FIFACoachBot

6.506 Midterm Report

Ian Gatlin

In this report, I first discuss the updated scope of the project and provide a brief overview of the tasks and challenges I've completed. Then, I will explain the work I have done so far, while addressing obstacles and challenges. I will wrap up with a schedule of the tasks to be done.

1. Updated Scope & Introduction

I've shrunk the scope of the FIFACoachBot. I now want to create an AI FIFA coach that tells a single player in a game where and when to pass or shoot the ball (rather than the attacking team). I also chose the teams Manchester City (myself) against Manchester United (computer opponent), with constant jerseys, formations, and players. The camera angle will also be at a constant perpendicular angle to the field when training the model. I chose this design to get the best chance of creating an AI coach that gives useful feedback to help me score goals.

In order to create an AI FIFA coach with graph neural nets, a lot of data preprocessing has to occur. Over the past month, I've explored computer vision libraries, image transformations, and color segmentation to create robust and accurate algorithms. These algorithms collect and create graph representations of the game that the model will use for training and to coach actions in different game scenarios.

Some of this work may not be useful in the model training portion of the project, but it may lead to interesting performance problems that I want to explore for the final report. Also, this work may be useful for developing more advanced features in the future. This work shows promise that the project's scope can be expanded to more general cases of FIFA 23.

2. Tasks & Obstacles

So far, I've worked on reliability transforming the video game screen into a graph representation of the game. As this process progressed, I found some useful ways to enhance the input data with explicit controller input and dynamic homography. Although I originally planned to have a first iteration of the graph neural net at this point, I hope that the extra work I did in preparing the data makes it easier to train an effective model.

Additionally, this preprocessing step has the greatest opportunity to discuss performance improvements, since in some use cases many color segmentation and edge detections need to operate on each frame to create an accurate graph representation. After creating a first working model, it will be important to revisit the data processing step to have interesting performance evaluations for the final presentation and report. To be clear, I am discussing a preprocessing step that happens on a camera angle that is not always perpendicular to the field. Figure 2 in section 2.2 shows the difference between these two angles.

In this section, some of the gameplay footage is from a YouTube video uploaded by Throneful [1].

2.1 YOLOv8 Bounding Boxes

I briefly explored implementing my own color segmentation to detect players, and quickly realized that using an existing library would be much easier. Drawing bounding boxes around a lot of examples is easier than trying to tune a generalizable color segmentation algorithm. YOLOv8 quickly emerged as a top candidate for use. This is because of its simplicity, documentation, and its use at my future employer. I found the "Ultralytics YOLOv8" YouTube playlist to be very useful [2].

I used Roboflow to create datasets that contained screenshots of the game with bounding boxes around items of interest on the game field.

I designated classes that I am going to use on the first neural network: ball, left_goal, right_goal, ref, team_player, team_goalie, opponent_player, opponent_goalie. The goals are differentiated so I can make it clear when I am attacking by manually setting which goal I am attacking in each half. This is important since I am only creating coaching recommendations for when my team is attacking. The ref class is created to make it clear that they are not players. There are some classes that I want to experiment with in the future: left_penalty_corner, right_penalty_corner, left_corner, right_corner, left_top_penalty_arc, right_top_penalty_arc.

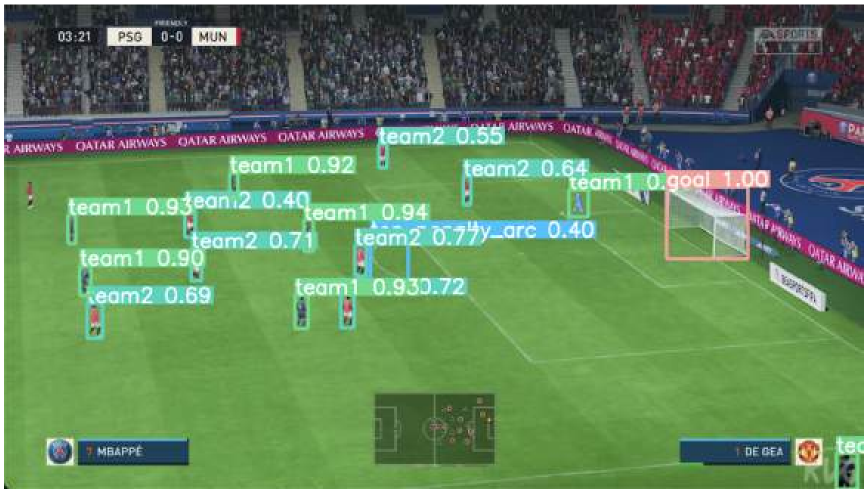

Figure 1. Bounding boxes generated from model trained from YOLOv8

There were some problems with the model detecting the ball on my initial attempt, but I am currently trying to fix this by choosing an all orange ball in the game, and by training it on many more instances of the ball. I have yet to test my this but there are instances of YOLOv8 being able to detect balls in soccer games in this video by Machine Learning With Hamza [3].

2.2 Homography Transformations into 2-D Plane

Once I reliably found bounding boxes around players, I chose to use the midpoint of the bottom edge of the box to represent the point where the player was located on the pitch. Now, I needed to transform these points into a 2-D plane that resembles an aerial view of the pitch. By doing this, I can construct a truer representation of the game state, since the raw angle of the camera is distorting the players' relative positions. To help me find these matrices, I used a method of mapping points on an image and plotting their endpoints as described in this Medium article by Francis Camarao [4].

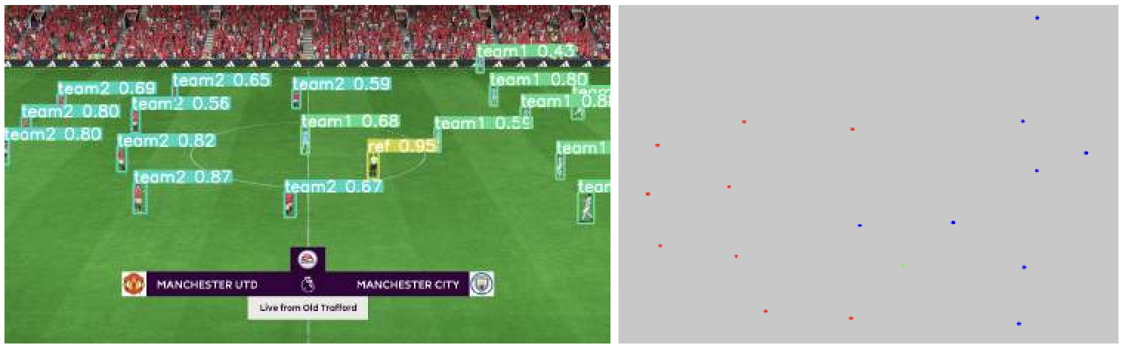

Figure 2. Bounding boxes turned into points and projected onto a 2-D plane

I first worked with the default camera setting in FIFA 23 on next gen consoles, "EA Sports GameCam." This camera setting does not stay perpendicular to the game field, therefore, different transformations were required to achieve consistent representations on a 2-D plane.

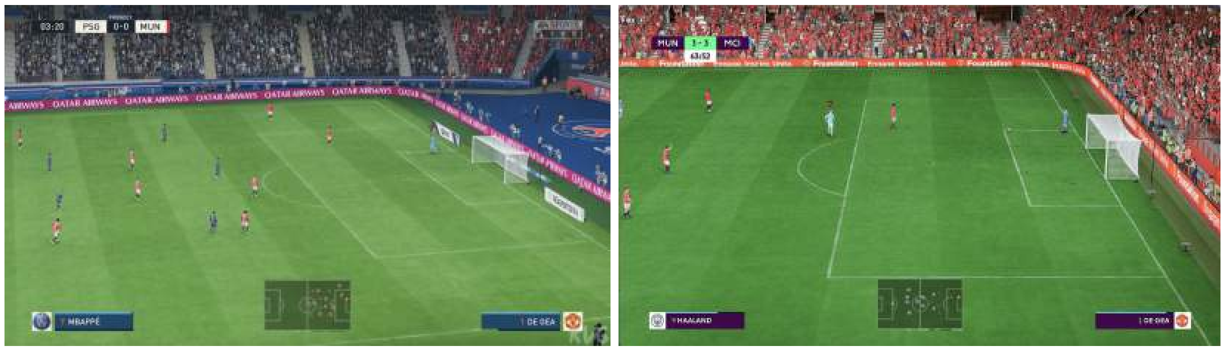

Figure 3. Differences of camera angles of the right end of the field in "EA Sports GameCam" and "default" camera settings

To figure out which transformation to apply, I found the angle of the advertising boards from the horizontal. To do this, I ran a large sobel horizontal edge detection convolution on gaussian blurred images of the red, green and blue color channels. Then I aggregated them by maximal value, and segmented the image by keeping points that were above the mean of all points with values greater than 80. Finally, I applied an edge detection convolution on the image to thin out the remaining lines.

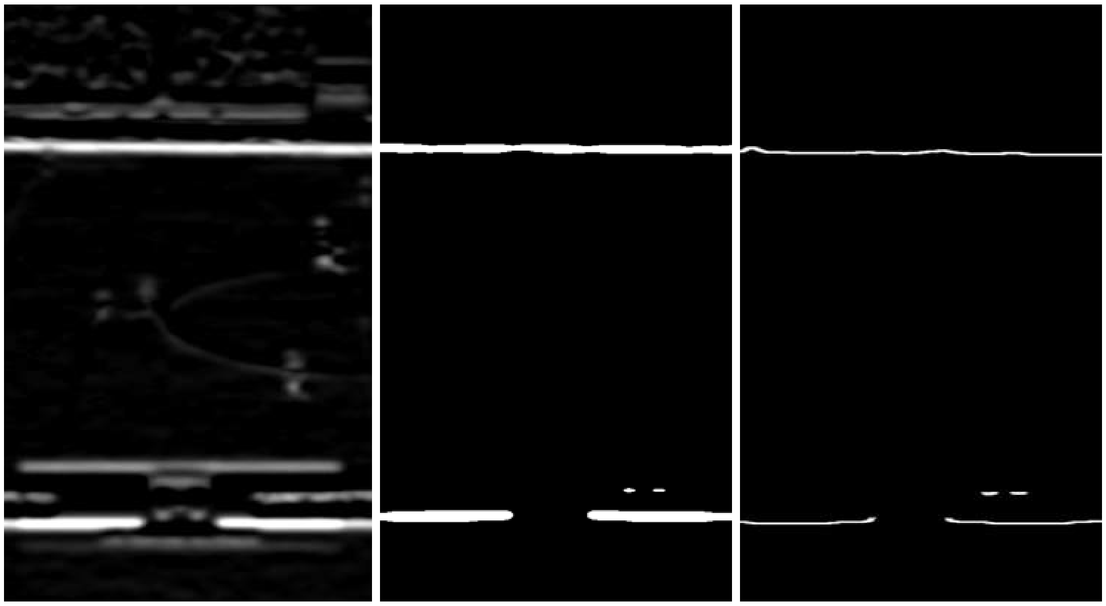
Figure 4. Results from maximal aggregation, segmentation, and edge detection thinning

To find the line that we could actually calculate the angle from, I used the HoughLines module from cv2 with incrementally decreasing thresholds from 1200 to 100 in increments of first 50 and then 100 until I found a line. Now that we know the angle of the field from the horizontal, we can plug that into a function that calculates our homography matrix. This complete process takes noticeably longer to process each frame compared to just streaming video though it. I would estimate this to be around a factor of 4-5.

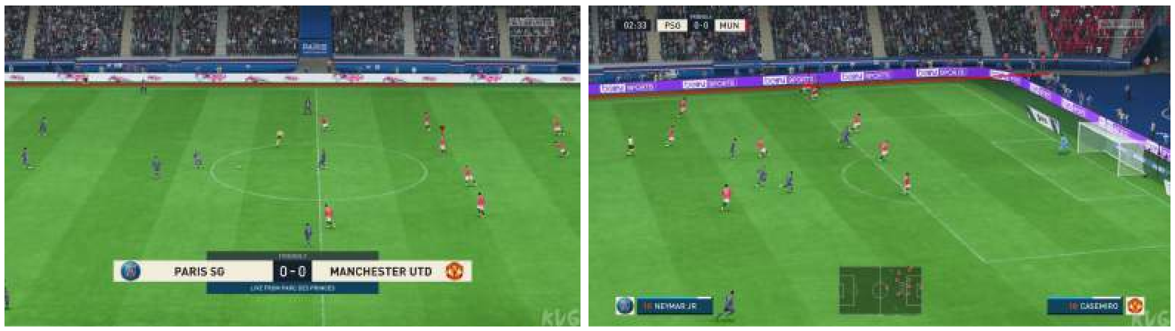

Figure 5. Red hough lines appear on the bottom edge of the digital advertising board

However, after doing this work, I realized that an easier solution would be to use game footage from the “default” camera angle (which isn’t actually the default camera angle on next gen consoles) that keeps the camera perpendicular with the field. Using this approach, the steps of color segmentation, finding hough lines, and calculating homography matrices can be replaced with a single static homography matrix.

Although this pipeline could be bypassed with this other camera setting, the approach of calculating dynamic homography matrices could allow amateur soccer teams who can only afford a single static camera to record game footage access to AI coaching by just adding a straight colorful rope along the opposite sideline of the camera (this rope would serve as the advertising board in FIFA 23). It may be important to go back and explore this algorithm to analyze and optimize its performance to be able to discuss an interesting performance aspect for this project. Some optimization ideas include determining which color channel is the most useful every 30 frames rather than running blurring and edge detection on each channel every frame, reducing the size of the images used, reducing the frame rate of the application, and only running one edge detection rather than two. These would permit for more responsive real time suggestions, but could diminish the quality of the data extracted from the algorithm. This would be interesting to explore.

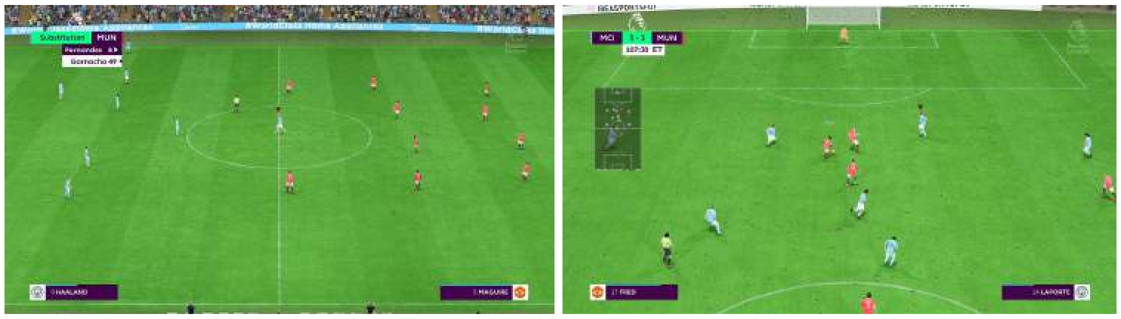
Figure 6. The “Co-op” and “End to End” camera settings

There are also other camera settings that may be useful to explore. The “Co-op” setting provides a more zoomed out camera angle, allowing us to see more of the field at once. The “End to End” setting shows the field on its other axis, which is a popular angle used across all sports footage analysis.

2.3 Graph Representation

I created a custom undirected graph class implemented with an adjacency list structure. I believe that a useful representation will include edges between every player on my team, between every player on my team and the goal, and between each of my players and the 3-4 closest opponent players. This will encode relationships between places where I can pass the ball, shoot the ball, or dribble the ball. It also encodes the likelihood of a teammate being able to successfully receive the pass through closeness of teammates to opponents.

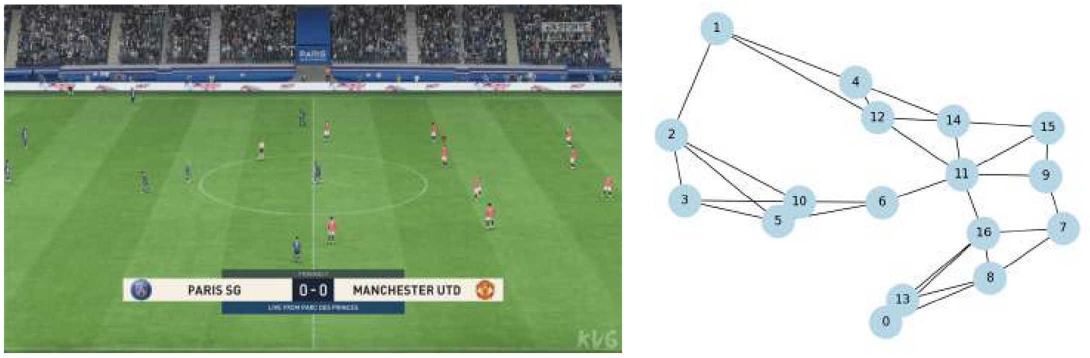

Figure 7. Graph Representation with edges from each player to closest 3 neighbors

I also think that encoding angles into these edges will be important to determine if passing lanes are available or not. I can simply assign an angle for every edge exiting a vertex, since relative edges from a single vertex are the most important to analyze. For example, say an opponent is standing between a player with the ball and a teammate, I should not pass it through that player. Those two edges would have the same angle, and therefore we have encoded that the passing lane is not open.

2.4 Additional Data & Labeling

I found out how to display an overlay of my controller input onto the screen with gamepadviewer.com. This is extremely valuable since my controller inputs are now automatically encoded into each frame. I use a white PS4 controller since color differences between parts of the controller are most pronounced.

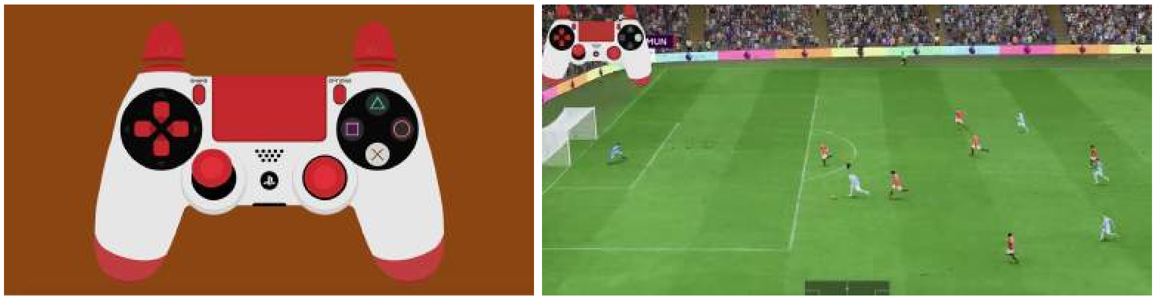
Figure 8. A controller input signifying passing upwards, and an input in game for a shot to the far post

I will still have to go though and label more general scenarios. I know I want to encode goals, goal scoring opportunities, and interceptions, but I will have to do some more digging into the graph neutral nets before deciding if more specific labels make sense.

2. 5 Obstacles and Major Changes

One of the most important challenges is figuring out how much FIFA data I need to collect and label for the model to understand what makes a good attacking play. I'm assuming that I will need a large data set, but I am not sure how large. Playing FIFA is not necessarily a problem, but the labeling process takes a lot of time. I don't think this will be a giant issue, but I just need to get started on this earlier rather than later.

A main reason why I don't have datasets with all of the new features discussed in this report is due to hardware issues. Since I am prioritizing capturing controller input and I have an Xbox, I have to use a streaming feature on the Xbox Windows application that has yielded very poor quality footage. Luckily, one of my roommates has a high quality gaming PC that I am going to collect my data on in the coming days. It is frustrating that the first model will not be very portable, but that will have to be addressed in the future. It is easy to collect high quality game footage with a capture card on Xbox or PC (with controller input), but very hard to record quality footage with an Xbox and controller input.

Another interesting challenge in portability is homography for different resolutions. Currently, I manually find matrices for different resolutions. It would be very cool to have this encoded in a function. This could enhance the performance aspect of the project since we could throttle the recording resolution quality during times when the CPU can't keep up with the incoming frames, which would help the model keep up with the game.

Finally, since I haven't started the GNN implementation yet, there are unknown problems yet to be faced. However, I am confident in being able to at least set up a neural net, and then explore the performance aspects discussed above. It would be optimal to have the performance aspects be related with the effectiveness of the model, and I believe that there is a high probability that goal is achieved.

3. Next Steps & Schedule

4/19: By the end of this weekend, have a labeled data set of two hours of high quality footage with controller input.

4/26: Have a neural network moderately tuned.

5/3:

- Continue to make changes to the graph representation to achieve better results. Explore doing computer vision on different parts of the field. Play with scope if needed.

- Collect some video data using the “EA Sports GameCam” camera setting. Tune pipeline to make graph representation in this setting be close to identical to the “default” setting.

5/10: Evaluate how speeding up performance of the dynamic homography pipeline results in better coaching recommendations from the model.

5.14 - Final Report and Presentation: Write and present findings about model and dynamic homography pipeline.

References

[1] “FIFA 23 Gameplay (PC UHD) [4K60FPS].” *YouTube*, Throneful, 27 Sept. 2022, www.youtube.com/watch?v=cgDlmvU2sA4.

[2] “Object Detection with Pre-Trained Ultralytics Yolov8 Model | Episode 1.” *YouTube*, Ultralytics, 12 July 2023, www.youtube.com/watch?v=5ku7npMrW40&list=PL1FZnkj4ad1PFJTjW4mWpHZhzgJinkNV0&index=9.

[3] “Computer Vision for Football Analysis in Python with Yolov8 & OpenCV.” *YouTube*, Machine Learning With Hamza, 11 Dec. 2023, www.youtube.com/watch?v=yJWAtr3kvPU.

[4] Camarao, Francis. “Image Processing Using Python-Homography Matrix.” *Medium*, Medium, 17 June 2023, medium.com/@flcamarao/image-processing-using-python-homography-matrix-a5da44f3a57b.