A modern approach for positional football analysis using computer vision

Conference Paper · September 2022

DOI: 10.1109/ICCP56966.2022.10053962

2 authors:

Mihnea Jurca
Technical University of Cluj-Napoca
2 PUBLICATIONS **3** CITATIONS
SEE PROFILE

I. Giosan
Technical University of Cluj-Napoca
25 PUBLICATIONS **281** CITATIONS
SEE PROFILE

2022 IEEE 18th International Conference on Intelligent Computer Communication and Processing (ICCP) | 978-1-6654-6437-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICCP56966.2022.10053962

A modern approach for positional football analysis using computer vision

Mihnea Bogdan Jurca, Ion Giosan
Computer Science Department, Technical University of Cluj-Napoca, Romania
Jurca.Vi.Mihnea@student.utcluj.ro, Ion.Giosan@cs.utcluj.ro

***Abstract*—In this work we aim to construct a robust pipeline for the sports analysis community in order to successfully extract useful information from broadcast football matches. We propose a fast and efficient solution, based on computer vision and machine learning methods and algorithms. Our solution provides a framework suitable not only for detecting, tracking and identify the roles of the players and staff, but also for mapping each player from their position as seen in broadcast images, to their absolute position on the field. In order to achieve this, we designed each module of the pipeline by comparing multiple solutions and choosing the most suitable ones taking into consideration the trade-off between performance and inference time. We managed to provide a system that can be used by anybody in the community by feeding a sequence of frames taken from a broadcast football match.**

I. Introduction

In sports, success is often on the side of the best trained team. The training strategy and in-game decisions are aided by analyzing a significant amount of information extracted not only from past games, but also from training sessions. Until recent years, match analysis was performed manually. The lack of technological capabilities and solutions made the automation of soccer analysis a difficult task. Therefore, the goal is now to be able to process data as accurate, as fast and as effortless as possible.

When it comes to positional data, the goal is to understand how the placement of the players in the field affects the performance of the team in the match. Positional analysis aims to answer questions like: Where are the players? How is the game influenced by the position of the players? From which positions did most of the key events happen? All these questions will be answered by providing a visual context to the matches. The broadcast of football matches has been a great breakthrough in the industry. As mentioned above, the data was processed manually and only in recent times was it stored and visualized in tools, as the one proposed in [17].

Our goal is to create a robust pipeline and give suitable solutions for each of its modules, in order to obtain a fully functional system which will detect, track and identify the players with respect to the team from which they belong. Also, the system should be able to map the position from the input frame, taken from a broadcast game as captured by a moving camera, to the absolute position on the football field, displayed as a birds eye view.

The pipeline will be composed of the following modules, which will be analyzed later in this paper:

- Player and ball detection
- Player and ball tracking
- Player team assignment
- Positional player and ball mapping

The pipeline of the system and the main steps are shown in Fig.1

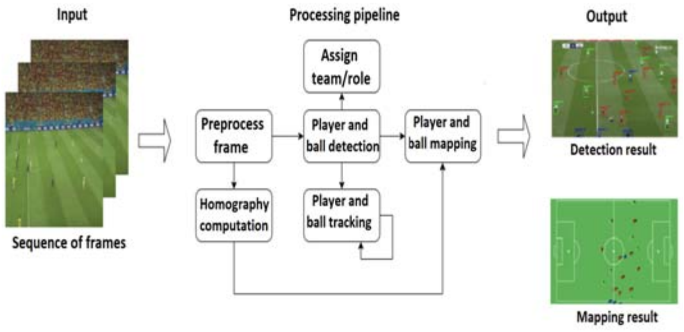


Fig. 1: Overview of the system

Similar work conducted in this domain brought contributions regarding different tasks. In [13], [16], [24], the subject of player detection and tracking was analyzed, while [8] is dealing with the homography transformation. In this paper, we are going to break down every module of the pipeline, explaining all the different methods and the reasons for choosing the respective solution, by presenting the results and testing metrics for each one of them.

The hardware resources required to gather data for this type of systems are expensive. This means that a high-resolution camera needs to be placed in a stable position, which should be able to capture the whole field. One of the research community obligations is to present solutions that can be further used by others, regardless of their limited technological capabilities. We accomplished to eliminate the resources problem by integrating efficient algorithms and coming with innovative solutions, in order to be able to use data from broadcast matches as system input.

Another aspect that we focused on is the capacity of the system to run in real time. Unfortunately, the researches that are currently available in this domain focus on solving the problem, but without taking into consideration the processing time required to analyze the data. In order for us to address this issue, the solutions were carefully chosen, paying attention to the trade-off between time and performance.

For the evaluation part, different metrics were designed for each module independently and experiments were performed on the pipeline, which provided optimistic results.

II. Related Work

A. Player and ball detection

The first implementations were based on color segmentation of the field and then extraction of information from the remaining blobs, such as in [10], [22]. The issue with these solutions is that they are sensitive when it comes to noise or edge cases, such as the players having the color of their jerseys similar to the color of the field. With the increasing popularity of modern object detection models, there is no surprise that we tried to come with a more robust solution and we preferred the approaches based on deep learning as in [4], [23], which tested different models on football-specific data sets. The conclusion that we have reached was that two stages models, such as masked R-CNN, are more accurate, but they are slower when compared to the YOLO models.

B. Player and ball tracking

The field of tracking, and more importantly of real-time tracking, is well studied. The most unfortunate problem that occurs in tracking, especially in the sports domain, is the problem of identity switching. Unlike pedestrians and vehicles that have predictable movements, in sports, changes of directions and occlusions happen very often. In [19], the tracking is done by extracting color information from the detected players and predicting the locations of the players in the next frames using the Kalman filter. In [3], the tracking of the players is done by using the SORT algorithm presented in [20] and the YOLOv2 model as detector. The SORT algorithm provides a good trade-off between the inference time and the performance, but in order to enhance its performance, we used a better detector.

C. Player and ball mapping module

The process of mapping objects between two different spaces is extremely common, not only in the sports industry. A homography matrix must be found in order to perform such operations. Often times, a semi-automated method is used in order to solve the homography matrix equation, or the methods rely on the fact that the cameras are keeping a fixed position throughout the game. Such approaches can be found in [12], [21], [5], [7]. In [9], in order to successfully compute the homography transformations, a deep network is used such that the features are extracted from the given frames, and then the homography is estimated. In order to extract more features, we used two different models: the line extraction model and the field segmentation model.

III. Implementation

A. Player and ball detection

The choice of a YOLO model was a natural one, given the fact that the detection must be as fast as possible and as accurate as possible. A downside in our case is that first YOLO models had a problem detecting small objects, or objects that are relatively close to each other. The model was chosen by looking at the trade-off between performance and speed, as shown in Table I. The data set used was the COCO dataset, because of the existence of the *person* and *sports ball* classes.

TABLE I: Performance comparison of different YOLO models [2], [11]

Model	Size	mAP	Speed(FPS)
YoloV3	608	33.0%	20
YoloV4	608	43.5%	23
Yolov5s	640	56.8%	98

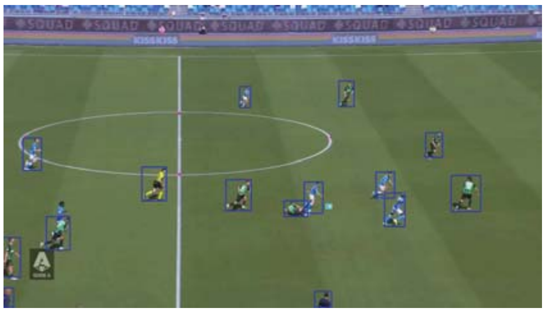


Fig. 2: Player and ball detection

Fig.2 illustrates examples of player and ball detection.

B. Player and ball tracking

The players and ball solutions were chosen differently, because of the different nature of each task. In a given frame, the number of players is equal or greater than one and there is only one ball object. The players tracking algorithm should support multiple object tracking and the ball tracker should deal with only one object.

For the MOT (Multiple Object Tracker) algorithm that is used for player detection, the SORT solution was a natural choice. We chose to go with an algorithm that uses the previous detection module, because of the good accuracy that is offered.

For the ball tracking mechanism, a layer of extra logic is needed in order to give a more accurate performance. In the context of football matches, the ball will be occluded multiple times by many players throughout the game. Also, because the YOLOv5 model was chosen in order to provide the detection, it will be hard to provide good detections when the ball size is small. This is why using the SORT algorithm will not be a good choice. Rather than depending on detections, an algorithm that uses ROI(Region of Interest) as tracking method should be chosen. The CSRT proposed in [14] algorithm works by training a correlation filter with compressed features and then looking in the near regions of the next frames in order to find a good match to the tracked object.

To come back to the occlusion problem, let's imagine the case in which a ball is totally occluded by a player. The algorithm will search in the near areas and will produce an error resulting in the loss of the object, or will most probably assign the tracking position to the part of the player that occluded the ball and will further track that part as considering it the ball. In order to avoid these situations, or other situations with the same outcome, a check is needed in order to catch these cases. At each detection, the difference between the distance of the position of the CSRT tracker and the detected bounding box will be compared to a threshold as follows:

$$\|\mathbf{bb}_{tr} - \mathbf{bb}_{d}\| < \epsilon_{ball\_distance}$$

where bb_{tr} is the centeroid of tracked ball bounding box at the current frame and bb_d is the centeroid of detected ball bounding box at the current frame; $\epsilon_{ball\_distance}$ is a constant value and was set to 5 in our work.

If the distance will be greater than the chosen threshold, the tracker will be reinitialized at the current detection position.

C. Player role assignment

The football game is played between two teams, each team having their players wear similar equipment. In a football field and near it, the following person roles can be found: players, referees, coaches, security personnel. Obviously, the players are the most important entities to detect, but the referees are also present in the football field as well as other additional personnel. If we think about a method in order to classify these roles, at first glance, the jersey color feature is the most obvious choice. In order to reduce the number of classes, we propose the following three classes:

- Players that belong to Team A
- Players that belong to Team B
- Extras

The *Extras* category is composed of the two goalkeepers of each team, referees, coaches and other personnel. Any of the following methods that are going to be discussed can be extrapolated to more classes.

The research was conducted in two directions, using supervised learning algorithms, or unsupervised algorithms.

1) Feature extraction: In order to classify the players, the detections obtained at the previous step are going to be fed to the classifier, after being resized to a constant shape of 180×90 pixels. The goal was to reduce the number of features and eliminate the noise that appeared from detections.

Noise in this case is given by the background that appears in each detection. This background has mostly the same color as the grass from the football field. In order to eliminate it, three approaches were followed: keeping the color space in RGB, converting it to HSV or converting it to Lab. After each conversion, depending on the color space, some threshold will be chosen in order to eliminate the green from the frames. The best results were obtained from the Lab model. We chose to threshold the values of the *a* channel, which ranges from (-128 to +127) and specifies the value of redness or greenness of the pixel. The chosen value was 0 and a mask was computed in order to determine the players region of interest.

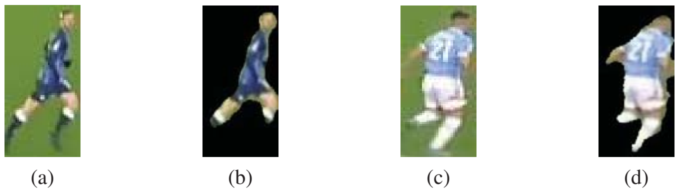


Fig. 3: (3a, 3c) Resized detection before performing the pre-processing step, (3b, 3d) Masked detection after applying the thresholding on the *a* channel

After the noise has been eliminated, we can reduce the number of features to be input in the classifier.

Our first try was to pass the mask through a pre-trained CNN, to be more specific in a VGG-16 architecture, in order to feed the PCA algorithm with the output features. The solutions performed poorly and the time that was gained from the CNN model was then lost in the PCA algorithm, in the process of feature refinement. Another drawback of this approach was the increased number of steps of the pre-processing pipeline.

The second approach was to compute a color histogram depending on the color space that was chosen. Both RGB and HSV color spaces were tested on. Even if this approach is promoted by so many papers, it has a big flaw that is scarcely discussed: the variations of color in the frame due to changes in illumination, posture of the players and other external factors. Another downside of a color histogram is that this representation leads to loss of information regarding how the R, G and B components of a pixel were originally related in its RGB representation. In order to address this issue, we kept the HSV converted frame and we computed a bi-dimensional histogram for each of the Hue and Saturation channels. The ranges were split into 10 bins, reducing the number of features from 16200 to 100.

2) Unsupervised learning: In order to predict the appartenance of a player to a team, an elegant solution would be to use an unsupervised learning algorithm that can classify the detection in the three given classes. Two algorithms were tested to perform this operation: K-means, DBScan.

The K-means will generate three centroids that will form the three clusters. The issue is that the centroids are chosen randomly at first and if, in a specific frame, the number of players from a team significantly exceeds the number of players of the other team, outliers will appear and the K-means algorithm will not be able to detect their classes.

On the other hand, the DBScan performs clustering based on spacial density. This means that it will perform better when encountering outliers and will be better suited for problems which have examples with more similar features. The algorithm performs poorly when illumination changes occur in the frame.

3) Supervised learning: A more robust solution would be to train a model such that, based on a training set, it will predict the label of each detection. In our case, the training set of each class will be represented by detections of a team's players.

Therefore, a suitable algorithm would be KNN (K-Nearest Neighbor), since the examples are closely related. Also, KNN is appropriate for small datasets, which is beneficial in our case, due to the fact that it is hard to extract a large dataset from a sequence of frames. If the data is gathered carefully the decision boundary can be learned in order to avoid edge cases or to successfully detect outliers.

The hyperparameter k signifies the number of neighbors to be taken into account when making a prediction and was set to 5, each neighbor having equal input into the decision making process.

D. Player and ball mapping module

The player and ball mapping module has the role to map the position of the detected player in a given frame, to a two-dimensional view point, which we are going to call the absolute position of the ball and players. In order to perform this mapping, the following steps must be performed:

- Field segmentation
- Line detection
- Homography computations

Both Field segmentation and Line detection models were implemented using the UNet architecture presented in [18]. The difference is made at the output layer, the field segmentation being a multi-class segmentation problem and the line detection a binary classification problem.

E. Field segmentation

This step is responsible with segmenting the frame received by the input sequence and outputting the masks for each of the declared labels. The labels that a pixel can take after segmentation are: field, big box,small box, center circle ,small circle, background (see Fig. 4).

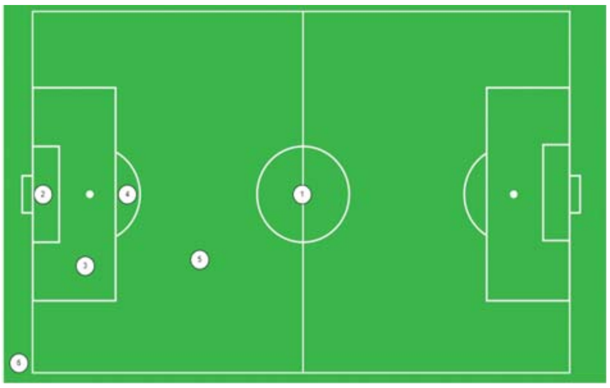


Fig. 4: Overview of field's regions of interest, where the following regions annotated by numbers are associated with the following classes 1 - *center circle*, 2 - *small box*, 3 - *big box*, 4 - *small circle*, 5 - *field*, 6 - *background*

The loss function used in the training process was the *categorical crossentropy*, and the optimization method used was *Adam*, with a learning rate of 0.01. The input is of shape $(batch\_size \times 224 \times 224 \times 3)$ and receives normalized data by dividing the given frame, element-wise, by 255. The batch size was set to 64. The output shape will be $(batch\_size \times 224 \times 224 \times 6)$, each pixel from each example having a probability vector consisting of 6 values, representing each one of the classes.

F. Line detection

The line detection module should predict a binary mask for the lines shown in figure 4.

The data is modeled as having two classes: pixel belonging to a line and pixel not belonging to a line. Having this in mind, the loss function chosen was the *binary cross entropy* the rest of the hyperparamters are identically set from the previous model. The output shape will be $(batch\_size \times 224 \times 224 \times 1)$, each pixel of each example having a probability value assigned to it, that represents the degree of belonging to line pixel class.

G. Homography computations

The homography module has the role to combine the two created sub-modules and to add the needed logic in order to compute the mapping between the original image and the absolute position of the interest object relative to the field.

In order to compute the mapping, three cases will be proposed, with respect to the field of view of the camera relative to the field. These three cases will be:

1) The middle line appears in the frame
2) The big box and the small box can be seen in the frame
3) The center circle and a part of the big box can be seen in the frame

Regardless of the case, the following steps will be done:

1) Get masks for each region of the field present in the frame
2) Compute the convex hull on each of the masks
3) Detect lines
4) Compute the lines using Probabilistic Hough Transform
5) Choose the case in order to find four points
6) Compute the homography matrix

In order to get the masks, the frame will be passed through the field segmentation model. The convex hull of each mask will be computed in order to get more information on each region of the frame. We can safely assume that if the area of a convex hull of a region is zero, then the respective class has no pixel attributed in the current frame.

The line detection step is similar to the field segmentation one, but this time the frame is passed though the line detection model. In order to obtain a parametric representation of the lines and to reduce the noise present in the detection, we are going to apply Probabilistic Hough Transform presented in [15] on the binary mask.

In order to successfully choose the case, the following questions must be raised: What does the computer see? What is unique in this scenario? By computing the two previous detections we have already gathered enough information in order to define the following case structure:

$$\begin{cases} \text{middle line case,} & \text{if } PA_{cc} > 0.02 \ \& \ B_{ml} = True \\ \text{big box small box case,} & \text{if } PA_{sb} > 0.02 \ \& \ PA_{bb} > 0 \\ \text{big box center circle case,} & \text{if } PA_{bb} > 0.002 \ \& \ PA_{cc} > 0 \end{cases}$$

where $PA_{cc}, PA_{sb}, PA_{bb}$ are the percentage ratios of the area determined by the convex hull contour of the center circle, small box, big box with respect to the entire frame; B_{ml} represents the presence of the middle line in the frame.

In order to determine if the middle line is present in the frame, we are going to compute the slope of each line and convert it to degrees, where we can safely assume the following formula:

$$85 < \|\theta_l\| < 95$$

where θ_l is the angle of detected line in degrees. The intuition behind this approach is that the camera is always positioned in the middle of the field, at some aleatory height. Therefore, the middle line will always appear in a vertical position.

1) Middle line detection case: In the middle line detection case, the four points that need to be detected are (see Fig. 6):

- bottom extremity of the center circle
- left extremity of the center circle
- right extremity of the center circle
- top extremity of the center circle or intersection point between the middle line and the top line.

It is trivial to find all four extremities of the center circle by detecting the extremities on the convex hull contour previously computed. An edge case that is often encountered is when the circle is partially hidden. In this case, we need to compute a projection of the right or left extremity with respect to the middle line. The point to be projected will be chosen by comparing the distance between the middle point of the field and the two extremities. If the difference between the two distances is greater than a set threshold, then the projection of the point with a smaller distance will be computed and the point will be introduced in the list of points that are going to take part in the homography matrix equation.

Lastly, in order for the homography to be as accurate as possible, the points used should be as far as possible one from another. This is why, if possible, the intersection of the middle line with the top line is preferred over the top extremity of the center circle (see Fig. 5).

In order to know what lines are eligible to be in the top lines set, the approach will be similar to the one of the middle line but the interval of the angle values will differ

$$175 < \|\theta_l\| < 185$$

The limits of the interval can be considered constant, because even if the camera moves, we are still in the case where the middle line is visible. Therefore, because the middle line is perpendicular to the top line and because the middle line's angle does not vary, the variation in the angle of the top line will also be small.

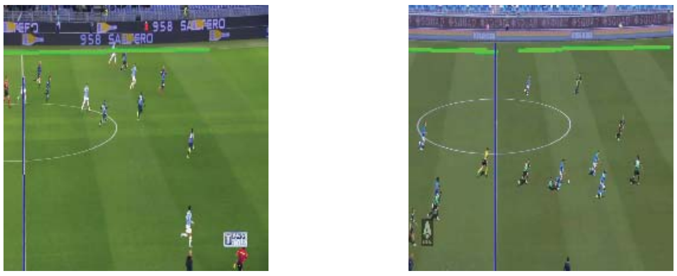


Fig. 5: Frames after middle line and top line detections

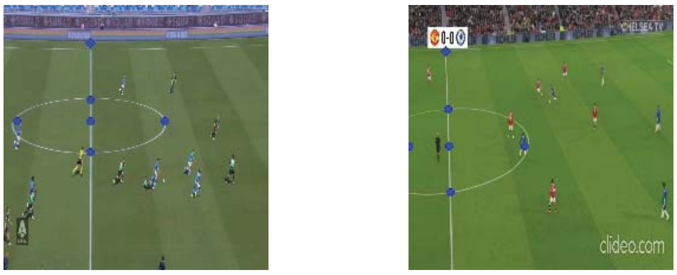


Fig. 6: Frames after keypoints detection - middle line case

Noise can appear from center circle ellipse, because the Probabilistic Hough Transform will sometimes detect the top part and bottom part of the circle as lines with a small norm. This can be solved by computing an exterior bounding box around the center circle and ignoring all lines that are present in the interior of the region.

2) Center circle and big box: The next case we are going to cover is when a part of the center circle and a part of the big box can be seen. The four points that are going to be computed are: the right or left extremity of the center circle, the top left or right point of the big box, depending on where the frame is positioned and the two projections of these point onto the top lines (see Fig. 7). The first step is to compute the extremities of the hull contour of the center circle and also to compute the bounding box around it, in order to eliminate the noise lines for the next step.

In the second step, the side of the field that is being shown will be found. For this, the lines obtained using the Probabilistic Hough Transform will be filtered with respect to their position, in order to be further processed. As mentioned before, some lines will be detected in the surroundings of the center circle. For this, any line that has an extremity in the center circle bounding box will be discarded. The angles of the lines that will pass will be saved.

K-means algorithm will be applied on the array containing the angles of each line, in order to classify the line by labeling them vertical or horizontal. The label that will be assigned to each centroid will be found by the following rules: if the centroid absolute value is closer to 180 degrees, then it will be considered the centroid for the horizontal lines, else it will be considered the centroid for the vertical ones. In order to determine if we look at the right or at the left side of the field, we are going to analyze the sign of the horizontal lines centroid. If it has a value greater than 0, then it will be assigned to the left side, otherwise to the right side.

The extremity of the big box will be computed as follows: if the left side of the field is shown in the image, then the extremity will be the top right of the contour of the big box, else the top left of the contour will be the extremity. Now, in order to reduce the noise, this point won't be introduced directly to the interest points. We are going to find the lines that compose the top of the big box and then intersect them with the closest line to the extremity that is included in the vertical lines. Of course that in order to find the top line of the bounding box, we need to look in the horizontal lines array.

Lastly, we need to compute the intersection of the perpendiculars to the top lines of both extremity points that we obtained. Unfortunately, this won't give us good results because of the perspective view of the frame. In order to solve this problem, for the center circle intersection with the top line, we are going to compute the intersection with the perpendicular and then an additional constant value will be added or removed from the x coordinates. The addition or the subtraction will be done depending on the left or right side of the field view. For the projection of the extremity point of the big box onto the top lines, the solution is simpler. Just compute the intersection of the found horizontal line and the top lines. Then, compute the mean of the coordinates.

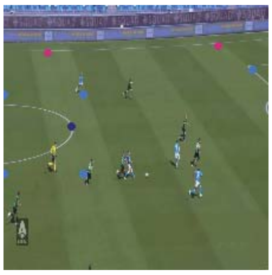


Fig. 7: Frames after keypoints detection and bounding box points - Center circle and big box case

3) Big box small box case: The last case to cover is when we can see the big box alongside the small box. The first step is to know the side of the field that we are looking at. In order to do so, we are going to compute the centroid of the contours of both small and big box. If the x coordinate of the big box centroid is greater than that of the centroid of the small box, then we are looking at the left side, else, we are looking at the right side. Likewise the previous case, we are going to compute the K-means of the lines angles.

The next step is to compute all the intersections between the horizontal lines that are in the interest region of the big box and the vertical line, because keypoints will surely be contained in the resulting points.

In the next step we can distinguish four cases, in order to know the points that we are going to append to the final array, in order to compute the homography (see Fig. 8).

- Right side of the field
    1) top extremity of the big box is available
    2) top extremity of the big box is not available
- Left side of the field
    1) top extremity of the big box is available
    2) top extremity of the big box is not available

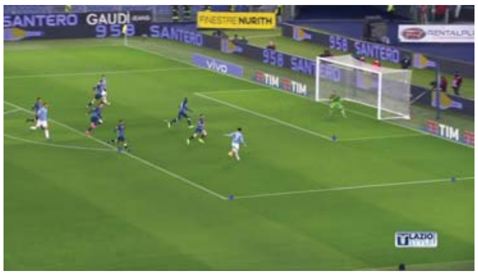


Fig. 8: Frame after key points detection. Big box and small box case

In other words, first we divide the problem in the left or right case of the field. Then, each case will have two sub-cases: if we see the top part of the big box or if we see the bottom part of it.

If we entered the right side top case, the goal is to find the exterior vertical line of the big and small box, but also the exterior horizontal line. In any given frame the top extremities will represent extremities of the box contours. In order to know what lines determine the boxes we are going to compute the intersection points between all vertical and horizontal lines and then the closest points to the contour extremities will be considered the real interest points.

Now that we have the exterior horizontal and vertical lines that delimit the big box and the small box, all we need to do is compute the intersection of the small box exterior lines with the ones of the big box and we can safely obtain the four desired points.

If we enter the second case, the process will be very similar, the difference will be that now we need to look at the bottom lines not the top ones.

In the left side top and bottom case, the computations will be similar, but now the x axis coordinates will be switched.

At the end of the method, the found points need to be returned with the correct coordinates in the original frame, so a scale correction must be performed because all the operations were performed with respect to the frames that were outputted by the two deep learning modules.

IV. Experiments

The experiments were done on 5 different matches from European championship. It is hard to perform an overall assessment of the pipeline, therefore we chose to isolate the experiments and results module-wise.

A. Player and ball detection

The evaluation was done by manually labeling the tested matches and performing the AP (Average precision) and AR (Average recall) with a threshold of 0.5.

TABLE II: Performance of the YOLOv3 and Yolov5 model of the player and ball detection

Model	Player AP	Ball AP	Player AR	Ball AR
YoloV3	0.873	0.919	0.846	0.524
Yolov5s	0.914	0.935	0.893	0.647

We compared the performances of two YOLO family models.

B. Player and ball tracking

In order to evaluate the player and ball tracking models, we are going to compute the mean MOTA (Multiple object tracking accuracy) that was presented in [1].

TABLE III: Performance of the player and ball tracking

Tracking Module	MOTA
Player tracking(SORT)	0.297
Ball tracking(CSRT)	0.226

C. Player team assignment metrics and results

TABLE IV: Performance of the player team assignment module

Classification Algorithm	Precision Team	Precision Extras	Recall Team	Recall Extras
kNN	0.876	0.852	0.861	0.834
k-means	0.674	0.573	0.642	0.560
k-means-FieldMask	0.719	0.684	0.698	0.653

An example of player detection and team assignment is presented in Fig. 9.

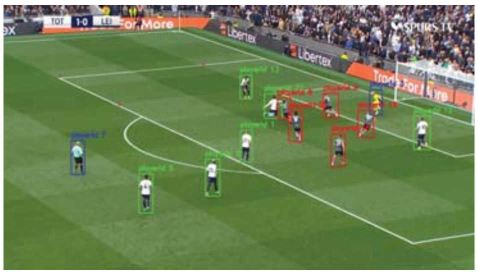


Fig. 9: Player team assignment and detection results

D. Positional player and ball mapping metrics and results

For the lines and field segmentation model, we compared the performances with a simple fully connected neural network that performs semantic segmentation. The training and test data set was taken from the data proposed in [6] and labeled by us accordingly.

E. Line detection model

TABLE V: Performance of line detection model

Model	Pixel Accuracy	IOU	Speed(ms)
UNet	0.917	0.191	101
FCNN	0.645	0.026	94

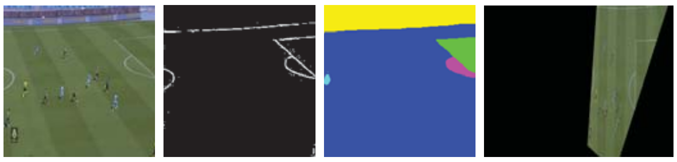
The IOU values are small since the lines of the field are extremely thin after frame resize, but this does not affect the final result, as we can see in Fig. 10b, 11b, 12b

F. Field segmentation model and homography computation

TABLE VI: Performance of field segmentation model

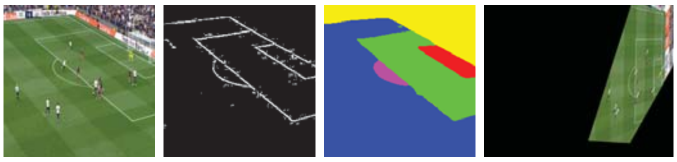
Model	Pixel Accuracy	Mean IOU	Speed(ms)
UNet	0.928	0.665	109
FCNN	0.878	0.359	103

The homographic representation of the original frame is obtained by combining the two segmentation models in the logic layer, where the homography matrix is computed using the key points that are found in one of the three cases presented (see Fig. 10, 11, 12).



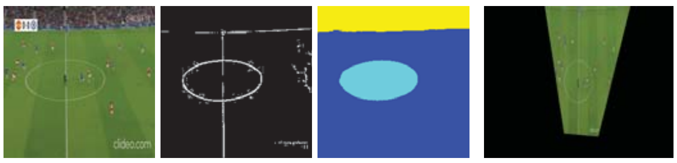
(a) Input frame (b) Lines detection (c) Field Segmentation (d) Homography transformation

Fig. 10: Example of homography computation pipeline (center circle and big box case)



(a) Input frame (b) Lines detection (c) Field segmentation (d) Homography transformation

Fig. 11: Example of homography computation pipeline (big box small box case)



(a) Input frame (b) Lines detection (c) Field segmentation (d) Homography transformation

Fig. 12: Example of homography computation pipeline (center circle case)

Lastly, we are going to show the mapping of the players and the ball on the football field from a bird eye view (see Fig. 13). This is obtained by putting together all the modules functionalities.

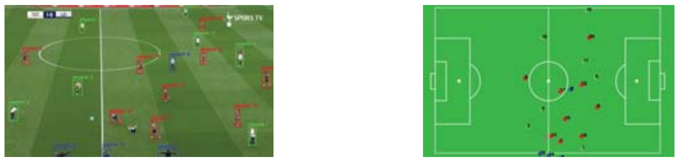


Fig. 13: Examples of player and ball mapping

V. Conclusions and future work

In this work we presented a robust pipeline designed to perform positional mapping and give a strong basis for the positional analysis work. We used modern solutions and more importantly, we did not ignore the trade-off between the performance and speed of each module. The metrics of each module yielded good results. The solution can be used on any given input sequence of frames that comes from broadcast football match.

We would like to refine this pipeline, especially on the mapping module side. Due to noises, distortions and other factors, the result of mapping the position of a player in a sequence of frames can seem unnatural (the movement of the player will not be homogeneous). As a future improvement, the state of the homography will be saved at each frame, in order to make a more accurate computation of the homography matrix.

References

[1] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[3] Matija Buric, Marina Ivasic-Kos, and Miran Pobar. Player tracking in sports videos. In *2019 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 334–340. IEEE, 2019.

[4] Matija Buric, Miran Pobar, and Marina Ivasic-Kos. Ball detection using yolo and mask r-cnn. In *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 319–323. IEEE, 2018.

[5] Dirk Farin, Susanne Krabbe, Wolfgang Effelsberg, et al. Robust camera calibration for sport videos using court models. In *Storage and Retrieval Methods and Applications for Multimedia 2004*, volume 5307, pages 80–91. SPIE, 2003.

[6] Mehran Fotouhi, Sadjad Fouladi, and Shohreh Kasaei. Projection matrix by orthogonal vanishing points. *Springer, Multimedia Tools and Applications*, 76(15):16189–16223, August 2017.

[7] Ankur Gupta, James J Little, and Robert J Woodham. Using line and ellipse features for rectification of broadcast hockey video. In *2011 Canadian conference on computer and robot vision*, pages 32–39. IEEE, 2011.

[8] Namdar Homayounfar, Sanja Fidler, and Raquel Urtasun. Sports field localization via deep structured models. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4012–4020, 2017.

[9] Namdar Homayounfar, Sanja Fidler, and Raquel Urtasun. Sports field localization via deep structured models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5212–5220, 2017.

[10] Yu Huang, Joan Llach, and Sitaram Bhagavathy. Players and ball detection in soccer videos based on color segmentation and shape analysis. In *International Workshop on Multimedia Content Analysis and Mining*, pages 416–425. Springer, 2007.

[11] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, Christopher-STAN, Liu Changyu, Laughing, tkianai, Adam Hogan, lorenzomammana, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Francisco Ingham, Frederik, Guilhen, Hatovix, Jake Poznanski, Jiacong Fang, Lijun Yu , changyu98, Mingyu Wang, Naman Gupta, Osama Akhtar, PetrDvoracek, and Prashant Rai. ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements, October 2020.

[12] Hyunwoo Kim and Ki Sang Hong. Soccer video mosaicing using self-calibration and line tracking. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 1, pages 592–595. IEEE, 2000.

[13] Jia Liu, Xiaofeng Tong, Wenlong Li, Tao Wang, Yimin Zhang, and Hongqi Wang. Automatic player detection, labeling and tracking in broadcast soccer video. *Pattern recognition letters*, 30(2):103–113, 2009.

[14] Alan Lukezic, Tomas Vojir, Luka ˇCehovin Zajc, Jiri Matas, and Matej Kristan. Discriminative correlation filter with channel and spatial reliability. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6309–6318, 2017.

[15] Jiri Matas, Charles Galambos, and Josef Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer vision and image understanding*, 78(1):119–137, 2000.

[16] Kenji Okuma, Ali Taleghani, Nando de Freitas, James J Little, and David G Lowe. A boosted particle filter: Multitarget detection and tracking. In *European conference on computer vision*, pages 28–39. Springer, 2004.

[17] Charles Perin, Romain Vuillemot, and Jean-Daniel Fekete. Soccerstories: A kick-off for visual soccer analysis. *IEEE transactions on visualization and computer graphics*, 19(12):2506–2515, 2013.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[19] Yongduek Seo, Sunghoon Choi, Hyunwoo Kim, and Ki-Sang Hong. Where are the ball and players? soccer game analysis with color-based tracking and image mosaick. In *International Conference on Image Analysis and Processing*, pages 196–203. Springer, 1997.

[20] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[21] Akihito Yamada, Yoshiaki Shirai, and Jun Miura. Tracking players and a ball in video image sequence and estimating camera parameters for 3d interpretation of soccer games. In *2002 International Conference on Pattern Recognition*, volume 1, pages 303–306. IEEE, 2002.

[22] Junqing Yu, Yang Tang, Zhifang Wang, and Lejiang Shi. Playfield and ball detection in soccer video. In *International Symposium on Visual Computing*, pages 387–396. Springer, 2007.

[23] Xinyi Zhou, Wei Gong, WenLong Fu, and Fengtong Du. Application of deep learning in object detection. In *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, pages 631–634. IEEE, 2017.

[24] Guangyu Zhu, Changsheng Xu, Qingming Huang, and Wen Gao. Automatic multi-player detection and tracking in broadcast sports video using support vector machine and particle filter. In *2006 IEEE International Conference on Multimedia and Expo*, pages 1629–1632. IEEE, 2006.