

INF573 - Report

Image Analysis and Computer Vision



École polytechnique
Academic Year 2023/2024

Prof. Mathieu Bredif

Claudio Arione - claudio.arione@polytechnique.edu
Riccardo Inghilleri - riccardo.inghilleri@polytechnique.edu

Contents

1	Introduction	2
2	Problems and Solutions	2
2.1	Challenge with Mediapipe	2
2.2	Integration with YOLO	2
3	Detailed Methodology	3
3.1	Advanced Player Detection and Estimation	3
3.2	Pose Detector Class	4
3.3	Ball detection evolution	5
3.4	Decision criteria for pausing the video and starting to predict	5
3.5	Data preparation and features extraction	5
3.6	Football Goal Identification	6
3.7	Clean architecture and ease of usage	7
3.7.1	Requirements	7
3.7.2	Running	8
3.8	Shot prediction model	9
3.9	Limitations	9
3.10	Possible improvements and future work	10
4	Results	12

1 Introduction

In the rapidly evolving world of technology, the intersection of computer vision and sports analytics has emerged as an exciting frontier. This crossroad promises not only to enhance our understanding of sports dynamics but also to revolutionize the way we predict and analyze key moments in games. Football's rapid decision-making, capable of altering a match's course, provides an ideal setting for advanced computer vision techniques.

Our project finds its niche in this dynamic environment. The core of our work focuses on the critical moments of a football penalty kick, a scenario where every movement, every decision counts. The primary aim of our project is to take advantage of the power of computer vision to conduct a detailed analysis of the poses of key players, specifically the attacker and the goalkeeper, during a football penalty. Beyond mere analysis, we ambitiously strive to predict the direction of the kick (expressed as a ternary prediction: left, center or right) and of the goalkeeper's dive (as a binary prediction: left or right). Those forecasts (and the final video's analysis itself) are displayed in a clear and visually pleasant manner, in order to be easily comprehensible by the end-users of the application.

Our report includes also a brief analysis on possible improvements and future developments of the application, as well as a step-by-step investigation on the creative process and the comparison with other possible solutions that led us to choose one path instead of another.

2 Problems and Solutions

2.1 Challenge with Mediapipe

To perform pose estimation, we chose Mediapipe created by Google because it is currently the most accurate and efficient technology with as many as 33 keypoints per person (and 3 keypoints in each foot), as opposed to other models such as the object detection's state-of-the-art YOLOv8 by Ultralytics, that have just 18 keypoints and only 1 keypoint in each foot. However, Mediapipe, while an advanced pose estimation tool, posed a significant challenge for our project because it was designed exclusively for single-person pose estimation. This limitation became evident when it came to a dynamic soccer scenario in which multiple players are simultaneously in action. Our project needed a solution that could accurately analyze the poses of both the attacker and the goalkeeper in the same frame, a real necessity for a comprehensive penalty kick analysis.

To deal with this challenge, we adopted the **divide et conquer** paradigm, dividing the main problem into two different subproblems, to be tackled one after the other. The details will be explained in the following section.

2.2 Integration with YOLO

To address the multi person pose estimation challenge, we turned to YOLO (You Only Look Once), a state-of-the-art object detection system. By integrating Mediapipe with YOLO, we developed a novel approach to extend Mediapipe's single-person pose estimation capability to a multi-person context. This integration involved first using YOLO to detect the positions (boxes) of the attacker and the goalkeeper as separate entities within each video frame. By isolating these entities, we could then apply an independent Mediapipe's pose estimation instance to each player. This creative workaround not only

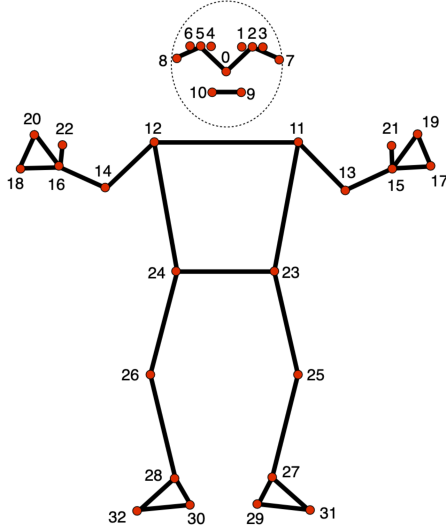


Figure 1: Mediapipe: keypoints

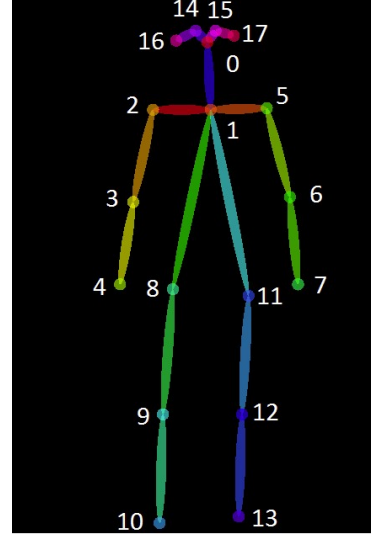


Figure 2: YOLO: keypoints

solved the problem at hand but also opened up new possibilities for pose estimation in complex, multi-person scenarios.

While seeming a quite straightforward solution, the implementation is a bit tricky and on the Internet there are actually no repositories implementing this workaround. Therefore, it's possible that we decide to publish it as a Python package, to help the community building similar project while waiting for MediaPipe to support multiple pose estimation. Of course the estimation of the keypoints is not as accurate as it would be using a model tailored for two persons and has problems when the identified person's bounding box is too small, but it's a nice workaround which correctly manages to solve an open problem.

3 Detailed Methodology

3.1 Advanced Player Detection and Estimation

The development of our system needed meticulous planning and a multi-faceted approach. Initially, we employed YOLO's robust object detection algorithms, renowned for their precision and speed, to identify the players and the soccer ball within video frames. This preliminary step was crucial, establishing the foundation for all subsequent analyses.

We decided to use YOLOv4 model, which is the newest YOLO model for which we were able to find the pretrained .weights and .cfg files online. In fact, successive versions of YOLO model are owned and deployed by Ultralytics, which makes them available open-source only in the sandbox of their ecosystem, and therefore not possible to integrate in medium-complexity projects like ours, which requires more than simple calls to the built-in functions.

We opted for the tiny version of the model because it represents a good compromise among having a good accuracy in detecting objects and the time used to process each frame, since both the accuracy and the time grows proportionally to the increase in size of the weights file.

To facilitate player identification, we devised a method that leverages the historical data of previously detected positions. When new frames are processed, our algorithm

evaluates the spatial proximity of the new detections to the known locations of the attacker and goalkeeper.

The specific detection of attacker and goalkeeper involves the subsequent steps.

1. Feed the YOLO model with the whole frame, making it return the coordinates of all and only the boxes whose content is established (by the algorithm) to be a person with a confidence higher than an arbitrary threshold.
2. Perform a non-maxima suppression on the boxes returned by the above algorithm, i.e., reduce multiple overlapping bounding boxes around the same object to a single bounding box with higher confidence. This helps to eliminate redundancy - and consequently to increase accuracy - in the detection results.
3. Among the detected boxes, identify all the ones whose position is "compatible" with the previous positions of either attacker and goalkeeper. This involves computing the distance of each new box from the center of the box containing the attacker in the previous frame. If the distance is below a set threshold, the box is included in the list of compatible ones. Of course, the same procedure applies to the goalkeeper.
4. Between the boxes in the - typically very short - list of boxes similar to the previous attacker, the biggest one is selected. Again, the same happens to the goalkeeper.

Note that this last filtering is often not needed, because the identification of similar boxes which takes place at the previous step really often leaves only one compatible box for the goalkeeper and one for the attacker. This last operation, however, can be useful to better handle relevant players' identification in case with "noisy" images (i.e., images with many other players).

Upon successful detection of the players, we deployed Mediapipe's pose estimation technology on the isolated sections of the image. This process involved a series of optimizations and adaptations, which were essential to ensure that Mediapipe performed effectively on the cropped images of each individual player.

A significant aspect of our approach was the extraction of key points from the attacker's pose. These key points were instrumental as they formed the primary dataset for training our XGBoost model, whose details will be given in a later paragraph. In summary, for each frame of the video all the keypoints of both the attacker and the goalkeeper are saved; the relevant ones are then passed to the classification model to compute a prediction.

Finally, alongside player estimation, the YOLO model played a crucial role in identifying the soccer ball. Its precision in determining the ball's position was integral, providing information needed to determine the exact moment in which the foot of the player impacts the ball, stopping the video at the right time.

3.2 Pose Detector Class

For each detected player, we execute a targeted pose estimation using a PoseDetector class.

The PoseDetector class, crafted using MediaPipe, showcased our commitment to capturing players' poses with high fidelity. It was designed with adjustable parameters to adapt to various image conditions and tracking requirements. The result was a quite detailed pose estimation that provided a wealth of spatial information about the players'

postures and movements. The class draws the detected pose landmarks and connections for visual analysis, if necessary (so before the attacker kicks the ball).

This class is fine-tuned to analyze the specific region of the image where the player is located. By cropping the original image around the player's bounding box, we reduce the computational load and increase the accuracy of the pose estimation.

3.3 Ball detection evolution

Our journey to implement ball detection within the system underwent several iterations. Initially, we experimented with a method that converted the image to grayscale and applied Gaussian blur to mitigate noise, setting the stage for the Hough Circle Transform to detect circles. While this technique demonstrated success in locating the ball within specific footage, it faltered across different videos due to the sensitivity of the Hough Circle parameters, which varied significantly from one video to another.

Subsequently, we explored color-based detection, operating under the premise that the ball's white color could be isolated through fine-tuning HSV values. This approach, however, was prone to overfitting; minor variations in the image, such as changes in lighting or size, resulted in the HSV values becoming unreliable.

The solution came with our decision to leverage YOLO, already proven effective in identifying players. By extending YOLO's object detection capabilities to recognize the soccer ball, we achieved a robust solution that consistently detected the ball across various videos. This method's adaptability and accuracy cemented it as the preferred approach, aligning with our system's overall methodology and ensuring consistent performance in ball detection tasks.

Also in terms of efficiency, this technique was the preferred one because it leverages the list of bounding boxes obtained in output from the convolutional neural network fed with the original image, so it does not require any additional step or image manipulation. The assumption we made is that the ball that the attacker is about to kick is the biggest soccer ball in the image. This is of course a more than reasonable assumption, because in a football matches there's only one ball on the field and, in general, it's almost impossible to see other big balls in places like the advertisement banners or the stands.

3.4 Decision criteria for pausing the video and starting to predict

A critical element of our system is the decision-making logic that determines when to halt the video. This is established by monitoring the proximity of the attacker's foot to the ball. If the euclidean distance between the keypoints of one the two feet (to account for both right-footed and left-footed players) and the ball falls below a specified pixel threshold, we infer that the ball is about to be hit, prompting us to initiate the predictive analysis and showing its results pausing the video.

3.5 Data preparation and features extraction

In preparation for model training, we extract and preprocess pose landmark features from the frames leading up to the kick. These landmarks, particularly from the attacker's feet, provide a rich dataset that encapsulates the dynamics of the run-up and the kick itself.

The data extracted from these critical moments are compiled into a feature array, which will later be used to train our predictive model.

We decided to use the position of all the parts of the body (and not only of the feet) because we believe that the whole posture of a football player before a kick can be indicative of the future direction of the shot. As already mentioned, this richness of information is essentially the reason why we chose to adopt the YOLO + Mediapipe approach: having a lot of keypoints for each player helps the model to be more accurate and more resistant to overfitting.

3.6 Football Goal Identification

A crucial component of our football penalty analysis was the accurate identification of the goalposts and the subsequent zoning of the goal area to predict shot placement. This process defines the spatial context in which the prediction model's output is visualized.

To achieve this, we developed an algorithm that employs several computer vision techniques. Our approach involved the following steps:

1. **Image Preprocessing:** We began by converting the original image to grayscale, in order to simplify the subsequent operations by reducing the data to a single intensity channel. Afterwards, we applied a Gaussian blur, which helps to reduce image noise and improves the edge detection performance in the following steps.
2. **Edge Detection:** The Canny edge detector was applied to the preprocessed image to identify the edges and the outline of objects within the scene.
3. **Line Detection:** With edges identified, we utilized the Hough Lines Transform to detect lines in the image.

We focused on detecting long vertical lines, as these are indicative of the goalposts' presence in the frame: for this reason we basically imposed a condition on the minimum angular coefficient that a line should have in order to be considered "vertical". Note that we didn't require the coefficient to be "exactly ∞ " - i.e., that the x coordinates of the starting and the ending point of the line were equal - in order to tolerate a slightly inaccurate perspective.

What is the Hough Lines Transform algorithm? The Hough Lines Transform is a technique used in computer vision and image processing to detect straight lines in an image. It's particularly useful for identifying lines even in the presence of noise, breaks, or gaps in the lines.

The Hough Lines Transform works in a parameter space defined by two parameters: ρ and θ . ρ represents the distance from the origin to the closest point on the line, and θ represents the angle between the x-axis and a line perpendicular to the detected line.

An accumulator array (or Hough space) is created, where each cell corresponds to a possible line in the image. For each edge point in the image, vote for possible lines in the Hough space it could belong to. After the voting process, a threshold is applied to the accumulator array. Cells in the array that exceed the threshold represent lines in the image.

4. **Goalposts Estimation:** Our algorithm then filtered these lines to estimate the positions of the left and right goalposts. We achieved this by setting up acceptability boundaries and iteratively comparing the detected lines with these estimated goalpost positions. The process involved calculating the distance between the lines and the estimated posts, and dynamically updating the estimates (as well as decreasing the acceptability threshold for future "candidates") as closer matches were found.
5. **Goal Area Zoning:** Once the goalposts were identified, we divided the goal area into three zones: left, center, and right. This division is fundamental for the shot prediction model, allowing us to assign probabilities to each zone, indicating where the ball is most likely to be kicked.
6. **Visualization:** Finally, we visually represented the model's predictions by drawing shaded rectangles within the goal area, each corresponding to one of the three zones. The opacity of each zone is proportional to the probability of shooting in that section. This not only helps in the interpretation of the data but also enhances the presentation for potential end-users, such as coaches or analysts.

The visualization, as seen in Figure 3, depicts a color-coded goal with percentages indicating the probability of a shot placement in each section.

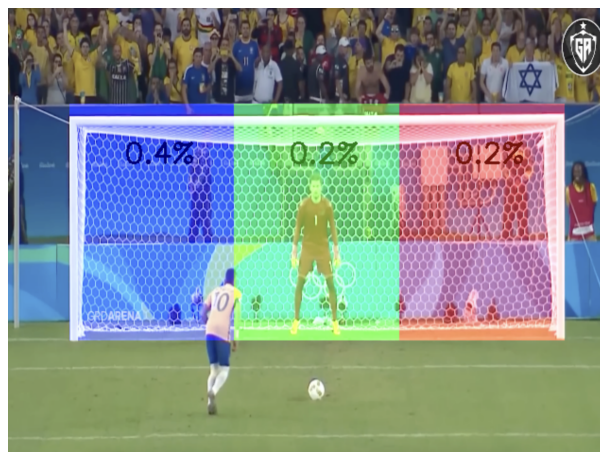


Figure 3: Football goal identification

3.7 Clean architecture and ease of usage


3.7.1 Requirements

1. Python 3.*

You can check if you have already installed python by opening the terminal `>_` and using the following command:

```
python --version
```

If you don't already have it, you can install it using the following linked guide [\[1\]](#).

2. Go to our github repository , clone it and go to the folder containing the **requirements.txt** file.

The following command will allow you to install all the needed libraries:

```
pip3 install -r requirements.txt
```

3.7.2 Running

To run the code and utilize the football shot predictor, follow these steps:

- **Running the Code:**

1. Open your terminal or command prompt `>_`.
2. Navigate to the directory where the **main.py** file is located.
3. To run the program, type:

```
python3 main.py --arguments*
```

followed by the specific **arguments** you want to use.

- **Arguments Explained:**

1. Use the following flag without additional arguments to train the model using the default video list:

```
-t or --train
```

2. Use the following flag without additional arguments to predict outcomes using the default video list:

```
-p or --predict
```

3. Use the following argument with the path to a file containing your custom list of videos for training or prediction:

```
-f or --filepath
```

4. Provide the path to the folder containing numpy arrays from previous training if you want to use them for prediction:


```
--training_data_load
```

5. Specify the folder in which to save new training data arrays:

```
--training_data_save
```

- **Clean Architecture:** The code is structured following the principles of clean architecture, with distinct separation between the prediction logic and the training logic.

- **Modularity:** Each function in *predict.py* and *training.py* handles a specific task, allowing for independent prediction and training.
- **Reusability:** The code is designed so that the training and prediction components can be reused or replaced without affecting the other parts of the system.

Remember, you cannot use both `-t` and `-p` simultaneously. Choose one mode of operation at a time. For further information and updates, refer to our GitHub repository  mentioned.

3.8 Shot prediction model

In our football analytics project, the chosen machine learning model for predicting the trajectory of the kick and the goalkeeper's dive is XGBoost. This choice was induced by XGBoost's efficiency in handling a large number of features and its capability for multi-class classification problems.

For the attacker's movements, we employed 330 features derived from the 33 key points across 10 frames preceding the shot, capturing the essential dynamics leading up to the kick. In contrast, for the goalkeeper, we streamlined the features to 165, using the same 33 key points but only from the 5 frames leading up to the dive. This decision is based on the fact that earlier frames do not contribute additional predictive information and can be omitted without loss of accuracy.

In labeling the data, we opt for a binary classification for the goalkeeper's dive direction and a ternary classification for the shot direction. Specifically, the shot can be classified into three categories: left, center, and right. The XGBoost classifier is configured with the `multi:softprob` objective, which suits our multi-class problem by providing a probability distribution over the three possible outcomes for each prediction.

Our implementation includes a bootstrap process to enhance the robustness of the model when the available training data is limited. We add Gaussian noise to existing samples to create additional synthetic training data, ensuring the model has sufficient examples to learn from.

The `XGBoostClassifier` class encapsulates the entire process, from initializing with the data path and number of classes to training the model and evaluating its accuracy. It also includes a prediction function that outputs the probabilities for each class, providing a nuanced forecast of the kick's likely direction.

3.9 Limitations

The proposed solutions has several limitations, many of which have been partially - but not totally - overcome. Almost the totality of the issues is related to the perspective from which a video is taken.

First of all, the fact that the each frame has to be processed by a neural network slows the video a lot. We decided to resize the image shrinking width and height in order to ease the computational load of the NN processing.

This problem could be partially overcome with the presence of a GPU that enhances and speeds up the computation. In addition, while the language in which the program is written is user-friendly and benefits from a rich library ecosystem, it also encounters speed-related challenges.

The workaround composed by the subsequent application of YOLO and Mediapipe has some problems when either the goalkeeper or the attacker are too small. This happens when the image to process is not big enough or the framing is such that the goalkeeper is too far from the camera objective. Indeed, the YOLO model needs to identify the people with enough accuracy, and afterwards the MediaPipe pose estimation has to be applied; the latter could fail also if the bounding boxes are too small for the algorithm to

produce any reliable results. The ball has the same problem: if the image is too small, the YOLO model is not able to identify it. These problems are evidently unsolvable and state once again that the quality of the outputs of our algorithm depends on the perspective from which the video is recorded. We tried to solve any related problem modifying the program and adding checks and edge cases handling, in order to make the system as resilient to noise as possible, but some of them were almost intractable. This is the reason we embraced this trade-off, i.e., acknowledging that the quality of our application is contingent on the chosen perspective.

Moreover, as for now, the default training set that we have used is still quite small, due to the difficulty in browsing the web and finding many videos recorded from the back. However, the algorithm is perfectly able to deal with an arbitrary dataset (of course matching the requirements) and then to enhance its prediction capabilities.

3.10 Possible improvements and future work

Each point represents a strategic direction that could significantly enhance the system's capabilities, making it an even more valuable tool in the field of sports analytics and strategy.

- **New version of the YOLO model** A more advanced YOLO model could be used to better identify the players even in situations of very noisy frames; assuming that the initial barrier to enter in the market and deploy a new, very advanced, YOLO model is unsurmountable, we would just have to wait that the .cfg and .weights files (or anything else, but similar) of a more advanced YOLO model are released on the market.
- **Leverage GPUs capability** Having a GPU could let us use more accurate version (even) of the same model without losing too much in terms of speed. In fact, while the time needed to process a frame with a standard YOLOv4 model (instead of our YOLOv4-tiny) with a CPU is so high that the video gets extremely slow, having a GPU would dramatically increase the speed of our program's output.
- **Enhanced Pose Estimation Accuracy:** New pose-estimation techniques could be deployed open-source in the following months (or years) and their usage would increase the accuracy of the pose estimation phase, and consequently of the predicted shot and dive probabilities.
- **Enhance Training Data:** Feeding our application with a large amount of videos compatible with the project's specifications would result in more reliable predictions.
- **Real-Time Analysis:** Advancing the system to operate in real-time would allow for immediate tactical insights during live matches, offering coaches and analysts a powerful analytical tool.
- **Goalkeeper Strategy Prediction:** Future versions could delve deeper into goalkeeper behaviors, analyzing patterns and predicting their strategic decisions.
- **User Interface Improvements:** Developing a more intuitive and interactive user interface would make the system more accessible to a broader range of users, enhancing usability and engagement.

- **Machine Learning Model Improvement:** Collaborating with experts in sports science and coaching could bring our project to a new dimension. In fact, in our model the features can be selected or weighted according to their relative importance.

4 Results

The culmination of these efforts resulted in a sophisticated system that could effectively analyze and predict events during a football penalty kick. This project not only addressed the initial challenge posed by the limitations of Mediapipe but also showcased the power of combining different computer vision technologies. The insights gained from this project, if further developed, could have significant implications for sports analytics, particularly in the strategic analysis of football games and the training of players. Our approach demonstrates how technology can be innovatively adapted to meet specific challenges, paving the way for future advancements in the field.