Tennis Autodistillation: Dataset Generation and Model Training

https://github.com/rafaelpadilla/tennis_autodistill

The Goal (pipeline)

https://github.com/rafaelpadilla/tennis_autodistill

https://github.com/rafaelpadilla/tennis_autodistill

Input video

Frame classification

Player detector

Court keypoints detector

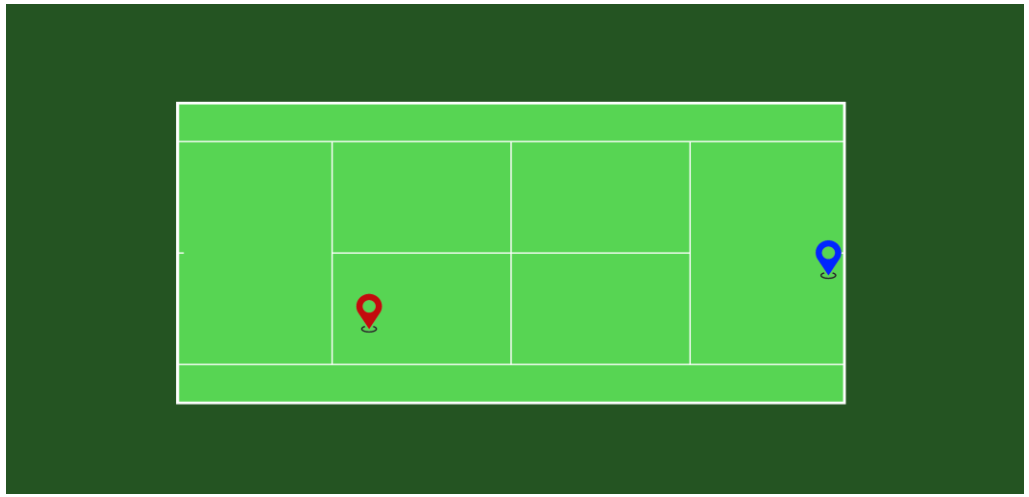
Compute homography

Annotated mini map

scripts/run_pipeline.py

$$\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

Input frame



Output data

Input video

Frame classification

yolo11n-cls

game_play

"an image showing the full playable area of a tennis court with camera positioned higher and centered, providing a full view of the entire court"

close_up

"a close-up shot of a tennis player"

partial_view

"an image showing partially the playable area of a tennis court, providing a partial view of the court"

Ignore1, ignore2

"anything else", "none"

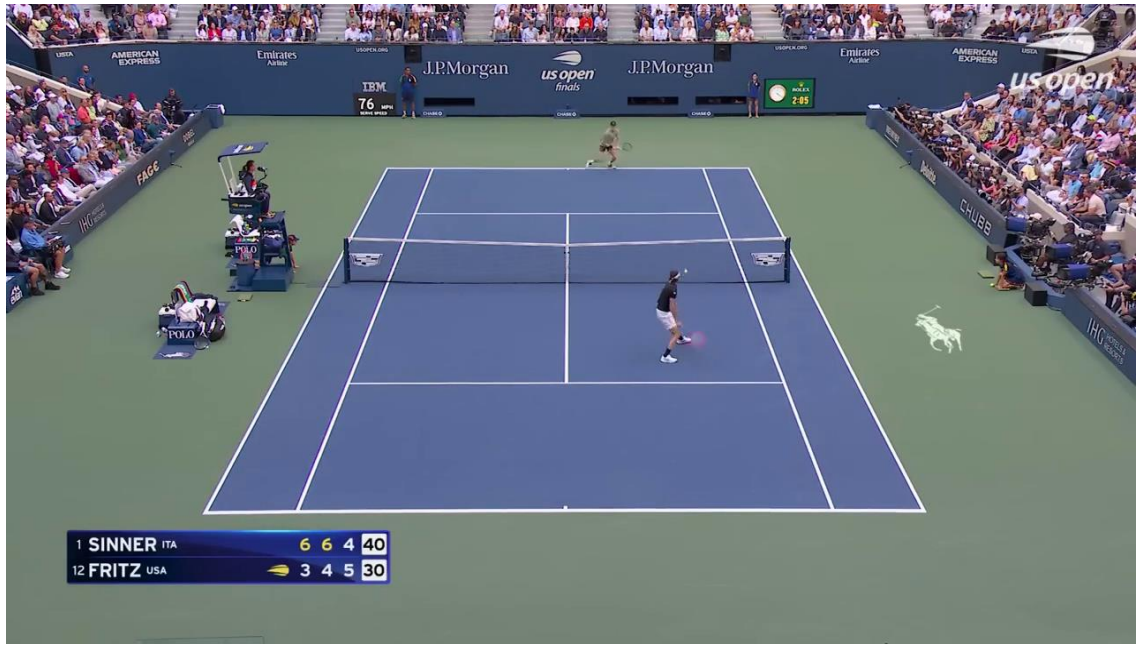
https://github.com/rafaelpadilla/tennis_autodistill

Input video

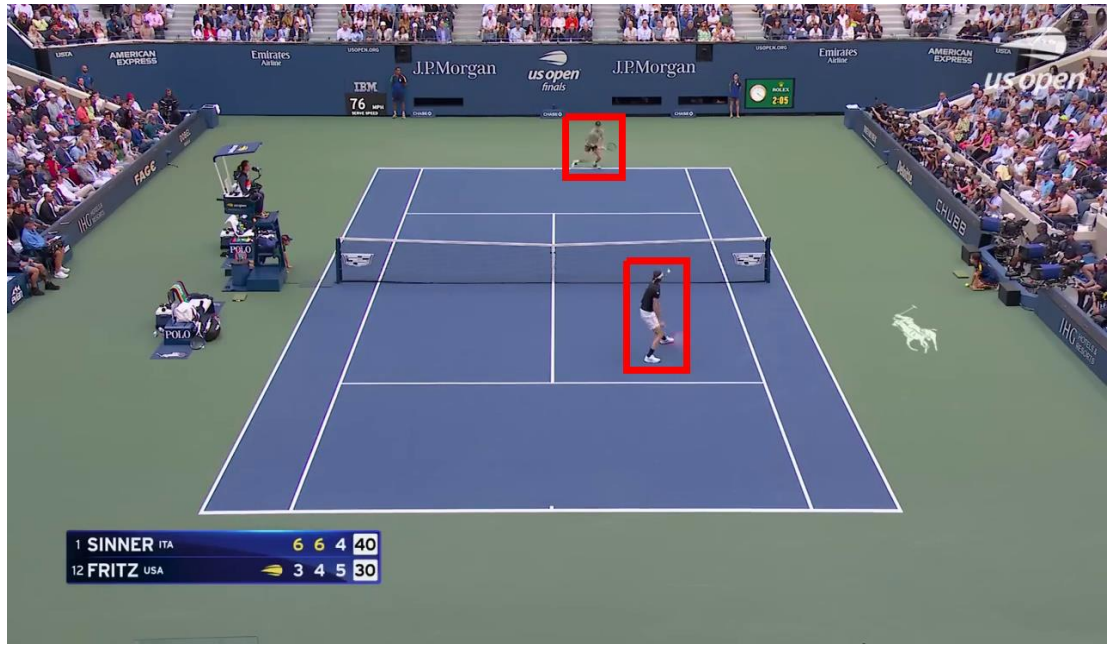
Frame classification

Player detector

Yolo11n.pt



Input frame



Output of the **player detector**: bounding boxes

https://github.com/rafaelpadilla/tennis_autodistill

https://github.com/rafaelpadilla/tennis_autodistill

Input video

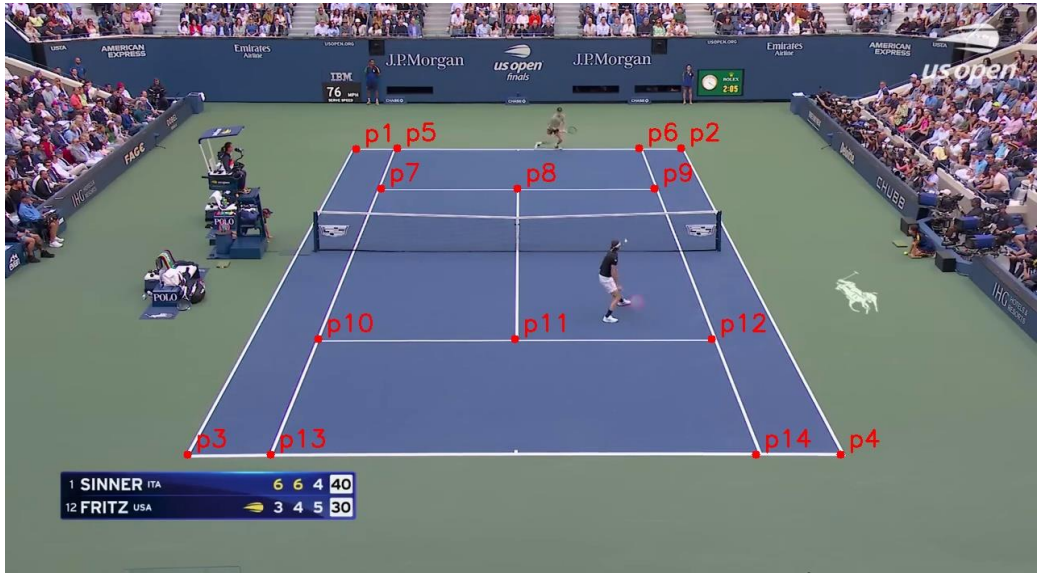
Frame classification

Player detector

Court keypoints detector

Yolo11n-pose

Input frame



Output of the **keypoint detector**: landmarks on the court

Input video

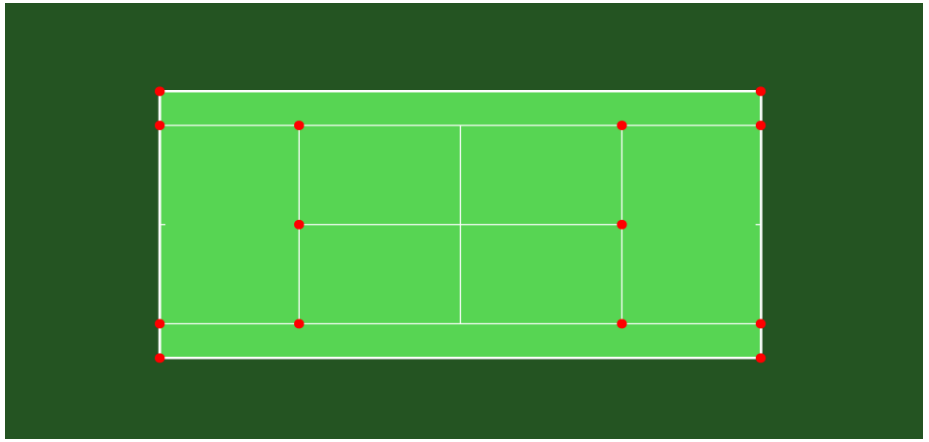
Frame classification

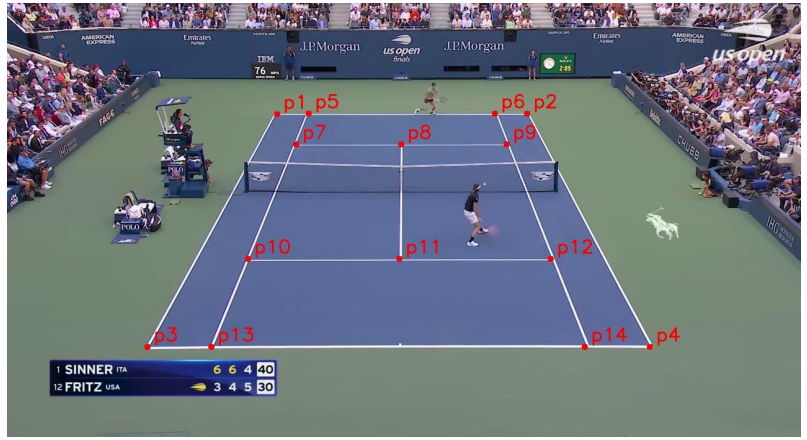
Player detector

Court keypoints detector

Annotated mini map

Compute homography





Inputs: annotated mini map & keypoints on the court

$$\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

Output: homography H

https://github.com/rafaelpadilla/tennis_autodistill

https://github.com/rafaelpadilla/tennis_autodistill

Input video

Frame classification

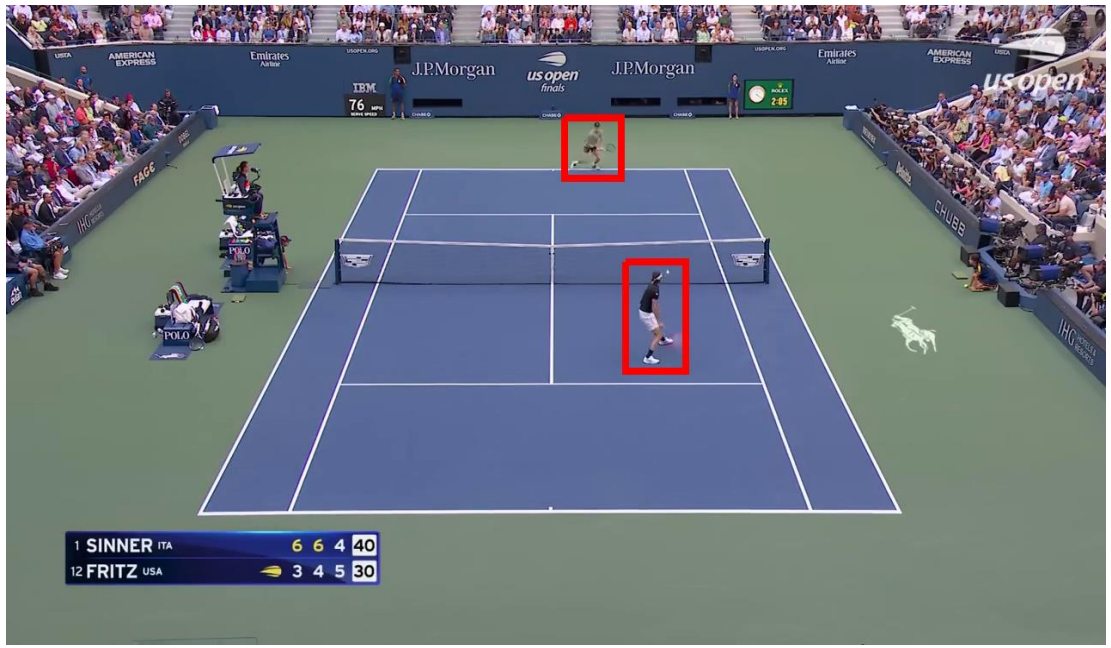
Player detector

Court keypoints detector

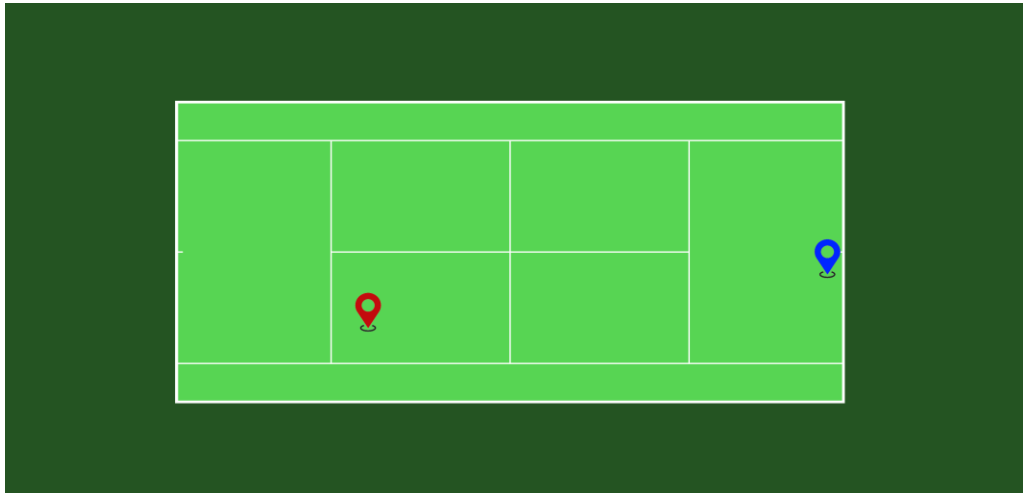
Annotated mini map

Compute homography

$$\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$

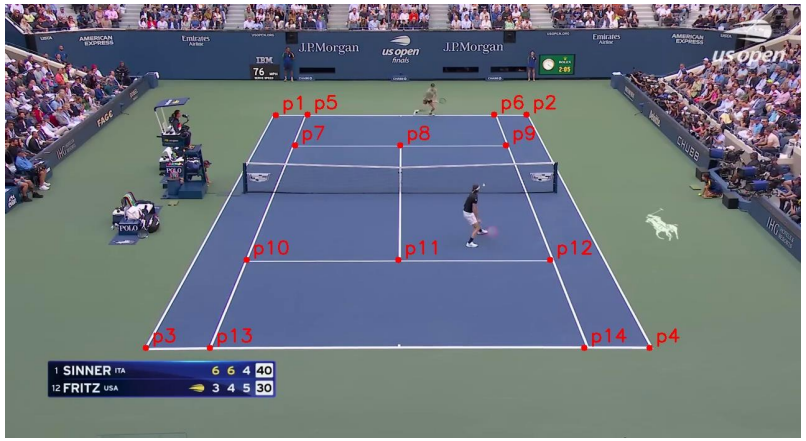


Input: bounding boxes with the players

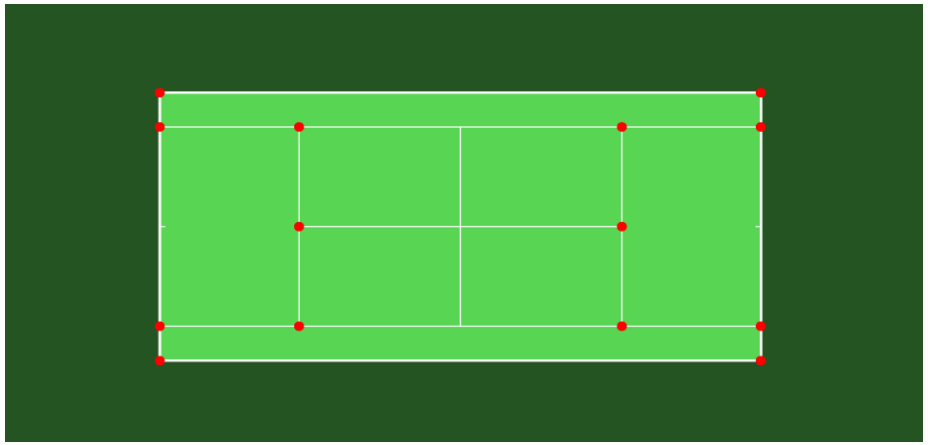


Output: projected (x,y) positions of the players

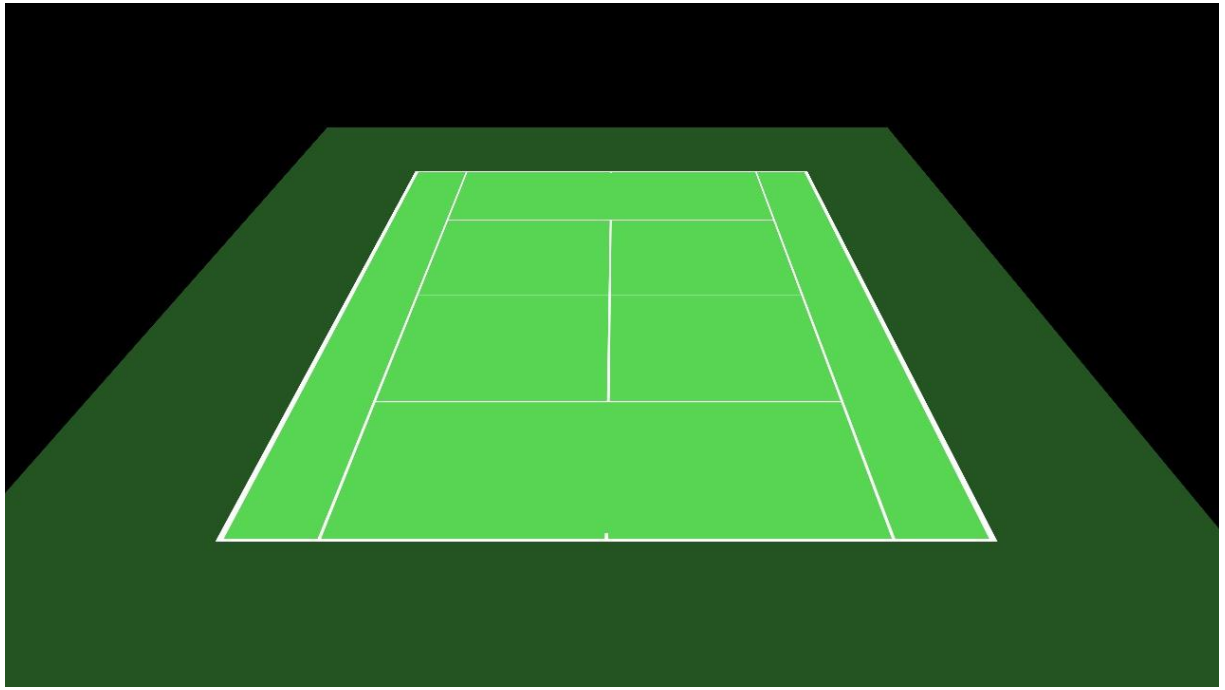
https://github.com/rafaelpadilla/tennis_autodistill



$$\begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}$$



Transformed mini map using inv(H)

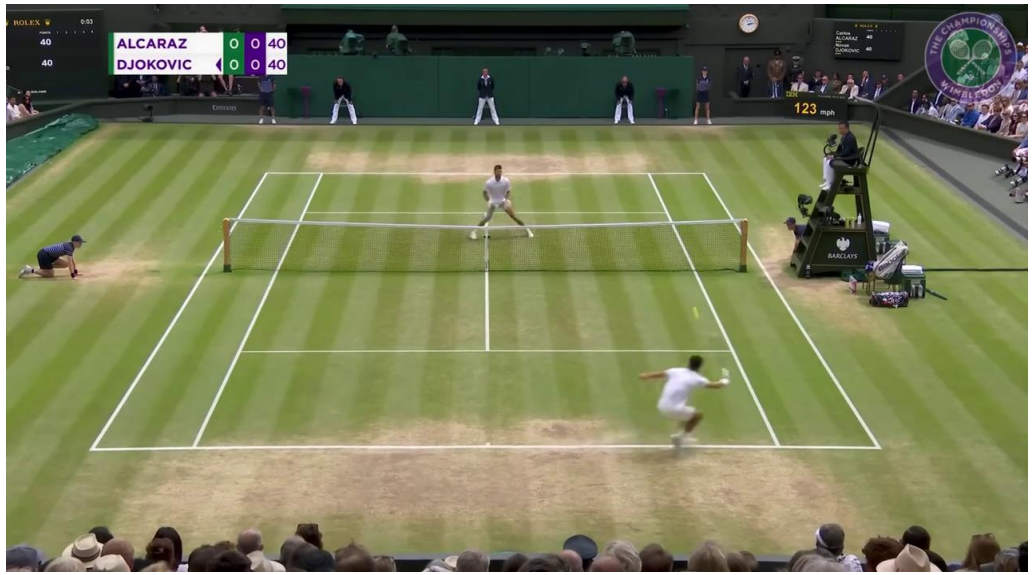


Dataset creation

https://github.com/rafaelpadilla/tennis_autodistill

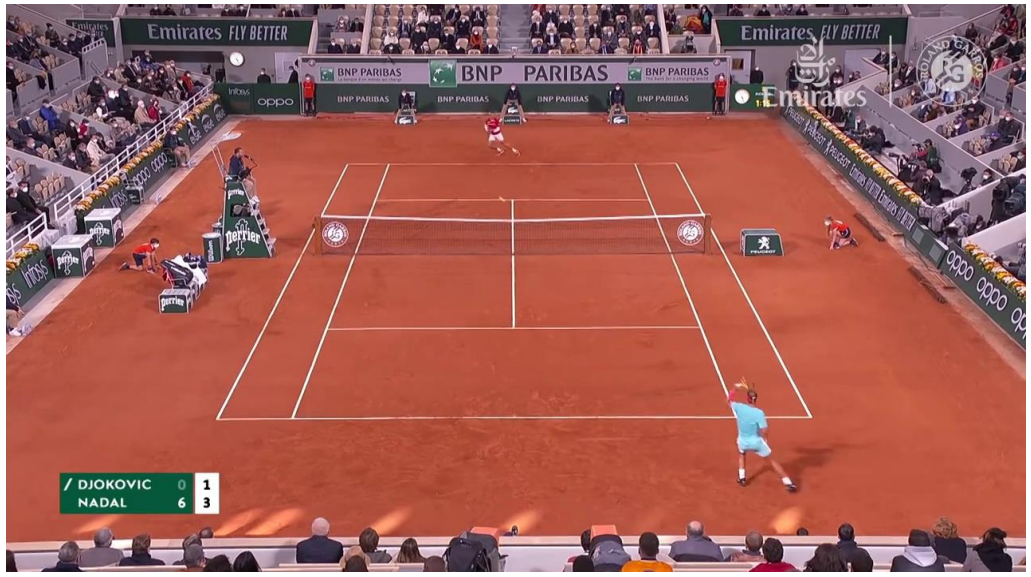
https://github.com/rafaelpadilla/tennis_autodistill

grass

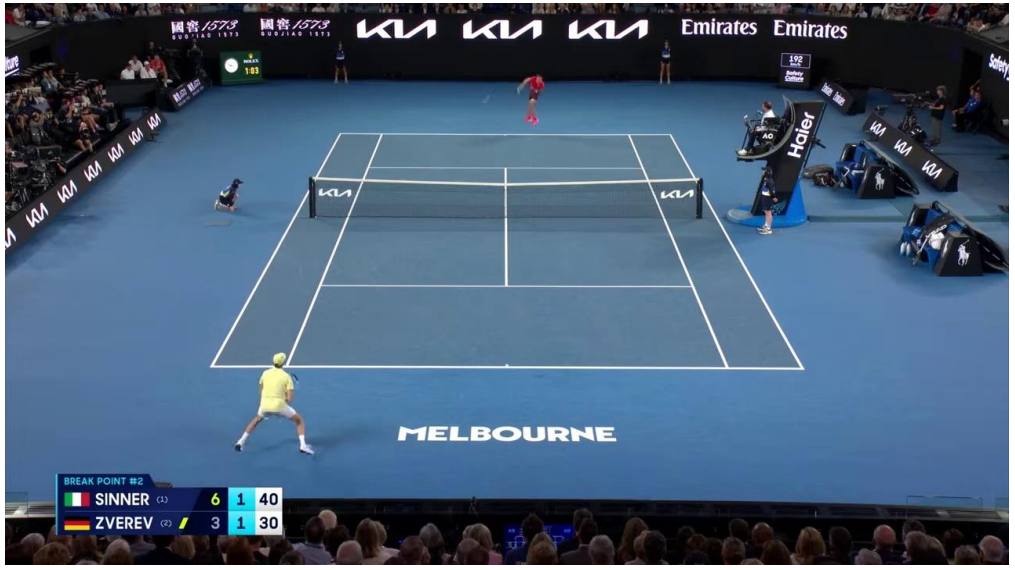


hard laykold

red clay



hard plexicushion



dataset/video_links.txt

Dataset

Hard laykold:
- US Open 2024 (Andrey Rublev vs. Grigor Dimitrov Extended Highlights) - hard_laykold-video_1
- US Open 2024 (Jannik Sinner vs. Taylor Fritz Extended Highlights) - hard_laykold-video_2

Red clay:
- Roland-Garros 2020 (Rafael Nadal vs Novak Djokovic) - red_clay-video_1
- Roland-Garros 2019 (Roger Federer vs Casper Ruud) - red_clay-video_2

Grass:
- Wimbledon 2024 (Carlos Alcaraz vs Novak Djokovic) - grass-video_1
- Wimbledon 2019 (Novak Djokovic vs Roger Federer) - grass-video_2

Hard plexicushion:
- Australian Open 2025 (Aryna Sabalenka vs Madison Keys) - hard_plexicushion-video_1
- Australian Open 2025 (Jannik Sinner vs Alexander Zverev) - hard_plexicushion-video_2

https://github.com/rafaelpadilla/tennis_autodistill

https://github.com/rafaelpadilla/tennis_autodistill

Frame classification dataset



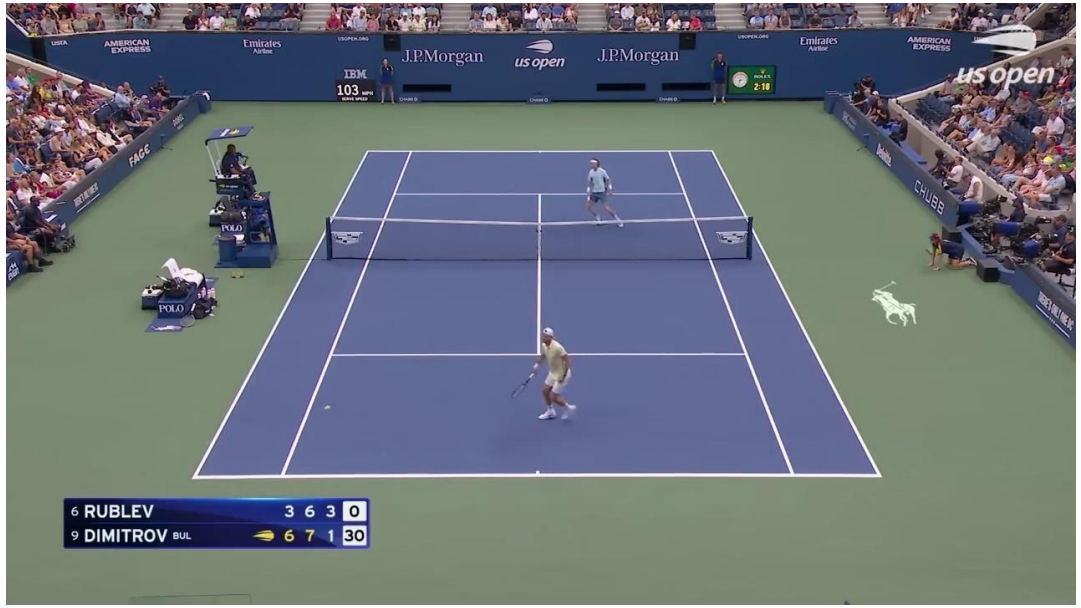


...

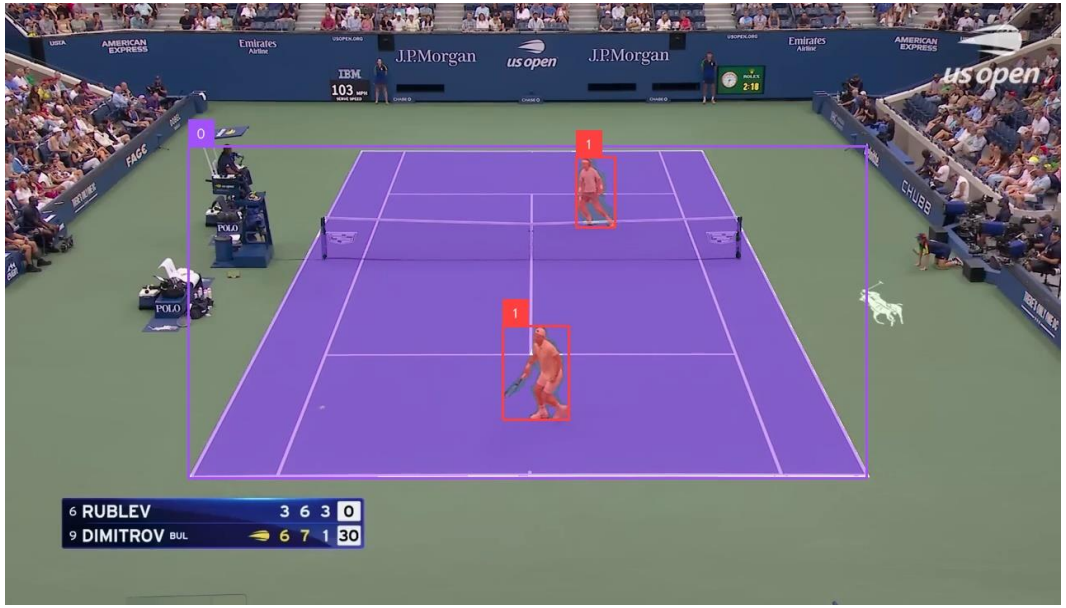
chatGPT

close up	"a close-up shot of a tennis player"
game play	"an image showing the full playable area of a tennis court with camera positioned higher and centered, providing a full view of the entire court"
ignore 1	"anything else"
ignore 2	"none"
partial view	"an image showing partially the playable area of a tennis court, providing a partial view of the court"

Player detector dataset



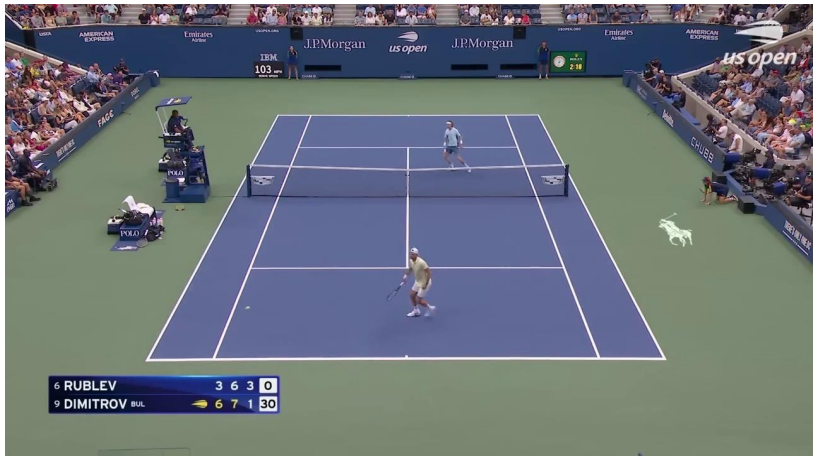
SAM 2



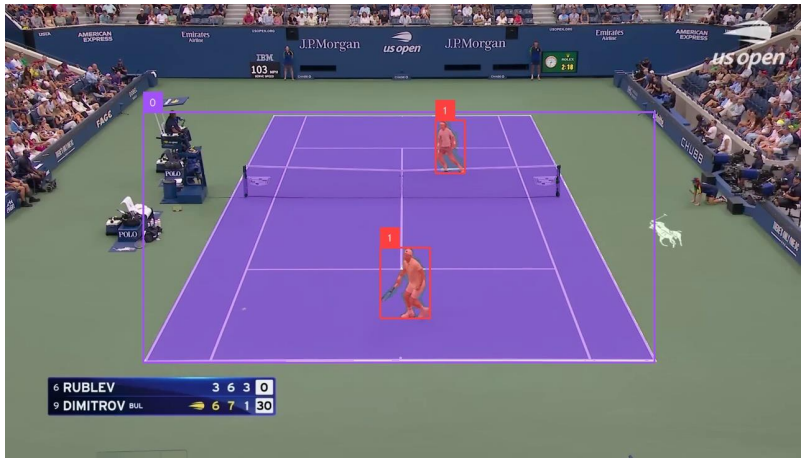
Detected masks converted to bounding boxes

https://github.com/rafaelpadilla/tennis_autodistill

Landmarks detector dataset



SAM 2



Mask of the court (playable area)

Estimated corners (playable area)

Computed 14 points of the court with homography

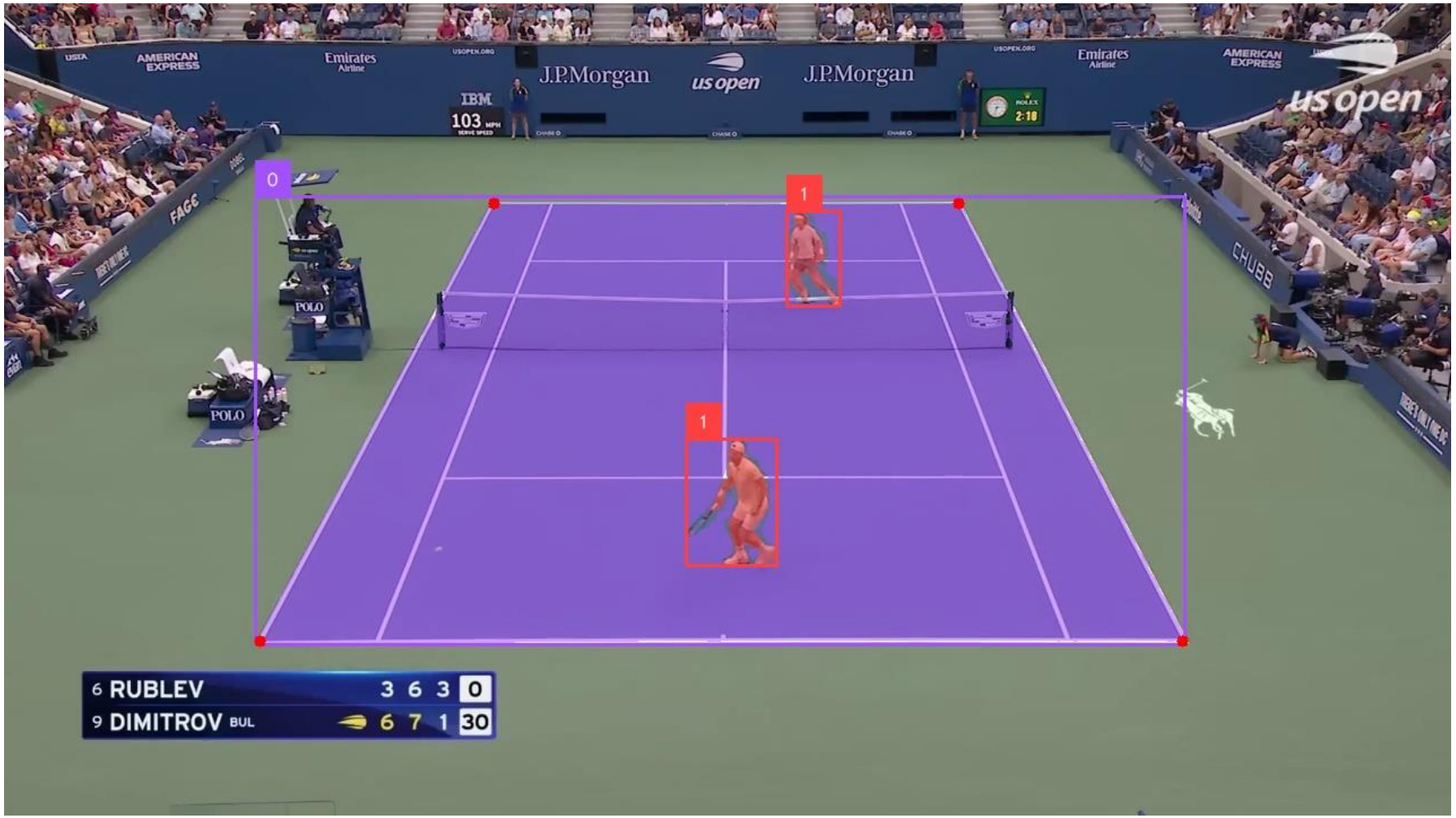
https://github.com/rafaelpadilla/tennis_autodistill

Identifying the points on the court

https://github.com/rafaelpadilla/tennis_autodistill

Identifying the points on the court

Based on the bounding boxes and the mask of the court, how to detect the corners of the court?

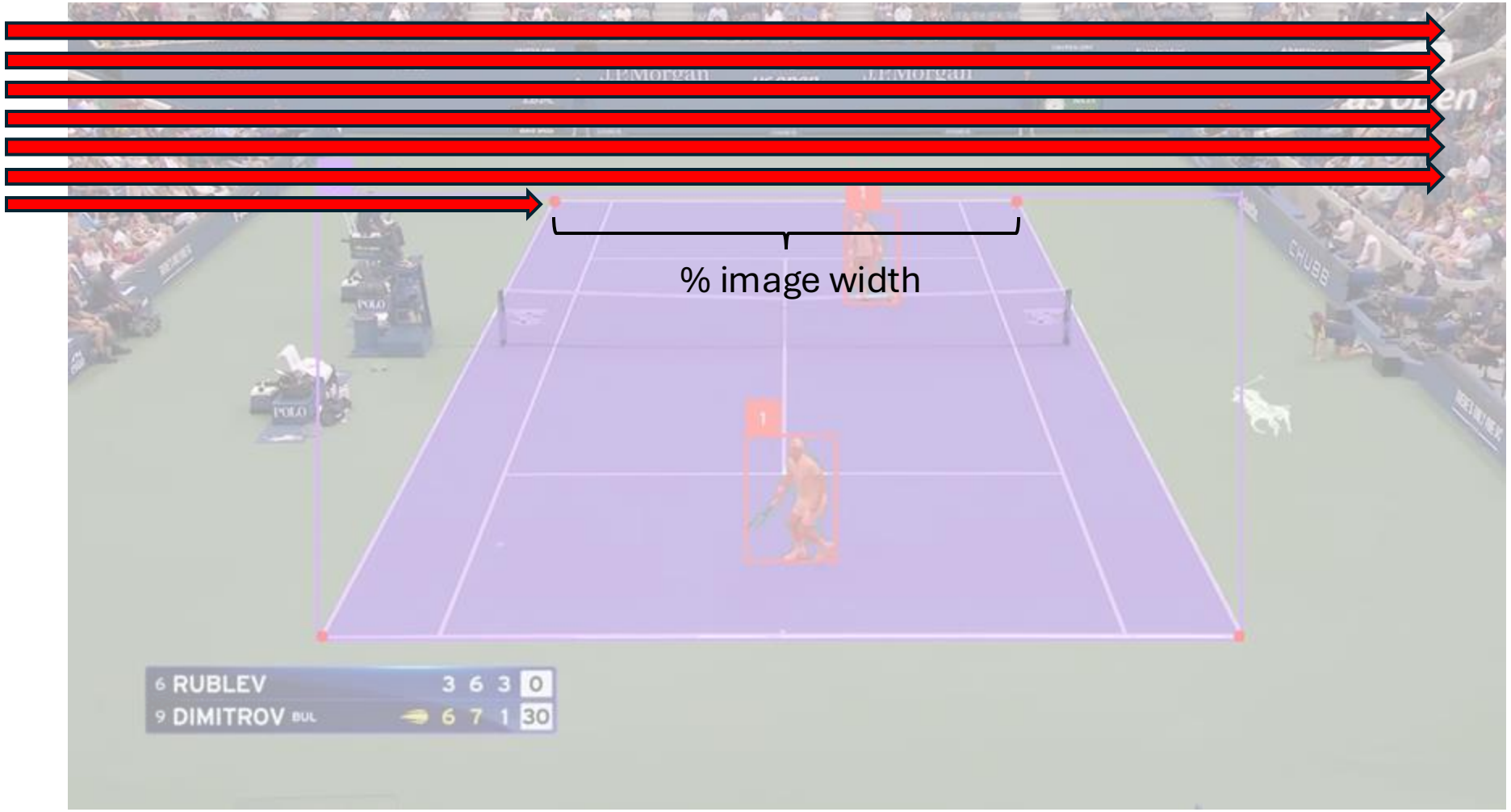


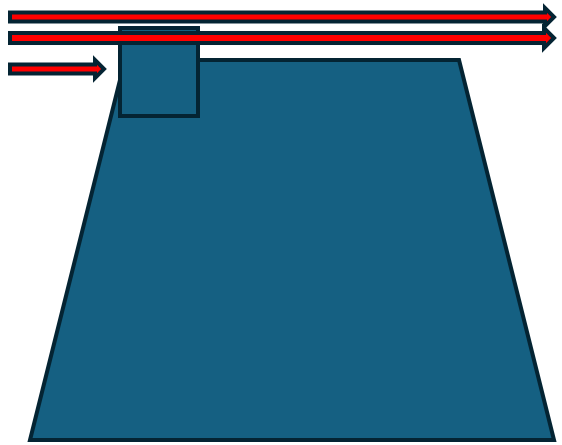
trapezoid

https://github.com/rafaelpadilla/tennis_autodistill

Identifying the points on the court

Based on the bounding boxes and the mask of the court, how to detect the corners of the court?



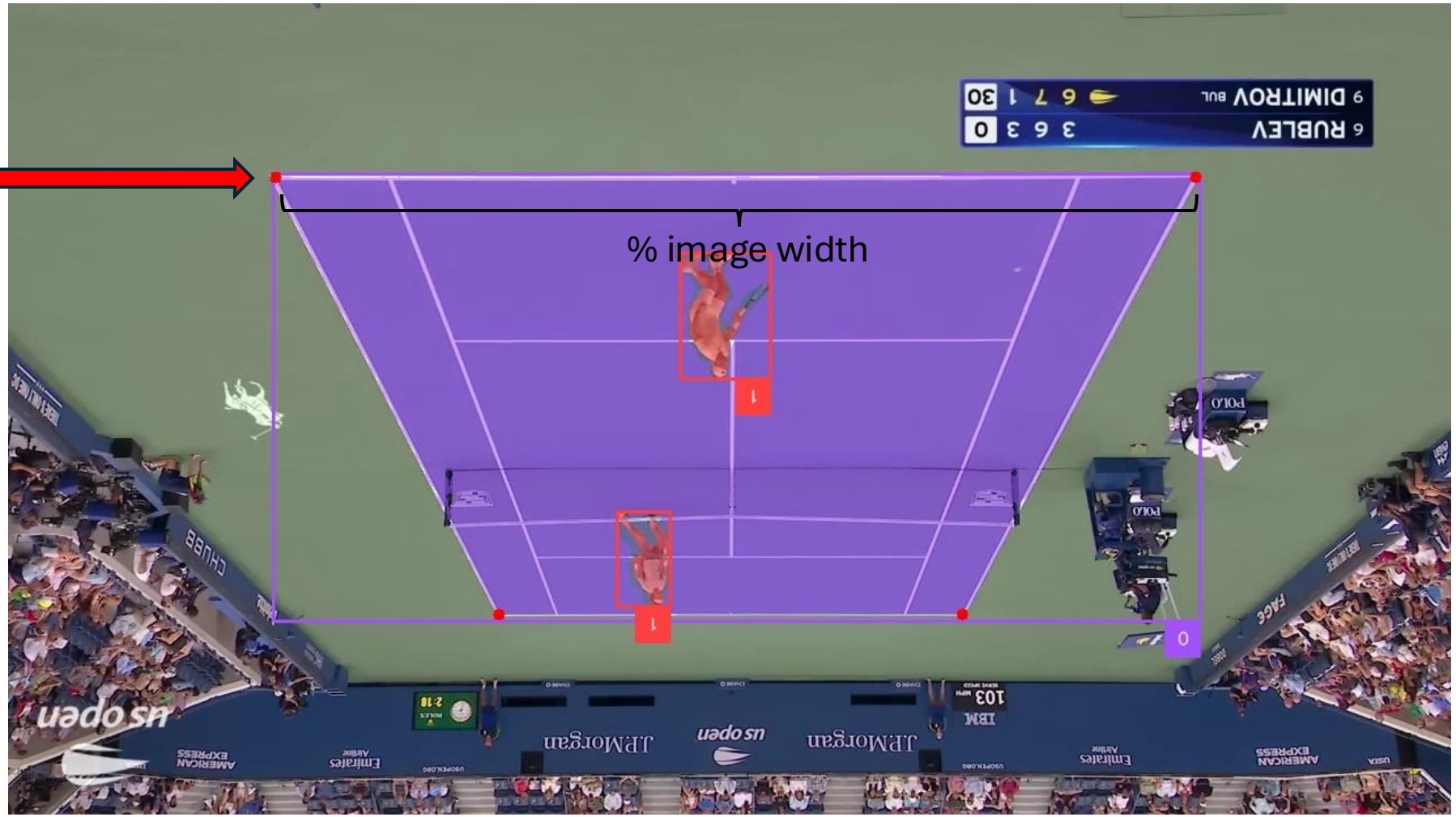


trapezoid

https://github.com/rafaelpadilla/tennis_autodistill

Identifying the points on the court

Based on the bounding boxes and the mask of the court, how to detect the corners of the court?

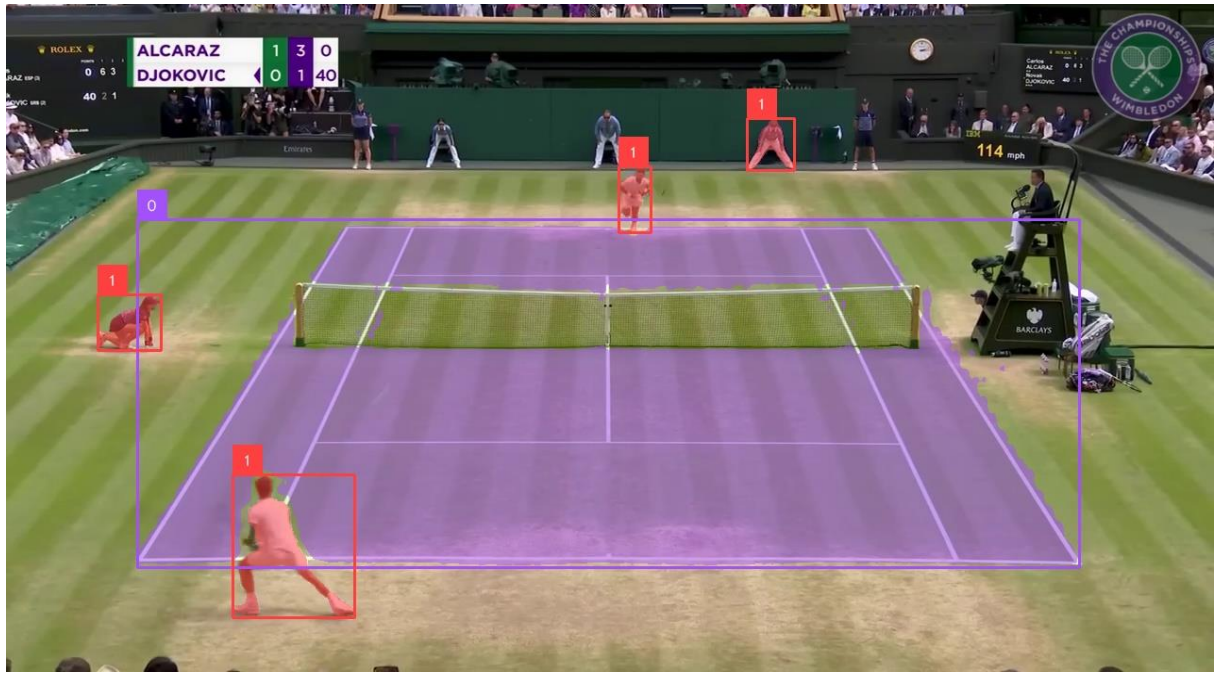


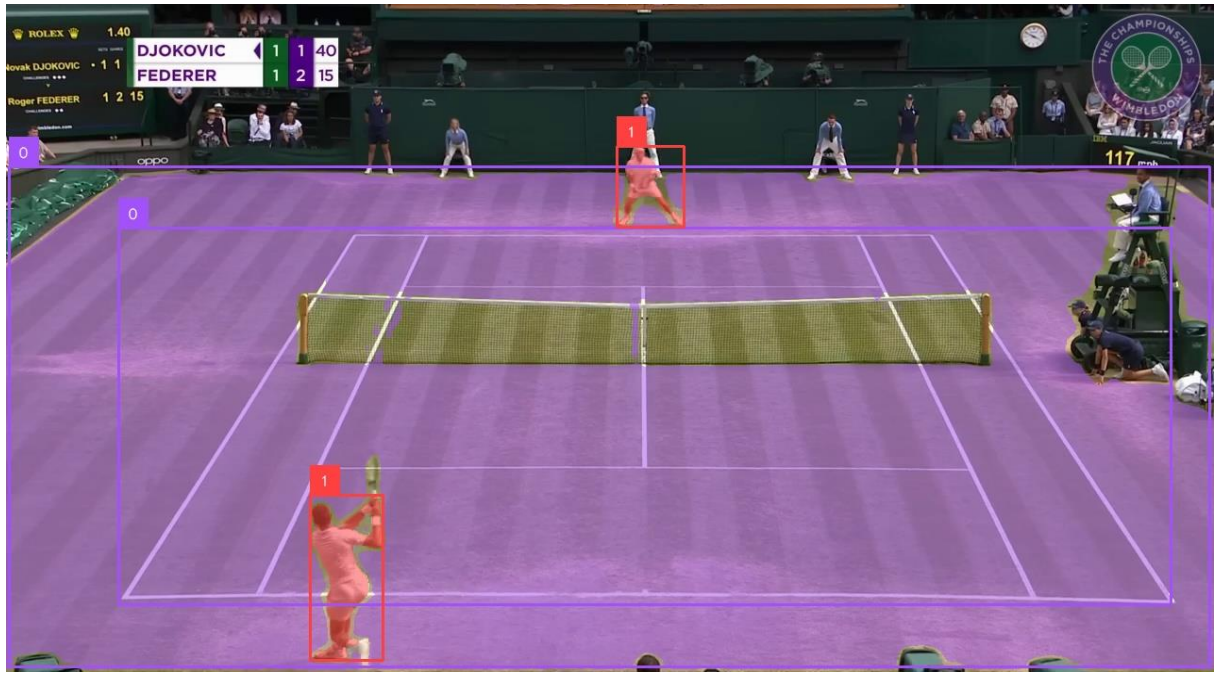
trapezoid

https://github.com/rafaelpadilla/tennis_autodistill

Identifying the points on the court

Filter 1: *Ignore images that do not have 2 players and 1 playable area*

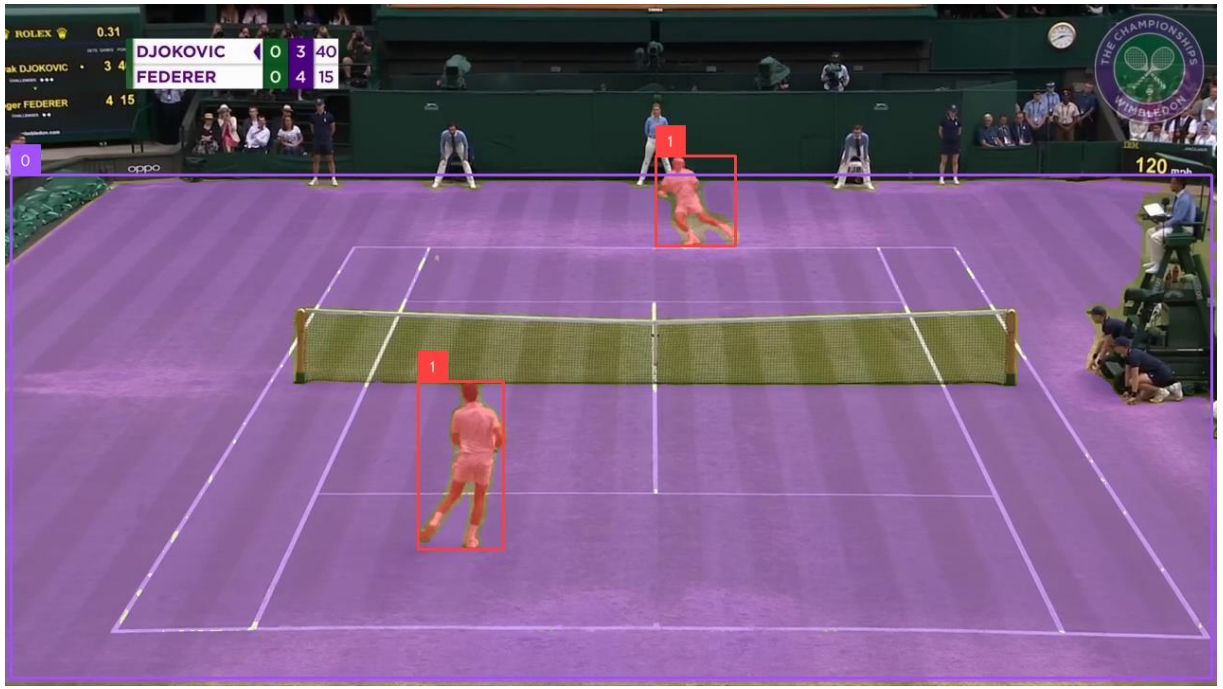


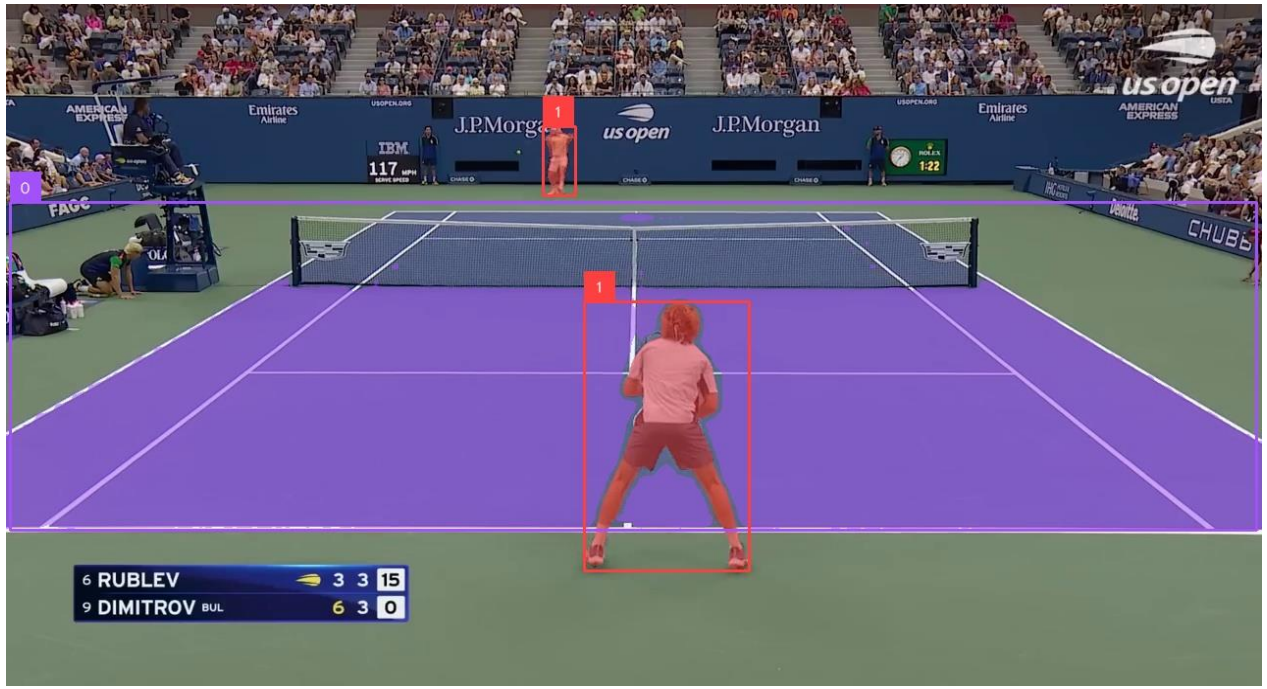


https://github.com/rafaelpadilla/tennis_autodistill

Identifying the points on the court

Filter 2: Ignore images that the playable area is touching the borders of the image

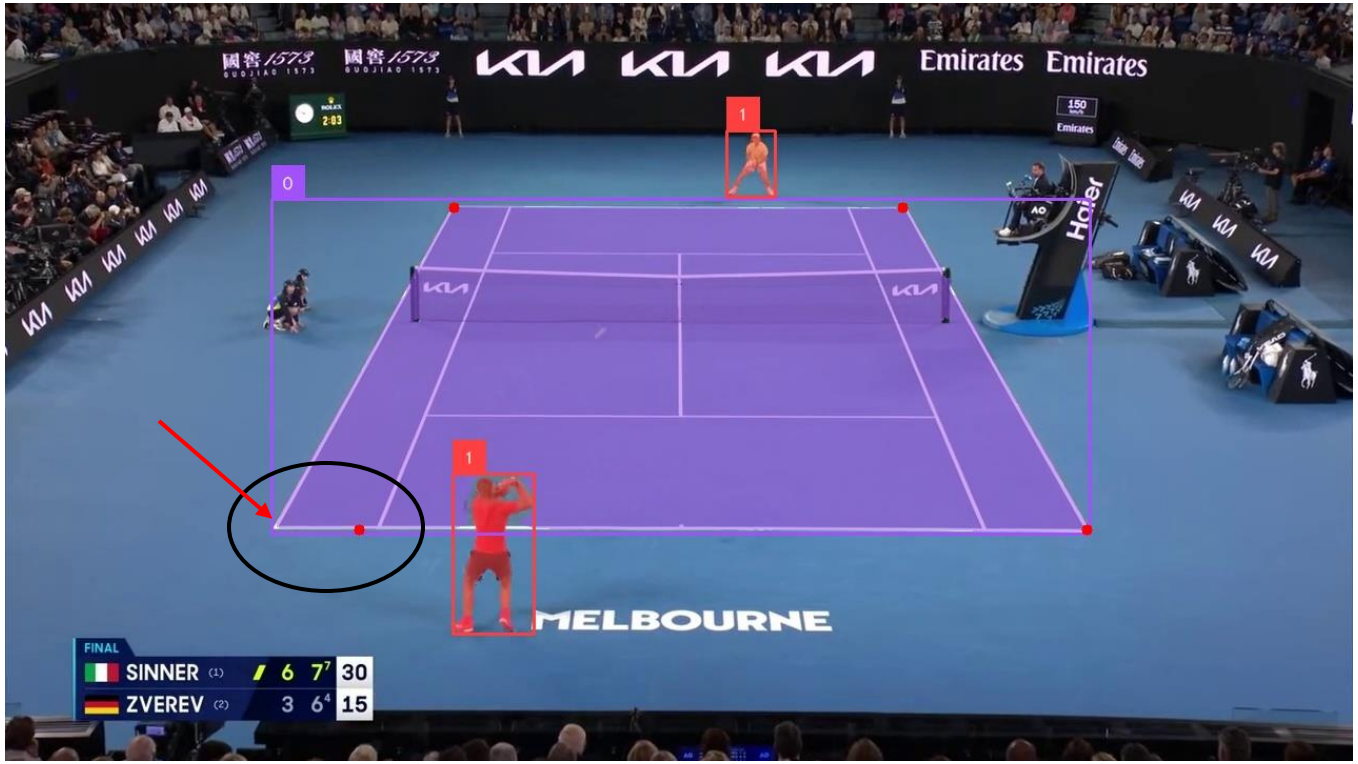




https://github.com/rafaelpadilla/tennis_autodistill

Identifying the points on the court

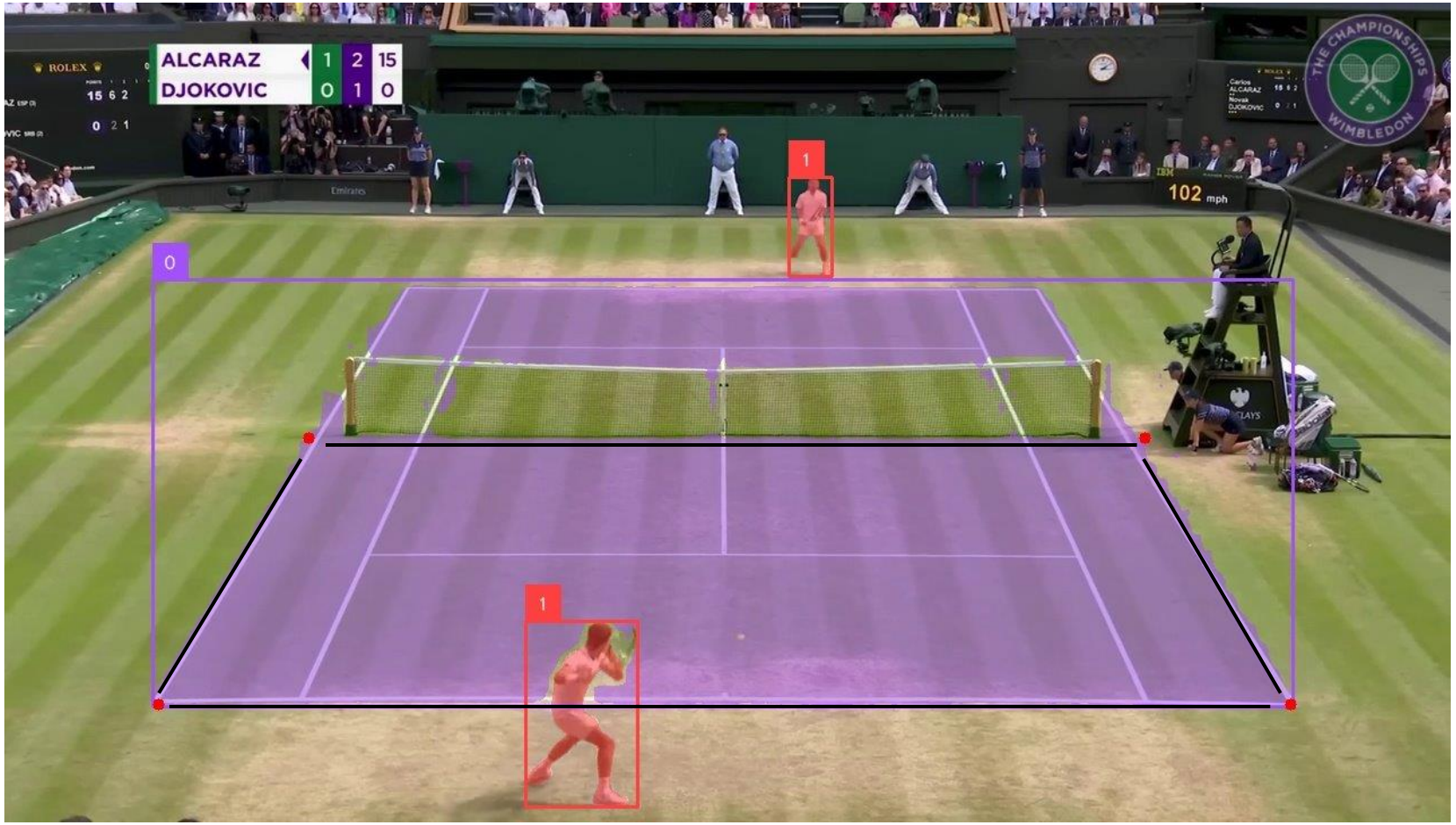
Filter 3: Ignore images that the bottom edges of the playable area (points 3 and 4) are distant from the corners of the bounding box of the playable area.



https://github.com/rafaelpadilla/tennis_autodistill

Identifying the points on the court

Filter 4: Ignore images that the playable are has considerable holes

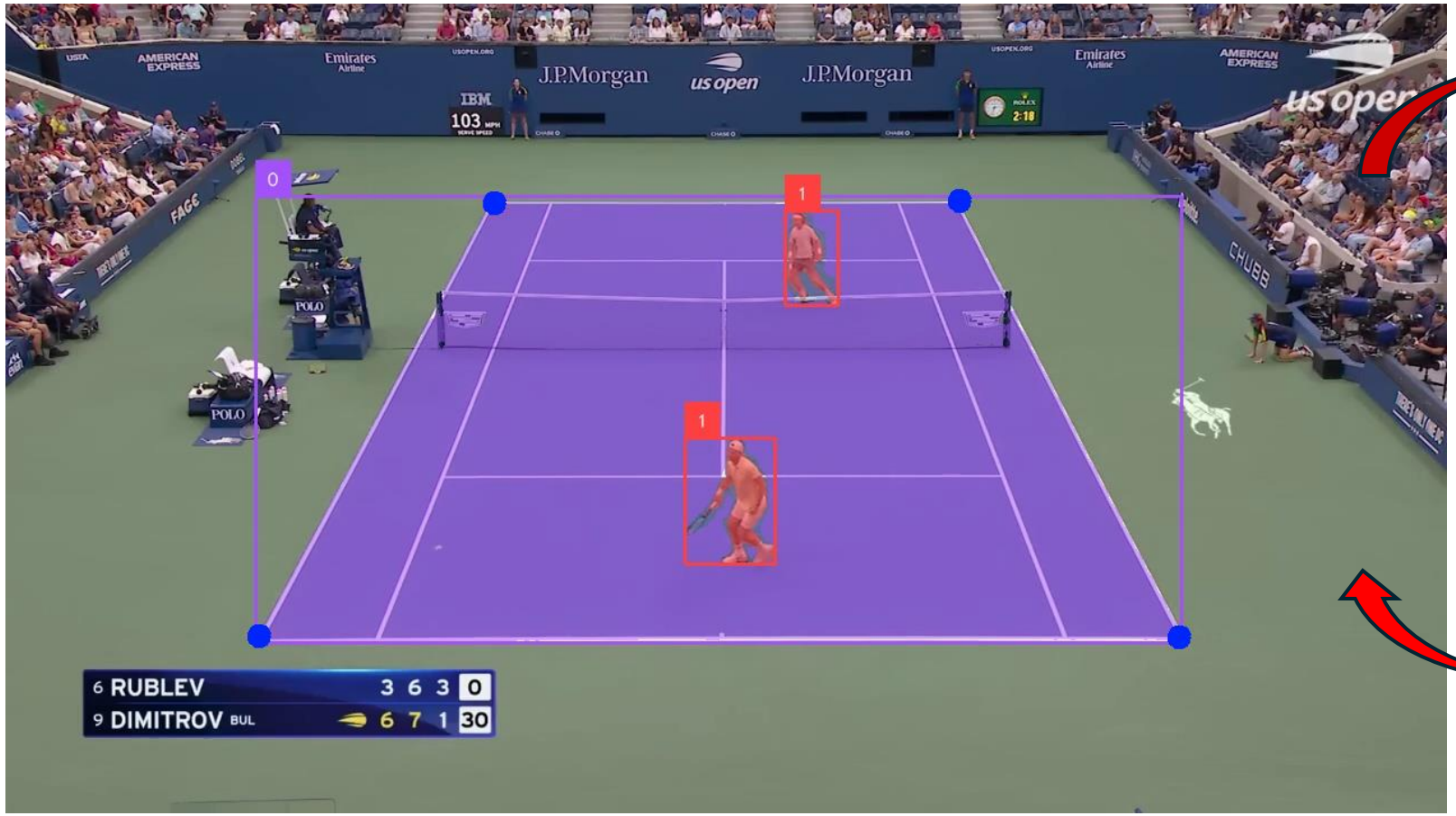


area trapezoid / area mask > threshold

https://github.com/rafaelpadilla/tennis_autodistill

https://github.com/rafaelpadilla/tennis_autodistill

How to obtain 14 points on the court with a mask?



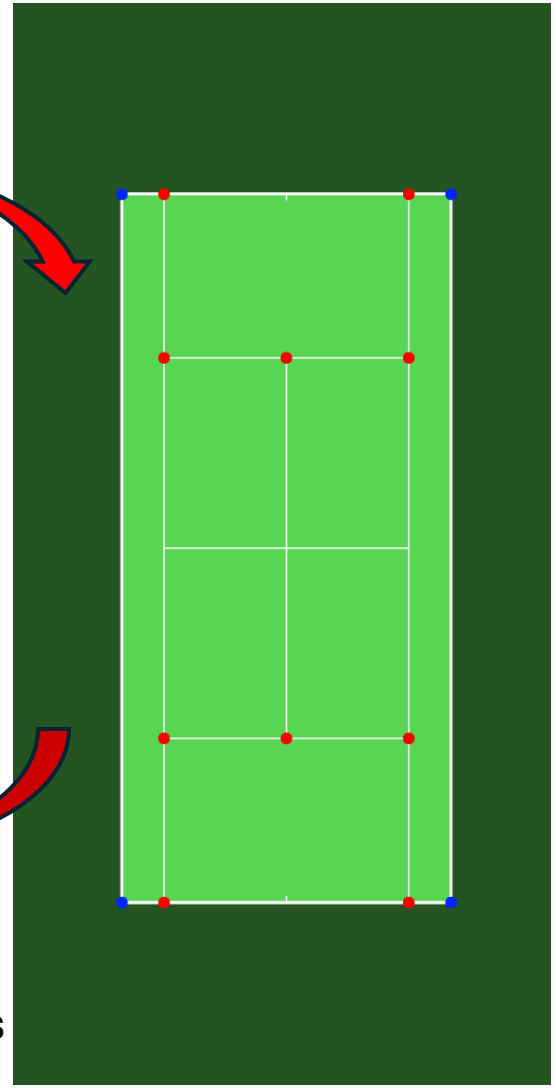
compute H using only the corners

H

H^{-1}

Project all 14points into the court

utils/coords_court_map.py



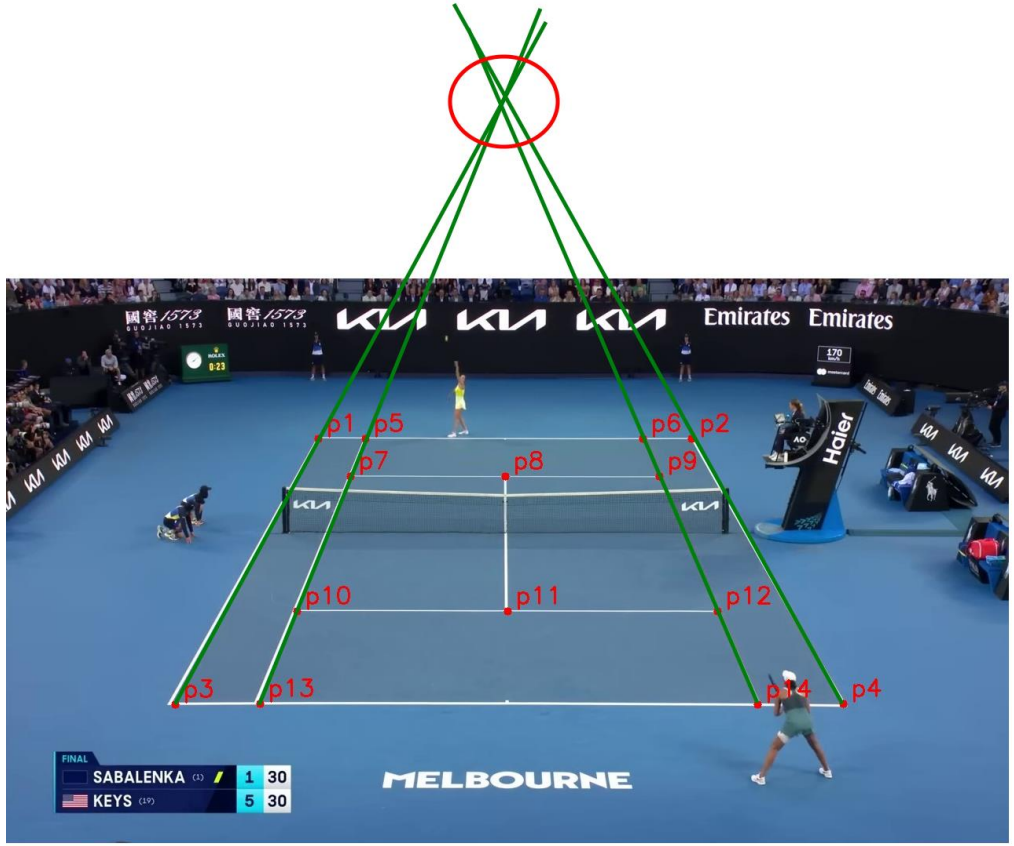
Improvements on Landmarks

Image-based

- Apply corner detectors (Harris)
- Corner descriptors

Geometric enforcement

- Parallel lines intercept each other on the same vanishing point
- Points should lay on the same line segment
- Trackers (Kalman Filter-based, optical flow, SORT, DeepSORT, etc.)



Due to inaccuracies in the detected points, the vanishing point cannot be computed precisely.

https://github.com/rafaelpadilla/tennis_autodistill

https://github.com/rafaelpadilla/tennis_autodistill

Examples of images automatically annotated