Detectis

1. Project Overview

This project aims to detect athletes "in action" during a tennis match. Two fundamentally different approaches were implemented: a classical computer vision method and a deep learning method. Both approaches were applied on the same video and were compared and discussed in detail.

Additionally, the detected athletes are subjected to keypoint estimation using a separate deep learning model. This allows for further analysis of their movements and positions during the match.

2. Approach

2.1 Classical Computer Vision Technique

2.1.1 Description

The first approach implemented involves using traditional computer vision techniques to detect objects in a video. It is crucial to consider the properties of the video for this approach. Specifically, in this case, the camera remains stationary while the players move within a constrained area on the court.

Based on these properties, a pixel-based approach using two consecutive video frames was implemented. More specifically, the pixel-wise absolute difference between two consecutive grayscale frames was computed. By applying a threshold to define the pixels that changed from the previous frame to the current frame, a binary foreground mask was generated.

To refine the foreground mask and remove noise, two morphological operations were applied. First, a closing operation was performed to remove noisy pixels from the foreground. Second, an opening operation was used to fill gaps. Using the contours of the foreground objects, the bounding boxes of the detected players were computed.

Additionally, to increase robustness, a specific area in the video was defined where athletes are expected to be. This helps to exclude any potential false detections of humans outside this area.

2.1.2 Other explored approaches

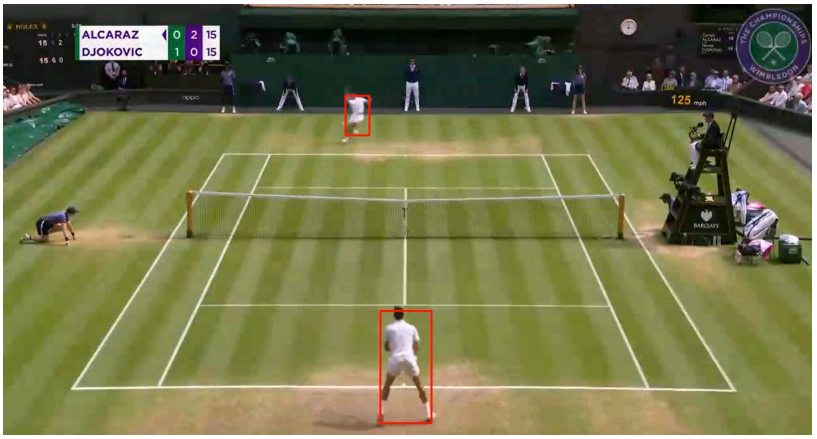
Other approaches were tested during the implementation process. One initial method involved using scale and rotation invariant template matching, where a template of the object of interest was matched in each frame. This template required regular updates to keep track

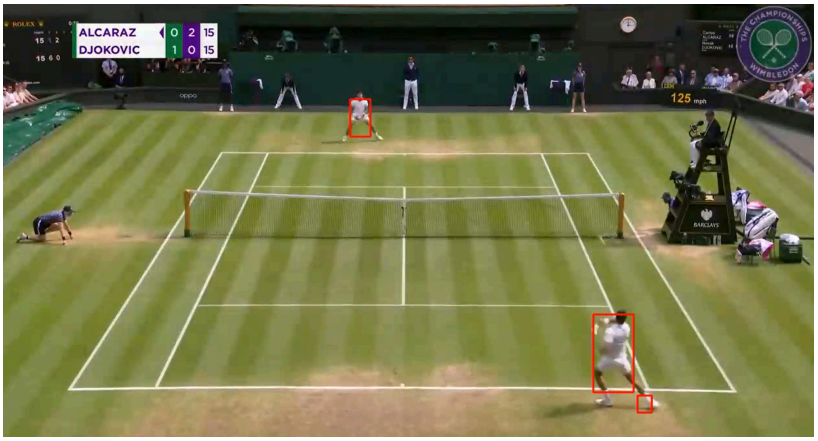
of the athlete's movement and body deformation. However, this update introduced a constant shift because the matching was not perfect. Over time, the template would become purely background, making it impossible to recover and leading to significant inaccuracies.

Another approach, similar to the final one, involved generating a reference background by averaging multiple frames throughout the video. The idea was to "erase" the athletes by using frames where the athletes were in different locations. Instead of computing the difference between the current frame and the previous frame, as in the final solution, the current frame was compared to this reference background. However, this approach was not as robust as the final method, leading to more false positives and less stable detection of the athletes.

2.1.3 Results

The results are very satisfactory, with both athletes being well detected. While the athlete further from the camera is not detected in a few frames, the athlete closer to the camera is consistently detected in all frames. However, a minor imperfection is noticeable in the frame shown in the second image below, where the foot of the closer athlete is detected as a separate object. This only happened on this specific frame and is due to the opening operation which was not able to merge the two objects (the body and the foot of the athlete).





This approach is highly efficient, with a computation speed of approximately 150 frames per second. It processes the entire video (about 8 seconds, or 202 frames) in under 2 seconds.

Notably, this performance is achieved on my personal PC, a Windows Surface without a GPU.

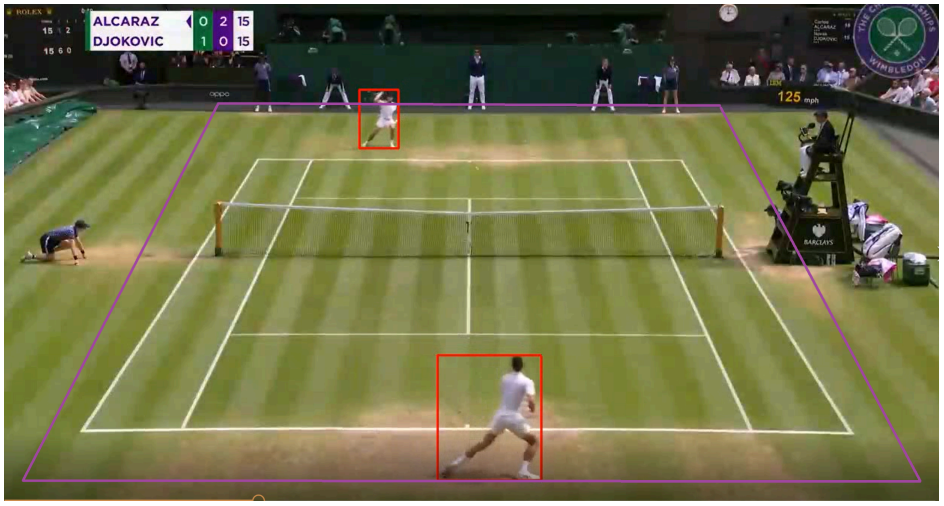
It is worth noting that, although the result is filtered to retain only objects within the court, this step is not essential for this approach. The outcome would have been the same without applying this filter.

2.2 Deep Learning Based Solution

2.2.1 Description

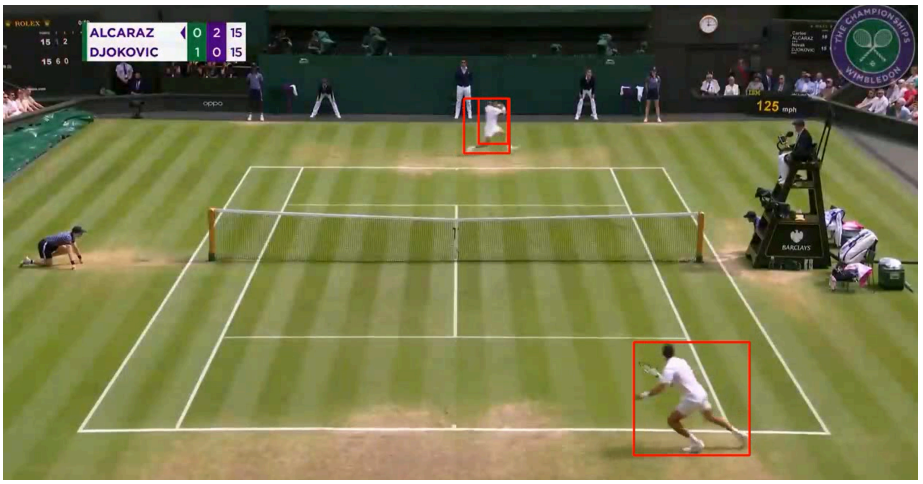
This approach utilizes a pre-trained, state-of-the-art deep learning model for object detection, specifically the YOLOv8 nano model. This lightweight version of YOLOv8 is chosen for its high performance relative to its scale and is well-suited for processing the provided video on a consumer laptop without a GPU.

Using this model, all humans in each frame were detected, including referees and spectators. To filter out these instances, a specific area was defined where athletes are expected to be, namely the tennis court. The filtering was achieved by checking if the centers of the detected bounding boxes fell within a polygon that delineates the boundaries of the tennis court (see purple lines in image below).



2.2.2 Results

This approach offers a stable solution to the human detection problem. The closer athlete was consistently detected in every frame, while the second athlete was detected in almost every frame. Minor imperfections are evident in a few frames, as shown in the image below, where the athlete is detected twice. To address this issue, a filter could be implemented to remove one of the overlapping bounding boxes.



Filtering out the referees and the spectator worked perfectly well by defining the area in which the players are expected to be. The average processing speed of this approach is 4.6 frames per second.

3. Comparison and Discussion

Both approaches provide effective human detection, each with its inherent advantages and disadvantages.

The deep learning approach offers an easily implementable, stable, and reliable solution, especially when using a pre-trained model. It yields more accurate human detection than the classical approach. However, the drawbacks include the relatively high computational expense and the design of these models to perform a wide range of tasks. In this case, the model can detect objects other than humans and is not specifically tailored to this use case, introducing performance overhead. This results in slower detection compared to the classical approach and makes it unsuitable for real-time applications in its current state.

The classical approach is highly efficient and can be easily used for real-time applications. However, it has a few disadvantages. First, the initial frame of the video lacks a previous reference, so no detection is performed on it. This could be resolved by using the second frame as a reference, but this solution is not feasible for real-time applications. Second, this approach relies on athletes always moving since it is based on differences between consecutive frames. The reference background approach, which could address this issue, did not work well for this specific case as mentioned above. Third, the classical approach requires manual fine-tuning. The threshold defining whether a pixel difference indicates movement or not and the kernel sizes for the morphological operations had to be manually adjusted and are tailored to this video.

In terms of performance, the classical approach outperforms the deep learning approach (~150FPS vs 4.6FPS). However, regarding accuracy, ease of implementation, and generalizability, the deep learning approach is more advantageous.

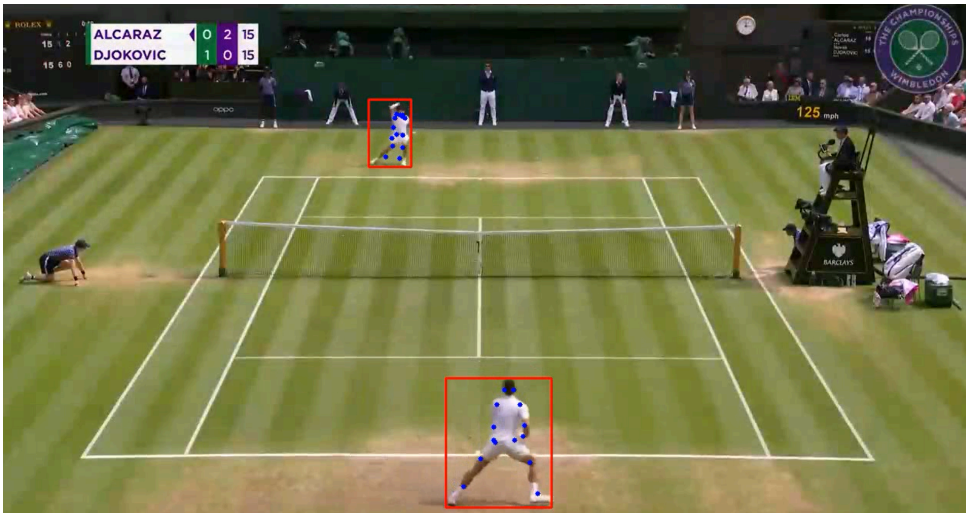
4. Optional Extension

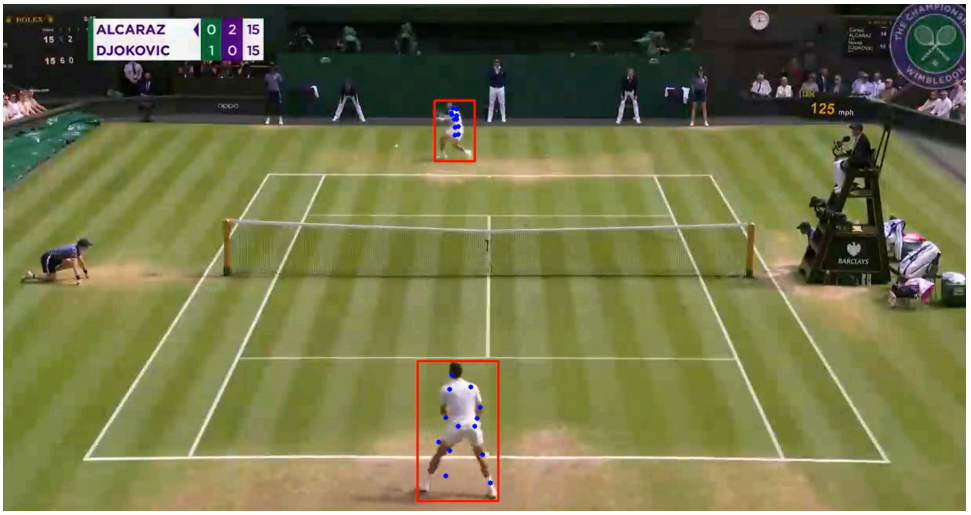
4.1 Additional Task Description

As an additional task, a keypoint estimator was implemented using the pre-trained deep learning nano YOLOv8-pose model. Similar to the other object detection model, this lightweight version was chosen for its high performance relative to its scale and is well-suited for processing the provided video on a consumer laptop without a GPU.

To minimize computation, predictions were performed on the padded bounding boxes retrieved from the human detection task. Using smaller input images for the model reduces the computational load and prevents detecting keypoints of referees and spectators, thus avoiding an additional filtering step. Once predictions were made, the keypoints were mapped back onto the original image space and drawn on the frame for visual feedback.

The predictions are satisfactory but not always very precise. This is assumed to be caused mainly by the low image quality of the patches. Below are two example images, one showing a good prediction (top) and one showing a less accurate prediction (bottom) (bounding boxes were predicted using the deep learning approach).





5. How to Run the Code

The requirements and the instructions on how to use the code can be found in the README.md file present in the project directory.