Detection of football player position on the field map using TV stream

Project Report

by Chloé Daems, Anne-Claire Laisney and Amir Mahmoudi

Abstract

Tracking and identifying players in sports videos from TV streams that zoom and move during the match has many applications. Indeed, live statistics in sports are important for the coach, the players, the journalists, and all passionate supporters, to analyze the game. To get those statistics, the first step is to create the best model to detect the players, the ball, and their live positions on the field. However, it is a difficult task. Indeed, the complexity comes from the fact that images are not structured data, and getting the computer to identify different parts of an image is difficult.
In this project, we want to ***create a first pipeline that can identify the player's position on the field. We added a constraint that is not to use any deep learning methods or pre-trained models****.*
This paper is organized into three main parts. First, we want to detect the players on the image. To do so, we used the SoccerPlayerDetection bmvc17 v1 public data set *and a common histogram of gradients (HOG) descriptor followed by an SVM. Secondly, we want to detect the edges of the field with a Canny Edge Detector method. Finally, we have to create a homography estimation to position the detections on a field map. To do so, we used our local CPU.*
This paper is mainly based on "Learning to Track and Identify Players from Broadcast Sports Videos" by Lu and al*, where we tried to implement some of their ideas.*

Keywords— Player detection, Field Detection, HOG detector, Canny Edge Detector, SVM, Soccer, Homography

1. INTRODUCTION AND MOTIVATION

Computer Vision is an interdisciplinary field that aims to automatically process and understand digital images from surveillance cameras, movie cameras, personal cameras that we can find everywhere now with the multiple streaming platforms (YouTube, Daily Motion, Instagram, TV ...). Many information sources can be exploited to simplify tasks.

The different businesses are interested in computer vision and specifically in image recognition. Indeed, with image recognition, you can, for example, identify if there is an anomaly in your store, if someone is stealing something, or if someone that isn't authorized in a place is in it. It can also yield massive information, like for a business in the fashion industry to scrap Instagram and identify what products are in the tendency to adjust their strategy.

Another application, which is the one we will exploit in the project, is **image recognition in sport and more specifically in football**. Data is now everywhere and is becoming more and more important in every sector including in sports. Having statistics on both the team and the player is really important for the coach to adjust its training and winning plan. To get them, there are multiple possibilities: we could directly put a tracker on the players or use video content. We chose the latest to implement and try to do the first step in gathering football data.

The critical part in image recognition is that image is, by nature, an unstructured type of data. We need to structure it to use a machine learning algorithm like the SVM. We added a constraint that doesn't use a derivative of a neural network or pre-trained model. The task will be divided into **three parts: getting the players' positions on the frame, extracting clear edges from the field, and finally estimating the homography matrices**. We will use a classic state-of-the-art solution for those tasks, with a HOG descriptor and an SVM for the player detection, a simple canny edge for the edges of the field, and some math for the homography estimation.

In this project, we want to take hands-on image recognition in a field that we like, football. We use the public SPD data set to train the HOG detector.

2. PROBLEM DEFINITION

The main problem we are trying to solve is the detection of the players on the field, from video inputs with different scales. This is binary image classification. This means it is an image considered as a player or not. Our data set is composed of 1,492 football images with their corresponding player bounding boxes. We created a new data set with 300 cropped images of the players and 300 random $128x64$ cropped images from each shot. We call them the *positive samples and the negative samples*. Hence, our data set for the player detection is composed of inputs $\{x_i, y_i\}, i \in \{1, ..., N\}, y_i \in \{0, 1\}$, with x_i the i-th review, y_i it's corresponding label and x_i of shape (128,64).

This format is perfect for the classical combination of HOG descriptor and SVM. To fit the SVM (Support Vector Machine) we use the classical loss of an SVM called the **SVM/hinge loss** that we want to minimize :

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \quad (1)$$

To evaluate our model, we will split our data set into a train and a test set. We will be able to see the accuracy and the f1-score of our SVM model. However, our evaluation will also be directly on the output image on an actual shot of a football match to see the model's performance.

3. RELATED WORK

3.1. SOD-MTGAN [6]

In this article, they introduce an end-to-end multitask GAN (MTGAN) to detect small objects in unconstrained scenarios. In the MTGAN, the generator upsamples the small blurred ROI images to fine-scale clear images, which are passed through the discriminator for classification and bounding box regression. To recover detailed information for better detection, the classification and regression losses in the discriminator are propagated back to the generator. Extensive experiments on the COCO data set demonstrate that our detector improves state-of-the-art AP performance in general, where the largest improvement is observed for small-sized objects.

It is a really interesting paper that could help improve the time complexity of a player position detector pipeline. However, as we can't use deep learning in the project, this is not applicable.

3.2. Deformable Part Model [7]

This paper describes a discriminatively trained, multi-scale, deformable part model for object detection using the PASCAL data set. They introduced a general framework for training SVMs with latent structure. They used it to build a recognition system based on multiscale, deformable models. Their system uses a scanning window approach. A model for an object consists of a global "root" filter and several part models. Each part model specifies a spatial model and a part filter. The spatial model defines a set of allowed placements for a part relative to a detection window, and a deformation cost for each placement.

The score of a detection window is the score of the root filter on the window plus the sum over parts of the maximum over placements of that part, of the part filter score on the resulting subwindow minus the deformation cost.

To learn a model they defined a generalization of SVMs called latent variable SVM (LSVM). An important property of LSVMs is that the training problem becomes convex if we fix the latent values for positive examples. This can be used in a coordinate descent algorithm. LSVMs allow for the exploration of additional latent structures for recognition.

An initial root filter is generated from the bounding boxes in the PASCAL dataset. The parts are initialized from this root filter. Now, as we are more interested in detection from a far view, we think a simple HOG detector will be sufficient for our project. However, for further upgrading, this could be a solution.

3.3. Track and Identify Players from Broadcast Sports Videos [1]

In "Learning to Track and Identify Players from Broadcast Sports Videos" by Lu and all [1], we have a complete pipeline for sports stream player detection and tracking. Even more, it can detect the faces of the player. They use a canny edge to detect the field's lines and then with the player detection, they remove the players from the field to have a better view of the lines. To estimate the homography, instead of using a SIFT descriptor to detect the interesting points, they chose to match the court model with the edges of the video frames. Then, they use the Iterated Closest Points (ICP) to estimate the homography.

Their approach to detect the players is to use Deformable Part Model (DPM), which will detect different features of the player to detect them.

This article is really interesting for us, as they do exactly what we want to do and even more. As our data set is less zoomed than theirs, we, however, think that a HOG + SVM detector might be sufficient for the player detection.

4. METHODOLOGY

Our project is divided in three main parts : Player detection, Field detection and homography matrices estimation.

4.1. PLAYER DETECTION

To detect the player, inspired by [1] we used a HOG descriptor followed by a classical Support Vector Machines (SVM) model using the open CV library[3].

Figure 1: Example of an image pre-processed using the HOG Descriptor

4.1.1 Pre-processing

First, we pre-processed the data set to apply the HOG descriptor. The data set [4] that we have contains direct shots of a football match with the coordinates of the players. However, this input is not right for what we want to do and corresponds more to an input for a deep neural network. To feed the SVM we need to get the shape, features description of what a player looks likes. As we want to feed a binary classification algorithm, we have to get a data set with positive and negative samples.

To get the positive samples, we cropped the football shots where the players are, giving us for 1,492 images approx. 30,000 players cropped images of size (128x64). We decided to reduce the number of positive samples by taking only 1,000.

Now, for the negative samples we decided to take random coordinates on the different shots, with one random cropped image per shot. We are taking into account the probability that the random did not crop a player. For the train set, we decided to take only 1,000 negative samples for a total training set of 2,000.

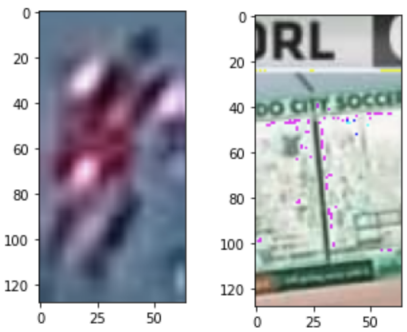

Figure 2: Example image from our positive samples (left) and negative samples (left)

4.1.2 HOG descriptor

The Histogram of Oriented Gradients is a feature descriptor that takes an image in input and outputs a feature vector that is more usable for training. The technique counts occurrences of gradient orientation in the localized portion of an image and focuses on the structure or the shape of an image.

For our project, we used the HOGDescriptor method from the *OpenCV* library that is a pipeline to get the HOG Descriptor from an input image. We used the default parameters for the HOG, with a window size of (8,8) which optimizes the trade-off between the computational time and the performance. We detailed the pipeline of a classical HOG method in fig. 4.

We extracted the HOG description of the positive samples and the negative ones giving us for each image a vector of size (3780,1). In fig. 3, you can see the HOG description of a player.

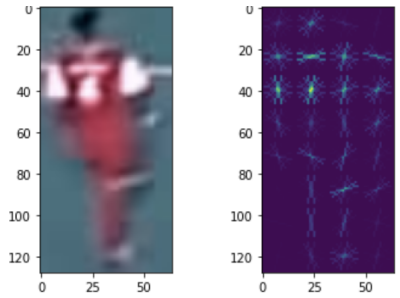

Figure 3: HOG descriptor of a player

4.1.3 Support Vector Machine (SVM) Model

To implement the SVM, we used intern SVM from the Open CV ML library. It is a classical binary classification task, SVM is the normal choice as there is a clear separation between the two classes and we have more features than samples. After a few fine-tuning, we resulted with the following parameters :

- Degree = 3
- Gamma = 0
- Linear Kernel
- epsilon = 0.1

To keep our model improving, we decided to double the training with, at mid-training, an addition of the ”hard negative samples”. The hard negative samples are the negative samples that were, at mid training, predicted as a player, so we add them a second time to the data set so that the SVM tries to categorize them better.

4.1.4 Test our classifier

To test our classifier we, on one hand, predicted the classes of the test set (see the result section) and on the other hand, tried it on an actual football frame with the detectMultiScale method from openCV which crops multiples times the input image to detect the players. Once we have the bounding boxes, we apply a Non-Max suppression algorithm to delete the bounding boxes that are overlapping to make only one.

To ameliorate our classifier, we created a ”jersey color detector” which will get the proportion of each team color in the bounding boxes. The advantage of this method is that it detects the team of the player - which is important information - and that it helps detect the false positive on the frame and delete it.

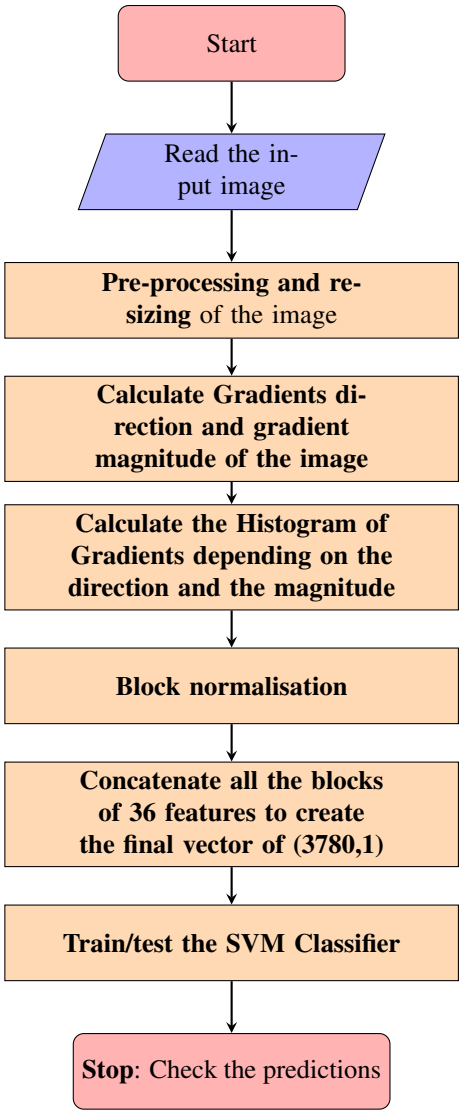

Figure 4: Workflow of player detection using HOG and SVM

4.1.5 Flow Chart

In fig.4 you can see our pipeline for the HOG + SVM training and testing.

4.2. FIELD DETECTION

In this part of the project, we want to detect the edges of the field cleanly so that we can use them for homography estimation.

4.2.1 The Canny Edge detector

Edge detection refers to the process of **identifying and locating sharp discontinuities in an image**. A Canny edge detector is a multi-step algorithm to detect the edges for any input image. We implemented a Canny detector from scratch to have full hands-on parameters. Our input image is an image that we took from the 2018 football world cup of the game between Belgium and Japan. In fig. 5, you can see our input image.

Figure 5: Input image for our Canny detector

The pre-processing of the input image starts with the removal of the noise using a **Gaussian filter**.

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}.exp(-\frac{x^2+y^2}{2\sigma^2}) \quad (2)$$

In our case, we used a Gaussian filtering with a sigma of 1.

Then, the derivative of Gaussian filter is computed to calculate the **gradient** of image pixels to obtain magnitude and orientation along x and y dimension.
At each pixel, the gradient value is:

$$|G_x^2 + G_y^2| \quad (3)$$

The orientations of the contours are determined by the formula:

$$\theta \pm \arctan(\frac{G_y}{G_x}) \quad (4)$$

Thirdly, the **non-max edge contributor pixel point is suppressed** considering a group of neighbors for any curve in a direction perpendicular to the given edge. Only the points corresponding to local maxima are considered as corresponding contours and are kept.

Lastly, the **double thresholding** method is used to preserve the pixels higher than the gradient magnitude and neglect the ones lower than the low threshold value. This requires two thresholds, a high and a low. In our case, we set

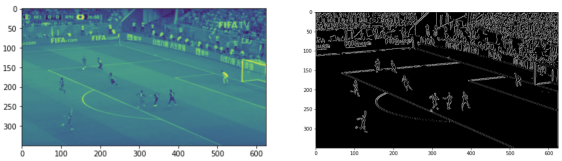

Figure 6: Football field image processed using the Canny Edge Detector

the low threshold to 0.05 and the high one to 0.08, which will be compared to the intensity of the gradient of each point.

For each point, if the intensity of its gradient is:

- Below the low threshold: the point is rejected
- Greater than the high threshold: the point is accepted as forming an outline
- Between the low and the high thresholds: the point is accepted if it is connected to a point already accepted

We also added a edge connectivity method that set the "low" pixels to "high" if they are next to a "high" pixel. This allows the detector to assure continuity in the edges. In fig. 9, you can see the flow chart of the canny edge detector. As you can see in fig 6. the canny image contains also edges we don't want like the edges of the player and the stadium.

4.2.2 Remove the noise

We need to remove the stadium and the players from the edges we detected to make the homography detection. To remove the player, as stated in [**?**], we use the player detection we have (see section 4.1) and delete them by setting all the pixels corresponding to 0. To remove the stadium, we had to create a mask corresponding to the green field (see fig X). We want to have a clear mask with the pixel of value

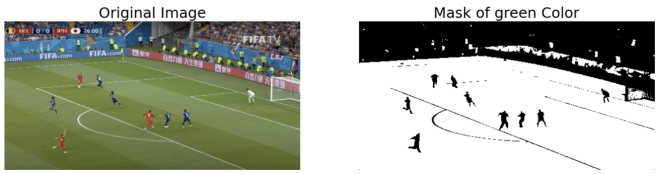

Figure 7: Green mask applied to the football frame

1 for the field space and 0 for the rest. To do so, we used the dilation function from *skimage*[5] on the inverse mask to delete the noise made by the stadium and then we inverse the new dilated mask and make another dilation to delete the players. You can see in fig. 8 our final image. Thanks to this new mask, we have a great look at the edges of our field.

4.2.3 Flow Chart

4.3. HOMOGRAPHY MATRICES ESTIMATION

The homography estimation is the last step for player position detection on the field and is a difficult task. Homography maps images of points that lies on a world plane from one camera view to another. In our case, the homography will place the field we see in the image on a field map (see fig. 10).

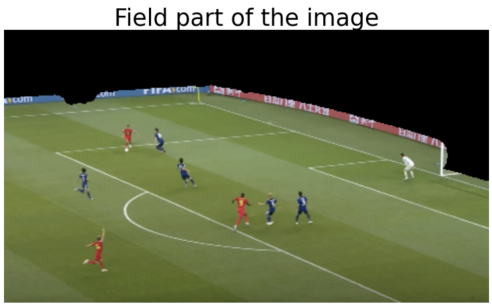

Figure 8: Green mask applied to the football frame

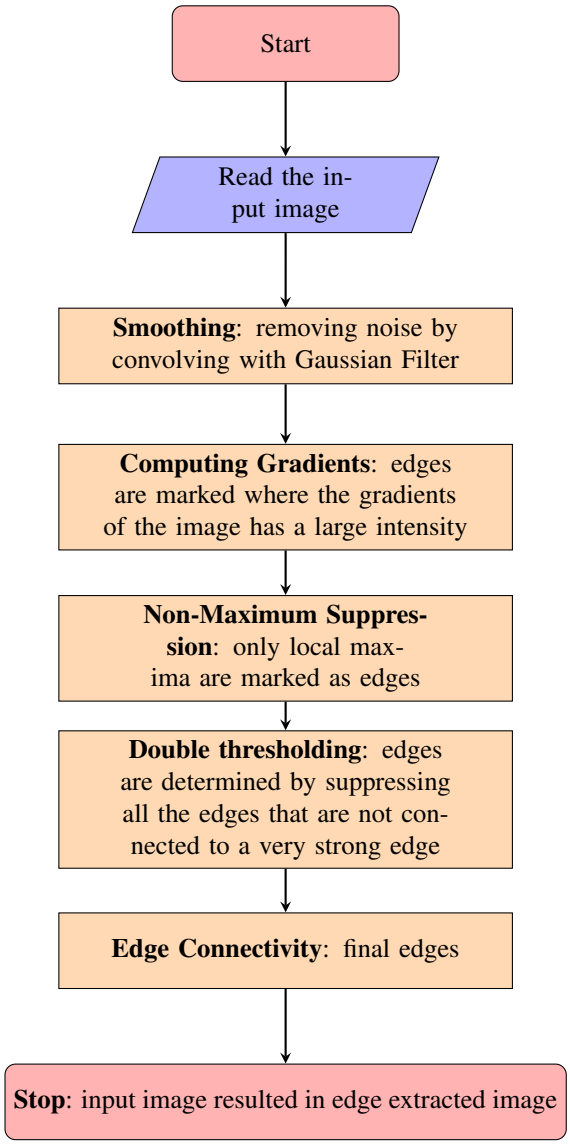

Figure 9: Flow Chart of the Canny Edge Detector Algorithm

4.3.1 Estimation of the homography

We need to transform the image coordinates (A) into court coordinates (B). Therefore, we will compute the

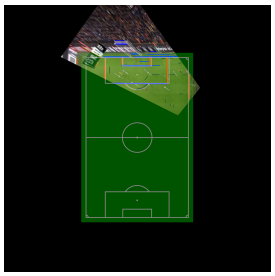

Figure 10: Example of a frame transposed on the field map

homography transformation from (A) to (B).

Let $p = [x, y]^T$ be a point in (B) and $p\prime = [x\prime, y\prime]^T$ its corresponding point in (A). We have the relation :

$$p\prime = \frac{1}{h_7 x + h_8 y + 1} \begin{bmatrix} h_1 x + h_2 y & h_3 \\ h_4 x & h_5 y \quad h_6 \end{bmatrix} = f(p; H) \tag{5}$$

where $f(p; H)$ is a nonlinear function, and $H = [h1...h8]$ is the homography.

Usually, homography is computed using the interest points of the image using a SIFT descriptor. However, this gives importance to the audience and the player position, giving bad results as it has nothing to do with the field portion. The Wu [**?**] paper suggests using the edges as interest points. However, they used a video and gave the first frame homography by hand to estimate the rest, which we didn't want to do. At this point of the project, we already have the edges computed by the Canny Detector, but our results are not as good as we would have expected. So what we state next is what we wanted to do but we didn't implement it.

Next, we would have computed the HOG feature of the edge map and used a KNN with all the possible homography matrices with their associated HOG vectors. We know a data set exists with that information however we didn't request access as we didn't go that far.

5. EVALUATION

To evaluate our player detector, we used the SPD public data set [4]. As stated in the problem definition section, we created our train and test set with cropped images of the players and random parts of the images to create a positive and a negative sample. Our train set is composed of 1,000 positive samples and 1,000 negative samples, our test set is composed of 500 positive and 500 negative samples. We obtained a quite good performance on the test set.

	Player	Not a Player
Player	492	28
Not a Player	0	464

```
Accuracy :  0.9715447154471545
Precision :  0.9461538461538461
Recall :  1.0
F1_score :  0.9723320158102766
```

Figure 11: Confusion matrix and different metrics of performance on the test set

As you can observe, the precision is not perfect and could be better with a data set containing more chosen negative samples. Indeed, we used random cropping, so we don't have every possibility of negative samples. However, we have a long computational time to execute all the detection pipelines, which can be explained by the size of our input image. When we rescale it to a lower shape, the pipeline executes faster.

In fig. 12, you can see the result of the player detection.

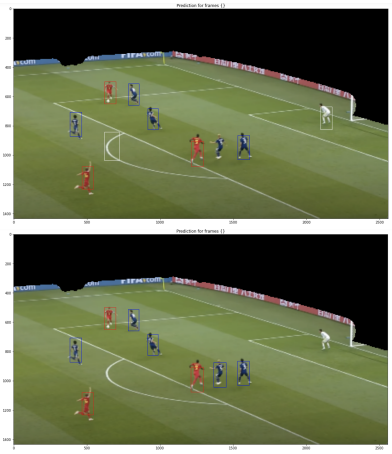

Figure 12: Result of our player detection with mask

As you can see, our detector combined with color detection works well to detect the player. However, we have a problem with the goalkeeper as it is white and some of the false detections captured a white field line. As stated before, this can be upgraded by changing the dataset. To palliate temporarily this problem, we decided not to detect the white shirts player.

In fig. 13, you can see our edge detector result. As you can see, we did not succeed to have a mask good enough to delete the advertising band around the field. That added to the player detection problem gave us a result that is not usable for a homography estimation which would have been the next step of the project.

6. CONCLUSION

This paper contains an unfinished pipeline for player detection and edge detection. We didn't succeed to create a

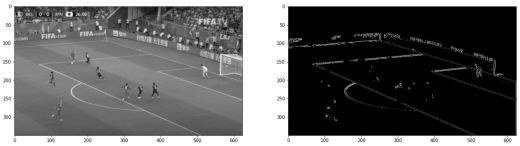

Figure 13: Result of the canny detector after removal of the player and application of the mask

method for homography estimation. However, we can see how we can improve our model. On one hand, we have to create a more reliable data set so that the SVM can learn on solid ground truth. Also, we maybe need to change our edge detection method or add some pre-processing before the canny edge.

We thought that not using computer vision would lower the computational time needed to process a frame. However compared to other projects, it seems that deep learning accelerates the process.

This project was really interesting as we learned to play with the different vision filters to achieve our goals and gives us insights into all the possibilities that computer vision without deep learning can offer.

References

[1] Lu, W.-L. (2011). Learning to track and identify players from broadcast sports videos (T). University of British Columbia. Wei-Lwun Lu, Jo-Anne Ting, James J. Little, Kevin P. Murphy.

[2] John Canny, ”A Computational Approach to Edge Detection”,6 november 1986

[3] Open CV library (https://opencv.org/)

[4] SoccerPlayerDetection bmvc17 v1 dataset (https://drive.google.com/file/d/1ctJojwDaWtHEAeDmB-AwEcO3apqT-O-9/view?usp=sharing)

[5] Skimage library (https://scikit-image.org/docs/stable/api/skimage.html)

[6] Small object detection via multi-task generative adversarial network. In Proceedings of the European Conference on Computer Vision (ECCV). 206-221, Yancheng Bai, Yongqiang Zhang, Mingli Ding, and Bernard Ghanem. 2018.

[7] A Discriminatively Trained, Multiscale, Deformable Part Model. Pedro Felzenszwalb, David McAllester, Deva Ramanan