Final Report for Syslab
Automated VAR Project

Saurav Banerjee, Amit Krishnaiyer

Due Date: May 23, 2022

Yilmaz - Periods 3 and 4

Table of Contents

Abstract

Offside, classified as a foul made in soccer when the player with the ball makes a pass to a player on his own team that is past the opposing team's last defender at the time the ball is being passed. Offside is one of the most difficult infractions in the game of soccer for refs to make accurate decisions in real time, since there are many instances of "close calls" when a player is marginally past the last defender, and the referee is not able to understand that just from his/her own view of the field, or because of the speed of the game. There are several methods currently towards attempting to solve this problem, most focused on using multiple angles to make decisions of offside quickly Our research looked at using Computer Vision methods such as Image Segmentation, Hough Line Detection, and K-Means clusters to expedite the process of automating just from one camera angle.

I. Introduction

Our research made an attempt to solve the issue of needing multiple cameras and angles in order to make a split decision on offsides. Currently the issue with VAR is that the process is time consuming, and needs specialized trained referees to be able to make decisions on offsides. VAR needs to be accurate, since whether a passage of play is offside or not is extremely important towards professional soccer, as incorrect decisions can lead to the loss of a critical game, and therefore a wrongful loss in a knockout tournament such as the Champions League, or World Cup.

While there are current solutions out there, our method was different because we were attempting to use only the broadcast view from the TV cameras. Tracking player positions in an exact fashion becomes much more difficult with the removal of multiple cameras, and coupled with the cost, is why VAR cannot be taken to lower leagues in soccer. The combination of using mp4 input and just using one angle are novelties to what has been done to replicate VAR for offside detection.

II. Background

There are several solutions that make an attempt to solve the problem of detecting offsides accurately through computer input.

One of these solutions is *Automated Offside Detection by Spatio-Temporal Analysis of Football Videos (*Uchida et al.). This paper specifically looked at input from multiple cameras, using them to paint an entire picture of the field, which helped track positions of the players and the ball, with ease. However, this is complicated since it required the usage of multiple cameras, and our project attempted to use one camera view only.

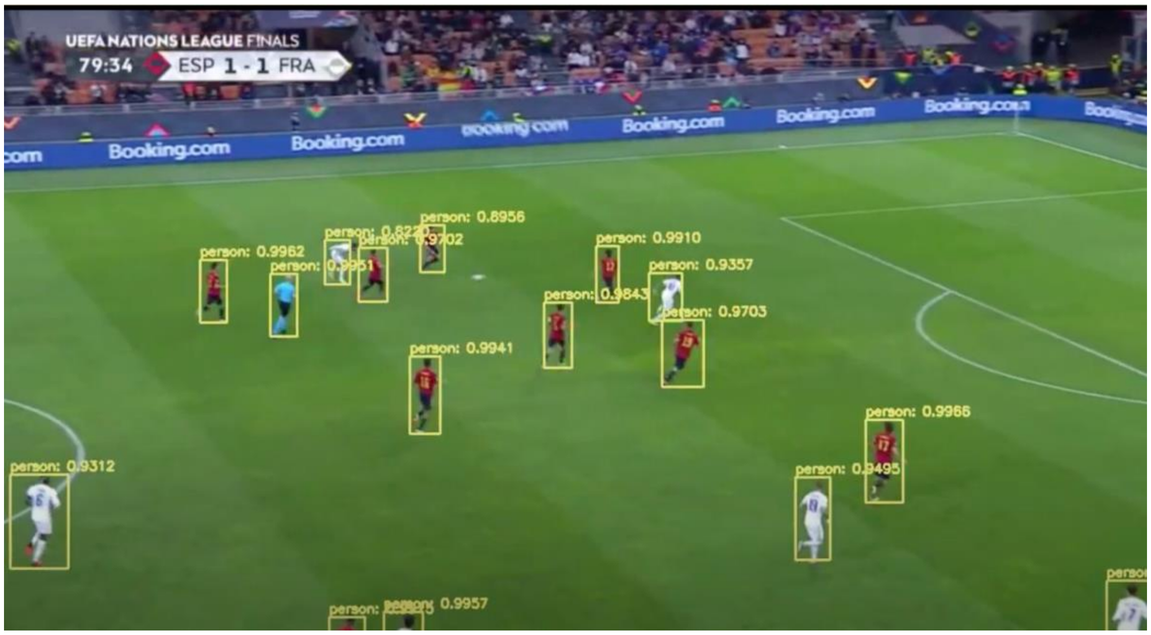
Another solution is *A Dataset & Methodology for Computer Vision based Offside Detection in Soccer* (Panse & Mahabaleshwarkar). This study used vanishing points and lines to compare positions of players to detect which players were offsides at the time. However, this method only used image input, which meant that humans had to input the right frame for when the ball was being passed. This limited the program when compared to our input, which was able to detect when the ball was being passed during the video which was inputted.

III. Applications:

The offside detector which we created can be applied to real life situations in soccer. Since it is based on input from just one camera angle, the detector could be applied to Lower Level leagues in the world that can't afford to use the higher-grade technology and cameras that leagues like the Premier League use. Our AutomatedVAR system can be applied to stadiums with clear boundary lines, from the TJ Soccer field, to fields in the 7th tier of the English Football Pyramid. This allows for cheaper usage without specialized referees who are the ones who are trained to draw the lines in the VAR room.

IV. Methods:

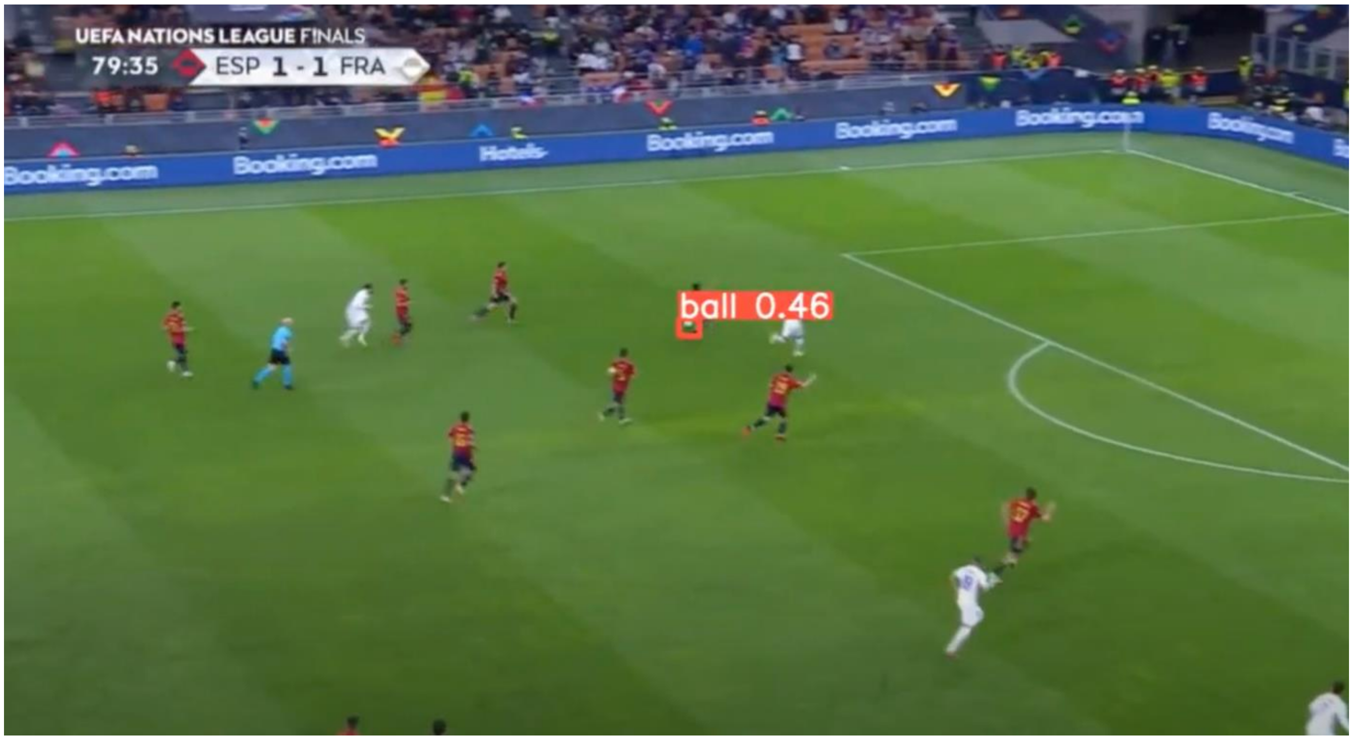
The input from our system is just an MP4 file with a video (TV View) of a certain passage of play taken from real-life soccer games. From there our first goal was to be able to track the positions of the players on the video in real time. To do this, we used YOLOv5 (Jocher et al.) and weights from Microsoft's COCO (Lin et al.) Dataset (Common Objects in Context) to be able to determine where the players were. We used COCO's weights for humans, and were able to successfully detect where in the picture the players were. For easier use for humans, we drew bounding boxes to show the results of the object detection.



However we also needed to track the position of the ball in order to detect who the player making the pass was, and when the ball left the players' feet. To do this we used YOLOv5 (Jocher et al.) as well, however we needed to train weights to do this task, as we couldn't find a dataset which did so. We trained our weights using YOLOv5, and created our own set of image annotations which we trained our ball tracking weights against.

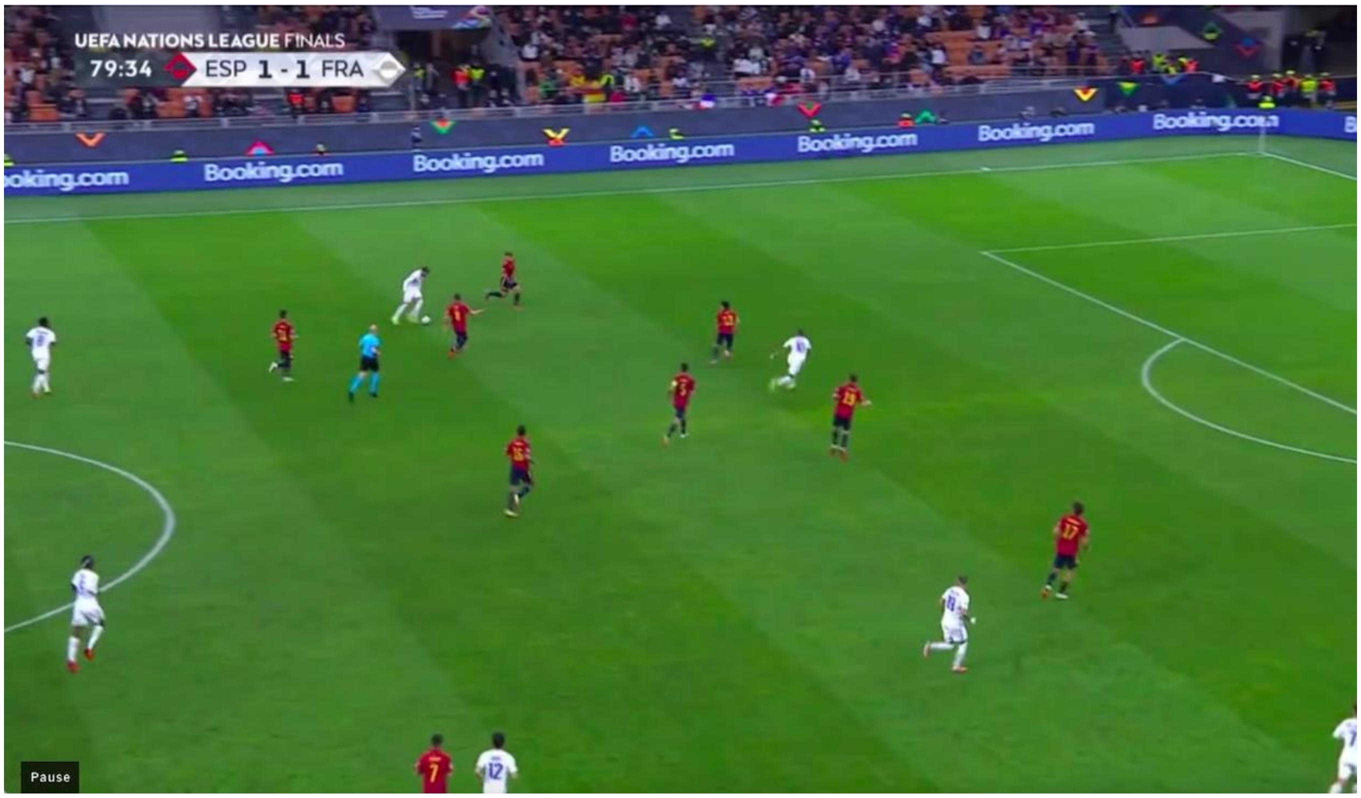
However it is important to note that tracking the ball was one of the more challenging parts of the project. Our first attempt at training weights to track the position of the ball was relatively inaccurate before we were able to fix it. We attribute it to not having a large enough training set, which we fixed by changing the size of the training set from around 400 to 800 on the second try.

The second training set we used, and the weights we trained on it gave us a much more accurate way to track the position of the ball.

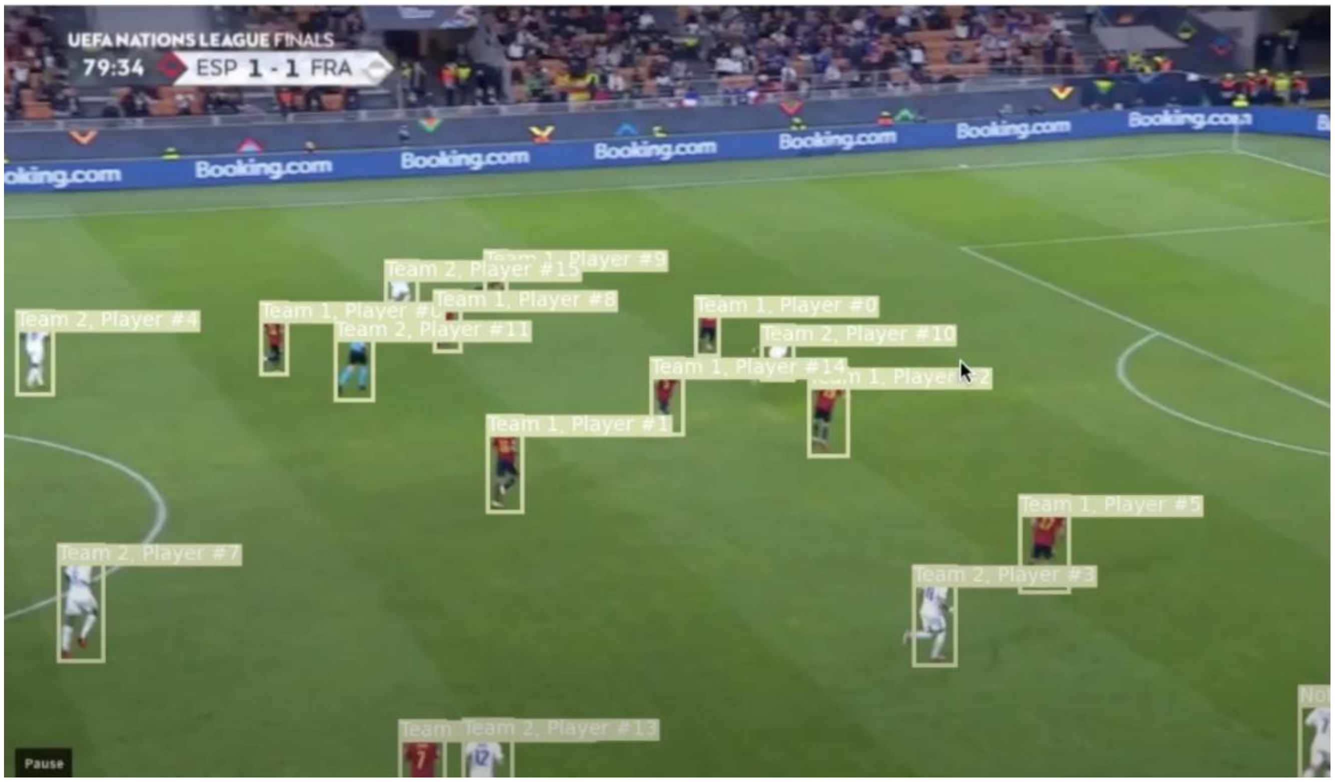


We were able to track the ball, albeit at lower confidence levels than humans. However this is relatively normal, since the ball is a small and fast moving object, and therefore much harder to track in the way YOLOv5 uses.

Given that we had the positions of the ball and players now, we could find a way to be able to tell when the ball is being passed. Our program went through the video, and using the positions of the players and the ball, and comparing between them, we could tell when the ball was in possession of the players or not. Given this information, we looked for the moment before the distance between the ball and the player closest to the ball suddenly changed, and that would be the frame that the ball would be passed at.

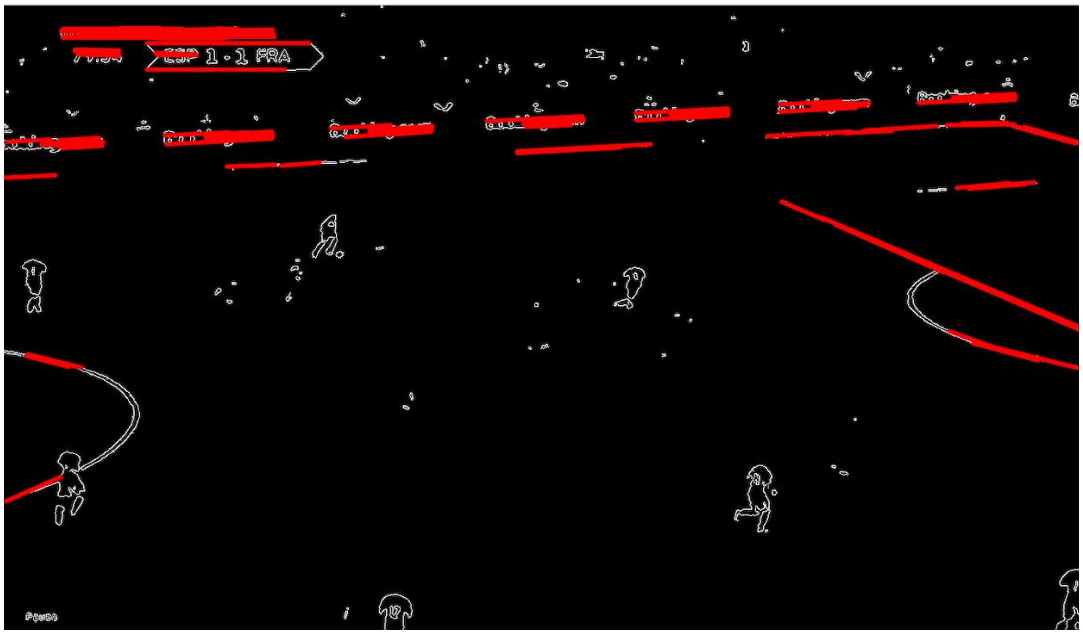


Given that we had the frame of when the ball was being passed, and the player making the pass, we needed to determine which players were on the defending team. We further separated the players into the two separate teams using K-means (OpenCV), a function built into the OpenCV module, which took the mean value of the pixels of each player and used that to figure out what team they were on. The K-means algorithm looked at the pixels in the bounding boxes of the players, then averaged them, determining the two teams based on these averages, meaning we could distinguish between the players on the different teams and label them accordingly.

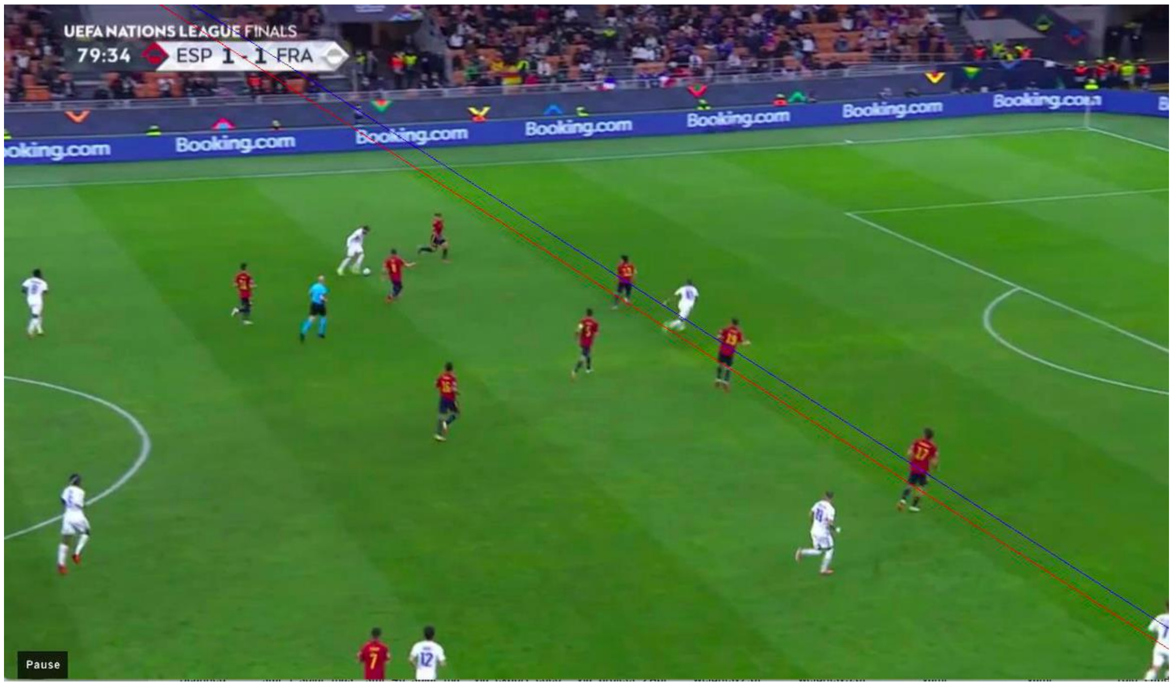


Given what teams each player was on, we could now determine where the last defender was. Our programs had the user input which player was receiving the ball. This gave us which team was attacking, which team was defending, and where the player receiving the ball was. From this, we had enough information to draw lines parallel to the players.

To draw lines parallel to the players, we first needed to segment the images using OpenCV.



Given that we now had a segmented version of the image, we could use Hough Line Detection (OpenCV) from the OpenCV library on the image. That would give us the values of either where the goal line was, or the 18 yard line. This meant that we could find draw lines to where the players were, since we had the other lines of the 18 yard box and the goal line.



Given the lines of where the player is, and where the last defender is, it is easy for humans to deduce where or not the pass was made while the player receiving the ball was in an offsides position

V. **Results**:

Our offside detector had some clear results at being able to detect offsides. While working for some cases for offsides, such as the one demonstrated in the *Methods* section, it did not work for all cases, and also took a large amount of time to run.

VI. **Limitations**:

Several limitations appeared while doing our project. The most alarming one we crossed was the amount of time that our project needed in order to make a decision on play. The program altogether took 10 minutes, which is way too long to make a decision on play without interrupting the flow of the game, since large breaks are part of the limitations of VAR. However much of the runtime was taken by the K-means process needed to differentiate between players of opposing teams, which meant the problem could be solved if we knew the colors of the players before we started the program. Additionally, one more issue that arose was that we needed either the goal line or the line from the 18 yard box for Hough Line Detection (OpenCV) to work.

VII. **Conclusion**:

Although it can't be applied to every image, our algorithm is one of the first that is able to make decisions on offsides from just the broadcast view video. This is a proof of concept that VAR can become semi-automated and that this process of making a decision on offsides does not need to be left to the human referees to make. With future work, this could at some point be implemented commercially, cutting technology costs for an entire VAR room.

VIII. **Future Work**:

There is still much room for improvement that future work could possibly address. One of these things is possible improvements is the run time of the program. One step that could be taken to significantly improve the run time is finding a way to replace the K-means section of the project. This would mean more user input into which colors the teams were wearing, but would take much less time, since the algorithm would no longer need the processing time used to determine the teams.

Additionally, more work could be done into converting the entire project into a GUI in which users could drag and drop videos in, and then later have it go through processing. This would result in a much cleaner interface for users, and give users not familiar with the code and command line input a much better chance at being able to use the software.

References:

Video Presentation: https://www.youtube.com/watch?v=HZvvLW67gfk&feature=youtu.be

Automated Offside Detection by Spatio-Temporal Analysis of Football Videos

Uchida, I., Scott, A., Shishido, H., & Kameda, Y. (2021). Automated Offside Detection by Spatio-Temporal Analysis of Football Videos. Proceedings of the 4th International Workshop on Multimedia Content Analysis in Sports.

A Dataset & Methodology for Computer Vision based Offside Detection in Soccer

Panse, N., & Mahabaleshwarkar, A. (2020). A Dataset & Methodology for Computer Vision based Offside Detection in Soccer. Proceedings of the 3rd International Workshop on Multimedia Content Analysis in Sports.

YOLOv5

Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Jiacong Fang, imyhxy, Kalen Michael, Lorna, Abhiram V, Diego Montes, Jebastin Nadar, Laughing, tkianai, yxNONG, Piotr Skalski, Zhiqiang Wang, … Mai Thanh Minh. (2022). ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference (v6.1). Zenodo.

OpenCV

OpenCV. (2015). Open Source Computer Vision Library.

Microsoft COCO: Common Objects in Context

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L. & Dollár, P. (2014). Microsoft COCO: Common Objects in Context

APPENDIX:

I. **CODE:**
https://drive.google.com/file/d/1fgqpaV5tStBDrChuuso1nBySLasWlUBc/view?usp=sharing

II. **Github Repository:**
https://github.com/Ameat77/AutomatedVAR