Image Analysis and Computer Vision final project

Lorenzo Aicardi and Gloria Desideri

Tuesday 25th June, 2024

Professor: Vincenzo Caglioti

Contents

1 Introduction

The project has the scope of detecting the bounce points of a tennis ball given a single-view video of a tennis match taken by the top camera. Moreover, the aim of the project is to find the true 3D coordinates of the points detected starting from the image. The project has a wide range of uses, such as improving performance analysis and helping coaches and players build strategies. Additionally, technology can increase the accuracy of umpiring calls with improved line calling technologies and enhance broadcasts with captivating visualizations.

2 State of the art

The current state of the art in detecting bounces involves pure machine learning techniques only; in particular, the model TrackNetV2 is often used. Bounce detection and 3D trajectory estimation are often performed using stereo vision which allows for triangulation. Furthermore, some studies assume a parabolic trajectory of the ball thus reducing the uncertainty of the ball's path. We propose a hybrid method that uses Deep Learning techniques to detect the ball trajectory and players, combined with analytical techniques to estimate the coordinates of the bounces and improve player detection. Our methodology also involves numerical methods to find the 3D bounce points despite the limitation of the single view.

Analytical methods to solve this problem involve leveraging the information of the Y coordinate change. Despite this approach being effective in detecting the bounces on the racket it has poor performances on the bounces on the field since the change in the Y coordinate isn't sharp enough. Our approach models the trajectory using the RDP algorithm and applies filters to the detected changes to accurately find the bounce coordinates.

3 Feature extraction

The very first step of the project was to try to extract as many features as possible from the moving objects involved. This includes tracking the ball, the court and the players. We first approached this problem with purely analytical methods based on compensating the micro motion of the camera,

then applying frame differencing to find the components of the players. After finding the parts of the images with a motion change higher than a threshold we found clustered the components and identified the denser clusters as the players. However, this approach was prone to large uncertainty, especially in the detection of the top player so we moved to a deep learning approach.

3.1 Ball tracking

In order to detect the ball we used a pretrained TrackNetV2 model. TrackNetV2 is a sophisticated deep-learning architecture designed specifically for tracking high-speed and tiny objects, such as a tennis ball, in sports applications. In order to improve tracking accuracy, its architecture combines the best features of both Convolutional Neural Networks (CNNs) and Deconvolutional Neural Networks (DeconvNets). A pre-trained VGG16 model is used for feature extraction in the first stage. For spatial localization, a DeconvNet is used after feature extraction. In order to determine the precise placement of the object within the frame, this portion of the network refines the extracted features. The network produces a heatmap as its output, which shows how likely it is that the object will be present at each pixel in the frame. The output of the DeconvNet is subjected to a softmax activation function, which yields a probability distribution over 256 grayscale values per pixel and creates the heatmap. This grayscale prediction is then transformed into a binary heatmap by applying a threshold, where the channel with the highest probability shows where the object is most likely to be. Lastly, to validate the frames and make sure the object appears only once, the Hugh Gradient Method is applied. If this is true, the object's location is represented by a central point that is found and saved for use in later frames, enabling continuous tracking of the object's movement.

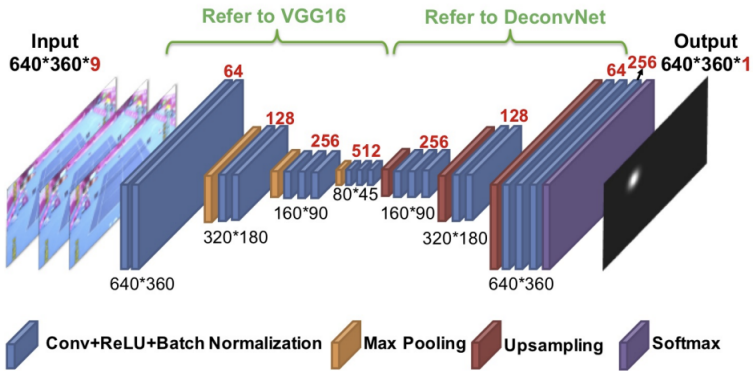

Figure 1: Tracknet

3.2 Court detection

In our experiment, a binary threshold is used to highlight possible court lines depending on their intensity once the input video frame has been converted to a grayscale image. Pixel filtering, which greatly improves line identification accuracy, is the next important stage. Each pixel is examined during this filtering procedure to see if it is a portion of a court line. The program specifically compares the intensity of each white pixel to that of its nearby pixels at a predetermined distance. The pixel is kept if the intensity difference is large enough in both the horizontal and vertical axes; if not, it is discarded. With this stage, only pixels that are probably going to be a part of the court lines are taken into account.
The Hough Transform is used to find lines in the binary image after the filtering step. The Hough Transform works by converting points in the image space to curves or lines in a parameter space, where each point casts a vote for a possible line that might cross it. Our method uses the Probabilistic Hough Line Transform, which picks points and, if they align with enough other points, extends them into line segments to detect lines. Even with noise and gaps present, this technique is very good at finding straight lines.

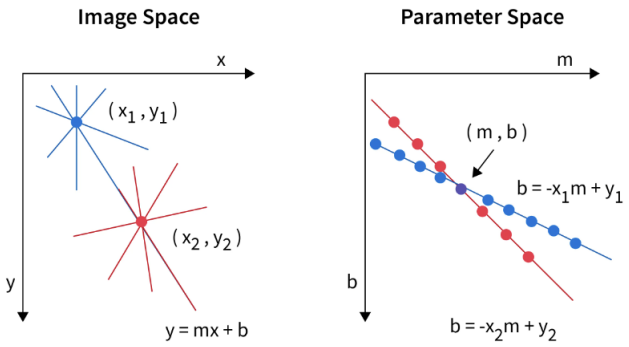

Figure 2: Example of the Hough Transform general Idea

Based on their orientation, the detected lines are further divided into vertical and horizontal lines. Vertical lines display the opposite pattern from horizontal lines, which have a larger disparity between the x- and y-coordinates of their endpoints. Combining lines that are near to one another and probably correspond to the same physical line on the court allows for even more detail. This merging procedure ensures that each court line is represented by a single, continuous line by grouping line segments based on proximity and similarity in orientation.

3.3 Player detection

In order to detect the players ResNet is used. ResNet-50 is a deep convolutional neural network with 50 layers, designed for image recognition and feature extraction. It includes residual blocks that allow the network to learn residual functions with reference to the layer inputs, facilitating the training of much deeper networks. These blocks use shortcut connections to jump over some layers, mitigating the vanishing gradient problem and enabling the training of very deep networks. The architecture consists of an initial convolutional layer, followed by four stages of residual blocks, each stage having multiple convolutional layers. Due to the fast movement of the players filtering of the bounding boxes detected is necessary.

To detect the bottom player we leverage the detected court: if the court has been detected in the previous step we use the bottom part of the court as Region Of Interest for the detection. The box with largest area detected

by ResNet is assigned to the bottom player. After the first detection the previously found boxes are used to determine a new restricted ROI. A margin is added to the previously found box to account for movement. The center of the previous box is calculated and the box, among the new detections, closest to the previous one is chosen as the new player box.

In the top part of the frame there are often multiple people which makes the task of detecting the second player difficult. Player 2 is identified by detecting all persons in the top half of the court, tracking their movements across frames, calculating their movement distances, and selecting the person with the most consistent and significant movement as player 2. This process helps in differentiating player 2 from other persons based on their activity patterns within the video frames. The top part of the court is used as ROI, all the bounding boxes are tracked across time using the SORT (Simple Online and Realtime Tracking) algorithm. The distance acress frame is calculated for each object and the player two is detected as the object with largest movement.

4 Bounce detection

The approach we used in detecting bounces consisted in modeling the trajectory of the ball on the image plane, and detecting sudden changes in it. At first we tried to simplify the trajectory using cubic splines, despite the smoothing producing good results in some sections of the trajectory it was unable to mitigate the noise of the network when the ball has an unstable direction. Even trying to increase the number of bases wasn't producing good results so we moved to the RDP algorithm.

4.1 Tracking changes in trajectory

We modeled the trajectory of the ball by using the Ramer-Douglas-Pecker algorithm; this choice was made in order to try to alleviate as much as possible the noise of the neural network. The RDP algorithm recursively divides the trajectory line. It is given the first and last points, which are automatically made to be kept; then it finds the point which is farthest from the segment formed by the line connecting the first and last point. If the farthest point is at a lower distance than a predetermined tolerance parameter ε, then it

is discarded, otherwise, it is kept. The algorithm calls itself by taking the farthest point and the first point and then the farthest point and the last point.

In the resulting simplified trajectory we looked for sudden changes by examining the angles: a smaller angle meant that the change in direction was sudden, candidating the point as a possible bounce.

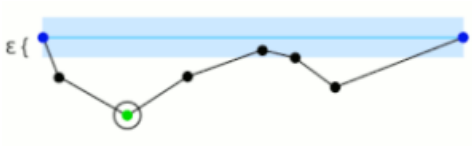

Figure 3: An example of an included point.

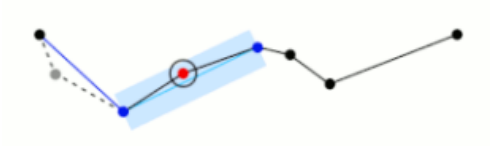

Figure 4: An example of an excluded point.

4.2 Filtering the changes in direction

The selected points were then filtered in order to differentiate them from peaks of the ball trajectory.

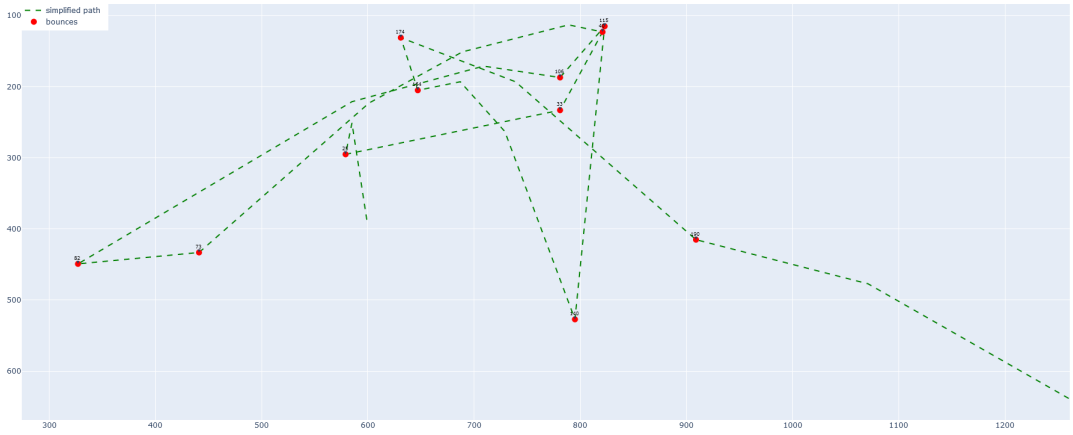

Figure 5: A sample graph showing the trajectory of the ball on the image plane, with detected bounce points.

The filtering was performed by examining the quadrants in which the trajectories were located. For each pair of lines intersecting in the bounce

point, the latter was used as the origin, and the two vectors spanning from the origin were considered. Then, we considered the following cases:

- If both the vectors appeared respectively in the first and second quadrant, then the bounce could either be the peak of the parabola trajectory or the player further from the camera hitting the ball. We differentiated the two cases by looking at the length of the vectors of the trajectory: if the vectors were long enough then it was reasonable to consider that the ball had been struck, so the point was marked as a bounce. If even just one of the vectors was short, then the point was excluded;

- If both the vectors appeared in the third and fourth quadrant, then the bounce was automatically confirmed, as it meant that either the ball hit the ground or that the player that was closer to the camera hit it;

- If the first vector appeared in a quadrant on the opposite side of the second (so for example, the first in the first quadrant and the second in the third quadrant) we decided to confirm the bounce if the angle formed was less than 180° (i.e., it was "convex"). This was confirmed by calculating the angle between the two vectors and the X axis, and according to the location of the two, if one angle resulted greater/smaller than the other, we would keep the bounce or not.

Furthermore, if the angle between two vertical trajectories was too small then the considered point corresponded to the ball being served. This case was considered not to be a bounce.

This approach was chosen in light of the limitations we had: first of all, only one view was available, making 3D reconstruction impossible. The ball moving as fast as it does prevented us from modeling its motion as a parabola, making considerations on its trajectory hard to verify. Finally, the neural network employed to detect the position of the ball in each frame suffered of severe noise, resulting in difficulty in detecting correct bounces.

4.3 Finding the camera center

The first step to retrieve the 3D coordinates of the bounce points is to find the camera center. We use the known positions of the corners of the tennis

field and the positions of the poles of the net to achieve this task. First, we find a homography that maps the four corners to their real coordinates. A perspective transformation is used to map points from one plane to another in a way that preserves the perspective.
The next step is to use the same homography to project the top of the poles (E, F) in the image to the ground using the same tomography. We call these two points E' and F'. The next image shows the image points and transformed points, E and F of the image, their ground projection and their true coordinates on the ground.

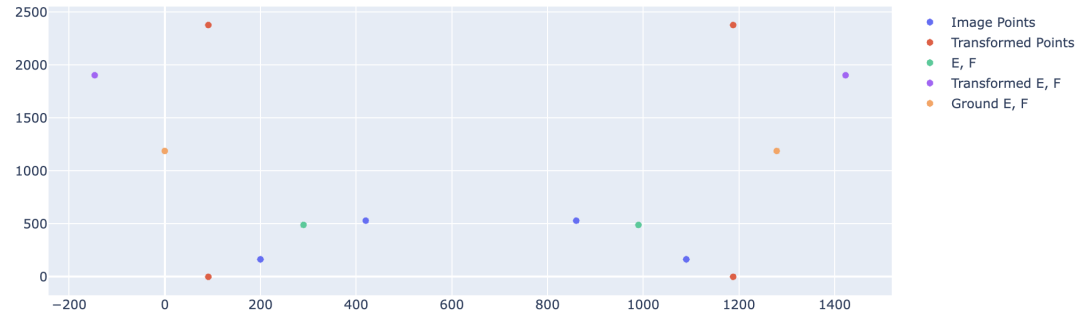

Figure 6: Corners on the image, corners rectified, E and F on image, E and F rectified, E and F ground projection

The last step is to draw two lines: one through E' and E, the other through F' and F. Then we intercept the lines to find the camera center.

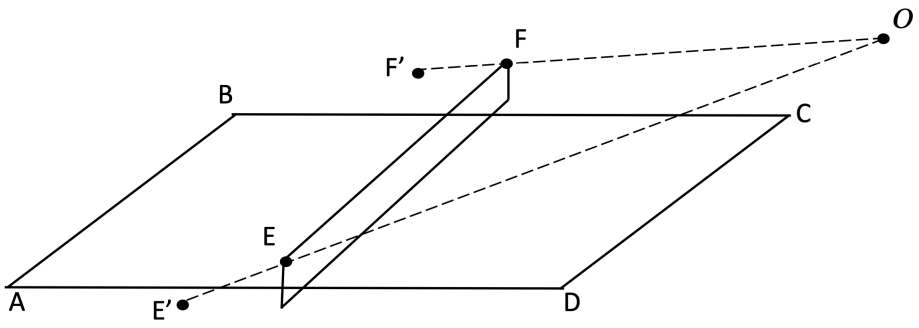

Figure 7: Finding camera center using top poles

To find the intersection of the lines we use their parametric representation. The following image shows the result

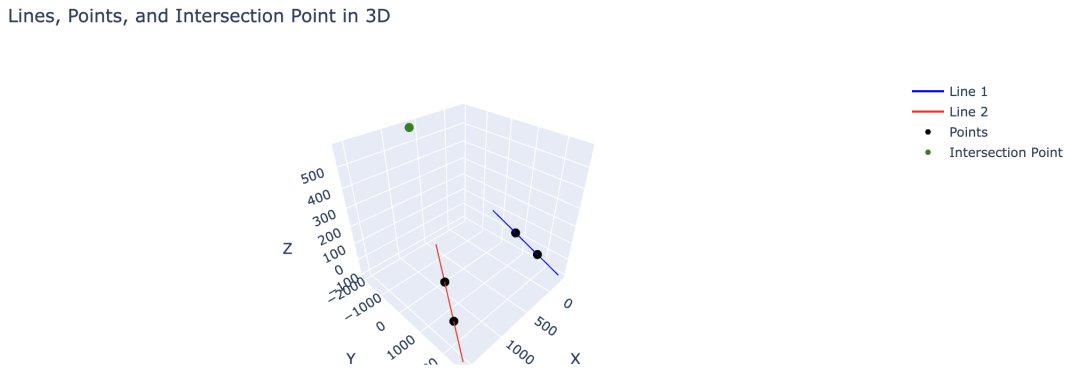

Figure 8: Intersection result

4.4 3D coordinates of bounces

To find the coordinates in 3D of bounces we use a similar approach. In this case, we know the coordinates of the camera center and the coordinates of the projection of the image of the bounce on the ground using the previous tomography. We assume the ball bounces on the player's plane when it bounces on the racket. We know the coordinates of the player's feet on the ground: they can be retrieved by applying the homography to the bottom left and right corner of the player bounding box. We use these coordinates as the base of a plane. We construct a line from the ground projection of each racket bounce to the camera center and we find the intersection with the player plane to know the 3D coordinate of each racket bounce.

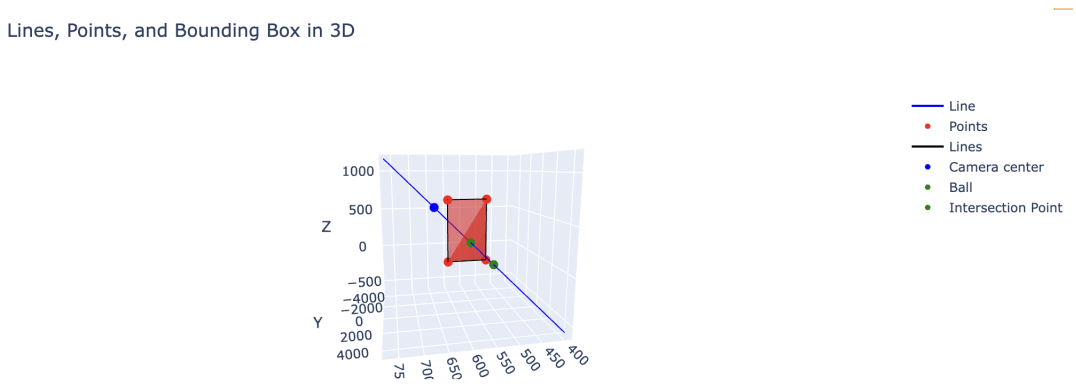

Figure 9: Intersection example

For the ground bounces it is only necessary to apply the homography since they belong to the ground plane.
We also implemented an algorithm that detects if a bounce is on the ground or on the racket. The algorithm detects whether a ball bounces on a player or a racket by analyzing the changes in the ball's vertical position (y-coordinate) over time. We use a variable to keep track of the expected direction of the ball, if the ball has changed direction over two consecutive bounces then a bounce on the player occurred.

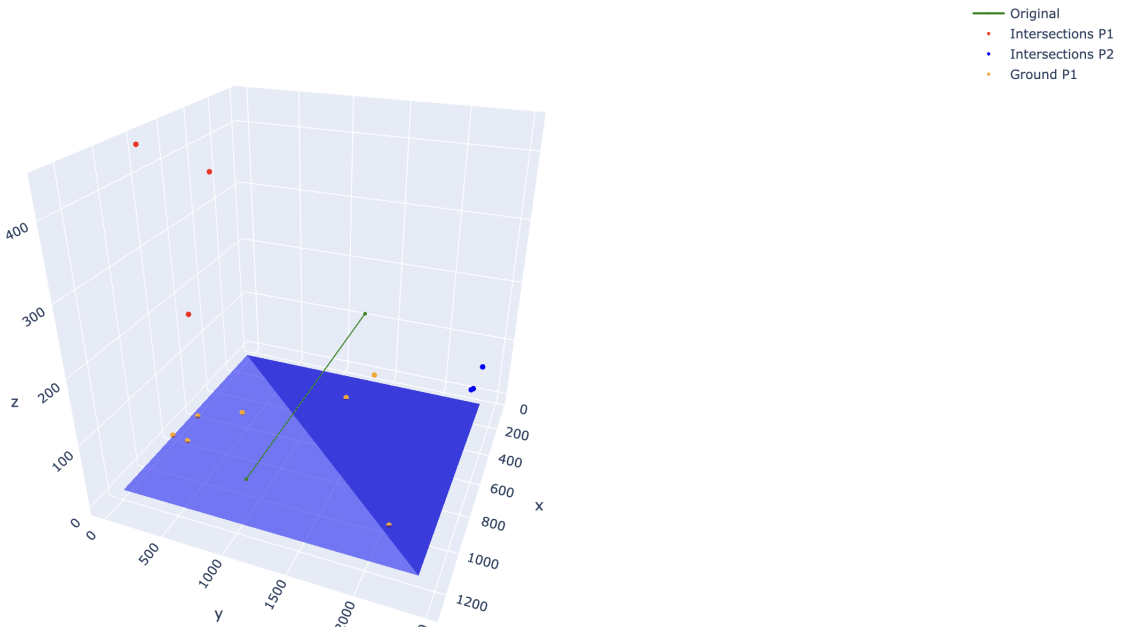

Figure 10: Projection results

5 The implementation

The code was developed in Python. Most of the image manipulations and calculations were carried out by the OpenCV library. We built a pipeline that performed the aforementioned operations as follows:

- At first, the video is given as input to the neural network. The output is a csv with every detected point on the screen in which the ball appears;
- Then, the trajectory is estimated using the Ramer-Douglas-Peucker algorithm;
- Each point in the estimated trajectory is then evaluated as a possible bounce point, as described before;
- The output of these two steps is a CSV containing the coordinates of the ball at each timestep and whether a bounce was detected.

- The pipeline estimates the court and player positions for each frame.
- the ball projections on the ground are estimated to build the minimap with the positions of the player and ball
- Finally the plot of the bounces in 3D is computed. If the camera center has not been found for the video the user is prompted to select the 4 corners and 2 poles

The computation of the trajectory and player position estimation are the most expensive steps. They both require the use of a neural network so they can benefit from the use of a GPU. Regarding player detection, however, the steps to find and filter the court lines for every frame and to estimate the top player positions remain computationally expensive.

6 Experimentation

The experimentation videos were taken from the WASB dataset. We used only the videos of tennis matches and only the ones taken from the top camera. In later stages WASB videos with poor quality resulted in bad results for player and court detection, thus we extracted some high resolution videos from long matches. To reduce the noise from the Neural Network a function to filter outliers was used so that if a detected coordinate was too far from previous points it was eliminated. We empirically found that a threshold of 50 was a good compromise for high resolution videos and a threshold of 100 was better for low resolution videos. In the earliest approaches, results were scarce, registering many points in which the ball wasn't bouncing as bounces. We analyzed the graph generated on the image plane to understand how the algorithm was evaluating bounces, and to uncover why multiple false positives were being detected. We noticed almost immediately that ALL sharp changes in direction were considered as bounces, meaning that also peaks in the ball trajectory were marked as such. We fine-tuned the algorithm basing ourselves on a large number of clips, allowing for an unbiased approach. Filtering out all the points in which the ball was at the peak of its trajectory was the final step in obtaining the results we hold now, in the way described earlier. In finding 3D coordinates of bounces we had to handle cases in which the top player was not found since it is the hardest object to detect. We also had to handle different resolutions of images: for high resolution images the

mapping was harder since the image size was bigger than the field size. In this case we found rescaling of the image useful.

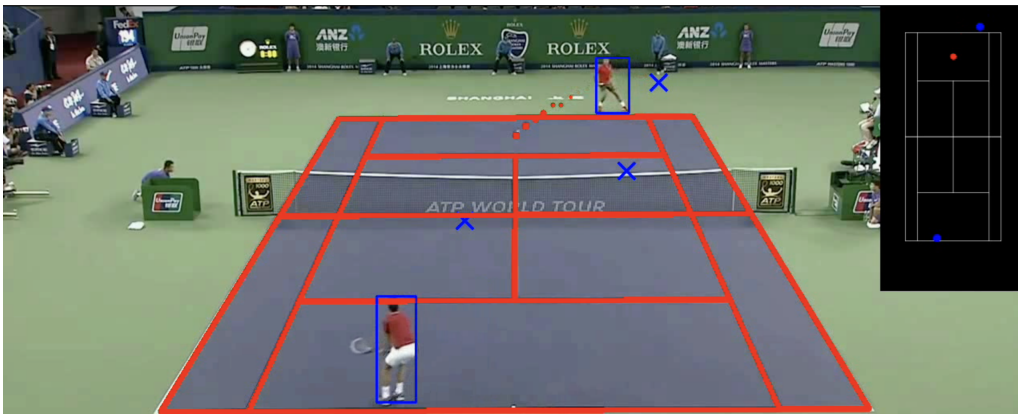

Figure 11: A frame from the output video showing bouncing points. The ball trajectory can be interpolated from the marked bounces.

Results vary according to the chosen tolerance parameter in the Ramer-Douglas-Pecker algorithm: the greater it is, the fewer bounces are seen, ignoring also noise coming from the neural network predictions while lowering it does the opposite. We empirically determined that a threshold of 15 was a good compromise. The procedure does not detect false positives on this tolerance parameter. Having less noise on the neural network would translate to better efficiency in the algorithm, allowing for lower tolerance values and detecting even more bounces.

7 Conclusion

We correctly managed to detect both the coordinates of the bouncing points in the image and in 3D. However, further improvements could be made to make the project better:

- Player detection is highly dependent on the court detection
- 3D accuracy depends on the points chosen for the homography. Detecting automatically the points of the court and the poles would make for better accuracy;

- Further reducing the noise of the neural network would make for better trajectory analysis.

References

[1] ArtLabsss. tennis-tracking. `https://github.com/ArtLabss/tennis-tracking`, 2021.

[2] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. *CoRR*, abs/1602.00763, 2016. URL `http://arxiv.org/abs/1602.00763`.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[4] Yu-Chuan Huang, I-No Liao, Ching-Hsuan Chen, Tsì-Uí Ik, and Wen-Chih Peng. Tracknet: A deep learning network for tracking high-speed and tiny objects in sports applications, 2019.

[5] Shuhei Tarashima, Muhammad Abdul Haq, Yushan Wang, and Norio Tagawa. Widely applicable strong baseline for sports ball detection and tracking, 2023.