POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Image Analysis and Computer Vision: Tennis Ball Tracking Project

Author:
Felipe Azank dos Santos
Felipe Bagni

Student IDs: 10919711, 10912321
Advisor: Prof. Caglioti Vincenzo
Co-advisor: Luca Magri
Academic Year: 2023-24

Contents

1 | Introduction

1.1. Description of the Project

Computer vision has revolutionized numerous fields by enabling machines to interpret and analyze visual data, and its impact on sports is particularly profound. In tennis, a sport characterized by rapid movements and precise timing, computer vision can provide critical insights into the dynamics of a match.

The project aims to track the ball and detect key moments when the ball changes direction, particularly during bounces on the court or when struck by a racket. This visual analysis will utilize video footage captured from cameras in classical positions.

By accurately tracking the ball's trajectory, the system will not only identify these critical events but also analyze the exact positions where bounces occur. This spatial information can provide valuable insights, such as the distribution of bounce locations, which can be used to assess player performance, strategize gameplay, or even enhance training techniques. The combination of precise ball tracking and bounce localization represents a comprehensive approach to understanding key aspects of the application of Image Analysis and Computer Vision.

1.2. Literature Review

This section provides an overview of the existing research and techniques related to ball tracking and event detection in sports video analysis. Key studies in computer vision have explored various methods for tracking moving objects, such as circle and shape detection and deep learning approaches like convolutional neural networks (CNNs). Previous work has demonstrated the effectiveness of these techniques in accurately identifying and following objects within video sequences, particularly in dynamic environments like sports. Furthermore, studies have highlighted the challenges associated with occlusion, motion blur, and varying lighting conditions, problems that can be seen further in the development of the project.

A brief literature review was developed in order to understand a technique that would suit the needs of the proposed project:

"Ball trajectory data are one of the most fundamental and useful information for game analysis. However, for some sports such as tennis, badminton, baseball, etc., the ball is not only small but also may fly as fast as several hundred kilometers per hour, resulting

in tiny and blurry images. That makes the ball tracking task becomes more challenging than other sports. In this paper, we design a heatmap-based deep learning network, called TrackNet, to precisely position ball of tennis and badminton on broadcast videos or videos recorded by consumer's devices such as smartphones. TrackNet overcomes the issues of blurry and remnant images and can even detect occluded ball by learning its trajectory patterns. The proposed network can be applied to other ball-based sports and help both amateurs and professional teams collect data with a moderate budget." - [2]

In addition to tracking the ball, various strategies were implemented to identify the specific frames in which key events, such as ball bounces and racket hits, occurred. These approaches relied on the detection of the ball's trajectory and velocity changes, for instance. Despite the success in detecting these moments, further refinement was needed to determine the precise location where the ball bounced on the court [3].

To address this, image rectification techniques were introduced. By correcting the perspective distortion inherent in video frames, image rectification enabled the accurate mapping of 2D image coordinates onto the actual 3D coordinates of the court. This process was essential for accurately determining the bounce points and fully solving the problem of tracking both the ball's movement and its key interactions during match. The literature behind this approach was based on what was seen during the Lessons and Practices of the discipline itself [1][4].

These studies provide a foundation for developing an advanced ball tracking system that not only detects key events but also accurately determines the position of bounces.

2 | Methodology

2.1. Tracking of the ball

As the first step, the ball tracking in our project was performed using TrackNet [2], a deep learning network specifically designed for tracking high-speed and tiny objects, such as balls in sports applications. TrackNet is a pre-trained neural network that excels at detecting fast-moving objects within video frames, making it ideal for our task of tracking the ball in sports footage.

TrackNet uses a series of convolutional layers to analyze each video frame, identifying the ball's position with high accuracy, even in challenging conditions like fast movements, varying lighting, and partial occlusions. Since the network is already trained on relevant sports data, it can accurately recognize and follow the ball without needing additional training on our specific videos.

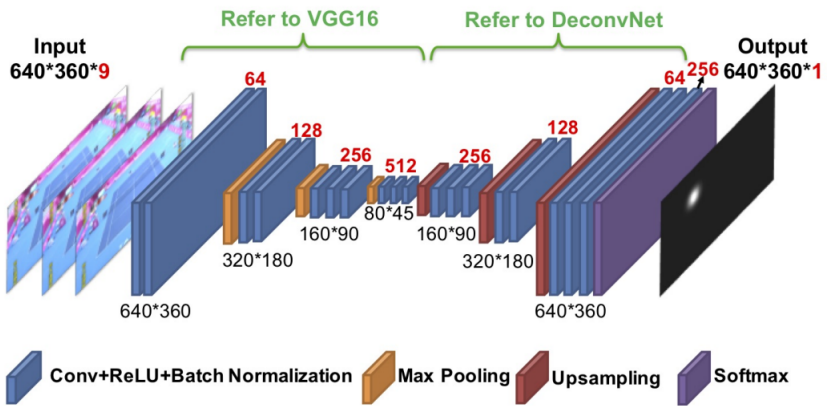

Figure 2.1: TrackNet architecture, as illustrated in the original paper [2]

In our system, TrackNet processes each frame of the video to detect the ball's location in real time, outputting the coordinates that describe the ball's position. These coordinates are then used later in the bounce detection process. By utilizing TrackNet, we leveraged a powerful, pre-trained neural network that simplifies the ball tracking process and provides reliable results for our analysis.

2.2. Trajectory Analysis and correction

After tracking the ball using TrackNet, we extracted the coordinates of the ball's position from each frame of the video. This provided a sequence of points representing the ball's trajectory during the game. To assess the performance of TrackNet, we analyzed how many frames the network correctly identified the ball and noted the frames where no detection was made.

For the frames where the ball was not detected, we used linear interpolation to estimate the missing positions, creating a smooth and continuous trajectory.

The interpolation between two known points (x_1, y_1) at frame t_1 and (x_2, y_2) at frame t_2 is calculated as:

$$x(t) = x_1 + \frac{(t - t_1) \times (x_2 - x_1)}{(t_2 - t_1)}, \quad y(t) = y_1 + \frac{(t - t_1) \times (y_2 - y_1)}{(t_2 - t_1)}$$

where t is the frame index where the ball's position is missing. This formula estimates the missing coordinates by linearly connecting the points before and after the gap.

By applying this interpolation, we corrected the ball's trajectory, ensuring a more complete and accurate representation of its movement, and increasing our confidence that the possible bounces identified were not detected due to lack of data.

2.3. Bounce moment identification

2.3.1. Derivative approach

The first approach to analyzing the trajectory of a tennis ball during a match involves starting with the discrete trajectory points, which represent the ball's position in space at specific time intervals. These points provide the ball's location in three-dimensional space, defined by coordinates (x_i, y_i), corresponding to its horizontal, depth, on each frame/time f_i.

To make the trajectory continuous, the discrete points are fitted with a smooth curve, as explained above, allowing for further analysis of the ball's motion.

A bounce can be detected by analyzing the vertical component $y(t)$, as a bounce occurs when the vertical velocity $v_y = \frac{dy}{dt}$ reaches zero and the second derivative $\frac{d^2y}{dt^2}$ is positive, indicating a local minimum in the vertical trajectory. These local minima reveal the moments when the ball hits the ground before bouncing upward.

Racket hits are identified by a sudden change in the magnitude and direction of the velocity vector $\mathbf{v}(t)$, reflecting the impact of the racket on the ball. These changes can be detected by calculating the magnitude of the velocity $|\mathbf{v}(t)|$ and observing abrupt shifts in its value. A significant jump in velocity, along with changes in direction, signifies the ball being struck by the racket.

This approach relies on detecting inflection points in the trajectory. Bounces correspond to vertical inflection points, while racket hits are indicated by sharp changes in velocity, offering a clear way to identify these key events in a tennis match.

2.3.2. Peak identification approach by neighbours values

After looking into the data, it was possible to identify that, perhaps, simpler solutions could be carried out in order to identify the occurrence of a sudden change of direction from the ball (indicating a bounce).

In order to do that, the X and Y coordinates were again analysed separately and then we used simple algorithms that identify "peaks" and "valleys" in the code. The algorithm (further seen in the implementation section), basically looks at the surroundings of each point and identify the plateau value that indicates the change of trend.

Later, in results, we will see that these "peaks" were filtered in order to avoid identifications of bounce from the same bounce occurrence

2.4. Image Rectification

Image rectification is a process used to correct perspective distortions in images, making objects appear as if viewed from a consistent and non-distorted viewpoint. By transforming the perspective of the image, rectification aligns objects to a common plane, which is particularly useful for accurate measurements and analysis.

In the analysis in question, the input video as well as traditional TV transmissions often present challenges due to the angled perspectives from which the footage is captured. These angles can distort the appearance of the tennis court and the trajectory of the ball, making it difficult to determine precise bounce points and other key details. The distorted view complicates the task of mapping the ball's position accurately on the court.

To address this issue, image rectification is applied to the footage to correct these perspective distortions. This involves transforming the images so that the court appears as though it is viewed directly from above, aligning the court lines and ball paths with their true positions. This transformation is achieved using a transformation matrix based on the camera's intrinsic and extrinsic parameters, which helps to remove the effects of perspective distortion.

While rectification itself does not involve tracking the ball in this study, it is crucial for accurately retrieving bounce points. Once the footage is rectified, the corrected images provide a reliable basis for identifying where the ball bounces on the court. This precise alignment alongside with the known standard court dimensions ensure that the bounce points can be accurately mapped to their real-world locations on the court, which is essential for detailed analysis and evaluation. Lastly, in order to perform the rectification, the court lines must be detected as a first step since the intersection between perpendicular lines and its parallelisms can be used for the rectification.

2.4.1. Court Identification

At a high level, the strategy of court detection involves first processing the input frame by converting it to grayscale and applying a threshold to isolate bright areas, which are likely to correspond to court lines. The next step is filtering out irrelevant pixels by taking advantage of the known structure of the court, based on the intensity of nearby pixels.

Once the relevant pixels are identified, the Hough transform is applied to detect lines in the image. These lines are classified into horizontal and vertical groups based on their orientation, and then the lines are further cleaned and merged to avoid duplicates or overlapping segments. The classification distinguishes lines as horizontal or vertical based on the slope and sorts them according to their position on the frame, filtering out those that do not align with the expected court layout.

After detecting and cleaning the lines, it is needed to find a transformation (homography) between a reference court layout and the detected lines in the frame. This is done by selecting pairs of horizontal and vertical lines, calculating their intersections, and comparing these points to a pre-defined court configuration. The goal is to find the best transformation matrix that maps the reference court layout onto the current frame. The transformation is evaluated using a scoring mechanism that measures how well the transformed court aligns with the detected lines in the frame.

Once a suitable homography is found, the court lines are overlaid on the original frame, allowing the court to be visualized. The detected court lines are also tracked across subsequent frames by applying the same transformation to new video frames. If the tracking of the court fails due to significant changes in camera position, the algorithm attempts to re-detect the court from scratch using the same process. This strategy handles the tracking by adjusting pixel search areas and refining the lines' positions based on pixel intensity, ensuring the court remains aligned even if the camera moves slightly.

The strategy employed to detect the court lines leverages a combination of image processing, geometric transformations, and pattern recognition based on known court structures to detect and track the court across video frames. The court detection is highly reliant on the structure of the lines and intersections, making the system robust as long as the court lines remain visible and the camera does not move drastically.

2.4.2. Rectification of the image

As seen in during lectures [1] and in the laboratory sessions, the rectification of the image was done by vector multiplications after we obtained the transformation matrix by using the parallel lines obtained during court identification. The process can be summed up by:

Given the original coodinates:

$$OG\_coords = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

The function `compute_rectified_coords` performs the following calculations to convert

the ball's coordinates to a rectified view:

- Convert the original coordinates (x, y) to homogeneous coordinates:

$$\textbf{OG\_coords} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Apply the transformation matrix M:

$$\textbf{rectified\_coord} = M \times \textbf{OG\_coords}$$

- Convert the resulting homogeneous coordinates to Cartesian coordinates:

$$\textbf{rectified\_coord} = \begin{bmatrix} x' \\ y' \\ w \end{bmatrix}$$

$$x_{\text{rectified}} = \frac{x'}{w}$$

$$y_{\text{rectified}} = \frac{y'}{w}$$

This helps us obtaining the coordinates of the bounce in the rectified image.

2.5. Ball absolute position identification

Finally, after tracking the ball, enhancing the path, identifying the bounce moments and rectifying the coordinates of the ball for these moments, the last step is to obtain the position of the ball with regards to the field position.

It's fairly simple to do that, all that is needed to do is to divide each coordinate of the tracked ball in the rectified region by the full length of the court in pixels. After that, we multiply this proportion by the actuall size in meter to obtain the absolute coordinate of the ball.

$$x_{abs} = \frac{x_{rec}}{1920\text{px}} \times 10.97, \quad y_{abs} = \frac{y_{rec}}{1080\text{px}} \times 23.77$$

where x_{rec} and y_{rec} are the coordinates of the ball during the bounce in rectified coordinates. Given that 1920 and 1080 are the sizes of the rectified court and 10.97m and 23.77m are the actual size of a tennis court.

3 | Implementation

3.1. Data Acquirement and Use case

In order to develop and test our system, we chose as main source of data the match between Jannik Sinner and Daniil Medvedev for the Australian Open 2024 Final. The full match video was obtained on YouTube and many "cuts" were made manually in order to select the "rallies" in which a considerable amount of bounces could be obtained.

3.2. Code Development

The implementation process was made using python for the tracking, bounce identification and plotting/creation of the results. Given the clear gain of using GPU for the execution of the predictions by the TrackNet, most of the process was held in jupyter notebooks on Google Collab (which supports GPU usage). The proceeding analysis after the tracking of the ball was made locally and the code was collectivelly developted by a GitHub repository `https://github.com/febagni/tennis-ball-tracker-computer-vision/`

3.3. Tracking of the ball

3.3.1. Network implementation

For the tracking of the ball, we used an existing implementation of TrackNet available on GitHub. This implementation served as a solid starting point, providing the core functionality needed to detect and track the ball.

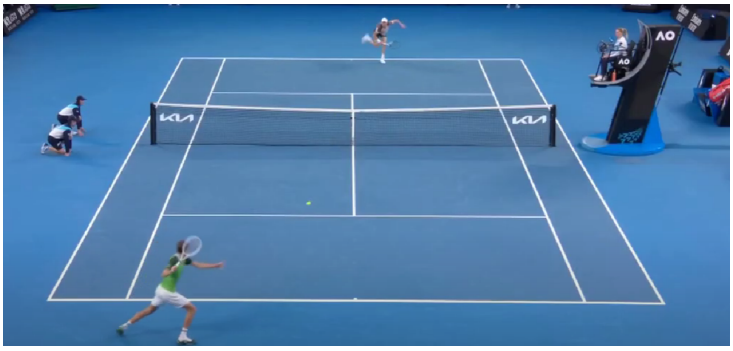

Figure 3.1: Cut frame obtained from YouTube accessible here

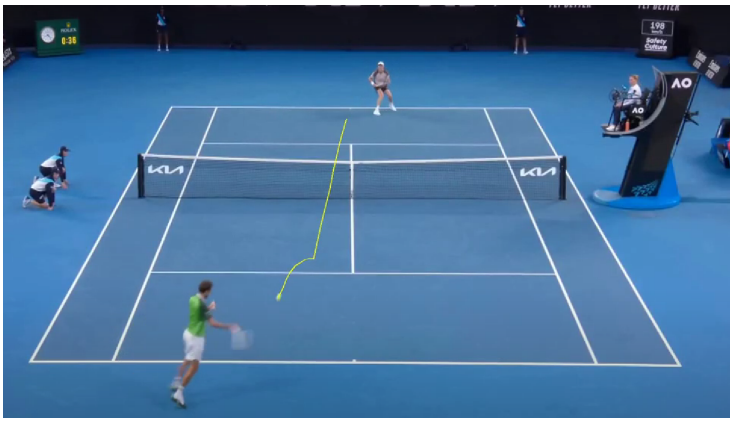

Figure 3.2: Frame of the video with drawing of the identified trajectory

We made several modifications to the original code to better suit our project requirements. Specifically, we adjusted the code to extract the full coordinates of the ball from each detected frame, enabling precise trajectory analysis. We also enhanced the implementation to generate output videos with the tracking results overlaid, allowing us to visualize the ball's path throughout the gameplay.

In addition to tracking the ball, the code was further adapted to detect the tennis court lines. This feature is essential for the subsequent image rectification process, as the detected court lines provide reference points to correct the perspective of the video frames.

Even though the decision of making the court detection at the same time as the tracking of the ball is computationally better. This was later reconsidered due to the reasons founded in the next section.

3.3.2. Choosing best input form

Given that, during the tracking data acquirement, the court detection was also being held. Its was possible to create three different tracking procedures:

1. Tracking the ball with the original video (as seen in Figure 3.1)
2. Tracking the ball with the court highlighted (Figure 3.3)
3. Tracking the ball with the rectified view (Figure 3.4)

The choice in order to proceed was based on how well the TrackNet was able to identify the ball. Considering that the ball was present in all the frames of the cuts we extracted, the comparison was simply the amount of frames in which the ball was correctly identified.

As seen in Figure 3.5, the choice was clear, in order to guarantee the best possible bounce analysis, we opted to use the coordinates obtained by the TrackNet applied on the original video, given that it identified the ball correctly in most of the frames.

After that, the coordinates identified were stored and used for subsequent analysis

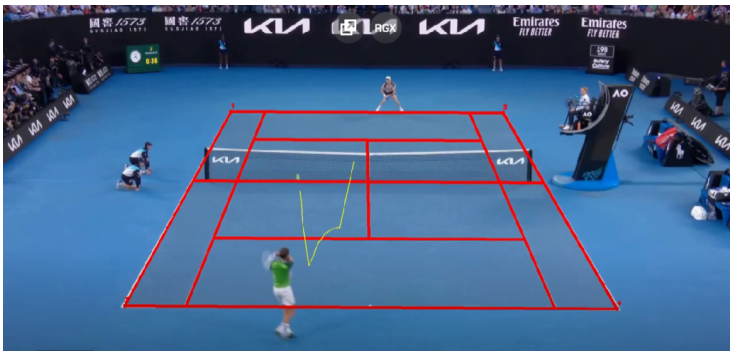

Figure 3.3: Frame example with court highlighted

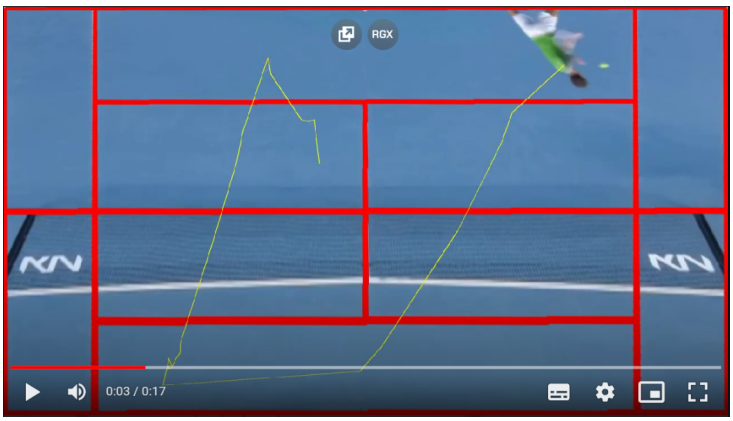

Figure 3.4: Frame example with court highlighted and rectification

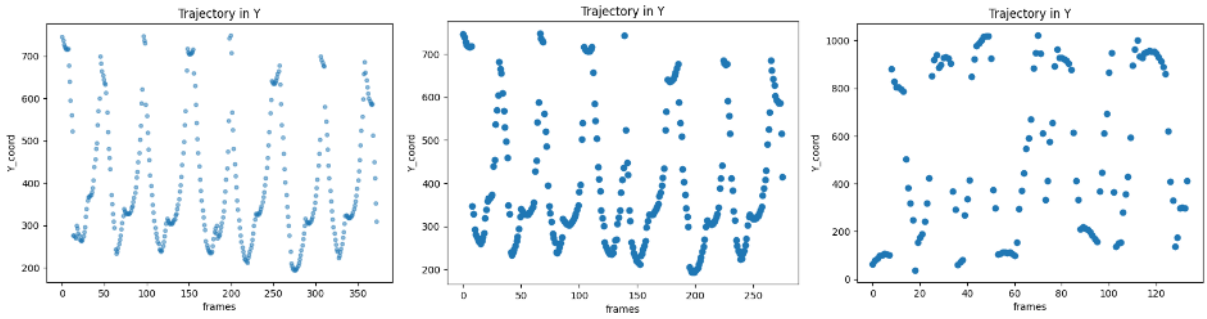

Figure 3.5: From left to right: approach 1 (83% of the frames identified), approach 2 (61%) and approach 3 (29%). The chosen approach was 1 (original video)

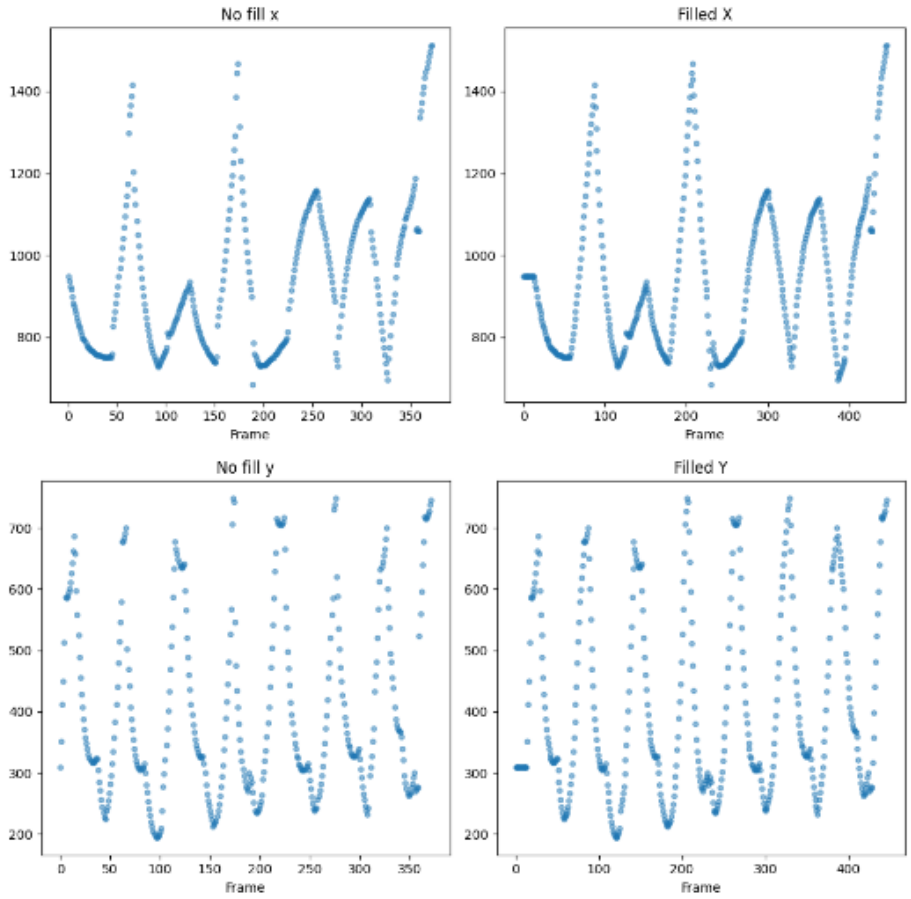

Figure 3.6: Comparison of coordinates after missing values imputation, smoothing the coordinates

3.4. Trajectory Pre-processing and Analysis

To analyze the ball's trajectory, we extracted the X and Y coordinates of the ball's position from each detected frame. However, due to occasional missed detections by the neural network, some frames had missing coordinate values.

Even though missing values is not a good thing, it's possible to see that the network fails to identify the ball mostly when the ball on high speed moving from one side of the court, in which there's no occurrence of bounces.

To address this, we applied linear interpolation to fill in these missing data points, ensuring a smooth and continuous trajectory representation, as explained in the previous chapter. The results can be seen in Figure 3.6 and seem to make good sense, without adding any erroneous breaks.

3.5. Bounce moment identification

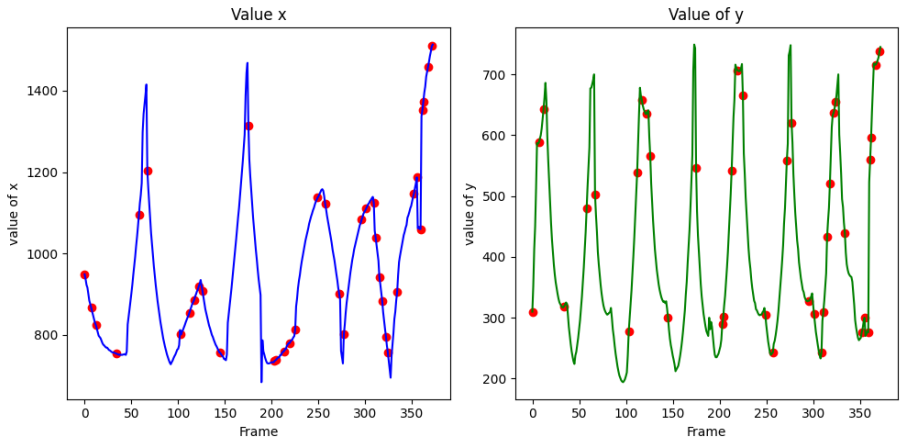

Figure 3.7: Graphs of the derivatives with the red points representing inflection points with the same frame for both x and y.

3.5.1. Derivative approach

The first analysis strategy revolves around detecting bounces of a tennis ball by analyzing its trajectory through the use of derivatives. When a tennis ball moves, its motion follows a predictable parabolic arc, determined by gravity. However, when the ball bounces, there is an abrupt change in this trajectory, which can be detected mathematically. This change is identified by examining the derivatives of the ball's position over time.

The first derivative of the ball's position with respect to time represents its velocity. By calculating the velocity in both the horizontal (x) and vertical (y) directions, we can observe how the ball moves frame by frame. A bounce is marked by a sharp change in the vertical velocity. Prior to the bounce, the ball's vertical velocity is negative, indicating it is falling. After the bounce, the velocity becomes positive, showing that the ball is now rising. This sudden shift in velocity is a clear indicator of a bounce.

The second derivative of the ball's position, which gives the acceleration, is also useful in bounce detection. Acceleration represents the rate of change of velocity. During a bounce, the velocity undergoes a rapid transition, leading to a spike in acceleration. By monitoring these spikes, we can pinpoint when the ball hits the ground.

In the notebook, the ball's trajectory is analyzed by first filtering the coordinates to remove any invalid data points. Once valid coordinates are obtained, the change in position between consecutive frames is calculated. This change corresponds to the ball's velocity. The first derivative gives insight into how the ball's velocity changes, while the second derivative (acceleration) highlights abrupt shifts in movement, particularly during a bounce.

The vertical (y) coordinate is especially important for detecting bounces because the bounce significantly alters the ball's vertical motion. The notebook computes the change in the y-coordinate between frames to determine when the ball transitions from falling to rising. This transition marks the point of a bounce.

So, the ideal here is to detect inflection points in the derivatives of both x and y, and when they coincide, the corresponding frame is of a ball bounce.

To deal with noise in the trajectory data, smoothing techniques like interpolation are applied. This makes it easier to identify bounce points by reducing minor fluctuations in the ball's motion. Once the data is smoothed, a threshold is applied to detect significant changes in the ball's velocity and acceleration. Frames where the y-velocity or y-acceleration crosses this threshold are likely to correspond to a bounce.

Visualization plays a crucial role in understanding the ball's motion. By plotting the ball's vertical position over time and overlaying its velocity and acceleration, it becomes possible to visually identify the frames where significant changes occur, and if they correspond to bounces.

The rationale for using derivatives in this analysis is that they provide a way to capture the ball's dynamics during movement. While the position data tells us where the ball is, the velocity and acceleration provide a deeper understanding of how it is moving and when it changes direction. This makes derivatives essential for detecting events like bounces, where sudden shifts in motion occur. Thresholding ensures that only significant changes are detected, filtering out minor noise or fluctuations in the ball's trajectory.

This method of detecting bounces through the calculation of the inflection points on both derivatives of x and y did not work as expected due to the irregular sampling of ball detection. The ball is not consistently tracked in each frame, resulting in gaps and inconsistencies in the data. As a consequence, when we compute the derivatives, the lack of smoothness introduces many rough edges, as we can observe in the provided image. These sharp fluctuations in the derivative values do not correspond to actual bounces but rather reflect the discontinuity in the data, making the bounce detection unreliable. This is shown on the Figure 3.7.

3.5.2. Peak identification approach by neighbours values

Given that the first approach did not generated satisfactory results. We opted for a simpler approach, using the `find_peaks()` function from the SciPy library[5] to identify changes in the direction trend of the ball.

The `find_peaks()` function was applied to both X and Y coordinate data to detect local maxima and minima, indicating significant changes in the ball's direction. This automated approach efficiently pinpointed crucial points in the trajectory, allowing us to analyze the ball's movement more accurately (3.8).

As seen in the text, the second approach generated goods results in terms of change of trend detection. As we will see in the "Results" chapter, these sudden change points indeed indicate frames in which a bounce occurred.

3.6. Image Rectification

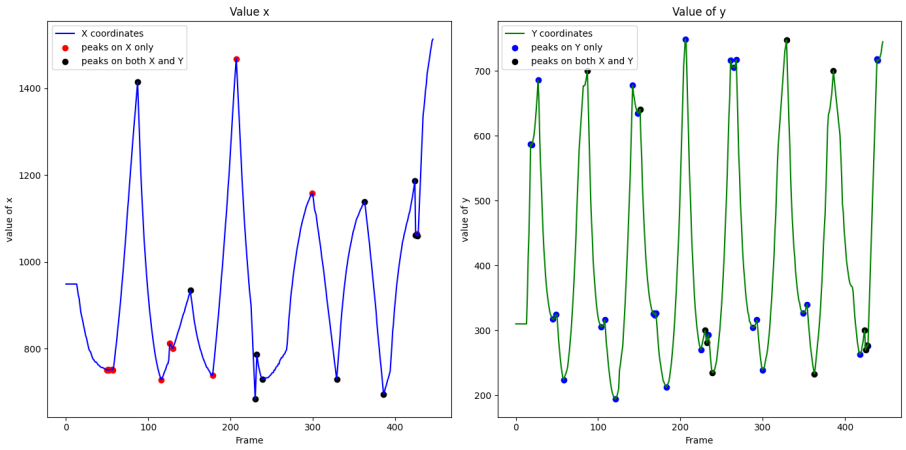

Figure 3.8: Peaks Identification

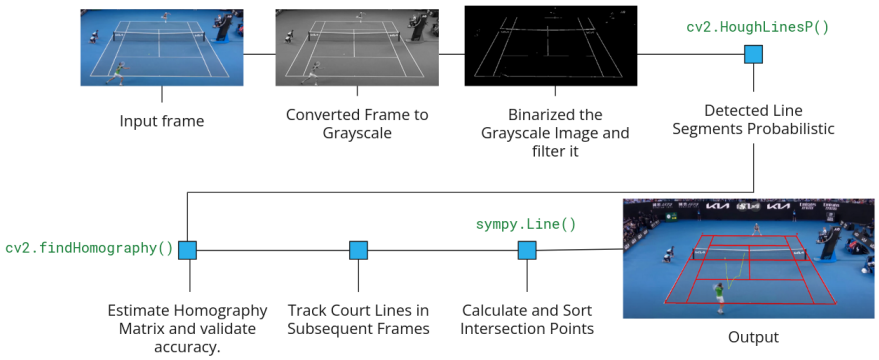

Figure 3.9: Court Identification Flux

3.6.1. Court Identification

The ‘CourtDetector‘ class is designed to identify and track the court in video frames using image processing techniques and computer vision methods. The process, which can be seen on Figure 3.9, begins with converting the input frame from BGR color space to grayscale using ‘cv2.cvtColor‘. This grayscale image is then binarized with ‘cv2.threshold‘, where bright pixels indicative of court lines are set to white, and other pixels are set to black. This binary image helps isolate potential court lines from the rest of the frame, preparing it for further analysis.

The next step involves refining the binary image to better align with expected court line structures. The ‘_filter_pixels‘ method achieves this by removing pixels that do not exhibit significant intensity changes relative to their neighbors. This intensity check ensures that only pixels with notable differences in brightness are retained, which helps in emphasizing the actual court lines while discarding irrelevant details.

With the binary image filtered, the ‘_detect_lines‘ method applies the Hough Line Transform using ‘cv2.HoughLinesP‘ to detect line segments in the image [3]. This method identifies lines by transforming points from the image space into a parameter space and

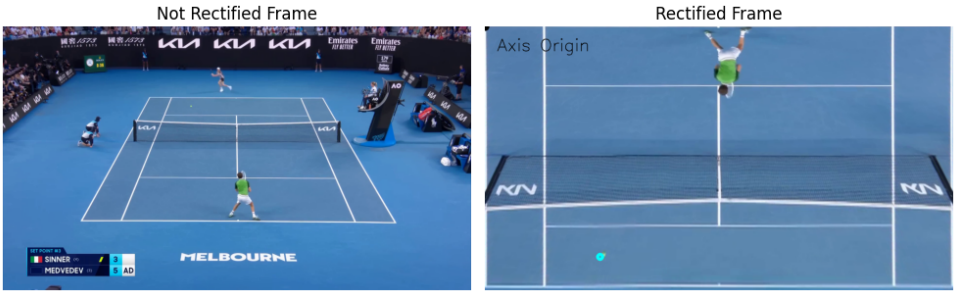

Figure 3.10: Input frame next to its rectified version. As it's possible to see, the ball's coordinate is correctly transformed to the rectified coordinate

detecting intersections in this transformed space. Parameters like 'minLineLength' and 'maxLineGap' are used to fine-tune the detection, ensuring that only meaningful lines are identified.

To map these detected lines onto a reference court model, the '_find_homography' method estimates a transformation matrix. This matrix is computed by examining pairs of detected horizontal and vertical lines, finding their intersections, and matching these with predefined court line positions. The homography matrix, computed using 'cv2.findHomography', is evaluated based on a confidence score to ensure accurate alignment of the detected lines with the reference court.

For tracking purposes, the 'track_court' method updates the position of court lines in subsequent frames by comparing new line positions with previously detected ones. If significant deviations are detected, indicating potential camera movement, the method recalibrates by re-detecting the court. The 'line_intersection' function calculates the intersection points of lines using 'sympy.Line', while the 'sort_intersection_points' function organizes these points for correct alignment. This approach integrates OpenCV for image processing and geometric transformations with SymPy for precise line calculations to achieve accurate court detection and tracking.

3.6.2. Rectification of the image

With the ball's coordinates and the identified moments when a bounce might occur, we then use the transformation matrix M derived from the parallel lines on the court and applied the matrix multiplications described in 2.4.2 with `numpy`.

$$M = \begin{bmatrix} 5.05 & 2.79 & -3847.30 \\ 0.02 & -3.95 & 3122.66 \\ -0.000005 & 0.0030 & 1.00 \end{bmatrix}$$

This matrix allows us to map the ball's coordinates to a vertical (rectified) view, accurately determining the position of each bounce in the transformed image (3.10).

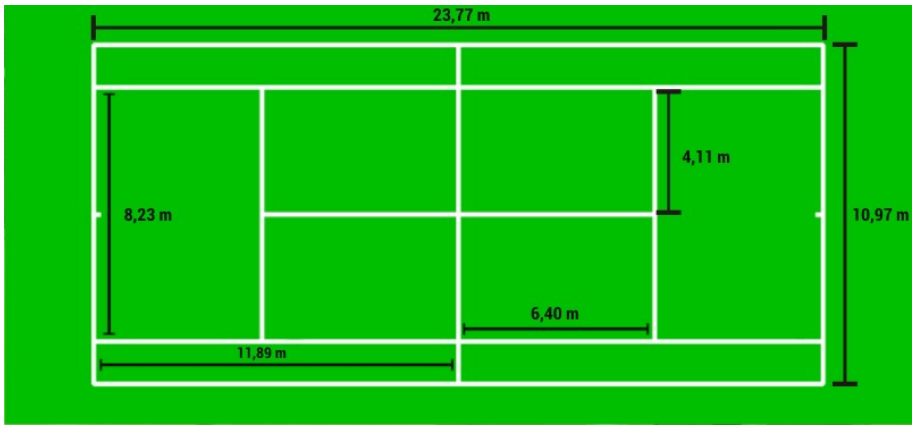

Figure 3.11: Model of tennis court that was used as parameter to the real life dimensions transformation

3.7. Ball absolute position identification

Just like explained in 2.5, the coordinates of the bounce were transformed for real size according to the schema 3.11

We will see in better detail the results and distribution of the bounces coordinates in the next chapter (alongside an evaluation fo the intermediate results).

4 | Final Results and Possible Improvements

The results were fairly discussed in the previous chapter, however, let's dive in the most important intermediate results and the final one as well as discussing the bottlenecks of the project and how to improve it.

4.1. Ball Tracking

As seen in the images, the coordinates of the ball obtained by the TrackNet showed a sufficient amount of accuracy, proving to be a good choice for the project. The best solution got a 83% ball identification with no misplaced coordinates.

The missing values, as said before, were indeed concentrated when the ball was at high speed and, therefore, not being to risky to the project, given that no bounce occurred in those phases. However, perhaps using the system with videos with a greater frame rate can help tackling this problem (even though the interpolation could fairly solve it).

4.2. Bounce Identification

For a better identification of results, a video is provided in the slides. However, it's possible to see that the system is clearly identifying the correct points.

During development, it was noted that some of the identified frames were pointing to the bounce occurrence, gathering a set of consecutive frames as "plateau". This was given once that we applied the `find_peaks()` separately fir X and Y and each one of them pointed the same bounce with slight difference of frames.

To correct that, we applied a "padding" mechanism, in which would take the consecutive (or almost consecutive) frames and replace it by only one frame. After some manual fine-tuning, the best result yielded the one that unconsidered identifications with 4 frames of distance

One considerable problem that proved to be a bottleneck relies on the fact that the bounce that happened on the court side of the player closer to the camera is not always identified by our system (As seen on figure 4.3).

This can be solved perhaps if we fine-tune the TrackNet for tennis matches as well as reducing the padding on the bounce identification.

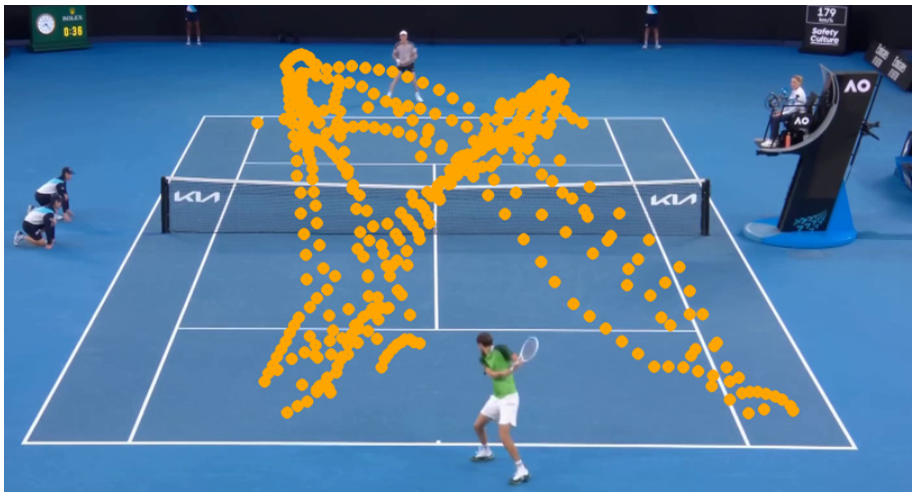

Figure 4.1: Frame with full trajectory

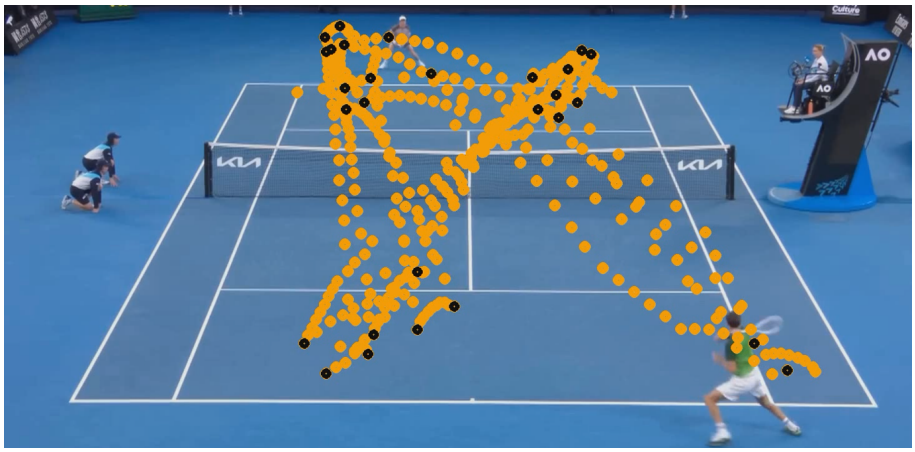

Figure 4.2: Full trajectory with the peaks in black

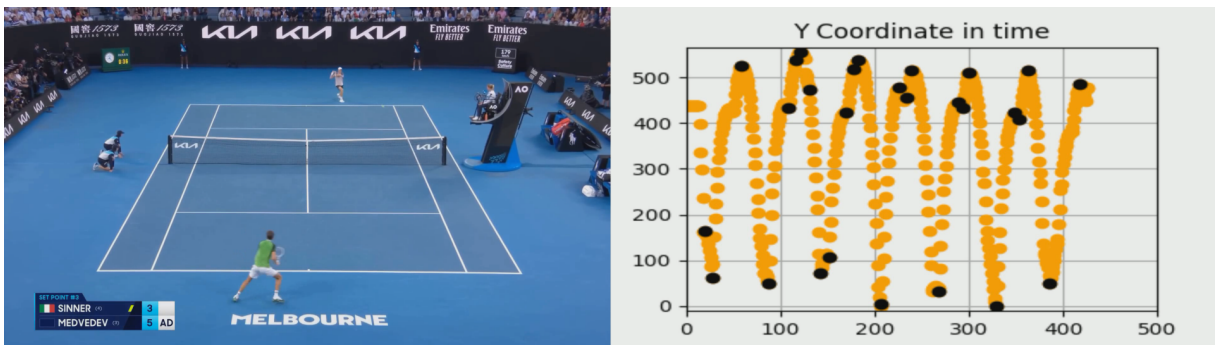

Figure 4.3: Result showing that the system fails to identify a bounce from Medvedev's side 4 times

Padding	Amount of Bounces Detected
no padding	54
1	37
4	28
5	21
real	28

Table 4.1: Detected bounces for different padding values

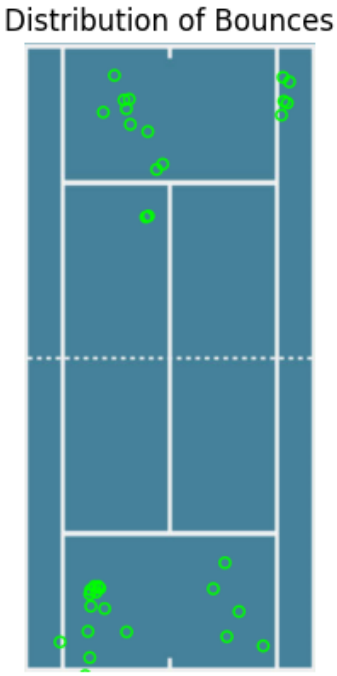

Figure 4.4: Bounce Distribution in absolute coordinates

4.3. Bounce position in real life

Finally, the expected result can be analysed by the distribution of the X and Y coordinates after the rectification and value correspondence is done:

Analysing the data, we can see that the identification makes a good amount of sense:

- All values are in general, inside the range of the of the court, with exceptions for some cases in which Sinner hits with his racket
- Coordinate X has a bounce position located in the sides of the court, which makes sense given that, during the cut, the bounces (and the entire "rally") happens in the extreme corners of the camera.
- Coordinate Y has no values in the middle of the court, which makes sense given no bounce occurs there. Most of the bounces do indeed happen close to the court limit (the common strategy in tennis).

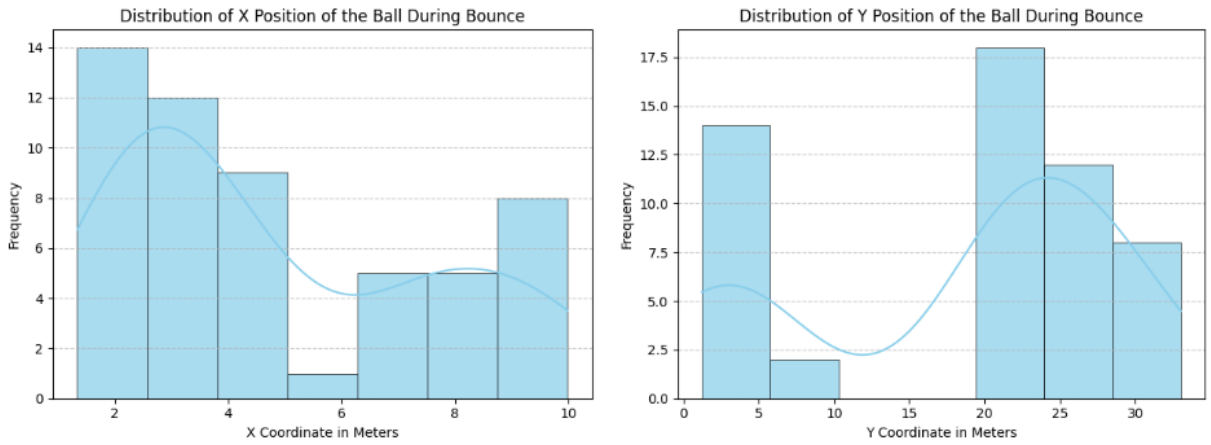

Figure 4.5: X and Y distribution in meters

4.4. Conclusion and Future

Finally, it's clear how new methods (such as CNNs) and classical geometry and projection can work together to develop cutting edge solutions for the Image Analysis and Computer Vision field, specially in the sports and industry fields.

it's also clear that the project has proved to be both challenging and rewarding, however there are still some improvements that can be done in the future, such as:

- Increase robustness of the system to make a good result in any tennis match recording
- Make the system less dependable of padding and frame rate
- Refactor the code used during the development in order to make it easier to other to use it

Bibliography

[1] V. Caglioti. 2d reconstruction (image rectification). Lecture Slides, 2023. Course: Image Analysis and Computer Vision.

[2] Y.-C. Huang, I.-N. Liao, C.-H. Chen, T.-U. İk, and W.-C. Peng. Tracknet: A deep learning network for tracking high-speed and tiny objects in sports applications, 2019. URL https://arxiv.org/abs/1907.03698.

[3] A. Liberman. Highlights generation for tennis matches using computer vision, natural language processing and audio analysis, 2022. URL https://openworks.wooster.edu/independentstudy/9837. Paper 9837.

[4] L. Magri. Image analysis and computer vision - luca magri's website, 2023. URL https://magrilu.github.io/iacv/. Accessed: September 4, 2024.

[5] SciPy Community. Scipy: Scientific computing tools for python, 2024. URL https://docs.scipy.org/doc/scipy/index.html. Accessed: 2024-08-29.