Tennis Ball Tracking

IACV

Felipe Azank and Felipe Bagni

Table of contents

01

Problem

02

Literature Review

03

Methodology

04

Implement
ation

05

Analysis & Results

06

Conclusions

01 & 02

Problem & Literature Review

Problem Description

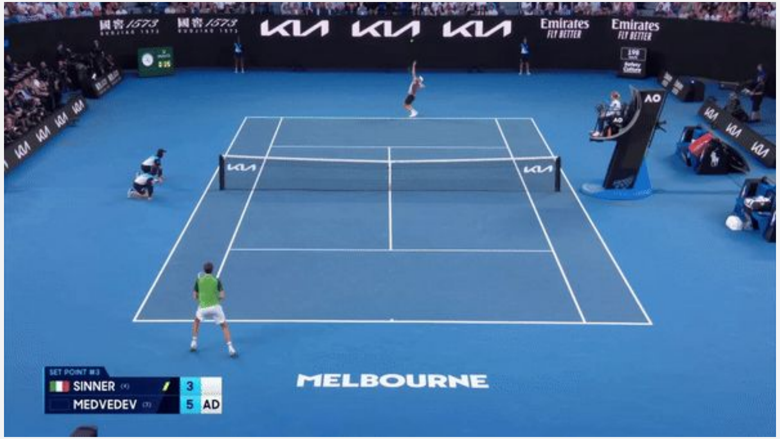
Objective

Given an input video of a tennis rally in a standard view, obtain:

- The time (frame) in which the ball bounces in the court or is hit by a racket;
- The position of the ball in those frames.

This allows, for instance:

- Valuable match insights;
- Assess player performance;
- Strategize gameplay;
- Enhance training techniques.



Literature Review

Ball tracking

Previous works have demonstrated the effectiveness of CNN techniques in accurately identifying and following objects within video sequences, particularly in dynamic environments like sports. [2]

Court detection

Court detection was done using the Hough Transform. In this project, the OpenCV function cv2.HoughLinesP() was applied to detect the court lines. These detected lines are categorized into horizontal and vertical groups, after which lines corresponding to the same one are merged. [3][5]

Image rectification

The literature behind this approach was based on what was seen during the Lessons and Practices of the discipline itself. [1][4]

03

Methodology

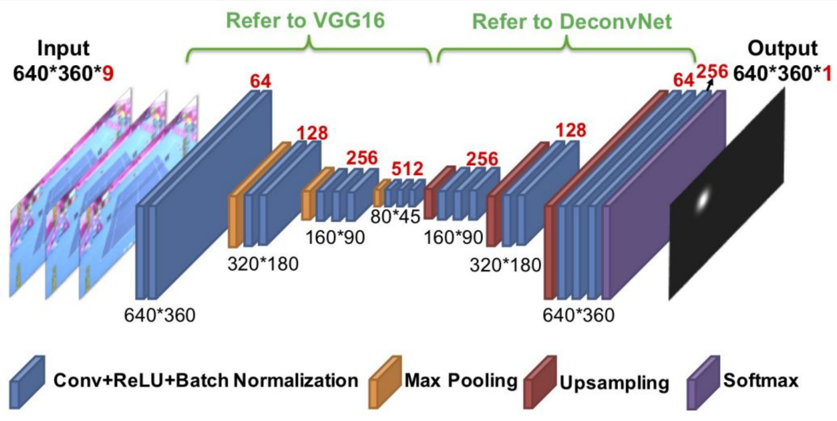
Tracking of the ball

TrackNet

In order to obtain the coordinates of the ball during the rally, we used TrackNet, a Convolutional Neural Network specialized on high speed ball detection in sports.

The output of this phase should be (for each frame):

- The coordinate X and Y of the tennis ball identified in that frame (w.r.t. the video, not the tennis court)



Bounce moment identification

Derivative Approach

A bounce is detected when the ball's vertical velocity (first derivative of position) reaches zero and its acceleration (second derivative) becomes positive, marking a local minimum as the ball hits the ground and bounces. Racket hits are identified by sudden changes in both the magnitude and direction of the velocity, signaling the ball has been struck.

Neighbourhood Values Approach

The second approach applied a simpler solution, basically looking at which values of the X and Y coordinates (separately) the data shows a local minima. After that, a filtering of the peaks was applied to avoid the identification of the same bounce more than once

The output of this phase should be:

- The frames in which a bounce or racket hit occurs

Court Identification

Input

Convert Frame to Grayscale

Binarize the Grayscale Image and filter it

Detect Line Segments

Estimate Homography Matrix and validate accuracy.

Track Court Lines in Subsequent Frames

Calculate and Sort Intersection Points

Output

Image Rectification

Step 1 – Transformation Matrix

The transformation from the original coordinates to the top view were obtained using a transformation matrix based on the camera's intrinsic and extrinsic parameters, which helps to remove the effects of perspective distortion.

This matrix is one output of the Court Identification

Step 2 – Coordinate Transform

With the matrix M we can then obtain the rectified version by:

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

And then, obtaining the final coordinates by:

$$\textbf{rectified\_coord} = \begin{bmatrix} x' \\ y' \\ w \end{bmatrix} \qquad x_{\text{rectified}} = \frac{x'}{w} \qquad y_{\text{rectified}} = \frac{y'}{w}$$

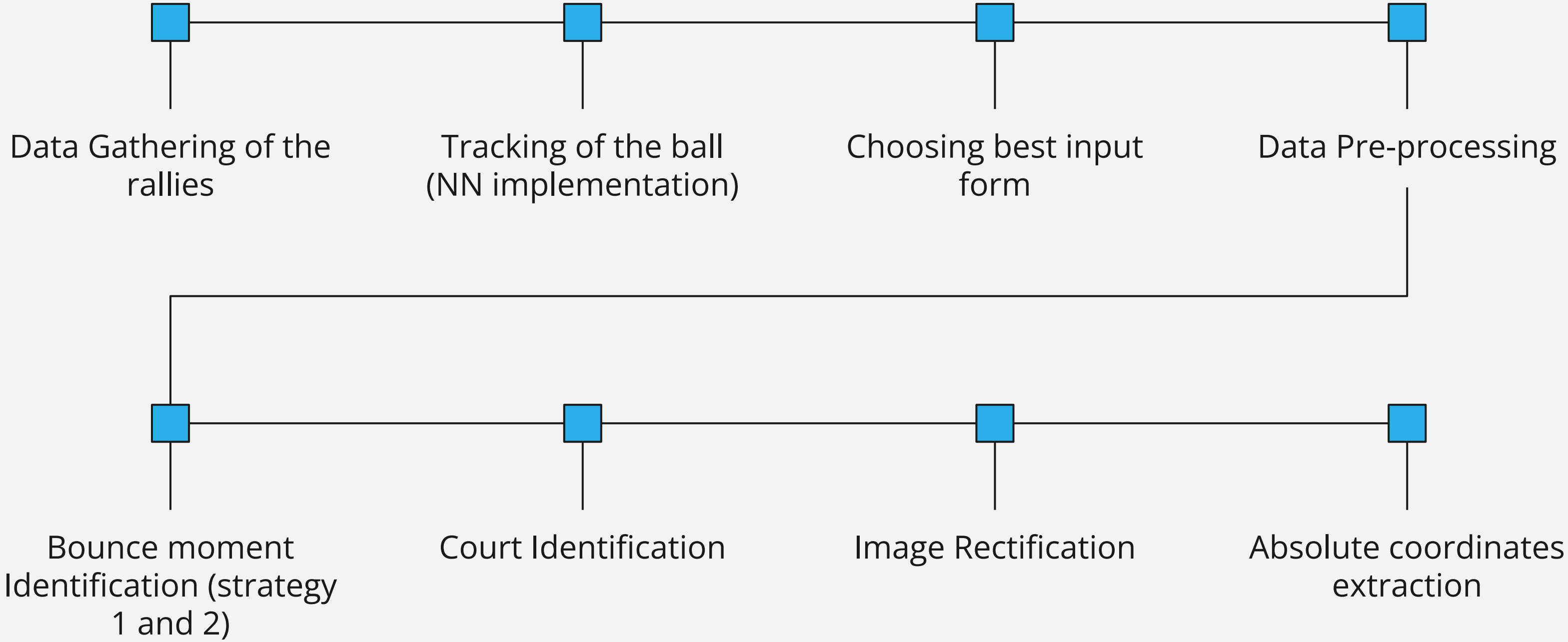
The output of this phase should be:

- The location of the ball in the rectified image on every frame in which a bounce was identified

04

Implementation

Implementation Summary



Data Gathering of the rallies

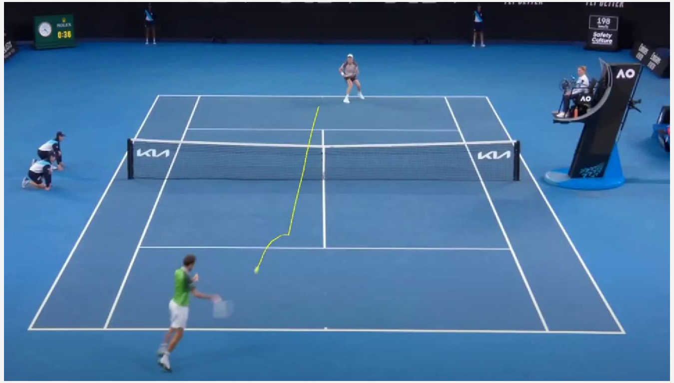
Main Data Souce

The main data source for the project was the match between **Jannik Sinner and Daniil Medvedev** for the Australian Open 2024 Final. The full match video was obtained in Youtube and many "cuts" were made manually in order to select the "rallies" in which a considerable amount of bounces could be obtained.

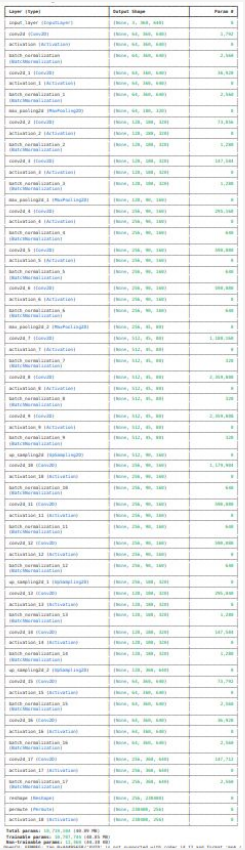
Ball Tracking and coordinates

TrackNet Implementation

For the tracking of the ball, we used an existing implementation of TrackNet available on GitHub. 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.



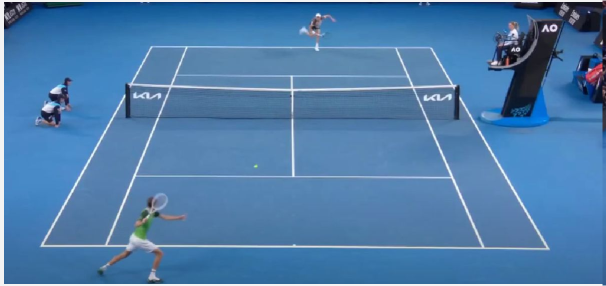
Total params: 10,719,104 (40.89 MB)
Trainable params: 10,707,744 (40.85 MB)
Non-trainable params: 11,360 (44.38 KB)

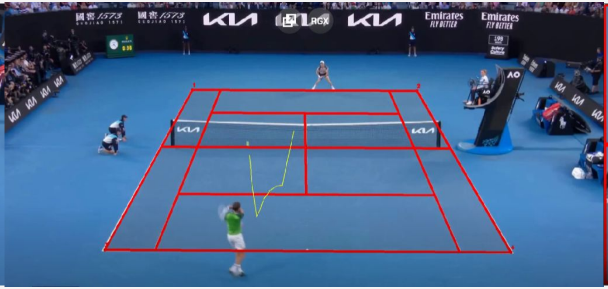


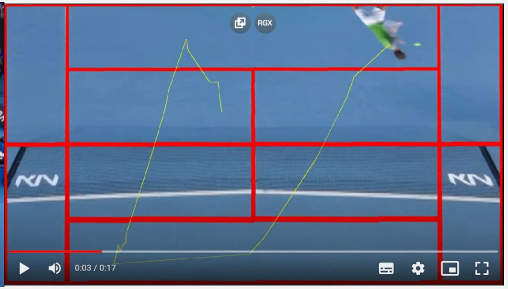
Choosing Best Input Form

Best input for the tracknet

During the implementation, 3 methods were tested as the input. The metric to choose the best one was done considering the amount of identified frames.

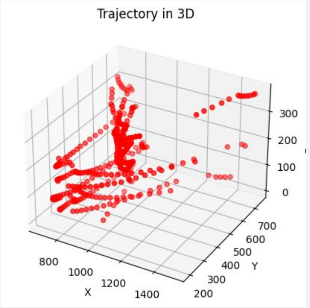


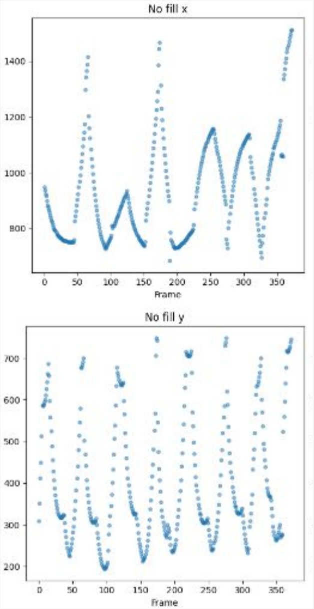


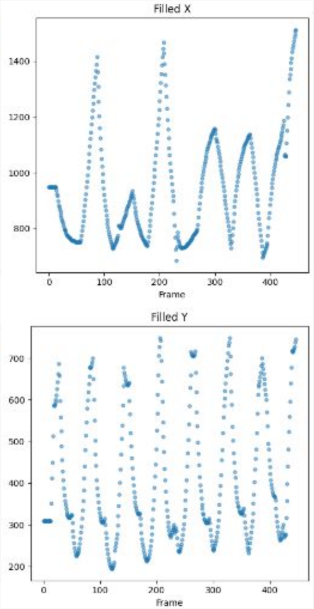


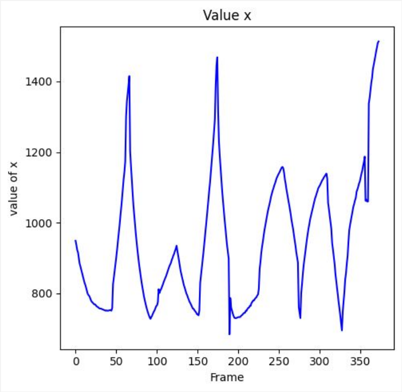
After some tests, the input that yielded the best results was the first one (given that it’s the most similar to the train test of the Tracknet).

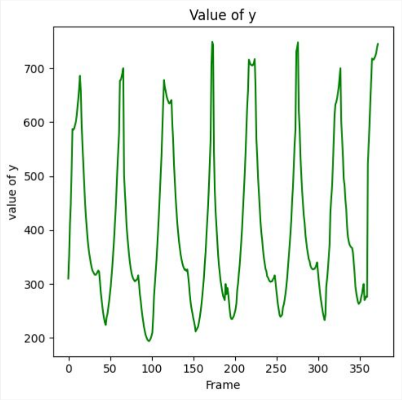
Data preprocessing







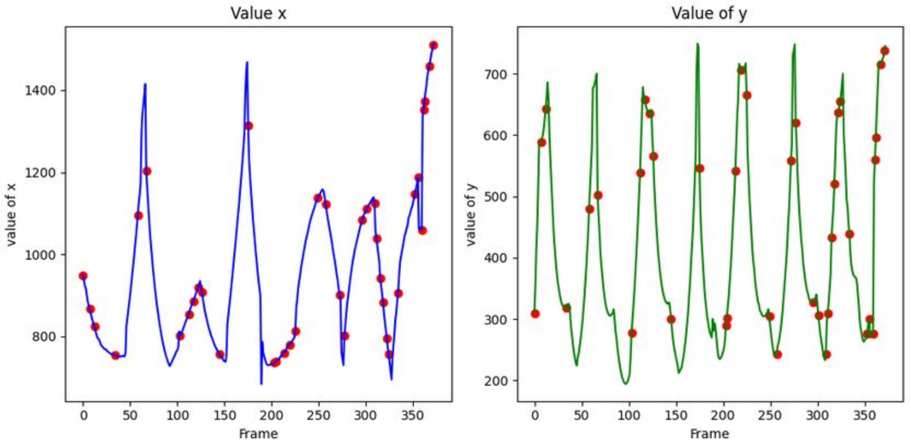




Bounce moment identification

Strategy 1

Using the gradient() function from NumPy to find the derivatives and find local minima using the change of sign from the second derivative of the positions where the first derivative equals zero.

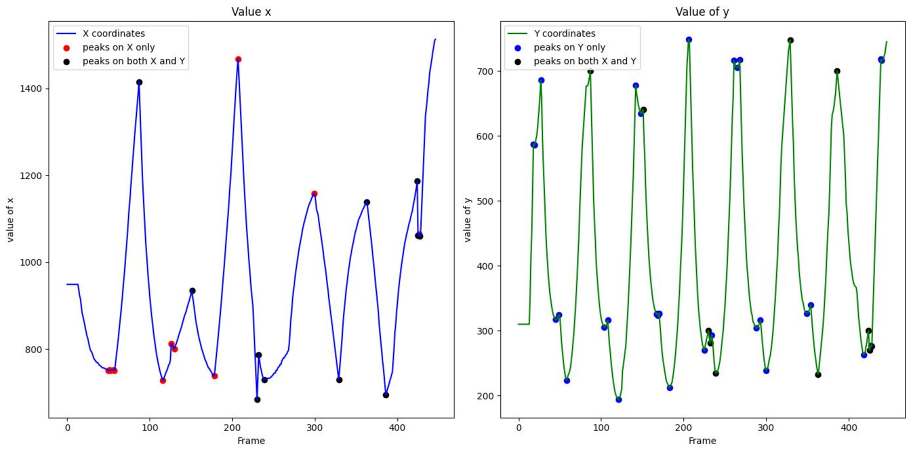


The results of this strategy were not as good as expected, due to inflection points introduced by fitting a continuous trajectory on the discrete points distribution.

Bounce moment identification

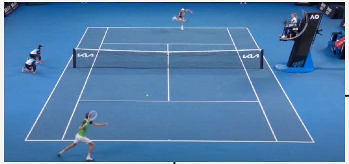
Strategy 2

Using the find_peaks() function from SciPy to find the peaks (trend change) in the coordinates and doing it again on the negative array to find the valleys of the data.

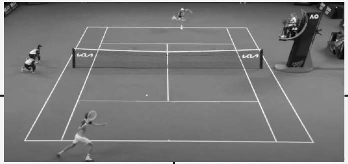


After obtaining the peaks, we applied a padding, ignoring peaks identified in a range of 4 frames.

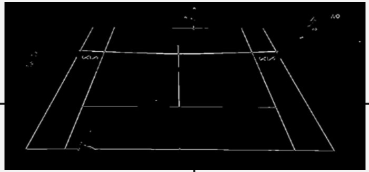
Court Identification



Input frame



Converted Frame to Grayscale



Binarized the Grayscale Image and filter it

`cv2.HoughLinesP()`

Detected Line Segments Probabilistic

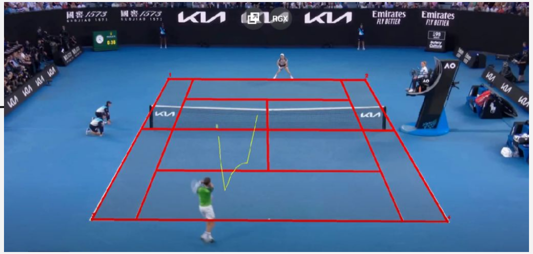
`cv2.findHomography()`

Estimate Homography Matrix and validate accuracy.

Track Court Lines in Subsequent Frames

`sympy.Line()`

Calculate and Sort Intersection Points



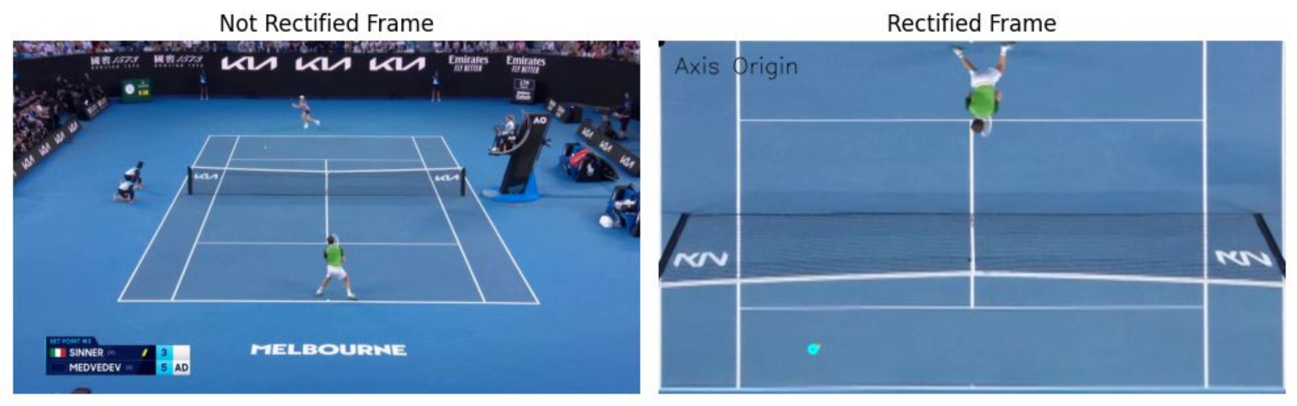
Output

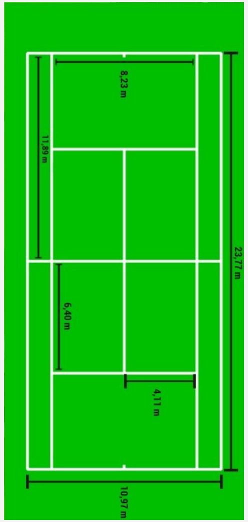
Image Rectification

From the output of the court detection, we could obtain the matrix

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

That generated the rectification:



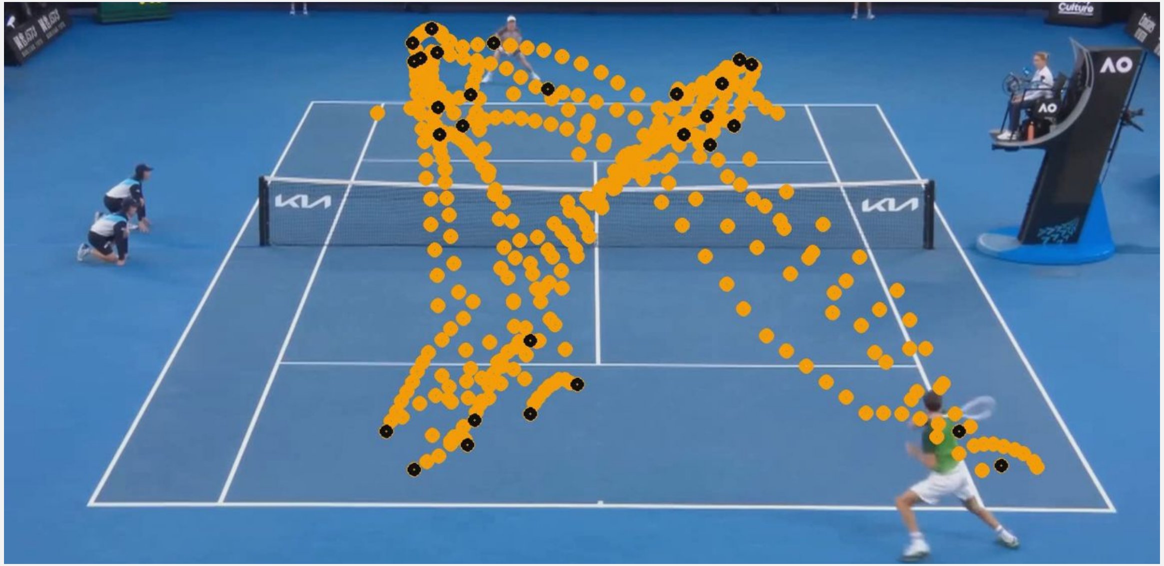


05

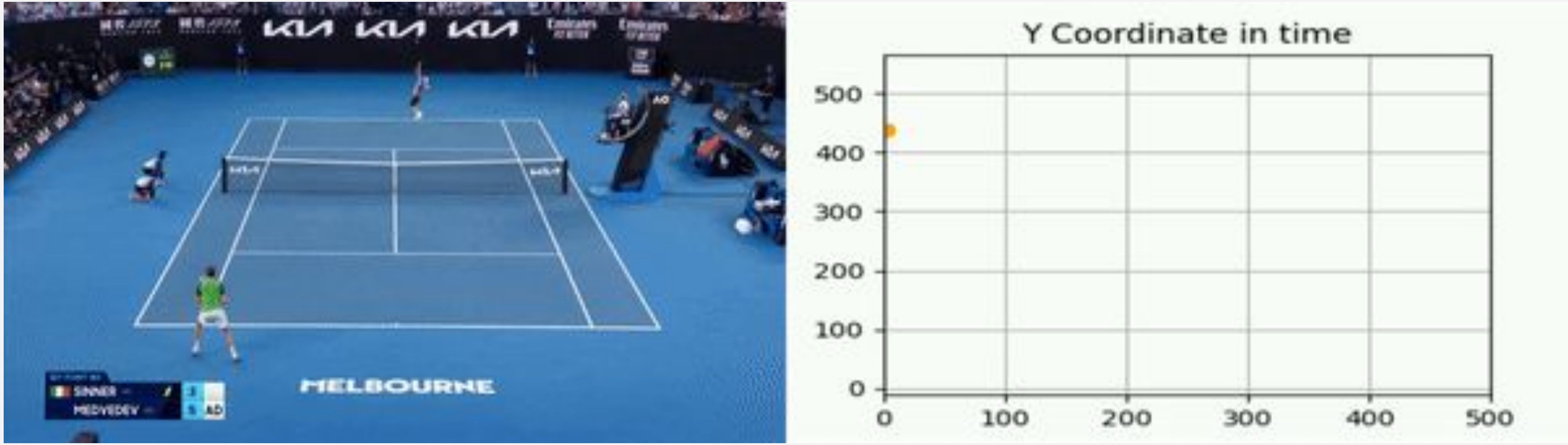
Analysis & Results

Tracking and bouncing results

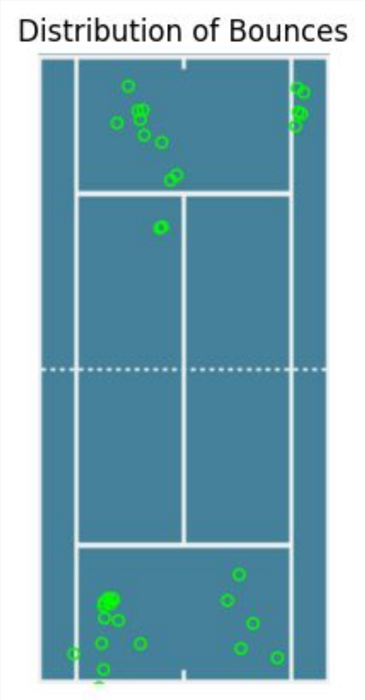
The tracking of the ball showed a quite good accuracy to the intuitive reality of the track. The bounces were correctly identified in the overall as well.

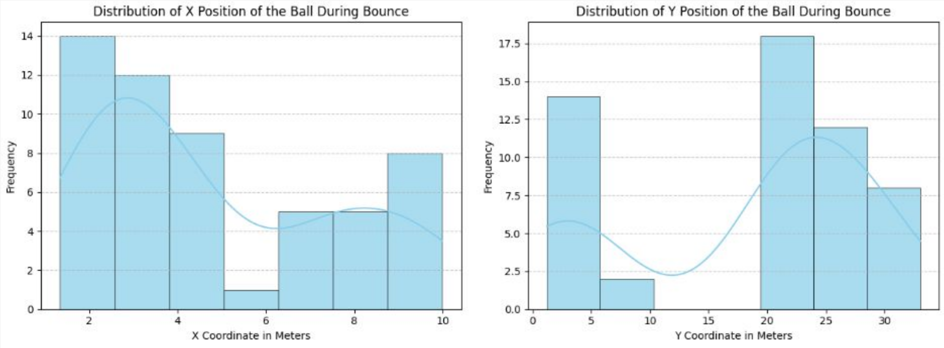


Tracking and bouncing results



Absolute position of the bounces





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

06

Conclusions

Conclusion

Key Takeaways

- The tracking of the ball was mostly satisfiable, as well as the data imputation;
- The bounce identification has a high precision (everytime it identifies a bounce, it’s a bounce);
- Traditional IACV techniques are really handy for current problems.

Improvements

- 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.

Future Features

- Further refactor the code used during the development in order to make it easier to other to use it;
- Use a bigger frame rate and better “find peak” strategy to improve the coordinate accuracy.

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.

Thanks!

Do you have any questions?