Internship Report at Sphaera

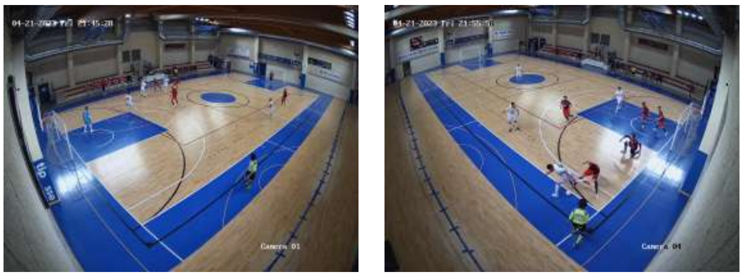
Gianluca Covini

September - February 2024

Introduction

The internship took place at Sphaera, an artificial intelligence startup specialized in the development of innovative computer vision techniques for the analysis of sports statistics and performances. Specifically, my project involved the development of an algorithm for identifying and tracking the players and the ball from recorded videos of a 5-a-side football match.

The starting point were videos of an indoor 5-a-side football match. The videos had been recorded by two fixed wide-angle cameras positioned at the right and left ends of the field. In order to produce a valuable product the cameras used were cheap cameras that eventually produced a distorted image that could not be analyzed with existing tools. Therefore, my aim was to develop a completely new algorithm that would allow the identification of the players and the ball, and the extraction of statistics.



(a) Left camera (b) Right camera

Figure 1: Views from the two cameras

The project roadmap was lined up as follows:

- Identification of players through bounding boxes;
- Tracking of the found bounding boxes;
- Team recognition;
- Merging algorithm for the two cameras;
- Ball detection;
- Projection onto the 2D view;
- Calculation of main statistics (ball possession).

Since it addressed an existing, open problem, the work was configured as a research project, where the development of the algorithm involved various skills acquired during my studies, particularly regarding programming, machine learning, computer vision, and statistical computing.

The internship activity took place remotely: the work was mainly conducted autonomously under the guidance and supervision of Mirko Messori, CTO of Sphaera, with whom I organized weekly video calls.

1 Players detection

The first step concerned the identification of the players. To do this, after a review of the literature, we decided to use the YOLOv3 network: a pre-trained neural network for the identification of the classes of the COCO (common objects in context) dataset. The network takes in account an image (as a 3d tensor) and gives back a list of bounding boxes, i.e., vectors containing the coordinates (in pixels) of four points, vertices of a rectangle enclosing the identified player, a class number, which identifies the object detected, and a confidence level, which determines the acceptance or rejection of the identified bounding box.

Specifically, we loaded the model using the Python library `opencv`, and applied it to the videos under analysis, filtering the results based on the returned accuracy and the type of the object. In particular, we limited ourselves to the *person* and *sports ball* classes.

The first result we noticed was the network's excellent performance in identifying people and, conversely, poor results in detecting the ball, which we set aside for the time being.

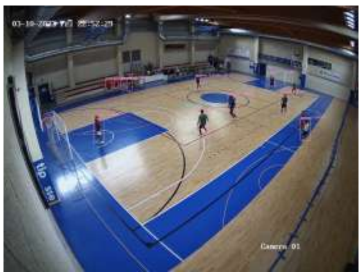


Figure 2: Results of the Yolov3

To improve the identification of players, we then added filters on the coordinates to eliminate the detection of people outside the field's range.

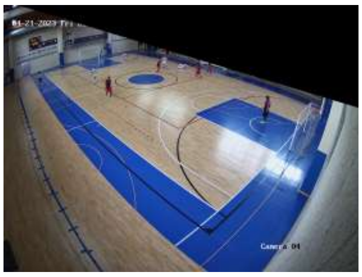


Figure 3: Example of a filter in the upper part of the field

In order to have more robust and accurate bounding boxes we implemented a non-maximum suppression through the OpenCV library, which unifies overlapping bounding boxes which identify the same object.

2 Team identification

A significant part of the work then focused on the team identification. Specifically, we approached the task as a classifing problem: the feature data were the color of the uniforms of people detected and the output was one of the five classes: Team 1, Team 2, Referee, Goalkeeper 1, Goalkeeper 2.

The image data corresponds to a 3d tensor, where each element reminds of a pixel whose colour was determined by a vector of three numbers in RGB coordinates

To address this type of problems various approaches are possible. For example, a possible solution we thought about is a ML-based classificator: in particular, we thought about implementing a 5-means clustering, taking in account the mean and the variance of the pixel colours for each bounding box.

However, we decided to follow a more classical computer vision solution: for each class we manually identified a range of colours (in RGB coordinates), and we wrote a script that counted for each bounding box the number of pixels in each of the established ranges and assigned the class with the highest number of corresponding pixels. The level of confidence for the classification was determined as the percentage of pixels in the bounding box within the selected range.

To promote classification uniformity, we adopted a post-processing technique: subtracting the background (for which an image of the field without players was used, as the cameras are static) from the original image.

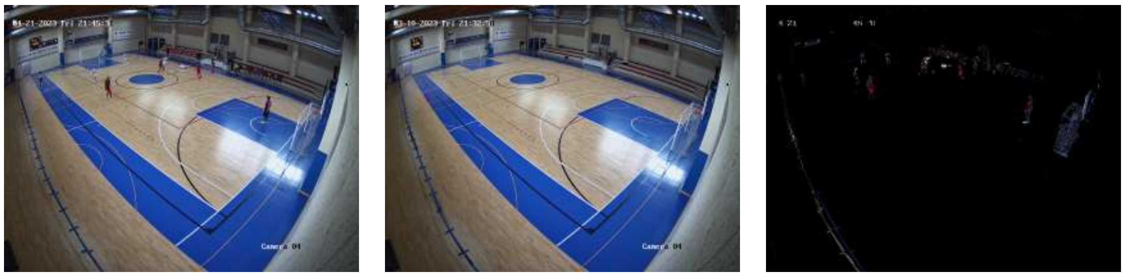

(a) Frame before background removal (b) Background image (c) Frame with removed background

Figure 4: Background subtraction example

In general, the team identification step required a long fine tune of the hyperparameters of the team range, which was mainly conducted manually. The result was indeed costly in term of time to find the good range and did not reach a high accuracy, in particular while classify the goalkeepers and the referee. These problems led us to explore another approach based on neural networks that we will discuss in the "open points" section.

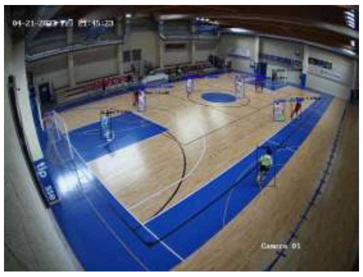


Figure 5: Example of team detection

2.1 Validation of the first two phases

The completion of these first two phases allowed for an initial validation of the work through the use of data of players and teams previously tagged by hand through the tool *Label Studio*.

To allow the comparison of the output of our script with data from label studio, it was necessary to define a data structure for our identification that was easy to handle.

We chose a Python dictionary, with the frame numbers as keys and, as values, list of lists, containing, for each bounding box: the list of the four coordinates, the team label and the confidence level of the team.

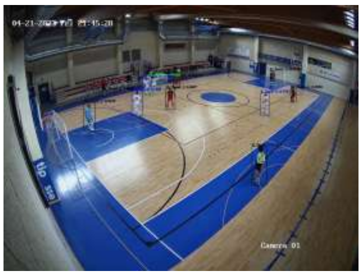
Consequently, we had to write a script that converted the annotations from the label studio format to our new data structure. Furthermore, to augment the validation data, we wrote a script that calculated the linear interpolation between each two vertices of two consequent bounding boxes, in order to obtain all the interpolated bounding boxes between two tagged manually.

The validation was then made by comparing the bounding boxes found by our algorithm and the one manually tagged and looking which ones overlapped: we considered bounding boxes of which the area overlapped a manual identified one for more than a certain percentage threshold, and, in that case, it was considered correct if the two identified teams were corresponding.

3 Players tracking

Once individuated the players, a side goal was to implement a first base for a tracking algorithm that keeps track of the detections. In particular, a tracking algorithm assigns a label to each detection and mantains the same label in the following frames if the object detected is the same. This step is useful in the perspective of detecting the player's number and calculate statistics over the single player: in fact, in that case it would be necessary to associate the detections of a single object over the entire video.

To do so, we tried two different libraries: SORT and DeepSORT. Both of them provide a tracking feature that takes in account the object detections made by the YOLO neural network and elaborates them assigning to each a number that should be the same if the object indentified is the same in different frames. The main difference between SORT and DeepSORT is that the DeepSORT library uses deep learning in order to better handle occlusions and different viewpoints. After some tries we realised that the DeepSORT, as expected, gave better results, so we adopted it. The implementation of the DeepSORT depends on three hyperparmeters: the maximum age of a track before it is terminated, the minimum number of hits required to establish a track and the Intersection Over Union (IOU) threshold for comparing bounding boxes. The baseline of the tracking was implemented by tuning these three parameters.


(a) First frame

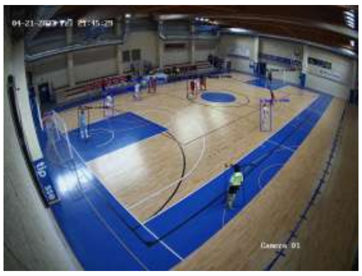

(b) Second frame

Figure 6: The number over the bounding boxes indicates the tracking ID

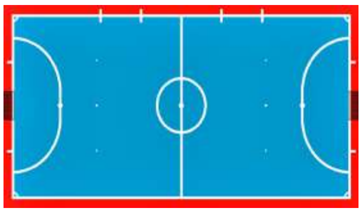
4 Bird-eye view

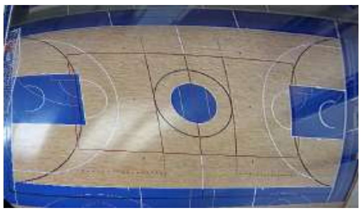
An important passage of the algorithm was the 2d projection of the found points, to build a "bird-eye view" of the match. The goal was to represent the players and the ball as points on a 2d image of the pitch. The bird-eye view is at the base of the statistics calculation, in fact it allows to have a representation of the players' and ball's movements without the distortion of a of the 2d projection (the video) of 3d objects. Furthermore, the 2d coordinate representation, being indipendent from the point of view, gave the possibility to merge the information of the left and right camera and also to use the euclidean distance on the plane to calculate statistics (useful for example to detect a ball possession or the run distance).

In order to achieve this, we used some tools from the Python library OpenCv. Given the coordinates of the bounding boxes on the original video, found in the previous steps, we projected it on a 2d pitch image through a matrix transformation. The matrix was obtained using the function cv2.getPerspectiveTransform() which takes in account 4 reference points from the 3d image and 4 points from the 2d image nad builds the matrix that sends the coordinates of each 3d point to the corresponding 2d point. The obtained points were then represented as small circles with color based on the team class.

In order to find the reference points we developped a debugging script that compared the bird-eye view of the empty pitch with the background image used for the 2d representation.

The result of this process is a video with a pitch image in the background and the moving dots representing the players.


(a) Background image used for the bird-eye view


(b) Bird-eye view of the empty pitch from the left camera

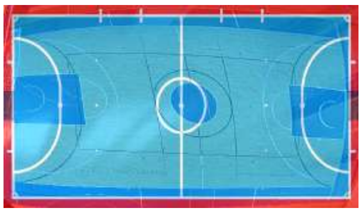
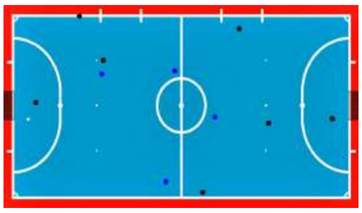

(c) Comparison of the two

Figure 7: Bird-eye view projection


(a) Bird-eye view result

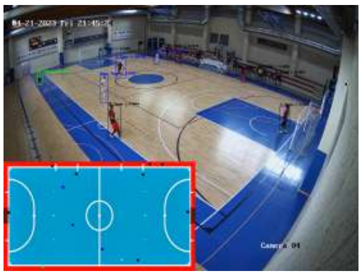

(b) Bird-eye view compared with camera view

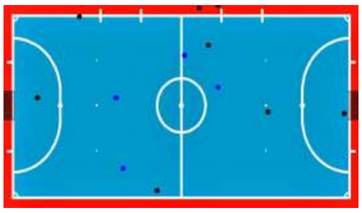
Figure 8: Bird-eye view results

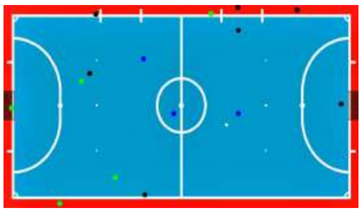
4.1 Merge of the two cameras

The bird-eye view made it possible to create a merging algorithm for the two cameras. In fact, the 2d projections from both cameras had the same point of view, so it became possible to compare the results of the two cameras. So, in order to obtain more robust data, we developed a merging algorithm. It took in account the 2d coordinates of the detections from both videos and combined them by taking the union of them. In particular, the algorithm considered two players identified by the different cameras as the same when the projected coordinates were nearer that a certain euclidean distance threshold. Whenever it found more than one detection with a distance lower than the threshold, it took the nearer one. Whenever it found the same player detected by the other camera it took the average coordiantes for the player; if it did not find a corresponding palyer it took the coordinates from one camera.

To obtain a better result and avoid problems related to occlusion and contrasts the merging of two players data was made only if the two players were identified of the same team by the two cameras.

The results were the coordinates of the players from both cameras, merged when the player identified was the same.


(a) Bird-eye view from left camera


(b) Bird-eye view from right camera

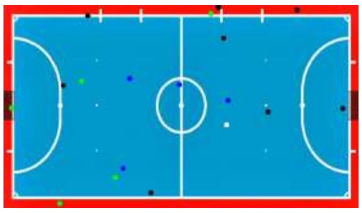

(c) Merged bird-eye view

Figure 9: 2d merge

5 Ball detection

Another important step consisted in the algorithm for the ball detection. We already noticed that the same approach of the player detection, using pretrained Yolov3 didn't work, so we had to try different approaches. We first tried a classical computer vision solution, based on a filtering process. We decided to work on black and white imagines and take advantage of the fact that the ball has a clear white color. First of all, we applied a mask to remove the background and set to black all the zones outside the pitch (similarly to what we did for team identification). Moreover, we set to zero also all the bounding boxes of the players, enlarged of a certain percentage so that it included the whole player. The aim of this preprcessing steps was to make sure that the ball was the only white object on the image. The individuation of the ball was then conducted by using the cv2.findContours() function which individuates all the objects with contours in the frame and taking the one with the biggest area (in order to eliminate most of the noise). Actually this first approach did not obtain the expected results, since it was still heavily influenced by the noise and the part of players that exceeded the bounding box. Moreover, it gave problems calculating when a player was in ball possess because the preprocessing steps eliminate also the ball when inside a player bounding box.

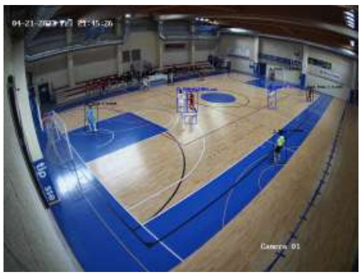
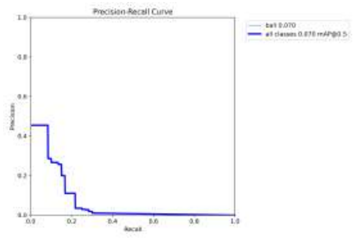


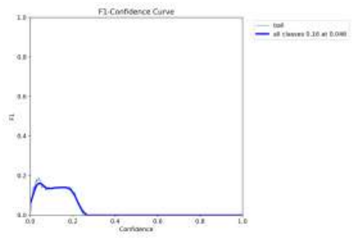
Figure 10: Often the ball (here the blue rectangualar) was confused with some reflections of the light on the pitch

In the end we decided to give up this approach and try a neural network solution based on the training of the Yolov8 network on custom data. In order to produce the data we needed we used the Label Studio tagging tool and tagged the ball for about 250 frames for each camera. We then tried different times the training of the network and found good results when the tags were near the ball not moving too much. In order to augment the data we also prepared a ball interpolation function that gave the interpolating bounding boxes of the ball, but was not indeed used since it has revealed to be not necessary.

Also for the ball we projected the found bounding boxes in 2d to obtain a bird-eye view and used a merging algorithm to make the detection more robust. In particular, taking advantage of the fact that we knew that we have only one ball on the pitch at a time, in the merging algorithm we took only the ball detected by the two cameras with the higher accuracy, which meant, in the majority of the cases, the ball detection from the nearest camera.



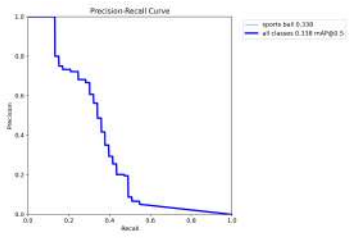
(a) Precision-recall curve



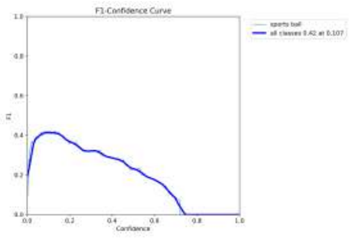
(b) F1 score

Figure 11: Results of the first neural network training for the left camera

The training of the neural network highly benefited of the augment of the number of data. Figure 11 show the results for about 100 tagged samples, of which about 75% used for training and 25% for validation. Figure 12, instead, show the latest results of the neural network training, after having tagged about 250 samples, and focusing on the most stable ones where the ball was not moving, which are also the most important when calculating ball possession. As we can see both the areas under the precision-recall curve and the F1 score became much bigger, meaning a more robust and accurate model.



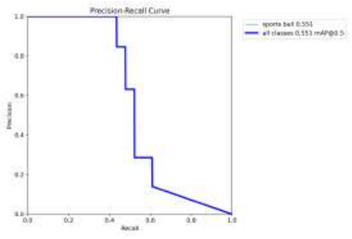
(a) Precision-recall curve



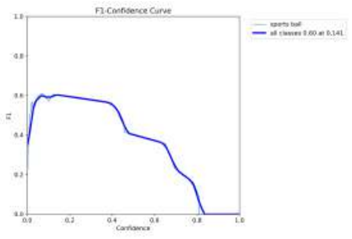
(b) F1 score

Figure 12: Latest results of the neural network training for the left camera

For the right camera we even obtained better results, due probably to the fact that in the training frames the ball was nearer to the right part of the pitch.



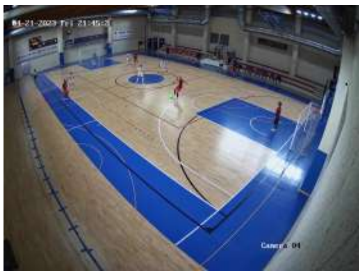
(a) Precision-recall curve



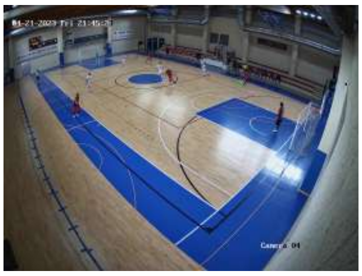
(b) F1 score

Figure 13: Latest results of the neural network training for the right camera

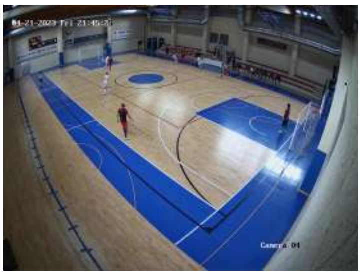
The results obtained in the end were good: the network finds the ball the majority of the time, with some noise but still being quite robust overall.



(a) The ball is correctly found even when there is reflection



(b) The ball correctly found



(c) Sometimes the results are a bit noisy

Figure 14: Ball detection

6 Possession calculation

The final part was the extraction from the detection of one of the most important overall statistics: the ball possession. We say that a team is in possession of the ball if the last touch of the ball was made by a player of that team. We wrote a script that considered a team in possession when the ball was detected withing a certain distance from a player of that team. In particular the distances where calculated using the 2d projected points and whenever we had more than one player under the threshold distance we considered the nearest one. The possession was then calculated as the percentage of frames a team has been in possession until the present time.

7 Video creation

The overall result of the project was the elaboration of a complete match. The output video is formed by the two 3d video from both cameras with the detection of players, teams and ball. In the bottom we overlaied the bird-eye view video and on the top the ball possession percentage. The video composition was made using ffmpeg commands.

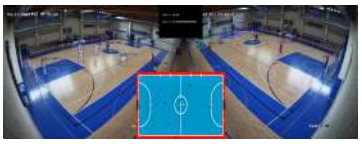


Figure 15: Frame from the final result

Further developements

Further developements involve technical imporovements that can still be done to the resulted code. In particular, the code would become more dynamical and easy to manage by switching to a object-oriented programming; in particular, an idea was to define the class BoundingBox.py with the attributes label, team, confidence, x, y, width, length.

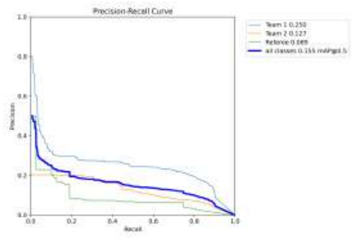
Another necessary developement is the creation of a pipeline that followed the entire creation of the overall video by giving in input the two videos from the cameras.

The overall code, moreover, need some improvements under the technichal point of view of speed which can be achieved by implementing the use of parallelization in the algorithm.

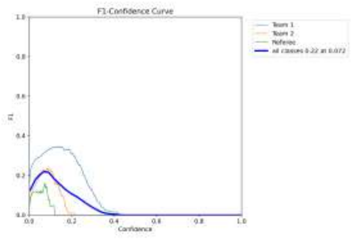
At last, in order to fasten the execution and make usable the algorithm a process of reduce the dimension of files would be useful, since a complete match analysed occupies about 6.5 GB.

Conclusions and open points

The project was conducted in a comprehensive way and arrived to a conlusion, which is the complete elaboration of a match with detection of players, teams, ball and ball possession calculation. However further developement are possible: one of the biggest problem of the so-developed algorithm is the low accuracy in the team detection. A possible alternative approach that should be investigated is the use of the Yolov8 NN trained on custom data with manually tagged teams. This approach was already tried during my internship, but it still need a deeper study with more data.



(a) Precision-recall curve



(b) F1 score

Figure 16: Neural network training results

As we can see from the plots the results are not great for now but probably augmenting the training data would be beneficial for the results.

Furthermore, also the ball detection algorithm could be improved by training the neural network on more and more various data.

Since the result gives an overall good behaviour but still is affected by a lot of noise, we would need probably some post-processing techniques in order to have more stable results, which would result in more reliable statistics.

The following step in the match analysis is the detection and stat calculation for the single player, identified by his player number, so that would be possible to calculate statistics on the single player: such as the average speed.

Another direction is the build of a model that identifies also the types of actions performed by the players (such as a goal, a passage, a save. . .).

Appendix: Technical considerations

The code was all written in Python, with and extensive use of the libraries named in this report. During this work I worked on a AWS virtual machine which gave me the opportunity to face for the first time also some more informatic and technical challenges.