JuggleTrack: Enhancing Soccer Skill Development Through Advanced Juggling Analysis

Kevin Fernandez, Marco Flores, Ruofan Liu, John Mejia
The University of Texas at Austin
Address
{kevinfdz, marco-a-flores, rliu17, johnmejia223}@utexas.edu

Abstract

In the realm of soccer skills, juggling is a fundamental technique that develops a player's control of the ball and enhances their overall abilities. This project seeks to create a tool to help soccer players improve their juggling ability. Users can track their progress in the fundamental soccer skill by filming themselves as the tool will measure both the number of touches and the duration of time the ball is kept off the ground. The system will utilize computer vision technology to track the ball and user as they perform the juggling. The project aims to help users grow their real-life soccer skills in a fun and competitive environment. In future works, we hope to expand the depth of this project by providing the users with feedback on how they can improve their juggling skills.

1. Introduction

Soccer is the most popular sport in the world with an estimated billions of fans. Its simplistic yet deep nature captivates people worldwide as almost anybody can play soccer, yet professionals are able to perform breathtaking and dazzling feats. Juggling is one of the most well known skills in soccer; one can often see both professional and casual players juggling. This is for good reason as improvement in juggling can offer improvement in a wide range of situations and skills such as first touch, receiving an airborne pass and passing/shooting an airborne ball in one touch. Furthermore, juggling is a skill that players can easily practice and improve on their own. For these reasons, juggling is one of the first things that players try to practice and improve in.

Recognizing the importance of juggling in the development of soccer ability, our tool seeks to improve the experience of practicing juggling. Our solution uses a combination of object detection and pose estimation to track and keep records of a user's juggling ability in multiple ways. In a given juggling attempt, our tool will count the number of successful touches in said juggling attempt and the duration of time that the ball is kept off the ground. Users can keep track of these statistics over time in order to track their overall skill in juggling over long periods of time. Furthermore, these tracked statistics give users an avenue to compete with their friends. We believe that this tool will greatly increase the enjoyment of practicing juggling and lead to faster growth in the skill. Furthermore, the creation of this tool opens the door for further developments such as providing feedback to users based on their estimated pose and the tracked ball.

2. Related Works

There is a lack of academic works that have attempted to solve the specific challenge of tracking the number of touches and duration of a soccer juggling attempt. Most projects involving computer vision and soccer involve tracking the ball or players during professional games as seen in [8] and [5]. Although these projects are interesting, they are not exactly helpful as the camera angle of a professional soccer game is far different from the camera angles we are expecting users to use.

However, there are many papers and resources involving object detection and pose estimation which proved useful in the development of our tool.

YOLOv5 [4] is one of the state of the art object detection and location algorithms, and it is trained with sports balls as one of the detected objects. Its lightweight nature is attractive as the closer we can get to real time object tracking, the better.

On the pose estimation side of things, Pinheiro et al. in 2022 used OpenPose [2], a popular open source pose estimation technology, combined with notational analysis in order to analyze penalty kick strategy [3]. Because of their success in using OpenPose to predict goalkeeper strategy, we will examine using OpenPose for pose estimation in order to give better scoring. We believe OpenPose is well

suited for our project because it is markerless and seems to have decent accuracy. Furthermore the paper demonstrates its efficacy, even with a single camera.

3. Methods and Results

3.1. Overview:

In our tool, we combined object detection and pose estimation in order to track the number of touches in a juggling attempt and the elapsed duration. For object detection, we used YOLOv5 for its state of the art performance and speed. For pose estimation, we tested both Mediapipe and OpenPose. We found that MediaPipe was unable to give us the accuracy necessary for our use case and that OpenPose was too slow to operate in real time. We track the number of touches by tracking the y-position of the ball which led to mixed results.

3.2. YOLOV5 (John, Ruofan)

3.2.1 Description

YOLO is an object detection approach proposed by Redmon et al. [6]. Contrary to popular approaches at the time which perform object detection through classification, YOLO approaches object detection as a regression problem of bounding boxes. YOLO involves a single Convolutional Neural Network which outputs both bounding boxes and class probabilities in a single pass on a given image.

We use a pretrained YOLO model, specifically YOLOv5s [4], a lightweight version of the fifth edition of YOLO models. This version was stated online as achieving good performance and being extremely fast. YOLOv5 has varying models that we were able to choose from, YOLOv5s and YOLOv5x, each having their own separate advantages and disadvantages. Ultimately we decided to incorporate YOLOv5s as it was a smaller model with already pretrained to detect soccer balls while YOLOv5x was a larger model which slowed down the process of detecting the object. Knowing that it was already pretrained with detecting soccer balls, it was the best decision to proceed with creating our juggling detection method. In figure 1, we can see how the pretrained model out of the box is able to easily identify people and a soccer ball and return high confidence values.

3.2.2 Results

YOLOv5 is able to detect the ball in most circumstances. Furthermore, since it returns a confidence value we are able to tweak the threshold for when a ball is detected and located. YOLOv5 worked better when it was used with video input as the source of the video. It was slower to output the final video image, but it was able to detect each frame with high accuracy if there was a ball in the frame. YOLOv5

Figure 1. Pretrained YOLOv5 Detecting People and Soccer Ball

Figure 2. YOLOV5 Failing to Detect Moving Ball

was not as successful in detecting the soccer ball using a live web camera. The soccer ball is a fast moving object which caused it to not be detectable in most cases in which it was moving. Using a live web camera would not be useful for our purpose since we require a model that is able to display and detect objects rapidly.

In certain cases, it runs into trouble. Specifically, when the ball is moving at a high speed and thus deformed in the camera's view, YOLOv5 can run into trouble detecting it. In Figure 2, we can see that blurring due to the high speed of the ball makes it so that the YOLOv5 model is unable to detect and locate it properly.

3.3. MediaPipe (Marco)

3.3.1 Description (Ruofan)

MediaPipe is a library developed by Google [1] designed to give people realtime on-device machine learning. Their models include a pose landmark detection model which can be used for pose estimation. It involves a Convolutional Neural Network and has an architecture similar to MobileNetV2 [7]. This means that its architecture is lightweight and based on inverted residual structure.

MediaPipe was desirable to us because its emphasis on performing well realtime was very attractive, and we believed that the lowering of accuracy would not be a huge hindrance as we were not doing anything intensive with the pose estimation.

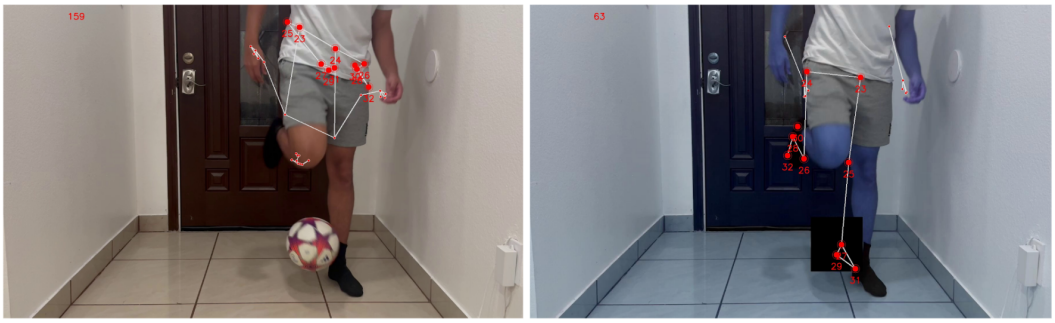

Figure 3. Mediapipe Struggles When the Ball is in Front of the Player. Can Sometimes be Solved with Preprocessing

3.3.2 Results

This method did show some results with the sample juggling videos with a caveat. However, the angle at which the camera would record the scene would affect the accuracy of Mediapipe pose estimation. The issue seemed to be related with when the ball would be directly in front of the juggler causing the estimation to be confused with creating the frame with the ball and player (see Figure **??**). A simple solution would be recording at a certain angle, but this limits the robustness of the tool as it would force users to accommodate the tool which would make them much less likely to use it.

Besides that, we also tried some more preprocessing experiments before executing the pose estimation of the player. This was mainly targeting reducing the influence that the ball might have on the method from blurring to completely masking the ball from the image.

From the results of the added preprocessing, the best results are seen from figure 3 and the Mediapipe pose estimation works as intended when comparing the images to each other. However with masking out the ball, there are cases where the pose estimation lags behind with the original unmasked image because the preprocessing can also corrupt information that is needed for Mediapipe. With new issues arising with experimenting and added time needed to process the pose it did not seem worth using Mediapipe if added delay and inaccuracy doesn't meet what we are looking for time wise and viability of information retrieved.

3.4. OpenPose (Ruofan):

3.4.1 Description:

OpenPose is an open source library developed by Cao et al. [2] for realtime Multi-Person 2D pose estimation. It uses a multi-part CNN to predict Part Affinity Fields to associate body parts with people in an image. It was used in [3] for the estimation of goalkeepers and penalty shooters to decent effect, so we decided to test it for our own use case.

We attempted to use an OpenPose model trained on the COCO dataset, specifically the pose_iter_440000 model.

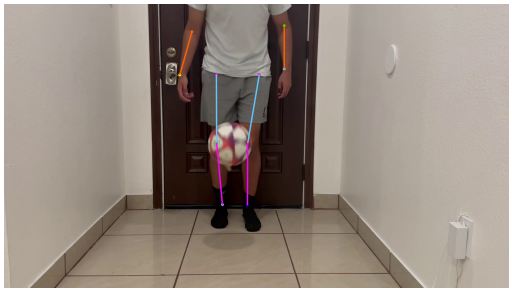

Figure 4. OpenPose Pose Estimation With an Obstruction

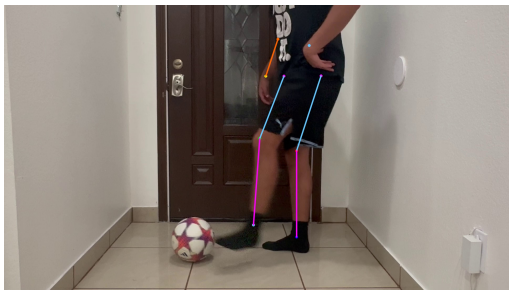

Figure 5. OpenPose Pose Estimation From a Sideways Angle

3.4.2 Results:

The results with OpenPose proved to be far superior than that with MediaPipe. The model is able to consistently estimate the joint locations and angles in the lower body, even when the ball is blocking part of the body (see figure 4 or at a side angle (see figure 5. This is a big contrast to our attempts at using MediaPipe which struggled to accurately measure joint positions and angles, even from a consistent angle with no obstructions.

However, the issue with OpenPose lies in its time to run. An average forward pass of the OpenPose model takes about 4 seconds, which is far too slow for any realtime system. This may be due to hardware limitations, but it is still an important consideration as we do not want the tool to be hardware limited. We did not have a chance to explore

many of the OpenPose models, so in the future we should look for lighterweight implementations in hopes of getting closer to

3.5. Object Tracking baesd Juggling Detection (Kevin, John):

3.5.1 Description

The method we used to detect a juggle of the ball is reliant on being able to consistently keep track of the position of the center of the ball. Using our selected ball-detection method, we isolated the point in the center of the frame which encompassed the ball, and this point would also represent the center of the ball. Once we have this point's position, we keep track of its position over time by storing its position in each frame in an array.

Once we have the array of the position of the ball's center, we look for any noticeable changes in its position on the y-axis. Since we know that in a successful juggle the ball would be falling down and then be kicked back up, we can look for the point in which the center's y-axis position goes from decreasing each frame to when it begins to increase. While in theory we could just look at the y-position in one frame and compare it to the one in the immediate next frame, this would not be as accurate since occasionally the frame that is used to calculate the ball's center shifts very briefly against how the ball is moving. This means that sometimes while the ball is falling, one of the positions saved in the array could be greater than the previous and following positions despite the ball still being in the middle of falling. In order to account for this, we take a look at the seven most recent y-axis positions saved in the array. From these seven, we check to see if the one in the middle (the fourth position of the seven) is the lowest value of them all. If so, then we take this to mean that the ball went from falling in the previous three frames to rising back up in the next three. This would indicate that the ball was juggled/kicked up again. Once this is detected, we increment a variable by one, which keeps track of the amount of times the ball has been juggled.

3.5.2 Results

Once we refined the method we had to test with different video inputs and test the yolov5 model to see the accuracy of the program we created. Figure 6 depicts an illustration of the program running with a live webcam. The benefits of having this run in a live webcam was so that we could eventually just need one single camera and have the program run through that and have live instant feedback. This makes it so that there can be a way to maintain scores, keep track of progress, and minimizes the amount of work needed to get the results of the program we created. The problem with

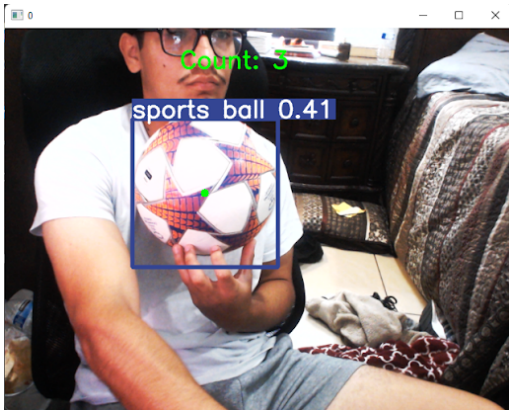

Figure 6. Live Webcam with Juggling Detection

this was that since YOLOv5 looks at every frame individually, thus to get the highest precision it has to look at every frame, but the program is not fast enough to detect the sports ball every single frame. Since it skips frames in detecting the ball, it does not have a high accuracy when detecting juggles as it has an aliasing effect where the middle of the ball appears to be in different positions once it gets detected again. For instances where the ball was manually moved up and down in the camera (see figure 6) it was erroneously counting those as juggles since it fell under the same constraint a juggle would be counted as. A better approach we found to this was giving the program an input to a video of a person juggling.

We achieved better results with the object detection when feeding the program a video file. This made the output display slower, but since the program was able to look at every frame, it outputted the video at a slower frame rate to ensure high accuracy of soccer ball detection. Figure 7 displays a person juggling the ball with a very high accuracy of sports ball detection which averaged a confidence level of 0.80. Since we stored the pixel coordinate values of the middle of the ball which can be seen by a green dot in figure 7, we were able to use these values to detect the trajectory of the ball. A juggle counts as when the ball looks at its previous 7 positions and detects a valley of valleys in decreasing then sudden change to increasing order. The accuracy of this juggle count works well when the soccer ball is detected, once the object detection starts to fail, then the juggle could be off, but in this experiment the object detection was working for 95% of the frames. Having the subject juggle the ball perpendicular to the camera also ensured that for any pose estimation, the ball would not interfere with obstructing the view of the player.

Another experiment involved the player juggling the ball in front of the camera. Since the ball was closer to the camera the object detection average a confidence level of 0.70, still good enough to not cause any aliasing issues as with the

Figure 7. Program Running Showing the Middle of the Ball and the Juggling Count

Figure 8. Program Running Showing Juggling Straight at Camera

webcam. Juggles were counted once the player kicked the ball and it displayed on the top left corner of the screen as the video was outputted. 2 problems arose from this scenario. One, the ball counter was detecting and counting bounces of the ball from when the ball was bouncing up and down on the floor. The second was that this view of being straight on the camera disrupted any possible pose estimation models since the ball was obstructing the view of the entire person.

3.6. Pose Estimation and Object Tracking based Juggling Detection (Ruofan)

3.6.1 Description

Although OpenPose proved to be too slow for realtime juggling detection, we decided to run some offline analysis using our OpenPose results in order to understand what an idealized version of our tool would look like.

First, we ran both YOLOv5 and OpenPose on our juggling videos and saved the y-positions of the balls and joint positions in files. Then, for each frame, we computed a line at the right knee of the juggler. This line would be the determining factor for whether or a not a juggle was counted. If the ball went from below the line to above the line (determined by the y position of the line), a successful touch was counted. Figure 9 shows a successful touch being counted after the ball goes above the knee line.

We decided to try this approach in order to solve the problem outlined in the previous section of erroneous bounces being counted as juggles. Bounces are very unlikely to be above knee level, so we felt this solution would work well. Furthermore, this approach is more robust to failures in object detection by YOLOv5. The previous approach requires the last 7 y positions of the ball which can prove to be detrimental if YOLOv5 is often unable to keep the ball detected for 7 consecutive moments. In comparison, the offline approach only requires the knowledge of the most recent detected y position. This means that as long as the ball is detected for one frame on the upward and downward path of the juggle, a touch will be correctly counted.

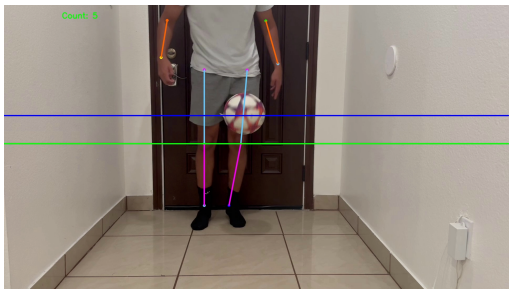

Figure 9. Offline Analysis Using OpenPose and a New Juggle Detection Scheme

Lastly, this approach theoretically does not need state of the art performance from the pose estimation model. There is no need for the knee line to be exact as it is just an approximate level at which the ball should reach when juggled. Mediapipe still proved to be too inconsistent for this, but it gives us hope that other lightweight pose estimation models may be able to implement this approach in real time.

3.6.2 Results

The offline analysis was able to best track juggling attempts as expected. The increased accuracy of OpenPose did increase the robustness as expected. Crucially, the new method was able to avoid the issue of counting erroneous juggles from manually moving the ball up and down and from the ball bouncing after a failed attempt.

There are still many limitations to this approach that need to be fixed in future iterations. For example, if a particular touch in a juggle barely moves the ball vertically, it is not likely to be counted as the ball will not go above the knee line. Figure 10 shows an example of this, where the ball was on the right side of the juggler, so the juggler gave the ball a touch that mostly sent it horizontally in order to center the ball again. Although this touch was completely valid, it was not counted by our algorithm as the ball did not achieve enough height to cross the knee line.

Other limitations go back to the limitations of other steps in our approach. In the example from the side, YOLOv5's inability to consistently track the ball still makes it difficult to properly count juggles as there are large portions of time

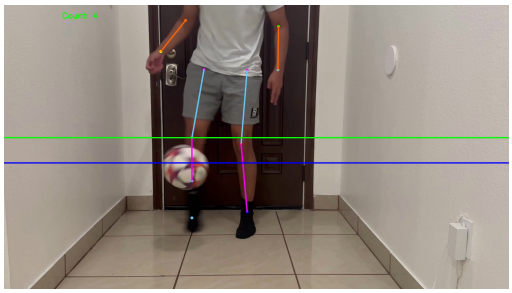

Figure 10. Low Juggles are not Counted

where YOLOv5 is unable to find a y-position.

3.7. Discarded Methods

3.7.1 Hough Transform

One of the implementations we tried and then disacrded was to do Hough Transforms with the OpenCV library to detect circles. A brief explanation is that the Hough Transform is a method that uses a given edge map for an image, and essentially draws a circle of a predefined radius, r from the center of every edge point on the edge map. By choosing the point drawn with the highest frequency, we can find a circle centered at that location with radius r. The Hough Transform was attractive because it is very fast and simple which would have been very beneficial for a realtime implementation of our tool.

This method had many issues when it came to ball detection. First, the ball deforms in the video as it moves at high speeds. This issue was also found in YOLOv5 but our Hough Transform implementation proved to be far too sensitive to this. The other issue is we are predefining the radius of the ball which is not robust to changes in distance between the camera and the juggler.

3.7.2 Motion Vectors/Optical Flow

The goal of this method is to extract the motion of the main objects we want to analyze. In this situation we would want to specifically focus on the ball. Advantages of this method would be that we iterate over frames and compute the flow of the features over frames. This can be achieved through optical flow on OpenCV (Lucas-Kanade method), where we pass parameters like a window to find the translated feature and minimum values for the translation. This approach operates under the assumption that there will be slight increments of movement for these features between frames.

Specifically, we wanted to use motion vectors to analze the movement of features frame by frame. This is achieved through a bounding box method to get the distinct object of the image and then removing the filters. More accurate processing is planned to first detect then mask out the objects not needed for the feature detection to carry out. This approach is mainly to have additional information for the directionality of images.

The main issue with this approach comes when objects make contact with each other. Sometimes a feature will ”abandon” an object when the object makes contact with another one. This is obviously problematic as during juggling there will be plenty of contact made between the feet and the ball.

4. Conclusion and Future Works (John, Ruofan, Kevin)

Future works that we found will benefit this experiment was to implement a deployable solution of this model. Currently it only works when given a video input or a webcam source to detect the sports ball. A deployable solution would comprise of an app where it will be easier to view progress as we originally intended and having an app using this method unlocks other possibilities where we would be able to have more minigames, scoreboard, and friends, which will host a competitive yet entertaining environment where players will be able to compete with their friends or themselves while increasing their own real life soccer skills.

To increase the overall effectiveness of detecting juggles an integration between pose estimation and object detection is mandatory. As we saw in the offline analysis, combining pose estimation and object detection proved to be able to solve many problems that we originally ran into such as false positive juggles. We believe that we can further improve our algorithm by experimenting more with pose estimation as we could do many interesting things such as compute distance from the feet to the ball as a way of determining a touch.

We also need to improve the object tracking in general as we ran into many issues further down the line after integrating pose estimation and object detection in the offline setting. At a certain point, many of the errors were caused by the fact that our YOLOv5 model was unable to locate the ball due to it moving at high speeds. This is due to a limitation of our approach as we are treating every frame as separate from each other and just resorting to object detection using a state of the art machine learning model to detect the object accurately. Some approaches that use Kalman filtering or optical flow may be very helpful here as it would help us incorporate the previous position of the ball to estimate the new position. Having a consistent track of the ball would boost the performance of our tool greatly.

Overall, we were able to track the movement of a ball, the person juggling, and the amount of times the ball was juggled with moderate success. Using OpenPose and YOLOv5 technology, we were able to use the data regarding the ball’s position in the frame to generally be able to determine when the player kicked the ball back up. Being able to do so allowed us to count the number of times the

person juggled the ball. However, there were some caveats in our implementation. The accuracy of our juggle detector is dependent on the position of the ball and player in relation to the camera, only works with video or webcams as the source of input, and counts a bounce as a juggle even if it was not caused by the person kicking the ball up. Even so, given specific player and ball positioning and video sources, our implementation was able to work quite well.

References

[1] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. Blazepose: On-device real-time body pose tracking. In *CVPR*, pages 6184–6193, 2020. 2

[2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43:172–186, 2021. 1, 3

[3] Guilherme de Sousa Pinheiro, Xing Jin, Varley Teoldo Da Costa, and Martin Lames. Body pose estimation integrated with notational analysis: A new approach to analyze penalty kicks strategy in elite football. *Frontiers in Sports and Active Living*, 4, 2022. 1, 3

[4] Glenn Jocher. Yolov5 by ultralytics, 2020. 1, 2

[5] Paresh Kamble, Avinash Keskar, and Kishor Bhurchandi. A deep learning ball tracking system in soccer videos. *Elsevier*, 2019. 1

[6] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016. 2

[7] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018. 2

[8] Wanneng Wu, Min Xum, Qiaokang Liang, Li Mei, and Yu Peng. Multi-camera 3d ball tracking framework for sports video. *The Institution of Engineering and Technology*, 2021. 1