even robust against occlusions from the players. RANSAC can still find the line with the most inliers even if there is a gap in the line. Occassionally, the full automatic camera calibration does not work due to the randomness of the RANSAC algorithm and some noise in the image. However, as long as a large camera zoom or pan does not occur, the previous camera calibration from a past frame can be reused. A slight downside to RANSAC is that running in 10 times per frame for each frame in a video can get very computationally expensive and it may be worth it to explore a downsampling scheme in the future.

Another downside to the algorithms outlined in this paper to detect the tennis court is that it does not work very well on grass or clay courts. When players play on grass courts, the grass near the court lines are often a light brown color due to the players killing the grass near those areas by sliding and stepping repeatedly on it. The light brown is very close to a white color and can generate a lot of noise which throws off the court detector. On clay courts, some of the clay gets kicked around and covers up some of the tennis court lines, making it hard to identify all the white court line pixels which RANSAC relies on to fit the lines.

Using the small dataset created for this paper, which labels the keypoints in on the tennis court, some metrics were able to be gathered on how accurate the keypoint detection was. By looking at the various mean squared errors in table 4, we can see that the one place that this method does struggle with is the net endpoints. These points are quite important as well since camera calibration relies on them since they are not in the same plane as the ground plane points. Throughout the video, the keypoints on the net line endpoints are not in one stable position but jump around in a small area around it. There are potentially several ways to improve this. One way to do so is to calibrate the camera

Keypoint	Mean square error
Bottom Left	0.833
Top Left	0.083
Top Right	0.167
Bottom Right	0.417
Net Line Left	2.667
Net Line Right	5.667

Table 1: Mean square error between labeled keypoints detected keypoints, performed on 1280 x 720 sized images

using only the ground plane points. Then using this calibration, we can find the position of where the net line pole meets the ground. This gives an extra constraint which can be used to get more accurate net line endpoints since we know the endpoints must along the pole holding up the net.

One of the other experiments that were performed included using a hough transform to detect lines from the cleaned binary image. The results received from this were not great as it did not appear to be as robust against noise and occlusions as RANSAC. However, maybe with more fine-tuning, a hough transform to detect lines could work well. Another attempt included uses a harris corner detector to find the keypoints directly. Even when the harris corner detector worked to find the keypoints in the ground plane, there was no clear route to finding the net line endpoints. Additionally, a player could easily occlude one of the corners, leaving the algorithm hopeless.

The results that I have seen so far show that TrackNet performs very well with the broadcast videos. It is able to track the ball throughout the frames accurrately. It does struggle when the camera pans slightly or zooms in or out. There is likely a way to help with this issue by cropping out the area of the court only and passing that to the TrackNet CNN. This would stop the tennis court from actually moving around in the image that are passed to it. This might be possible by using the tennis court detector that was created for calibrating the camera. Another potential solution could be to use an optical flow algorithm to detect when the camera has moved and adjust what is passed to the CNN accordingly.

5. Conclusion

Overall, robustly detecting a tennis court turned out to be a much more difficult task than expected. There is a lot of noise in the image around the tennis court which makes it hard to detect which lines correspond to tennis court lines. Additionally, the camera tends to pan around as the rally is in progress, greatly reducing the amount of assumptions you can safely make about the tennis court's position. The algorithm described in this paper works quite well at certain

grand slam tournaments such as the U.S. Open and the Australian Open where the game is played on a solid court with deterministic colors.

There is a lot of opportunity for future work with this project. In order to get more robust tennis court detection with all different surface types, it is likely that a convolutional neural network will perform best. The main challenge with this is trying to acquire training data or finding a way to do self-supervised learning.

Another possiblity for increasing the robustness of the tennis court detector is to try to track where the tennis court is located throughout the frames. It is fair to make an assumption that the tennis court will be in the center of the image at the start of the video. Then in consecutive frames, we can make better guesses on where the keypoints might be based on the previous positions of the tennis court.

It also seems that the image processing algorithms used before RANSAC was applied could be improved. If there are white pixels in the crowd or surrounding the court, it can very easily throw off court detector. A reasonable improvement that could be made is to look at luminance instead of RGB value for detecting where the tennis court pixels might be.

With a very robust tennis court detector, there are many algorithms which can be built on top of that. One of them could be to estimate the 3D path of the tennis shot. This can be done using the prior known physics of the tennis ball, similar to the work that was done in [6, 5]. With the 3D trajectory, you can start gathering statistics to build datasets on the type of hitting patterns that certain players use.

Performing tennis rally analysis with computer vision from single view broadcast videos is a very challenging task but allows you to learn a lot about different areas such as camera models, image processing, and data fitting. Due to this, it is a great project for someone looking to improve their computer vision skills.

6. Code

https://github.com/John-Boccio/Tennis3DTracker

References

[1] J. Han, D. Farin, and P. H. de With. *Generic 3-D Modeling for Content Analysis of Court-Net Sports Sequences*. Springer, 2007. 2, 3

[2] K. Hata and S. Savarese. Cs231a course notes 1: Camera models. 2

[3] Y. Huang. Tracknet: Tennis ball tracking from broadcast video by deep learning networks. Master's thesis, National Chiao Tung University, April 2018. 2, 3, 5, 6

[4] Y. Huang, I. Liao, C. Chen, T. Ik, and W. Peng. Tracknet: A deep learning network for tracking high-speed and tiny objects in sports applications. *CoRR*, abs/1907.03698, 2019. 2, 3, 5, 6

[5] E. Ribnick, S. Atev, and N. P. Papanikolopoulos. Estimating 3d positions and velocities of projectiles from monocular views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):938–944, 2009. 3, 7

[6] L. Shen, Q. Liu, L. Li, and H. Yue. 3d reconstruction of ball trajectory from a single camera in the ball game. *iSCSS*, 2015. 3, 7