Video Preprocessing for American Football Formation Recognition

1st Kimi Wright
Electrical and Computer Engineering
Brigham Young University
Provo, Utah
ksw38@byu.edu

2nd Shad Torrie
Electrical and Computer Engineering
Brigham Young University
Provo, Utah
shad.torrie@byu.edu

3rd Benjamin Orr
Electrical and Computer Engineering
Brigham Young University
Provo, Utah
benjamin.orr@byu.edu

4th Dah-Jye Lee
Electrical and Computer Engineering
Brigham Young University
Provo, Utah
djlee@byu.edu

2024 Intermountain Engineering, Technology and Computing (IETC) | 979-8-3503-7307-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/IETC61393.2024.10564283

***Abstract*—American football analytics and in particular formation identification is important to gaining a competitive edge. Currently, most of this football video annotation process is done by hand. With the recent advancements in computer vision and artificial intelligence, automated sport analysis systems are gaining popularity. One major challenge in current football systems is constraints on camera location. In this work we attempt to overcome issues of camera location by performing a perspective normalization method. This entails using conventional computer vision algorithms to locate various lines and markers on the football field. This information is then used to normalize points into a camera position agnostic perspective. This preprocessing will then allow for a neural network to be trained to identify formations and preform other football analytics.**

***Index Terms*—OpenCV, American Football, Sports Analytics, Neural Networks, Preprocessing**

I. Introduction

American football involves frequent setting up in formations; therefore analytics, (formation identification in particular), to learn how these formations work and how opposing teams use them are vital to success. Currently, analytics are often done manually, which involves hours of work by the football coaches hand reviewing footage.

This work focuses on the preprocessing of data points in an image to transfer them into a camera position agnostic perspective. This preprocessing is a crucial piece of a larger system that will automatically analyze football footage and extract information such as formation recognition and player tracking data. We use a camera in a fixed position close to the 50 yard line. Sample images of our camera pointed straight and turned right and left are shown in Figure 1 respectively. We identify points on these images to create a camera agnostic view such as bird's eye view. Our work in this paper focuses on line detection, number recognition, and perspective transformation, and to place players in a virtual football field that can then be used in future works to perform formation classification and implementation of other football analytic tools.

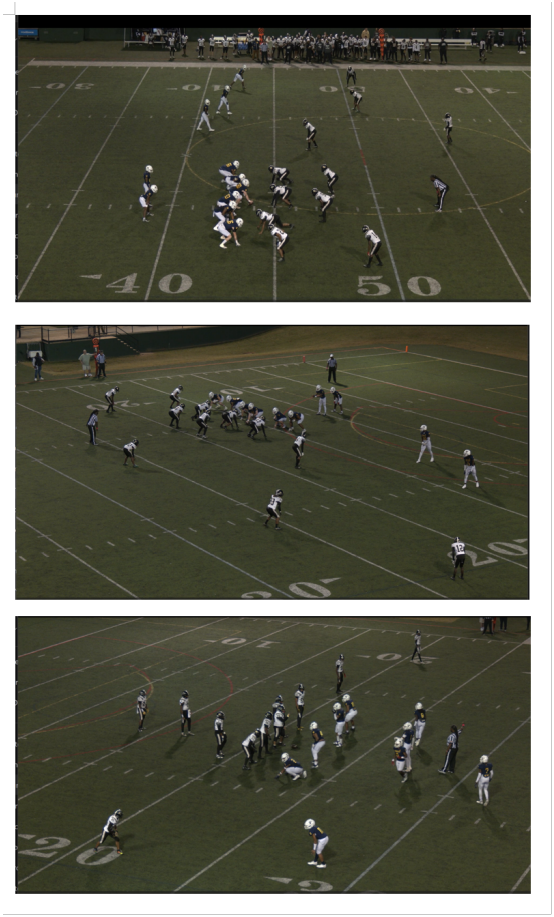

Fig. 1: Examples of the variation of camera angles in football game footage. Top: pointed straight, Middle: turned right, Bottom: turned left.

II. Background

Some estimates put the sports industry at USD 885 million in 2020 [1], and sports analytics has been of interest to both fans and coaches looking to predict how teams will do, determine how to improve teams, and develop strategies. Computer vision has recently found countless applications in the sports realm. These include ball tracking, athlete training, and automation of broadcast cameras [2]. In addition to traditional methods, deep learning techniques have also proved to be an invaluable tool in accomplishing of these tasks. Cust et al. demonstrated examples of this in their review of model development for sport-specific movement recognition [3]. Most sports games are already filmed, but because broadcasts contain irrelevant events, like people celebrating, zooming in on referees, or advertisements, they are difficult to work with. This leads researchers to either make them more usable, for instance develop highlight tracking or advertisement removal [4] or taking data separate from the broadcast video by putting in multiple cameras [5] or placing sensors on players [1].

In this paper, our focus is on utilization of deep learning for automated annotation and analysis of American football games using standard football footage. Designed to save time, money, and energy of coaches and athletes, systems like these can provide major benefits throughout the sports world. One system that does this is shown in [6], but we believe that accuracy can be further improved by identifying player locations before pipelining data to the neural net. An example of a system designed for this task can be seen in our previous work [7]. In this specific architecture, the authors detail three major system modules: player localization, player labeling, and formation identification—each of which require a clear, unobstructed view of the athletes.

As such, camera pose and placement is a nontrivial challenge in automated sports analysis. Naik et al. discuss the necessity of using the appropriate number of cameras in the correct positions for play field extraction [8]. In [7], the ideal camera placement is described as being high enough such that all players are visible by the camera (i.e. directly above the play). Given that this is not always possible, we approach this challenge with methods of video preprocessing—something that has shown promising results in the work of [9].

III. Methods

Having our camera at the 50 yard line puts our videos at an angle and obstructs parts of the field, as can be seen in Figure 1. Therefore, before analysis it is important to convert the positions of the players to a camera agnostic view so the data for formations remains consistent. Football fields must be made to certain specifications, including the size of the field, 120 yards long and 53.3 yards wide; the yard lines, which are all parallel and spaced 5 yards apart and every 10 yards; and the numbers, which mark the distance to the end zone. These consistent features allow us to accurately determine where players are on the field. We begin by finding the lines using the Hough lines formula, which helps us determine the relative positions of the players. The numbers can be found by finding and sorting other contours, and then we recognize these numbers by training a neural network. The numbers help us determine where we are on the field. Figure 2 depicts our method as a flowchart.

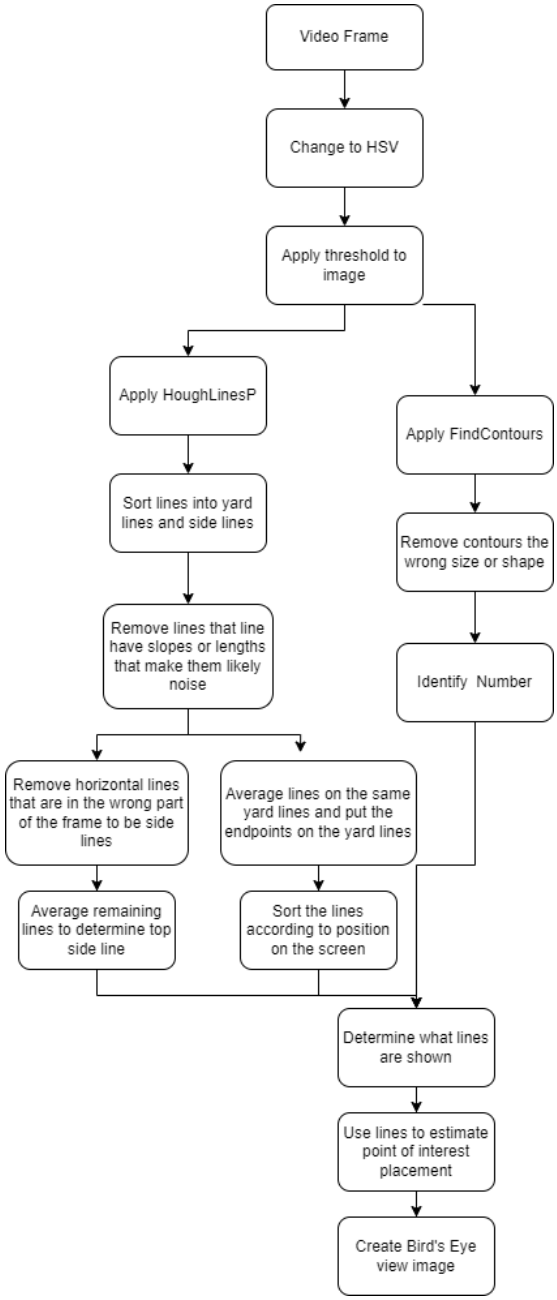

Fig. 2: Video preprocessing method for American Football Formation Recognition.

A. Frame Preprocessing

The Hough lines formula and functions for finding contours work best when images are black and white and have high contrast. Through trial and error we found that using the value channel of the HSV version of the image has the strongest contrast between the green field and the white markings.

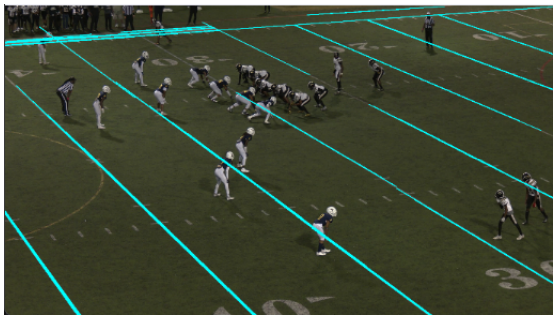

Fig. 3: Lines detected using HoughLinesP shown in light blue on the original color image.

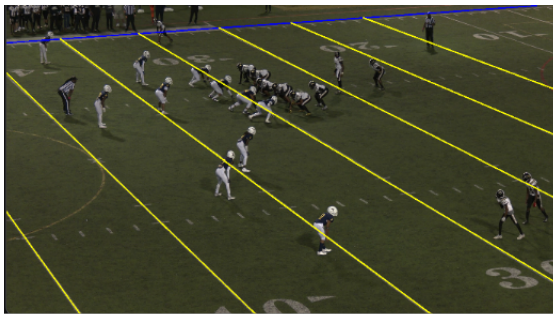

Fig. 4: Relevant lines after line processing. Yard lines are shown in yellow, sidelines in dark blue.

After extracting the value channel, we took the threshold of the frame, leaving a black and white image for use in our processing.

B. Line Detection

After processing a frame of our video, lines can be recognized by the popular OpenCV formula HoughLinesP. Figure 3 depicts the lines found by this formula in light blue on the color version of the image. The lines found by this formula are stored in the form $[[x_1, y_1, x_2, y_2]]$, with (x_1, y_1) and (x_2, y_2) being the endpoints of the line. From our angle at the 50 yard line, yard lines appear closer to vertical and sidelines closer to horizontal. Therefore, we can sort lines with a steep slope as likely yard lines and lines with a shallow slope as likely sidelines. As can be seen above the sidelines in the stands of Figure 3, lines not relevant to markings on the football field are also found, but because the dimensions of a football field are well established, these can be sorted out using parameters of length, location, or slope. Additionally, the lines are thick, so multiple lines are found at every line on the football field; after determining which lines correspond to each field line, these lines are averaged together.

The bottom sideline is frequently obscured or not visible from our camera angle, so we focus on finding the top side line. Most near horizontal lines we need to sort out come from the stands or the crowd at the top of the image. Fortunately, because the top sideline is so thick, most near horizontal lines found will be on the top sideline.

$$y_{intercept} = y_{h1} - s * x_{h1} \tag{1}$$

Thus, we can sort out which lines identify irrelevant features, (e.g., the stands), or noise by finding the y intercept of these lines using Equation 1, where s is the slope, and removing any lines with an y intercept that deviates too far from the mode. The remaining lines are averaged to determine the top sideline. To add stability, this line is also averaged across many frames.

$$s = \frac{y_{h2} - y_{h1}}{x_{h2} - x_{h1}} \tag{2}$$

We can then estimate the bottom sideline by determining the slope of the top sideline using Equation 2 and estimating the bottom sideline as being parallel to the top sideline.

Because there are less extraneous features on the field itself, yard lines can be sorted from noise by leaving out all lines that are below a certain length or have slopes that deviate too far from the expected slope. The leftover lines can be combined into yard lines by use of a rolling window across the center of the screen. Any lines within the same window of a few pixels are combined into an average, creating a more accurate singular yard line. Because of areas of faint paint or players obstructing parts of the line, these lines tend to be shorter than the true yard lines. To find the true endpoints, we find where the lines would intersect with the top and bottom sidelines if the yard line were infinite in length.

$$u = \frac{(x_{b2} - x_{b1})(y_{a1} - y_{b1}) - (y_{b2} - y_{b1})(x_{a1} - b_{x1})}{(y_{b2} - y_{b1})(x_{a2} - x_{a1}) - (x_{b2} - x_{b1})(y_{a2} - y_{a1})} \tag{3}$$

$$x_{int} = x_{a1} + u(x_{a2} - x_{a1}) \tag{4}$$

$$y_{int} = y_{a1} + u(y_{a2} - y_{a1}) \tag{5}$$

Using the yard line as line a and the sideline for line b we first find u as a courtesy variable using Equation 3 and then use u in Equations 4 and 5 to find the x (x_{int}) and y (y_{int}) coordinates of these intersections, respectively. The yard line is then extended to have endpoints at these intersections on top and bottom sidelines. Finally, the yard lines are sorted by their position on the screen so they can serve as reference for other points of interest on the screen. In Figure 4 the sideline is depicted in dark blue and the yard lines are depicted in yellow.

C. Number Recognition

Finding the lines helps us establish relative details of the image, but tells us very little about what part of the field is actually shown. To find that information we recognize the numbers found next to the yard lines using the same processed frames we used to find the lines. Another OpenCV function, FindContours, detects color changes in the image, and allows us to find the white numbers on the black background of our thresholded frame. However, this also captures noise and miscellaneous contours unrelated to number recognition. Trying to identify these extraneous contours greatly slows

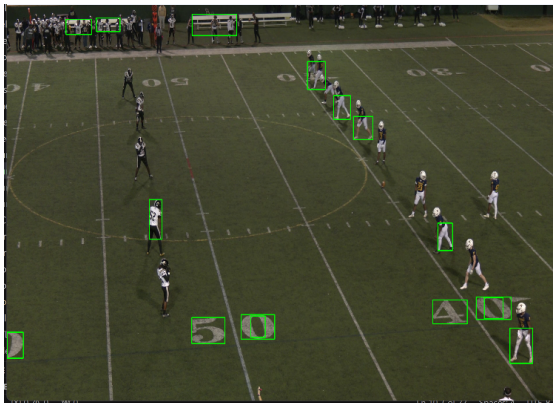

Fig. 5: Numbers detected in a football field, with some extra contours also detected.

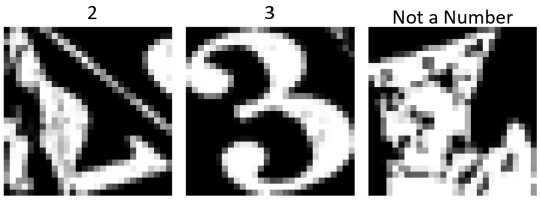

Fig. 6: Normalized Numbers for Neural Network use. Normalized numbers are fed to the neural net, and a label of "Not a number" or the recognized number is returned.

down computation time but adds no value. Thus we set a minimum number of white pixels or contour area in a contour for it to proceed to the next step. This greatly reduces the noise processed; however, if large pieces of a line are identified as a contour then they may meet the contour area requirements. These are removed by setting a required proportion; contours that are too long and skinny are removed. This still leaves some extra contours, such as players in light-colored or white uniforms, but this pairs the contours down to a manageable number for recognition. These found contours are given a bounding box for visibility that can be seen in Figure 5.

Once the numbers are found, they can be identified. Due to the font and tilt of the numbers on a football field, premade Optical Character Recognition (OCR) algorithms like Pytesseract struggle to identify the numbers. However, number recognition is a solved problem. Many solutions already exist for the similar problem of handwritten numbers, such as those using the famous dataset MNIST. Our requirements are lower than many configurations made to work for MNIST, as we only need to identify numbers 1-5. The number 0 also occurs on the field, but because it occurs next to every number no added information comes from recognizing it. Additionally, because many numbers are displayed on the field at the same time and from frame to frame, errors are easily corrected, giving us a high tolerance for error. Thus, it was easy to adapt a neural network written for MNIST to our purposes. We made our dataset by using our finding and sorting contours code, normalizing these images to the same dimensions as

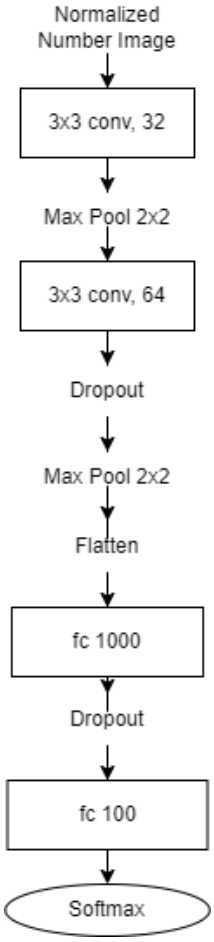

Fig. 7: Neural Network structure for number recognition.

the MNIST dataset, and then hand labeling the images. An example of this is shown in Figure 6. We created a dataset of normalized number images derived from our videos containing 500 images of either zeros or non-numbers, and 120 numbers 1-5, which is a ratio similar to what it will see analyzing actual footage. We trained our model from scratch using a randomized training set of 496 of these images. We tested the model on the remaining 124 images, and attained an accuracy of 96%. This accuracy will be improved as we build a larger dataset with more balance ratios of numbers to non-numbers.

D. Transformation to Bird's Eye View

Once we have identified the lines and numbers of the field, we have the information needed to normalize points into a camera position agnostic perspective. Using intersections between the yard lines and sidelines, we could use the the standard OpenCV functions getPerspectiveTransform() and warpPerspective() to get us an image of a transformed field, as shown in Figure 8. However, this is not actually as helpful to formation identification as finding individual player locations on the image and calculating where they would be on a virtual field.

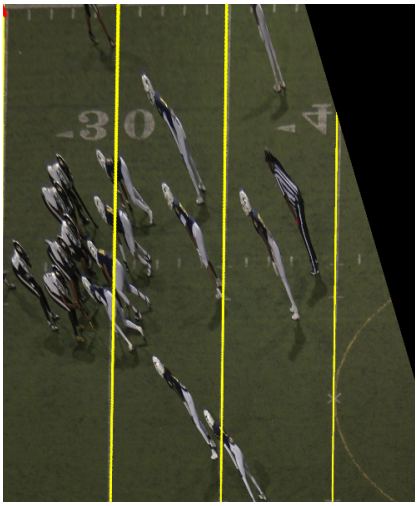

Fig. 8: Version of the field warped to bird's eye view using standard OpenCV functions getPerspectiveTransform() and warpPerspective().

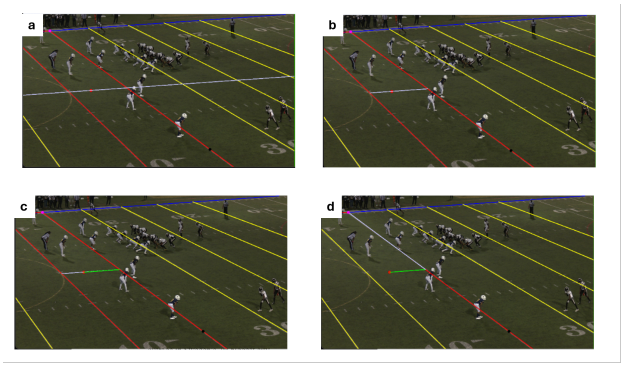

Fig. 9: Reference lines to determine the position of the point of interest. a) Reference line through point of interest, parallel to the sidelines shown in gray. b) Line segment of the reference line between the yard lines shown in gray. c) Green line between the point of interest and the right yard line, used to find where the point is between the yard lines. d) Yard line with portion between the point of interest and top sideline shown in gray, used to determine where the point is between the sidelines.

The point of interest, referred to here as (x_p, y_p), can have its relative location in a camera agnostic view, (x_c, y_c), calculated by comparing it to the yard lines and sidelines. To do so we begin by drawing a reference line through the point that is parallel to the sidelines stretching across the whole image, as shown in Figure 9.a. This code defines lines by its endpoints, so the reference line found in terms of endpoints (x_{p1}, y_{p1}) and (x_{p2}, y_{p2})

$$y_{p1} = s(x_{p1} - x_p) + y_p \tag{6}$$

$$y_{p2} = s(x_{p2} - x_p) + y_p \tag{7}$$

Because the line is nearly horizontal, we can ensure that it stretches across the image by choosing the x values $x_{p1} = 0$ and $x_{p2} = image\ width$. We can then use these values in Equations 6 and 7 to give us y_{p1} and y_{p2}. Consolidating these values, we have our reference line $(x_{p1}, y_{p1}), (x_{p2}, y_{p2})$.

Given a reference line, we can find which yard lines are the closest to the points of interest. We begin by using Equations 3 and 4 to determine the x-value, x_{int}, of the intersection of the reference line with each yard line. We then find the two minimum values of $x_{int} - x_p$ to determine which yard lines the point of interest is closest to. The yard lines found to be closest to the point of interest are shown in red in Figure 9.a.

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{8}$$

To find where the point of interest is between the two nearest yard lines we calculate the length between their intersections with the line through the point of interest using the standard formula for the distance between 2 points, given in Equation 8. The line between the two intersections is shown in Figure. 9.b; this gives us the distance between the two lines. We then use the same equation to calculate the distance between the point of interest and the intersection on the right side, shown in green in Figure. 9.c. We can then divide this line by the distance between the two lines, to give us our estimate of the horizontal distance.

To find where the point of interest is between the two sidelines we calculate the distance between the intersections of the right yard line with the top sideline and bottom sideline, giving us the total length of the yard line. Then we calculate the distance between the intersection between the right yard line and the line through the point of interest and the intersection between the right yard line and the top sideline, shown in gray in Fig. 9.d. By dividing this line by the length of the yard line, we can determine where the point is between the sidelines.

Due to the angle of the image, a line actually parallel to the sideline is not parallel in the image. Therefore, by using a line going through the point of interest that is parallel to the sideline introduces some error. However, these estimates do not affect the end result of our code because player positions are not measured to match the field, and for formations it is only their approximate relative positions to each other that are important. Figures 10 and 11 show different formations and their estimated placement on the bird's eye view on the field. In addition, the visible lines are shown in red on the bird's eye view graphic in both figures.

IV. Results

Our method allows us to estimate the placement of points of interest using a video taken from a camera angle in the stands to a bird's eye view. Looking at each frame individually, we process each frame by taking the threshold of the value channel and using the modified frame to identify lines and contours, which can then be sorted into field lines and field numbers. Using these lines and numbers we can pinpoint the

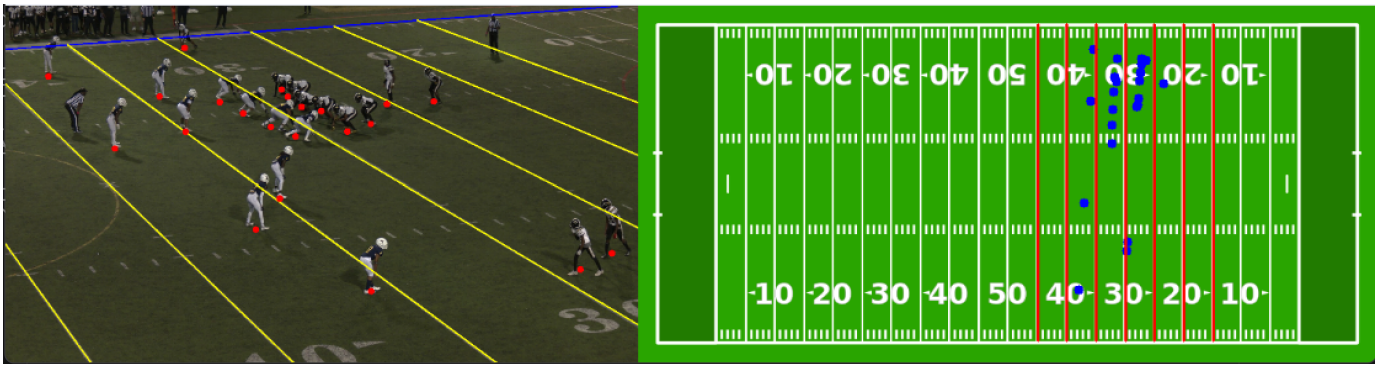

Fig. 10: Formation transformed into camera agnostic view.

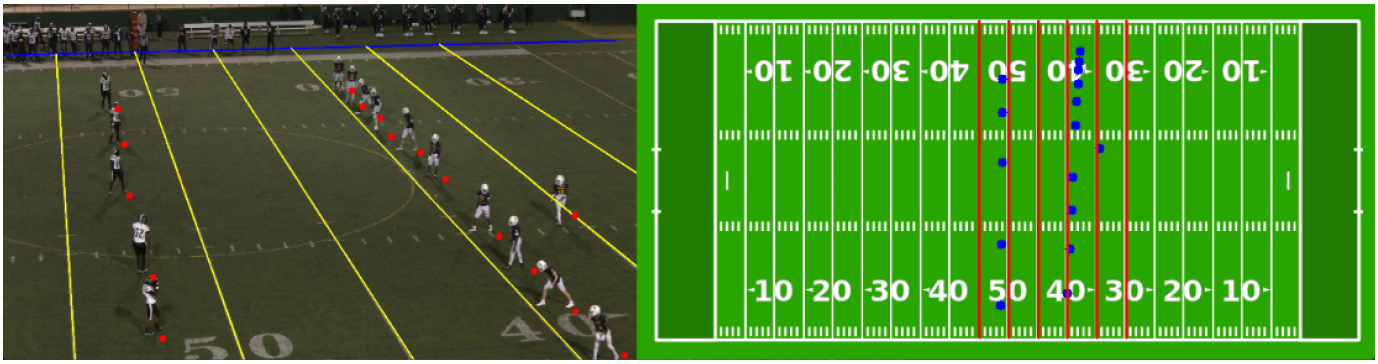

Fig. 11: Kick-off formation transformed into camera agnostic view.

location of different points of interest on the field, which can then be placed in a virtual field as shown in Figure 10.

The placement of the points achieves horizontal accuracy to a decent degree but is frequently off on its vertical measurement. However, due to the nature of how football players cluster, that is sufficient for formation detection.

This program achieves better results at certain angles and videos, and more work is needed to improve the overall robustness and integration of the code.

V. Conclusion

In conclusion, through field number and line detection, we can change points of interest from a video frame to a bird's eye view to allow formation recognition based on the placement of the points. This perspective normalization is crucial for future formation identification systems. Our future work will involve improving robustness, integrating player detection using YOLO to find our points of interest in the frame, and finally integrating them into a neural network for formation recognition.

References

[1] I. Ghosh, S. Ramasamy Ramamurthy, A. Chakma, and N. Roy, "Sports analytics review: Artificial intelligence applications, emerging technologies, and algorithmic perspective," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 13, no. 5, p. e1496, 2023.

[2] G. Thomas, R. Gade, T. Moeslund, P. Carr, and A. Hilton, "Computer vision for sports: Current applications and research topics," *Computer Vision and Image Understanding*, vol. 159, 04 2017.

[3] E. E. Cust, A. J. Sweeting, K. Ball, and S. Robertson, "Machine and deep learning for sport-specific movement recognition: A systematic review of model development and performance," *Journal of sports sciences*, vol. 37, no. 5, pp. 568–600, 2019.

[4] T. D'Orazio and M. Leo, "A review of vision-based systems for soccer video analysis," *Pattern recognition*, vol. 43, no. 8, pp. 2911–2926, 2010.

[5] S. Barris and C. Button, "A review of vision-based motion analysis in sport," *Sports medicine*, vol. 38, pp. 1025–1043, 2008.

[6] K. Zhou and J. Galbraith, "Automatically labeling offensive formations in american football film using deep learning," *Journal of Student Research*, vol. 12, no. 1, 2023.

[7] J. Newman, A. Sumsion, S. Torrie, and D.-J. Lee, "Automated pre-play analysis of american football formations using deep learning," *Electronics*, vol. 12, no. 3, p. 726, 2023.

[8] B. T. Naik, M. F. Hashmi, and N. D. Bokde, "A comprehensive review of computer vision in sports: Open issues, future trends and research directions," *Applied Sciences*, vol. 12, no. 9, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/9/4429

[9] I. Atmosukarto, B. Ghanem, S. Ahuja, K. Muthuswamy, and N. Ahuja, "Automatic recognition of offensive team formation in american football plays," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2013.