Master's Thesis: Deep learning for ball tracking in football sequences

Auteur: Lhoest, Alexandre
Promoteur(s): Van Droogenbroeck, Marc
Faculté: Faculté des Sciences appliquées
Diplôme: Master: ingénieur civil en science des données, à finalité spécialisée
Année académique: 2019-2020
URI/URL: http://hdl.handle.net/2268.2/9074

Avertissement à l’attention des usagers:

Tous les documents placés en accès ouvert sur le site le site MatheO sont protégés par le droit d'auteur. Conformément aux principes énoncés par la "Budapest Open Access Initiative" (BOAI, 2002), l’utilisateur du site peut lire, télécharger, copier, transmettre, imprimer, chercher ou faire un lien vers le texte intégral de ces documents, les disséquer pour les indexer, s’en servir de données pour un logiciel, ou s’en servir à toute autre fin légale (ou prévue par la réglementation relative au droit d’auteur). Toute utilisation du document à des fins commerciales est strictement interdite.

Par ailleurs, l’utilisateur s’engage à respecter les droits moraux de l’auteur, principalement le droit à l'intégrité de l'oeuvre et le droit de paternité et ce dans toute utilisation que l'utilisateur entreprend. Ainsi, à titre d'exemple, lorsqu'il reproduira un document par extrait ou dans son intégralité, l'utilisateur citera de manière complète les sources telles que mentionnées ci-dessus. Toute utilisation non explicitement autorisée ci-avant (telle que par exemple, la modification du document ou son résumé) nécessite l’autorisation préalable et expresse des auteurs ou de leurs ayants droit.
University of Liège - Faculty of Applied Sciences
Year 2019 - 2020

Deep Learning for Ball Tracking in Football Sequences

Graduation Studies conducted for obtaining the Master’s degree in Data Science Engineering by Alexandre Lhoest

Supervisors
Marc Van Droogenbroeck
Bruno Wery

Jury
Marc Van Droogenbroeck
Bruno Wery
Gilles Louppe
Acknowledgements

First of all, I would like to thank especially my internship supervisor, Mr. Bruno Wery, for his daily guidance and his many advice throughout this work. I am grateful to him for having introduced me to 3D modeling, as well as for his disponibility at any time for any inquiry.

I am also grateful to my promoter, Professor Marc Van Droogenbroeck, for his supervision, the follow-up meetings and the global expertise that he transmitted to me.

I would like to thank all the staff of the Deltatec company for their hospitality, and for allowing me to work in flawless conditions, even during the encountered health crisis.

Thanks to my family and friends for their presence and their kindness.

I am grateful to my parents and my sister for their continued encouragement.

I would like to thank my partner, Hanna, for her unconditional support, especially significant in this difficult period of health crisis.

Finally, I would like to thank those who helped and supported me throughout the five years of studies from which this thesis is a conclusion.
Abstract

Nowadays, football is by far the most popular sport in the world. Its huge audience obviously attracts many fields, including the field of game analysis. In recent years, deep learning has brought progress to automated match analysis but a challenge remains strenuous: the ball tracking on football sequences. Indeed, the ball is often represented by few pixels on the football videos and many other object have a similar appearance. In addition, the ball can be occluded by a player or in front of a line with similar color. All these circumstances make the ball tracking a challenging objective.

In this work, an offline ball tracking system based on deep learning for soccer sequences is implemented.

The system is divided into two main parts: a ball detector followed by a ball tracking method. The detector is based on a deep convolutional neural network for image segmentation. A dataset composed of semi-synthetic images representing a synthetic ball in front of a real frame from a football video sequence is created to train the network. The objective of the detector is to identify ball candidates on each frame of the sequence.

Then the tracking method treats all the candidates of the sequence to generate candidate trajectories with the exploitation of the Kalman filter. A score is assigned to each trajectory thanks to the use of a deep neural network for image classification. The trajectories with the highest score are selected as ball trajectories. Finally, a cubic spline interpolation estimates the ball positions between the trajectories.

Evaluated on sequences of professional competitions, this ball tracking system allows an identification of the ball position on more than 80% of the images of most of the evaluated sequences, without having any false detection. Moreover, the evaluation highlights the ability of the system to estimate the ball positions during occlusion on a few frames.
## Contents

### 1 Introduction

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Introduction</td>
<td>3</td>
</tr>
<tr>
<td>1.2 Related works</td>
<td>5</td>
</tr>
<tr>
<td>1.3 Working Plan</td>
<td>9</td>
</tr>
<tr>
<td>1.4 Methodology</td>
<td>14</td>
</tr>
<tr>
<td>1.5 Objectives</td>
<td>15</td>
</tr>
</tbody>
</table>

### 2 Ball detector

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Introduction</td>
<td>16</td>
</tr>
<tr>
<td>2.2 Dataset Creation</td>
<td>16</td>
</tr>
<tr>
<td>2.3 Neural Network Creation</td>
<td>25</td>
</tr>
<tr>
<td>2.4 Evaluation</td>
<td>27</td>
</tr>
<tr>
<td>2.5 Training Set Improvements</td>
<td>33</td>
</tr>
</tbody>
</table>

### 3 Ball Tracking

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Introduction</td>
<td>35</td>
</tr>
<tr>
<td>3.2 Trajectory Creation</td>
<td>35</td>
</tr>
<tr>
<td>3.3 Trajectory Score</td>
<td>38</td>
</tr>
<tr>
<td>3.4 Trajectory Selection</td>
<td>44</td>
</tr>
<tr>
<td>3.5 Trajectory Expansion</td>
<td>45</td>
</tr>
</tbody>
</table>
6 Discussion 48
6.1 Structure Summary 48
6.2 Result Evaluation 49
6.3 Additional Processing 51
6.4 Future Works 54
6.5 Conclusion 55

A Appendix A 60
A.1 Description of the Company 60
A.2 Marketing Strategy 61
A.3 Human Resources Strategy 61
A.4 Recruitment Strategy 62

B Code labeling 63

C Code for cropped images creation 64

D Tables of non-optimal results 65
1 Introduction

With 250 million players in over 200 countries, football is by far the most popular sport in the world. Between the World Cups, the Champions League, the Europa League, and all the Championships of each country, football is watched by billions of people all across the world. At the last World Cup, a total of 3.572 billion cumulative viewers watched the matches on television or digital platforms, and 1.12 billion people watched the final.

This enormous audience obviously generates a thriving economy for football, which would represent 400 worldwide in turnover. Many fields are interested in football, including the field of game analysis.

In the past decade, the analyzes of football matches were more and more in demand. In fact, matches’ results and statistics are usually intended for several types of applications. The coaches need the statistics of the opposing team to set up new strategies, the bettors increasingly request the probabilities of the matches’ results to know on which team to bet on, and the analysts use positioning information to facilitate the explanation of the tactics. Many high-level analyses can be performed through the help of computer vision.

Computer vision plays a crucial role in exploiting the information given by the football videos. The automatization of tracking and detection of objects on the videos facilitates the analysis of the active entities or the identification of infringement during a match. Computer vision can help many applications such as shot classification, ball possession, or event detection. A piece of crucial information is needed for all these applications, the ball position.

Figure 1: Example of a sequence frame filmed with the principal camera
The automatization of the ball position in long-shot football sequences is a challenging task for many reasons. The long-shot sequences include the videos filmed with the principal camera, placed in the middle of the field length to provide large views of the game. Figure 1 illustrates an example of a video frame filmed with a principal camera. First, the ball is very small on long-shot football video frames and can be confused with other similar shapes within the image. On a frame of size 960 x 540 pixels, the ball is often represented on around 10 x 10 pixels. Moreover, the ball can take a high speed, which provides it a blurry and elliptical aspect. In addition, during a match, the ball is often occluded by players, or in front of lines that frequently have the same color. Finally, the colors of the ball differ from one match to another, or due to the lighting conditions. The ball size perceived on the video varies depending on the distance between the ball and the camera. Figure 2 shows several representations of the ball on frames filmed by a principal camera.

![Figure 2: Representations of the ball on 30 x 25 pixels images from video sequences filmed by a principal camera with a resolution of 960 x 540 pixels per frame.](image)

Different methods have been used to produce automated ball tracking systems. In recent years, the use of deep learning improved the performance of the tracker. In an offline ball tracking, all the sequences can be analyzed at once, meaning that additional inter-frame information can be used to track the ball on the entire sequence.

In this work, an offline ball tracking in football sequences is proposed. The tracker is divided into two main parts. First, each image of the sequence is analyzed separately by a ball detector to identify a list of ball candidates. The ball detector is based on a convolutional neural network for image segmentation. To train the network, a dataset of semi-synthetic images is created. Each image of the dataset represents a modeled ball in front of a real frame from a football match video. The creation of semi-synthetic images allows simple expansion of the database, controllable variance of the ball represented in the training, and fast and effortless labeling. The second part of the system consists of using all the ball candidates of the sequences at once to exploit inter-frame information. The method uses the Kalman filter to generate candidate trajectories from all the detected candidates. A huge amount of wrong candidates is removed by keeping the trajectories of
Then, the remaining trajectories are ranked by using a neural network for image classification. The neural network evaluates the confidence of each candidate to belong to the class “ball”, the trajectory score is simply the average confidence value of each of the candidates constituting the trajectory. Finally, the trajectories with a score above a threshold value are considered as the ball trajectories and a cubic spline interpolation is used to estimate the ball positions on the frames between the trajectories.

The remainder of the Master thesis is organized as follows. Section 2 resumes the researches on the ball tracking challenge, while section 3 establishes a synthesis of the methods encountered, a methodology of the accomplished work and the expected results of the system. The ball detector and the tracking system are presented in sections 4 and 5 respectively. Finally, section 6 discusses the obtained results, the limitations of the ball tracking system and acknowledges future potential works.

This thesis has been carried out during an internship at Deltatec company. A strategic analysis of the company is described in appendix A.

The main code of the ball tracking system written in C++ is presented in the external appendix. However, this appendix is only available for the jury of the Master thesis.

2 Related works

In recent years, many researches have focused on the problem of ball tracking and have brought progress to this challenge. The previous studies propose several works based on different methods to detect and track the ball. Some works are based on basic information such as the different colors and the curves observed on the image, while other ones use advanced models by employing convolutional neural networks or a combination of methods for detecting the ball. The authors have used information given by the previous frames to track the ball such as the trajectory, the position, and the velocity of the ball.

This section presents several methods used in papers that contribute to this progress.

The first papers exploiting ball detection use basic methods of image processing to retrieve pertinent information in order to detect the ball.

The paper [6] proposes a real time ball detector for long-shot frames. This method is based on 4 steps. The first one is a background subtraction; it consists of converting an image with an RGB model to a binary model by selecting the pixels of the initial image where the green is preponderant to red, which is preponderant to blue, and convert them to 0. The remaining pixels are converted to 1 to get the binary image. However, this step can regrettably make the players, the lines or the ball disappear. Therefore, the next step consists in applying a Sobel gradient on the initial image and add it to the binary image in order to recover the features. The third step applies the Hough transform on the obtained...
binary image to identify the lines and eliminate them. At the last step, the ratio of area and perimeter of each object is calculated to compute a threshold value and eliminate the remaining unwanted objects. With the hypothesis that the player and the ball have a large density of pixels, the objects that have their ratios below a threshold value are eliminated.

This method is fast, not hard to implement, and does not require a dataset. However, it is not robust to some cases that appear regularly on the football field such as the occlusion or the touch of the ball by a player and the location of the ball on a line.

Likewise, the authors of the paper [25] introduce a method based on the curvature of isophotes, the curves with the same pixel intensity. The algorithm first performs a background subtraction based on the difference of the current frame and a reference image, which is updated at every time step. Afterwards, the curvature of each isophote is computed to create sets of pixels with the same curvature. The sets with a curvature compliant with those of the expected ball are selected. Three pixels are randomly chosen to compute the parameters of the circle and to perform evidence checks in order to determine if the circle can be considered valid. The checks determine if the radius is similar to the expected one, if the circle contains a curvedness local maximum, and if another edge pixel is close to the circle. The candidates are analyzed by a voting algorithm based on the distance between the edge pixels and the center to select the best one and to verify if it is genuinely in the image. The circle is adjusted by a linear error compensation due to the estimation of the circumference with only 3 pixels.

This work presents a method that limits the computation time. However, this method uses a background subtraction and it is not robust to occlusion.

In [33], the ball is tracked by a Kalman filter and a template matching method. To manage the occlusion, when the ball is touched by a player, it is considered that the player has the ball. The ball is searched around the player for the next frames until it is detected and tracked again.

In recent years, the possibilities of the convolutional neural networks (CNN) have been exploited by several works, bringing a huge progression in the ball detection task.

The paper [18] proposes a ball detector in long shot videos, called DeepBall. It consists of a CNN that takes as input a video frame and produces a ball confidence map, which is a scaled down image of the input frame that encodes the confidence of the presence of the ball at each location. The characteristic of this neural network is the concatenation of the three firsts convolutional blocks fed into the final classification layer. This method has several advantages: the neural network processing the entire image at once in a single pass through the network, and the efficient reduction of the computational time. Moreover, Deepball can be used with images of any size and have good results. However, Deepball is
made for ball detection, this technique does not take into account temporal and physical information that can increase the accuracy of the method for ball tracking, since the ball confidence map is created only from the current video frame.

The paper [19] introduces an alternative method, called FootandBall. This time, the CNN architecture is inspired by the feature pyramid network [23]. This architecture uses both low-level features, allowing precise location of the objects, and high-level features, improving the discriminability of the ball detector from the first convolutional layers and the higher convolutional layers, respectively. This architecture improves the accuracy of ball detection and, in addition, allows the detection of players by a player confidence map and the position of the bounding boxes.

In [30], the authors propose an algorithm to classify a patch of a tennis game image as “ball” or “not ball”. The neural network takes as input a small patch of the image and determines if the patch contains the ball or not. This approach offers high accuracy values, suggesting its feasibility. However, the limitation of this technique is the small patch taken as input. As no information is given during the patches’ delimitation, the patches should overlap each other for one contain the entire ball. Since the size of a patch is relatively small, the number of patches to search the ball in the whole image can be huge. This method is more appropriate when the search region is reduced.

Some authors also decided to use alternative methods to increase the ball tracking accuracy.

The algorithm proposed in the paper [36] overcomes the hard challenge of a ball classification by evaluating if an object trajectory is a ball trajectory, instead of evaluating whether an object is a ball. The algorithm selects ball candidates by applying a sieve with some characteristics such as the size, the shape, and the color. The ball candidates are ranked from 1 to 3 depending on the probability to be a ball. The ranking is made by two kinds of features: the appearance features such as the circularity, the color, the size, and the isolation of the candidates. The trajectories of the candidates are computed by a Kalman filter. Each trajectory is indexed with a score composed of the length of the longest consecutive candidate sequence of rank 1, and 2, the ratios of the candidates of rank 1 and 2 and the length of the trajectory. The trajectory with the highest score is taken and the other trajectories that overlap this one are removed. This operation is repeated until there is no trajectory with a score higher than a threshold value anymore. Finally, the trajectories are extended by using a Kalman filter. This method achieves a ball detection accuracy of 96.1% on sequences filmed with a moving camera.

This method can be integrated with any kind of ball detection to improve its accuracy. Moreover, it can be used to detect several soccer game events such as a ball touched by a
player, the possession, the pass, and shoots.

To improve their models, recent works use a combination of different methods in order to increase the ball tracking accuracy and decrease the computation time.

The authors of the paper [22] present a scheme for ball detection and ball tracking in broadcast videos filmed with a moving camera. For ball detection, the algorithm uses a weighted graph to hold multiple hypotheses for the location of the ball on several frames. Each node of the graph represents the score of a ball candidate, while each edge of the graph represents how likely the two nodes of consecutive frames correspond to the same object. The score of a node is based on the circular variance, while the score of the edge is based on the Euclidean distance between the two nodes, with the size and the gray level similarity of the two nodes. Ball detection is made by finding the optimal path of the graph, while the score of a path is the addition of the score of the nodes and the score of the edges that it contains. After the detection of the ball, the tracking is made by a Kalman filter to predict the location of the ball on the next frame and filters the tracking result in the current frame with the use of the ball position and velocity. The template matching is used to obtain observation, the template is updated after each detection. If the ball is not detected in a number of consecutive frames, the ball is considered as not tracked and the detection procedure is run again. This method increases the robustness of the ball detection by the use of the optimal path of the weighted graph. Moreover, the execution time is very low since the Kalman filter allows to process a small portion of each frame. However, even if the experiments show acceptable results in general, the precision and recall decrease drastically when the ball is occluded by a player or follows a line.

Another method introduced in [37] considers the basketball detection as a segmentation problem. Moreover, to provide a heat map that shows the probability of the ball at each position, the algorithm exploits the ball dynamics by adding the difference between two consecutive frames as input, since the ball should move rapidly compared to the entire image due to a shot, a dribble or a pass. The ball is selected with a detection rule that uses the information that there is only one ball in the image. Compared to the universal mask R-CNN, the introduced method increases the accuracy and reduces drastically the computation time, allowing real-time ball detection. However, the ball has often a high velocity in basketball due to the repetitive dribbles when a player has the ball. Therefore, even if the method of taking two consecutive frames increases the performance of the ball tracking in basketball, this technique is not necessarily suitable for ball detection in football videos.

The tracking algorithm presented in the paper [14] is based on several distinct steps. First, a background extraction is applied by creating a set of the median of successive
frames and taking the median of this set. The background obtained is compared to the current frame to detect blobs. A small CNN classifies the blobs as a ball, a player, or the background. In the next frame, the search region depends on the previous one: if the ball is detected, the ball candidate is searched around the last position of the ball. If no ball is detected, the algorithm estimates that the ball is occluded by a player and therefore starts to track the player in the search region to retrieve the ball in the future frames. If the ball was occluded for a long time, the search is applied on the full frame. Finally, if the ball is considered out of frame, the search is made at boundaries for the next frames. The ball is considered tracked if there exists an intersection of the inflated bounding boxes on several successive frames.

The method of blob detection is efficient with a fixed camera. Unfortunately, since extracting blobs with a moving camera can be very difficult and inaccurate, this method seems to be inappropriate concerning this project. However, decreasing the search region and computing the intersection of the inflated bounding boxes improves the performance of the tracker and decreases the computation time.

3 Working Plan

3.1 Synthesis

To tackle the challenge of ball tracking, several distinct steps are required for a good approach to the problem. First, and especially with the use of a neural network, a set of images must be provided to evaluate the algorithm and train the neural network. Moreover, since the network needs to evaluate its estimation of the ball detection, the images in the dataset must be labeled with the true position of the ball. Such a dataset is very difficult to find and is therefore often created.

The next important step to reach high performance with the use of a CNN is the choice of its architecture. The CNN is made of different kind of layers which all have their importance. The CNN can have an infinite number of architectures, but not all architectures are appropriate for the challenge. The goal is to find an architecture that can reach high performance in a reasonable execution time.

When the models give an output, the results must be evaluated to estimate the performances of the algorithm. Since the human feeling is not sufficient to express a good measure of the performances, some evaluation metrics must be introduced. Several well-known metrics exist and evaluate different kinds of performances. The evaluation metrics must therefore be chosen according to the performances that we are looking for.

Finally, the ball tracking in a video sequence is a huge challenge that is very hard to
implement with simple methods of ball detection. To achieve better performance, it could be interesting to use the information given by the set of frames of the video instead of considering each frame independently.

All these steps are discussed in the subsections below, in the light of the information given by the related works.

### 3.1.1 Dataset creation

The works which only use classical methods of image processing do not need a dataset to train their model. However, the generation of an accurate dataset without inconsistency is an important step for models using a convolutional neural network. For a good learning of the network, a large dataset with variety is needed to avoid overfitting. A coherent labeling is also crucial for expecting the high performance of the model.

In [14], they designed their own dataset by cropping online available images to get 1500 images for each ball, player and background class, including blurry and occluded objects for the robustness of the CNN. Since this CNN is used to classify blobs, the dataset does not need long shot views.

In [18, 19], the dataset used is the ISSIA-CNR Soccer Dataset, which is publicly available. This dataset contains long shot views of the football field taken by a camera recording at 25fps. The authors used 20,000 manually annotated frames, which 7000 contain the ball. Although the dataset is publicly available, it cannot be used for commercial purposes.

In [6], 116,385 images (patches) have been collected during tennis matches and training sessions, which 65% and 15% have been used for the training set and the validation set, respectively. 27,600 patches have been manually labeled as “ball”, while the remaining patches have been labeled as “no ball”. The dataset created contains ball images under different conditions, such as behind the net, in the hands of players, blurred due to a fast shot or in the corner of the image, to be more robust to real scenarios.

In [37], the dataset contains 280 basketball scenes. For each one, two consecutive frames with a delay between 30 and 40 ms were captured. At least half of the ball is visible. They use a k-folds, with k=7 in this case. This method uses 1 fold for the test, 90% and 10% of the remaining folds for the training and validation set, respectively.

Considering the difficulty to find labeled images, most of the authors decided to manually label the dataset. Two methods can contribute to this purpose: the creation of labeled synthesis images and data augmentation. In [18, 19], data augmentation is used
to overcome the lack of annotations. The transformations applied are random changes in brightness, contrast, saturation or hue, horizontal flip, random cropping, and random scaling. In [18], this data augmentation allows to increase the average precision from 0.792 to 0.877.

### 3.1.2 CNN architecture

In general, the authors use small architecture since the number of different classes is small but also to avoid the tuning of a huge number of parameters.

In [37] the authors use the ICNet [38] implementation, which is publicly available, but they modify the three input resolutions to better handle the small size of the ball at the lowest resolutions. Moreover, the input layers were adapted to handle the 6 channels input data. A Softmax is applied on the last layer to output the heatmap of the ball. Figure 3 illustrates the architecture of ICNet.

![Figure 3: Network architecture of ICNet][38]

The architecture of DeepBall [18] is inspired by single pass object detection SSD [24] and YOLO [29]. The architecture uses the hypercolumn concept [10] to correctly classify fragments of the scene containing objects similar to the ball by paying attention to the larger visual context. Some unnecessary components such as multiple anchor boxes are removed to increase the performances.

The CNN uses three convolutional blocks that decrease the spatial resolution and increase the number of channels to process the image. The output of each convolutional block...
is concatenated and fed into the last convolutional layer, followed by a Softmax layer to output the confidence map. The architecture is illustrated in figure 4.

In paper [19] written by the same authors, they have created a new architecture by implementing a feature pyramid network to use both low-level and high level features. This time, the CNN is composed of 5 convolutional layers that decrease the spatial resolution and increase the number of channels. Before the concatenation of the 4 last layers, a 1x1 convolutional layer is applied to decrease the number of channels for each convolutional layer to the same value. This architecture increases the accuracy in ball detection obtained by DeepBall. The architecture is illustrated in figure 5.
The loss is computed by a cross-entropy between the ground truth and the ball confidence map for each pixel. The networks use a gradient descent technique with an initial learning rate to 0.001. The learning rate is decreased by a factor 2 every 40 epochs with a batch size of 4 and 150 epochs in [37], and decreased by a factor 10 after 50 epochs with a batch size of 16 and 75 epochs in total in the papers [18, 19].

3.1.3 Tracking system

The structures previously illustrated allows the detection of the ball on each image separately. However, there is a huge amount of inter-image information that can be exploited with the use of different tracking systems.

For the first image, since no information about the ball position probabilities is available, the ball is searched on the entire image. However, for the next frames, the information given by the previous frames can be used to facilitate the ball detection of the current frame. In [14], the authors use the intersection over union (IoU) of bounding boxes to track the ball, which expresses the area of the intersection of the bounding boxes divided by the area of their union. If the IoU of the bounding box of 2 consecutive frames is larger than 0 in M consecutive frames, the ball is considered detected. When this is the case, the search region is made around the ball. This new search region could be computed giving the distance of the camera, maximum ball speed, and camera movement. If several balls are detected in the image, the ball position could be chosen with the help of the trajectory or with the highest IoU value. In [14], if no ball is detected, the ball is considered occluded by a player, therefore the players in the search region are tracked. If the ball is not detected in N consecutive frames, ball detection will be made in the entire image.

In [33, 22, 36], the authors decided to use a Kalman filter to approximate the ball position of the next frame and a template matching to retrieve it. This method is generally fast but not robust to occlusion or location of the ball in front of a line.

3.1.4 Evaluation metrics

To choose a result evaluation measure, the objective must be clearly defined. In this work, the purpose is to track the ball throughout a sequence. Therefore, the problem can be seen as a segmentation of the image, with the aim of object detection on each image and classification of the detected object. The evaluation must be carried out according to the type of problem selected.

To evaluate the results of a model that handles a classification problem, well-known evaluation metrics which are usually used. These evaluation metrics are computed from the confusion matrix, in which the line corresponds to a real class and each column corre-
sponds to an estimated class. The value of a cell represents the number of elements of the class of its line that are estimated as the class of its column.

With the confusion matrix, several metrics can be computed. For example, the accuracy is the number of well classified elements of a target class divided by the total number of elements of this target class; sensitivity, recall or true positive rate (TPR) is the number of elements of the class considered as “positive” that are correctly estimated, divided by the number of elements for the class “positive”; and the precision is the number of elements of a class that are correctly estimated divided by the number of elements of this estimated class. The precision expresses the proportion of positive identifications that are correct and the recall expresses the proportion of real results that are correctly identified.

In [37] they use the receiver operating characteristic (ROC) curve to evaluate their model. The ROC curve plots the true positive rate as a function of the false positive rate, while varying the detection threshold. The best model is the model that reaches the point of the true positive rate equal to 1 and a false negative rate equal to 0.

In [18, 19], to evaluate their results for a detection task, the authors compute the average precision (AP) to summarize the shape of the precision-recall curve by computing a mean of 11 values of precision at a specific recall value. The recall value is modified by varying the threshold value that classifies a pixel as a ball pixel or a background pixel.

3.2 Methodology

After the familiarization with the services made available within the company DeltaTec and the reading of the state-of-the-art concerning ball tracking, let us establish a working plan to get into the challenge of ball tracking. The general idea of this project is to divide the work into two parts: a ball detector and a tracking system.

Detecting the ball from a long-shot sequence of a football game is a complex task to implement. Several visual features of the ball can differ from a match to another one, such as the colors of the ball or the distance of the camera to the pitch, varying the size of the ball in the image. During a match, the circular shape of the ball can become blurry and elliptical due to a high speed, as well as locate in front of a player or a line that has often the same color as the ball.

Concerning the detector, it must be robust to a large number of situations that appear on a football sequence, as well as a large number of features that differ between the sequences. Basic methods of detection such as methods based on the circular Hough transform or the curvature of isophotes express some difficulties to detect the ball when its features have a large variance, and during the situations previously mentioned.

Considering these challenging conditions, the ball detection method chosen is based on a deep convolutional neural network in order to exploit the robustness of the deep learning.
The creation of the detector based on a neural network can be separated into three distinct steps: the creation of the training set, the creation of the network with his training, and the evaluation of the test set results. However, the training set must be created in parallel with the test set. Indeed, the training set must be produced according to the evaluation that will be performed. For a consistent evaluation, the accepted situations, as well as the range of the features of the ball and the sequences must be identical for the training and the test set.

The detector analyze the frames of the sequence separately to estimate the position of the ball on each of them. However, it is conceivable that the detector will not find the location of the ball on every image, especially when the ball is occluded. Moreover, it is plausible that the detector makes wrong detections, or detects a ball where there is none.

A good way to manage these situations is by using the information provided by the entire sequence. For this reason, a tracking system that handles the outputs of the detector and the temporal information of the sequence to improve the consistency of the detector is introduced. In other words, the tracker will use all the outputs of the detector of the sequence to determine which of the detection seems to be correct and which seems to be wrong. In addition, the tracker will provide a continuous detection by retrieving the trajectory of the ball to expand it in order to estimate the ball location in the frame where the ball was not detected.

The neural networks created in this work were implemented using the deep learning framework Caffe [12]. Developed by Berkeley AI Research (BAIR) and by community contributors, Caffe takes its benefits from its speed, its modularity and its expressive architecture.

The training of the neural networks were made on DIGITS [5], a Deep learning GPU training system developed by NVIDIA. Digits offers a graphical user interface allowing to design and train the neural network with simplicity, as well as evaluate the results with the help of loss function graphics and the representations of the different layer outputs of the network.

3.3 Objectives

The purpose of this Master thesis is to lighten the domain of ball tracking using deep learning. This subject is a hard challenge with many parameters to handle. At the end of this project, a perfect ball tracking is not expected. The objective is to initiate the creation of a tracking system, evaluate the option chosen for its creation, define the limitations of the system, and assess in which situations the ball tracking obtains expected results. Lastly, the objective is to identify several potentially proficient improvements to upgrade the tracker.
The ball tracking is separated into two mains parts with distinct aims:

- First, the detector must detect as many balls as possible, even if a few wrong detections appear. Considering that the ball is the positive class, the objective of the detector is to maximize the number of true positives, while maintaining an acceptable number of false positives. The maximum number of false positives accepted is hard to fix because it depends on the capability of the tracker. As a result, considering the obtained results seen in related works, it is expected to detect almost all the balls in not complex situations, such as when the ball is on the grass without a huge blurry aspect or when the ball is far from a player. The detection of a majority of the balls with a blurry aspect or close to a player is also expected. Considering the false positives, they should contain only ambiguous cases such as football shoes, heads or line corners.

- Then, the tracking system must select the real balls between all the detections generated by the detector. This time, the objective is to not have false positive anymore, while keeping the number of true positive as large as possible. The expected result is to keep basic trajectories such as a rolling ball on the grass with a low speed, and the majority of trajectories with a significant blurry aspect. It is also expected that the interpolation system becomes able to define an appropriate position when the ball passes in front of a line.

4 Ball detector

4.1 Introduction

The ball detector created in this project is based on a convolutional neural network for image processing. This network takes as output a frame of a football sequence implemented in a RGB color model and outputs a confidence map to identify the location of the ball. The confidence map has the same size as the input, but contains only one channel with values that vary between 0 and 1, where the value 1 means the highest confidence to find the ball. Finally, the detector has a segmentation task, it must decide for each pixel of the confidence map if it belongs to the class “ball” or the class “no ball”.

4.2 Dataset Creation

Due to the actual difficulty to find a large labelled dataset, the option to label the datasets ourselves was chosen. However, labeling a huge amount of balls for image segmentation is a meticulous and time-consuming work. Considering this drawback, an alternative method has been considered: the creation of semi-synthetic images. The objective is to add a
synthetic ball in front of a real frame from a football video sequence. Indeed, the synthetic modeling of the ball has several advantages:

- Once the modeling and the animation of the ball done, the number of generated images can be as large as desired. Contrary to the use of real images that can be difficult to find, especially when particular situations are required, the enlargement of the training set with new images of such situations can be made effortless in a trice.

- By using synthetic images, the variance of image features can be controlled. In a neural network, the situations where the ball will be detected depend strongly on the images used during the training. However, by using real images of available datasets, the situations and the features covered by the network are not easily controlled. For example, on the real available images, if the balls with red faces are underrepresented or inexistent, the network will not be adequately trained to detect these balls and the rebalancing of the training set can be arduous. While synthesizing balls, the balancing of the training set is under control and the accepted range of features can be uniformly covered.

- During the creation of the semi-synthetic images, the synthesized ball representation is placed over a real image. Since the ball is placed externally, it is effortless to store the position of the ball and to execute an automated labeling of the images. This automated labeling allows a considerable gain of time.

While the training set is made of semi-synthetic images, where a synthetic ball is added to a real frame from a football sequence, the test set contains real frames with real balls. The objective of the detector is to detect the real ball on football sequences, so that the results that will be evaluated contain unmodified original images.

All the modeling, animation, and rendering were made on the software Cinema 4D. Developed by the company Maxon, Cinema 4D is a tool intended to render, texture, animate and model 3D objects. Training the network on synthetic ball representations to test on real ball representations is a hard challenge, the semi-synthetic image must represent a realistic scene and the modeling of the ball must have enough details to look like a real ball. This experiment will clarify the option of using the modeling for ball detection, but also define with which precision it is possible to achieve with this option.

4.2.1 Targeted Parameters

Before creating the dataset, the targeted visual aspects of the ball must be explicitly defined. The accepted range of features describing the rendering of the ball on the image must be identical for the training set and the test set, even if one contains synthetic balls.
and the other real balls.

According to [32], there are 5 types of parameters in the physical world that change the visual aspect of an object: the object, the scale, the movement, the illumination and the appearance of the object. After a visualization of around thirty sequences, accepted ranges of the variation of these parameters have been chosen to describe at best the ball aspects. These ranges are defined below:

- The targeted objects are balls of the form of an icosahedron, a polyhedron with 12 regular pentagonal faces and 20 regular hexagonal faces. The targeted balls have their hexagonal faces in white and do not have restrictions on the color of the pentagonal faces, except that they must all have the same colors. In addition, to enlarge the range of targeted objects, spherical white balls with colored lines are also taken into account.

- The size of the ball must be between 6 x 6 pixels and 12 x 12 pixels.

- During a football game, the ball moves in many places; it can be located on the grass or in front of a player or a line. In addition, it is common for the ball to be in the air, thus being able to pass in front of the public or billboards. All these situations are taken into account.

- Lighting conditions vary regularly from one soccer video to another. In theory, it is simple to vary the illumination of the ball when it is represented by the HSV channels, the Hue Saturation Value channels, by assigning the minimum value to the S channel and varying the V channel. With the brightness of the sun, the ball can have a yellowish appearance. This aspect can be simulated by giving a value around 60 for the H channel and varying the S channel. In practice, the rendering of the ball with different lights has been simulated by representing the ball through the RGB channels and varying them between large values.

- Due to a high speed, the ball can have a blurry elliptical appearance. The detector will only consider the blurry aspect behind the ball where the length is at maximum two times larger than the diameter of the original ball. Indeed, since the ball has a diameter of around 22 cm, it means that the ball moves around 44 cm during the exposure time. Considering that a video of 25 fps has a shutter speed of 1/50 second, the maximum ball speed considered is 22 m/s, i.e 80 km/h. This maximum ball speed covers all ball movements except high powerful shot, that can reach 129 km/h. [13]

### 4.2.2 Background Creation

Before handling the modeling of the ball, the images to use as background behind the synthetic ball have been determined. To create the training set, the company Detlatec provides a set of 10 000 images from football videos of around 40 different matches. These
videos were filmed with the principal camera in a resolution of 960 x 540 pixels. However, since the size of the ball on the image is around 10 x 10 pixels, less than 0.02% of the pixels on the image represent the ball. These images have been used to train a neural network for image segmentation, it will be very complicated for the network to converge. To help the convergence of the network, the option of cropping the images before using them for the training set is chosen. On the other hand, the results on the training set and the future use of the detector will be made on complete images. Cropping the images on the training set will modify the balancing between the segmentation of the ball and the rest of the image during the training. This modification can result in an overestimation of the pixels belonging to a ball during a detection on a whole image. However, since the principal objective of the detector is to optimize the number of detected balls, this rebalancing should not harm this optimization.

Hence, from one initial image, three cropped images of the size 288 x 160 pixels are generated from random positions. However, since the initial images belong to videos from football matches, some cropped images contain a ball. All the cropped images containing balls are removed manually. Finally, 9 000 cropped images without ball have been generated. Figure 6 illustrates the creation of the cropped images from a long-shot image.

Before the use of the cropped images, a modification has been made on them. The contrast of the image varies by using a gain parameter, that has an impact on the apparent brightness, of a random value between 0.7 and 1.3. The objective of this modification is to create diversification on the images of the training set to be more robust to any luminosity received by the camera.

All the generated images have been assembled to create a video that will be used as a background behind the animation of a synthetic ball. The generation of cropped images is implemented in the code on the appendix C written in python with the use of Pillow [8] and OpenCV [28] libraries.

4.2.3 Ball Modeling

First of all, a model of the synthetic ball needs to be created. Different rendering effects are applied externally on this model to have all the different ball characteristics previously explained to constitute the training set.

The ball is modeled from an icosahedron where a spherical deformer has been applied to give a swelling effect. The model is then smoothed by applying a modeling effect on it, which has the consequence of subdividing the object and rounding it off. The modeling of the ball model is presented at figure 7.
Figure 6: The creation of the cropped images from an initial long-shot image

Figure 7: Overview of ball model

4.2.4 Model Parameters

Now that the model is created, its parameters can be modified to create a varied dataset.

- Considering the colors, all the hexagonal faces must be in white. Since the colors are represented in a RGB color model, the three channels have received a value of 250. However, the pentagonal faces can have any color. A random function was therefore created to set random values on their 3 channels. Due to the random function, the pentagonal faces have a different color for each frame of the animation. The balls with lines are created by applying a noise of type sema with a large scale to the white ball. The color and the seed of the noise also randomly have different lines of any colors in the training set. Figure 9a illustrates a white ball with colored faces, while figure 9b illustrates the ball with colored lines.
• To vary the size of the ball, a scale variation is applied uniformly on the ball. On each frame, the variation parameter multiplies the scale of the object by a random factor included in a defined amplitude. This amplitude is delimited to vary the size of the ball between 6 x 6 and 12 x 12 pixels.

• To change the position of the ball on the image, another variation is applied to the position of the ball. This variation allows the ball to have a random position around the original location. Thanks to this, the ball can be anywhere on the created image: in front of the grass, a line, a player or the public. This vibration proceeds also a rotation of the ball, allowing the change of the positions of the visible faces.

• In theory, to simulate the illumination of the ball in hue, saturation, value (HSV) channels, the white colors can be represented by varying the saturation channel value between 0% and 20% along with a value around 60° on the hue channel to simulate a yellowish ball. In addition, the value channel can vary between 80% and 100% to simulate different lighting conditions. In practice, the simulation of these different illuminations was made by varying the values of the RGB channels so that the ball color perceived on the images is in the range defined below. In addition, Cinema 4D allows to create spotlights to illuminate the ball. The spot creates light effects on the ball; this induces a non-uniform illumination of the ball for a visual rendering close to real. The position of the spot is changed on each frame to illuminate the ball from different positions to create variety between each frame of the animation.

• The vector motion blur is an option in Cinema 4D, allowing to simulate the motion blur of an object. The vector motion blur is based on the object’s speed and the shutter angle of the camera to determines how much blur can be captured in the film. A parameter is available to varying the strength of the blur. The vector motion blur was used on the ball to create a blurry aspect. The strength parameter is set randomly in a range on each frame to vary the elliptical appearance of the ball in the accepted scope. The vector motion blur is not used on all images to keep distinct balls to represent ball without speed. Figure 9c shows an image with a blurry ball.

In addition, for the sake of realism, the shadow of the ball is also simulated. This was created by the following scheme, resumed in figure 8: first, a simple ball without animation is created and a plane is placed below this ball. Then, a spotlight is created to illuminate the ball with the option of generated shadow. As illustrated in figure 8a, the shadow of the ball is visible on the plan. The spotlight is animated by a vertical movement to vary the size of the ball shadow and a circular movement around the ball to modify the position of the shadow. The animation of the shadow overview on the plane is saved. Next, the shadow video is loaded as a texture on the plan. Now, the shadow of the ball can be seen.
on the plane without the need of the spotlight, as illustrated in figure 8b. A property alpha is applied on the plane to render the part of the plane that does not contain the shadow invisible. Figure 8c shows that only the shadow of the ball is now visible. The last step consists of applying a transparency on the plane with a varying strength to vary the visibility of the shadow, to simulate diverse lighting situations, to integrate it in front of a background image.

However, the football field is often lighted by spotlights, creating 4 shadows around the ball. To simulate this situation, all this method is repeated, but instead of using one spotlight to create one animated shadow, by using 4 spots around the ball equidistant from each other. Figure 9d illustrates an image of the training set containing a ball with one shadow, while figure 9e illustrates one with 4 shadows around the ball.

Since the values of the parameters are automatically set, the parameters could be badly chosen compared to the background scene. For example, if the background image represents a scene filmed from a large distance, the ball size should be around 7 x 7 pixels. However, the ball size can be automatically set to 12 x 12 pixels, creating a mismatch between the ball and the background. All the generated images have been seen separately and the inconsistent images have been removed. This inconsistency was decided by a human eye.
After generating all the images, the corresponding labels must be created. The label chosen is a binary image of the same size where the position of the ball is represented by pixels of the value 1, the other pixels have the value 0. To do so, the ball model is replaced by a white disk, the image of the background and shadow are removed, to get a black background. Since Cinema 4D allows to save the labels only on a color image, the conversion of the colors’ labels into a grey image must be made externally. However, due to the blur of the ball on several images, the ball contains different shadows of white. To binarize the grey image, a threshold must be chosen to set the pixels with a value below this threshold to zero and the pixels with a value equal or above this threshold to 1. The conversion was made by the code in the appendix B written in python with the help of the Opencv library [28]. Figure 10 illustrates the generation of a label.
Figure 9: Overview of images of the training set containing a ball with (a) colored faces, (b) colored lines, (c) a blurry aspect, (d) one shadow, (e) 4 shadows.

Figure 10: Labelization of an image. The images represent (a) an image of the training set, (b) the label in grayscale image, (c) the label in binary image.

4.2.5 Training Set Content

In total, 8 000 pictures have been created to fulfill the training set. The partitioning is described in the table below.
<table>
<thead>
<tr>
<th>Colored faces</th>
<th>Colored lines</th>
<th>1 shadow</th>
<th>4 shadows</th>
<th>Number of images</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>1000</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>1000</td>
</tr>
<tr>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>2000</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>1000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2000</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>8000</strong></td>
</tr>
</tbody>
</table>

Table 1: Training set content

4.3 Neural Network Creation

4.3.1 Neural Network Structure

The proposed neural network architecture, inspired from the paper [19], is illustrated in figure [11]. It uses convolutional layers to process the image in a single pass through the network, for the sake of computational effectiveness. The network is based on the design pattern of the Feature Pyramid Network [23]. First, the input image (a) is scaled (b) to have all the values in a range between 0 and 1 to accelerate the training. Then, the image is processed by 5 convolutional layers (c) (e) (g) (h) and 2 max-pooling layers (d) (f) that produce feature maps with decreasing spatial resolution by two for each max-pooling layer. Afterwards, feature maps from the higher pyramid level are upsampled by a deconvolutional layer (i) (k) to retrieve the same size as feature maps from the lower level so the maps from both higher and lower levels can be added (j) (l). This architecture allows to use both low level features from the first convolutional layers and high level features from the last convolutional layers. The first convolutional layers provide information for a precise spatial location of the ball, while high level features have bigger receptive fields to provide additional context to improve the accuracy of the detection. This additional accuracy is especially significant since the ball is a small object on the image and can have an appearance similar to objects in the images such as shoes, heads, or a part of a billboard. Finally, a block of 3 convolutional layers (m) (n) (o) processes the resultant features maps to produce one feature map and a sigmoid function (p) generates the final confidence map (q). Even if it is not illustrated in the figure for the sake of clarity, an activation function is applied after each hidden convolutional layer of the network to achieve a stronger learning [16]. The activation function chosen is the Relu because of its high computation speed and its handling of the problem of gradient vanishing [34].

The numbers of outputs after the convolutional layers and the kernel sizes were defined to have a simple network to avoid the overfitting and a lack of generalization. By setting a number of input of 10 to each convolutional layer, the total number of parameters is around 12 000. Since this number is of the same order of magnitude as the number of
images in the training set, the overfitting should be avoided.

4.3.2 Neural Network Training

Considering the loss function for the training of the network, a euclidean loss between the output confidence map and the label is computed, since it is commonly the default choice of loss function for image processing [11]. The euclidean loss function on Caffe is defined as follows:

\[
E = \frac{1}{2N} \sum_{n=1}^{N} \| \hat{y}_n - y_n \|_2^2
\]

where \( N \) is the number of the output confidence map, \( \hat{y}_n \) is the value of the \( n^{th} \) pixel constituting the computed confidence map and \( y_n \) the value of the \( n^{th} \) pixel constituting
The network is trained using a stochastic gradient descent technique. The initial learning rate is set at 0.001 and decreased according to an exponential decay policy. At each iteration \( t \), the learning rate \( l_r \) is equal to

\[
l_r = l_r 0 \times \gamma^t
\]

where \( l_r 0 \) is the initial learning rate, \( \gamma = 0.975 \) is a parameter chosen empirically that varies the learning rate at every iteration and \( t \) is the current iteration.

The dataset is split to have 85% for the training and 15% for the validation validation.

A key point of the training is that the network must converge to a good local optimum. Indeed, even if the training images are cropped from long-shot images, the ratio between the pixels belonging to the “ball” class, represented by a value of 1, and to the pixels belonging to the “no ball” class, represented by a pixel value of 0, remains very low. Consequently, the network converges to a local optimum by setting all the pixel values to 0. To get out of this bad local optimum, the method illustrated in figure [12] that was suggested by my internship supervisor Bruno Wery, was used as follows. At the beginning of the training, a max-pooling layer with a kernel size of 16 was added before the computation of the loss. The purpose of the layer is to divide the scale of the confidence map and the ground truth label by 16 in order to increase the ratio of “ball” class pixels to “no ball” class pixels. Since this ratio is higher, the network is more able to output a confidence map with “ball” class pixels. After the convergence of the network, the model is re-trained by initializing the weights with those previously computed but, this time, the kernel size of the max-pooling layer is set to 8. This reduction of the kernel size allows to increase the number of pixels on the confidence map by 2 in order to have a more precise loss computation. These steps are repeated until the kernel size of the max-pooling layer is equal to 1, which is equivalent to calculate the loss on a confidence map with the same size as the initial image.

4.4 Evaluation

4.4.1 Test Set Creation and Evaluation Method

As said previously, the test set contains only real frames from long-shot soccer videos to evaluate the results of the model on cases that it is likely to encounter during its use. To label the data in the test set, each frame in a soccer sequence will be assigned manually the location of the ball on the frame. This labeling was made by a software provided by DeltaTec. However, there is a mismatch between the label provided by the detector that
(a) Use of a max-pooling of kernel size equal to 16 to divide the image size by this number

(b) Use of a max-pooling of kernel size equal to 8 to divide the image size by this number

(c) Use of a max-pooling of kernel size equal to 16 to divide the image size by this number

(d) Use of a max-pooling of kernel size equal to 2 to divide the image size by this number

Figure 12: Illustration of the method implemented to facilitate the convergence of the network by using a max-pooling layer
outputs a confidence map where using a segmentation labeling and the ground truth label of the test set that provides the location of the center of the ball in the image. To solve this mismatch, a bounding box label is considered. In fact, it is noticeable that a neural network for object detection could be used instead of a network for image segmentation since the ball location is defined with a bounding box. However, the network for image segmentation presents several advantages. The principal ones are the computational efficiency, the accuracy by delimiting the ball, and the fact that it can be more general than object recognition and detection [9].

First, let us focus on the change of labeling of the ground truth labels. To do so, at each ball location defined, a bounding box is created around this position. The height and the width of the bounding boxes remain of the same predefined size for all the ball locations for the sake of simplicity. The width and the height chosen are equal to 12 and 10 pixels, respectively. The width is slightly bigger than the height because the ball has often a horizontal velocity on the image, inducing a horizontal elliptical aspect. The size of the bounding box is also slightly bigger than average ball size to handle potential elliptical aspect and some imprecision of the ball location due to the manually labeling.

Now, let us consider the creation of bounding boxes to replace the segmentation labeling of the detector. The first step is to use a threshold that assigns which pixels of the confidence map are belonging to the ball. To optimize the number of detected balls, the small threshold value of 0.1 is selected. Therefore the sets of pixels of a value higher than 0.1 establish the ball candidates. Next, the smallest rectangle including the candidate is fulfilled to extend the candidate to a rectangle form. Finally, the bounding box is represented by the edge of the rectangle candidate.

Two types of evaluations can be performed on the ball detection, the classification and the matching tasks. The first is based on the number of detections, such as right, missed or wrong detections, while the second is based on the exactness of the detection location. The evaluation of the results focuses only on the classification task for specific reasons. First, as said previously, the ground truth labels are manually annotated and the size of the bounding box around the ball is the same for all the balls in the training set. This induces some imprecision of the ground truth label. In addition, since the ball is relatively small in the image, the IoU between the 2 bounding boxes could highly vary with small offset. For example, for bounding boxes of size 10 x 6 pixels, if there are a perfect match of the bounding boxes, the IoU is equal to 1. However, if the ground truth bounding box is shifted of only 1 pixel down due to a human imprecision, the IoU becomes 50/60 = 0.83. This example shows that the IoU of the ground truth bounding box and the detected bounding box can vary significantly due to the imperfection of the manual labeling.

Considering the classification task, it is provided by the IoU of both ground truth and detected bounding box, since it is the common use. The threshold value for the IoU is commonly set to 0.5 [31]. However, this value was slightly decreased to 0.4 in this evalu-
ation to compensate for use of an identical bounding box size for all ground truth labels, which leads to some imprecision.

4.4.2 Result Analysis

The evaluation was made on 7 soccer sequences of 80 to 350 images. The sequences were made from the principal camera with a resolution of 960 x 540 pixels per frame and with a frequency of 25 fps. Each image selected for the evaluation contains the whole ball visible. Hence, the frames where the ball is partially or totally occluded and where the ball is out of frame are removed from the evaluation. As said previously, the goal of the detector is to get the higher number of true positives and keep a number of false positives acceptable. Therefore, the training set was evaluated by computing the number of true positives, the number of false positives as well as the true positives rate and the number of false positives.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of frames</td>
<td>282</td>
<td>270</td>
<td>211</td>
<td>188</td>
<td>311</td>
<td>212</td>
<td>59</td>
</tr>
<tr>
<td>Number of true positives</td>
<td>264</td>
<td>237</td>
<td>207</td>
<td>183</td>
<td>281</td>
<td>196</td>
<td>59</td>
</tr>
<tr>
<td>True positive rate</td>
<td>0.93</td>
<td>0.88</td>
<td>0.98</td>
<td>0.97</td>
<td>0.9</td>
<td>0.92</td>
<td>1</td>
</tr>
<tr>
<td>Number of false positives</td>
<td>1006</td>
<td>1620</td>
<td>558</td>
<td>521</td>
<td>1170</td>
<td>394</td>
<td>327</td>
</tr>
</tbody>
</table>

Table 2: Results on the first test set

By looking at the results, it is observed that the detector achieves the expected number of true positive which is less than 10 false positives per image for each sequence of the test set. However, during the creation of the detector, the results have been analyzed to understand its limitations and improve its performances. For example, the choice of adding images with multiple shadows in the training set was taken after a large number of miss detection of the ball in the test set. In addition, it was noticed later that some images used as background to create the semi-synthetic images of the training set come from the same matches than the sequences used in the test set. Consequently, the test set is not totally independent of the neural network training. To fix this issue, a second test set of 6 sequences with the same parameters has been created. Therefore, the first test set was used to analyze the results and to improve the training of the detector based on missed detections, while the second test set was used to evaluate results on data independent of those used for training. Figure 3 shows the obtained results of the evaluation made on the second test set.

Even with a slight diminution of the number of true positive, the ball remains detected in most of the cases. Considering the number of false positives, it increases until it exceeds 10 false positives per image on a sequence but should remain acceptable in general. However, the high number of wrong candidates can afflict the identification of the real ball. If
Table 3: Results on the second test set

<table>
<thead>
<tr>
<th>Sequence</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of frames</td>
<td>254</td>
<td>258</td>
<td>93</td>
<td>456</td>
<td>223</td>
<td>139</td>
</tr>
<tr>
<td>Number of true positives</td>
<td>239</td>
<td>215</td>
<td>82</td>
<td>419</td>
<td>192</td>
<td>119</td>
</tr>
<tr>
<td>True positive rate</td>
<td>0.94</td>
<td>0.83</td>
<td>0.88</td>
<td>0.92</td>
<td>0.82</td>
<td>0.86</td>
</tr>
<tr>
<td>Number of false positives</td>
<td>912</td>
<td>2925</td>
<td>241</td>
<td>2106</td>
<td>687</td>
<td>860</td>
</tr>
<tr>
<td>Number of false positives / Number of frames</td>
<td>3.59</td>
<td>11.33</td>
<td>2.59</td>
<td>4.61</td>
<td>3.08</td>
<td>6.19</td>
</tr>
</tbody>
</table>

it is the case, a method to address this issue could be the increasing of the threshold value that split the pixels to both class “ball” and “no ball”, in order to decrease the number of false positives.

Now, let us focus on the missed detection, the false negatives. Figure 13 illustrates several images where the ball is not detected. In fact, most of the missed ball detection can be stowed in a few situations: when the ball is front of a line, a player, or the public. Figure 14 resumes the partitioning of the missed ball detection into the different situations, where the case “others” contains missed detection that do not correspond to any situations appointed. Note that more than the majority of missed detections are due to a ball position in front of a player, which is an arduous situation to handle and which is often encountered in soccer sequences.
Figure 13: Several examples of missed ball detection on the sequences of the second test set
4.5 Training Set Improvements

To improve the training of the model, different investigations have been made.

As a reminder, the background image was cropped to facilitate the convergence to a local optimum that detects the ball. However, the network is evaluated on whole images. Therefore, the network should achieve better results by being trained on data with identical size to the test data, and identical ratio of “ball” class pixels to “no ball” class pixels. To assess this issue, the experiment of continuing the training of the model on whole images was realized by using the network previously produced as a pre-trained model. To create the dataset, the initial images provided by Deltatec were modified to replace the real balls by synthetic balls. To do so, the real ball present on the initial image was hidden manually with the help of the software Gimp by adding grass in front of it, and a synthetic ball was added with the method previously explained. The training parameters are the same as those used with the cropped images. However, after the training of the network, the evaluation exhibits a lower number of true positive. These are resumed in the appendix 6. The most likely cause of these disappointing results is that, even if the network is pre-trained, having a so low ration of pixels belonging to the class “ball” strongly encourages the network to reconverge towards the optimum where all the pixels are set to 0. In other words, the loss when all pixel values are 0 is smaller than the loss of the pre-trained model on whole images, due to the adding of a large number of pixels belonging to the class “no ball”.

After the result evaluation, it was noticed that two specific situations of missing are often encountered: when the ball is in front a line or a player. To decrease the number of
missing detections in these situations, two methods have been applied. First, the objective of increasing the number of images containing a ball in front of a line is considered. To do so, the experiment of modeling synthetic lines was investigated. The purpose is to add images in the training set containing a synthetic ball in front of a synthetic line. To vary the render of the lines on the dataset, some parameters were randomized:

- Since the width of the line are between 10 and 12 cm and the ball have a diameter of 22 cm, the width of the line varies from 45% to 55% of the diameter of the ball, to simulate the different line widths encountered on a soccer field.

- A random factor is applied to the color of the line to have different saturations. This is used to simulate changes in line illumination and texture. A noise is also applied on the line to add some perturbation to the uniform color of the line.

- Right lines were created, as well as curved ones to synthesize the lines present on the field.

Figure 15 illustrates several images containing a synthetic line.

![Figure 15: Semi-synthetic images containing modeling ball and line](image)

After adding these new images in the dataset, the network is trained again with using as initial weights as those previously obtained. This new model is evaluated against the previous one. Compared to the results formerly obtained, the new results do not have a higher number of true positives on the testing set, nor in the case when the ball is front of a line. These disappointing results could be due to a lack of realism of the synthesized...
lines and these should be improved to expect an increase of the number of ball detection. The result obtained on the first dataset with this model is illustrated in figure 7. Since the results are not those expected, the new network was no longer be considered.

The second method implemented consists of, during the modeling and the animation of the ball, removing the random position of the ball and placing it manually in front of a player or a line. Unfortunately, manually placing the ball on the image is a time-consuming step, inducing a low number of images generated, 250 images of each situations. However, adding 250 images on the training set containing already 8000 images weakly affects the training. For this reason, a second training set has been created that contains only the new images. The pre-trained network will therefore be re-trained on this new training set with a very low initial learning rate of $10^{-7}$. Note that this method greatly improves the number of true positive on the testing set. The weights of the network used in the result analysis were therefore those obtained after the training on the second training set.

5 Ball Tracking

5.1 Introduction

The ball detector previously introduced outputs a list of ball candidates on each frame of a soccer sequence. Even if the ball is often detected by the detector, a large number of wrong detections appears amongst the list of candidates. Therefore, a tracking system is set up to select which of the candidates happen to be real soccer balls. To do so, the tracking system uses the information given by the ball candidates of the entire sequence. Inspired by the paper of X. Yu et al. [36], the tracking system can be split into three distinct steps. First, the continuity of candidate detections is represented by the creation of trajectories along the sequence. The purpose of this step is to identify which candidates are detected for a long period of time and to eliminate the candidates detected for a short period. Second, the trajectories of the remaining candidates are assessed so that a score is assigned to them. The score defines the confidence with which the trajectory will be considered as a ball trajectory. Afterwards, a method is considered to decide which trajectories are kept according to their score and their space-time position. Finally, the remaining trajectories are extended to estimate a plausible position of the ball when it has not been detected. These 4 steps are explained in sections 5.2, 5.3, 5.4 and 5.5, respectively.

5.2 Trajectory Creation

The ball is a small object that is similar to many others on an image of a soccer sequence. It is therefore difficult to differentiate it from other ball candidates with great confidence. This is why the creation of trajectories is introduced. The aim of this task is to identify the same object on successive images, and then to define its position at a specific moment. The creation of trajectories replaces the challenge of identifying which balls are among the
candidates, by identifying which trajectories are propitious to a ball trajectory among all
the trajectories of objects. In addition, some bad candidates were detected by the ball
detector because it looked like a ball in an image due to their placement on this image or
a blur effect. In this case, this object should not be detected as a ball for a long time.
Therefore, the created trajectory of this object is relatively short. To eliminate the trajec-
tory of these wrong candidates, the potential trajectories of less than 5 successive frames
are not taken into account.

The chosen trajectory creation method is based on a Kalman filter. This algorithm pro-
vides estimates of unknown variables given the measurements observed over time [15]. The
evolution of the state from time $t-1$ to time $t$ is defined by the process model as:

$$
x_t = Ax_{t-1} + w_{t-1}
$$

where $A$ is the state transition matrix and $w$ the process noise vector.

In addition, the relationship between the state and the measurement at the time $t$ is
described by the measurement process as:

$$
z_t = Hx_t + v_t
$$

where $H$ is the measurement matrix and $v$ the measurement noise vector.

In this work, a first order dynamic model is employed, such that:

$$
x = \begin{bmatrix} x \\ y \\ v_x \\ v_y \\ bb_{width} \\ bb_{height} \end{bmatrix}, \quad z = \begin{bmatrix} x \\ y \\ bb_{width} \\ bb_{height} \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}
$$

where $(x, y)$ denotes the center of the ball bounding box, $(v_x, v_y)$ denotes the velocity
of the ball and $(bb_{width}, bb_{height})$ denotes the size of the bounding box. The process and
measurement noise vector $w, v$ are considered as Gaussian noise, such that $p(w)\sim\mathcal{N}(0, Q)$,
and $p(v)\sim\mathcal{N}(0, R)$. The process and measurement noise are of the form:

$$
w_{t-1} = \begin{bmatrix} 10^{-2} \\ 10^{-2} \\ 10^{-2} \\ 10^{-2} \end{bmatrix}, \quad v_{t-1} = \begin{bmatrix} 10^{-4} \\ 10^{-4} \\ 10^{-4} \end{bmatrix}
$$
The covariance matrix of process noise and measurement can therefore be deducted from \( \mathbf{w}_{t-1} \) and \( \mathbf{v}_{t-1} \), such that:

\[
\mathbf{Q} = \begin{bmatrix}
10^{-2} & 0 & 0 & 0 & 0 & 0 \\
0 & 10^{-2} & 0 & 0 & 0 & 0 \\
0 & 0 & 10^{-2} & 0 & 0 & 0 \\
0 & 0 & 0 & 10^{-2} & 0 & 0 \\
0 & 0 & 0 & 0 & 10^{-2} & 0 \\
0 & 0 & 0 & 0 & 0 & 10^{-2}
\end{bmatrix},
\mathbf{Q} = \begin{bmatrix}
10^{-1} & 0 & 0 & 0 \\
0 & 10^{-1} & 0 & 0 \\
0 & 0 & 10^{-1} & 0 \\
0 & 0 & 0 & 10^{-1}
\end{bmatrix}
\]

The Kalman filter works on the two following steps: the prediction and the update [35]. The prediction step allows to predict the state and the corresponding uncertainty \( \mathbf{P} \), knowing the previous states and the equations that define the movement. The update step combines the predicted state with the measurement at the current time step to refine the state estimate. Figure 16 illustrates an overview of the way the Kalman filter works.

![Figure 16: Overview of Kalman Filter System](image)

The trajectories creation is implemented as follows. First, a Kalman filter is initialized by the first candidate encountered. Then, if the bounding box predicted by the Kalman filter has an intersection with a candidate among those detected on the next frame, it is added to the candidate’s trajectory and the filter is updated with this new measurement. These steps are repeated until the prediction of the Kalman filter does not have an intersection with a candidate on the next frame to fulfill the trajectory of the candidate. At this moment, all the candidates are removed from the list of detected candidates and the trajectory is added in a trajectories pool. The next trajectories are created with the remaining candidates on the list.
Note that, since the Kalman filter is initialized with one candidate, the initial speed is set to 0. Therefore, the first prediction has the same position as the first candidate that has initialized the Kalman filter. The Kalman prediction, therefore, does not intersect the ball detection in the next image if the ball has a high speed. To address this issue, the width and the height of both predicted and detected bounding boxes on the second frame are enlarged by 8 pixels. This allows to intersect candidates that have their ball center at a distance of about 24 pixels around the initial position of the Kalman filter. By considering that the ball with a diameter of 22 cm, is represented by a diameter of 8 pixels on the image, and the sequence is filmed with a frequency of 25 fps, a ball movement of 24 pixels between two frames corresponds to a ball speed of about 16.5 m/s. Therefore, for the first prediction, the Kalman filter is capable to track a ball that has a speed of about 60 km/h at maximum.

The trajectories creation is implemented in the function `ballfinderkf_finder_kf` of the main code in the external appendix. The implementation of the Kalman filter is made with the help of the Opencv library [28].

The use of the Kalman filter allows to continue the trajectory of a candidate even if no candidate has an intersection with the predicted bounding box on a frame by simply skipping the update step. This allows to handle the situations where the ball has not been detected on a frame due to an occlusion of a short period below a player that does not modify the ball trajectory, for example. From now on, when the candidate is not detected on less than 3 consecutive frames and there is an intersection between the bounding box predicted by the filter and a candidate on the next frame, the predictions of the Kalman filter are used to complete the candidate’s trajectory. The trajectory is considered finished when there is no intersection on 3 consecutive images. An example where the Kalman filter handles such a situation is illustrated in figure [17].

### 5.3 Trajectory Score

Now that the list of all candidates of the sequence is converted into a pool of candidate trajectories, the objective is to determine which trajectories correspond to a ball trajectory. To do so, a score is attributed to each of the trajectories. The score determines with which confidence the trajectory is considered as a ball trajectory.

The first idea was to use the pixel values of the confidence map given by the detector. Indeed, each candidate had a score given by the mean value of the pixels that represents the candidate on the confidence map. Therefore, the score of a trajectory is given by the addition of the candidate scores constituting the trajectory. The addition of the candidate scores was used instead of the mean to promote long trajectories. However, this trajectory score method has several limitations:
Figure 17: Example of the creation of a trajectory with a Kalman Filter. The bounding boxes of color black, red and blue represent respectively the candidate selected in the trajectory, the candidate detected by the ball detector and a candidate estimate by the Kalman filter. In the figure (a), the black bounding box is the position of the ball in the previous frame. Since there is an intersection between the detected and predicted bounding boxes, the Kalman filter is updated with the detected ball position and this one is included in the trajectory, as shown in the figure (b). The figure (c) illustrates that there is no detected ball that overlaps the estimated ball position. Therefore, the Kalman filter makes a new prediction without update in the next frame, as shown in figure (d). This time, the prediction bounding box is overlapped with one from the detector. Hence, the figure (e) illustrates that the detected bounding box is included in the trajectory, as well as the estimated bounding box for the previous frame since no ball was detected.
• First, some people dressed in white in the public are confused with the ball by the detector. However, it is common that the public is visible for a long time on the sequence, making their trajectory very long. Because of this, the score of this trajectory is wrongfully large.

• The trajectories that contain candidates predicted by the Kalman filter are disadvantaged since the ball was not detected by the neural network, which means the confidence map pixels values are low for these candidates.

• If no ball candidates are included in the list of ball candidates, it is because the detector expresses difficulties in differentiating them from a real ball. As a result, many wrong candidates have unjustly high scores.

Due to these expressed limitations, it appears that the method used does not attribute the highest scores to the real ball trajectories with the expected precision.

To address this issue, another method is considered. This time, instead of using the pixel values of the confidence map, the score of each candidate is computed with the help of a neural network for image classification. Indeed, a classifier is created that takes as input a cropped part of an image corresponding to a candidate bounding box and return the class of the image among the two possible classes, “ball” or “no ball”.

5.3.1 Training Set Creation

To train the classifier, a labeled dataset must be created. This dataset contains cropped images of soccer game videos of size 24 x 24 pixels. The training set images are created using the ball detector on 7 soccer video sequences. Indeed, the bounding boxes of the ball candidates outputted by the detector are resized to get a cropped image of 24 x 24 pixels. Then, the images containing an entire visible ball with parameters in accordance with the range emitted in the section 4.2.1 are manually labeled to the class “ball”, while the others are manually labeled to the class “no ball”. However, since the detector returns a bigger number of wrong ball candidates, the number of images in the training set belonging to the class “no ball” is consequently larger than the images belonging to the class “ball”. This class imbalance problem can lead to a difficulty of the classifier to label the images of the “ball” class. To address this issue, a data augmentation technique is used to increase the number of “ball” class images. The technique used consists of flipping horizontally each “ball” class images to double the number of images and rebalance the two classes. Finally, 2,424 images are generated belonging to the class “ball”, while 5,055 images are generated belonging to the class “no ball”.

5.3.2 Classifier Structure

The structure of the classifier, inspired by the network AlexNet [20], is represented in the figure. Since the processed images have a small size and only two classes are possible, the
classification task is simplistic and can be handled by a non-complex network. Therefore, only three convolutional (b) (f) (i) and two fully-connected learned layers (k) (n) process successively the image (a).

![Neural network for image classification structure](image)

After each hidden learned layer, an activation function ReLU (c) (f) (i) (k) is used since it allows an acceleration of the learning time [20]. After the first ReLU function, a Local Response Normalization (LRN) (d) is applied to carry out local contrast enhancement that allows the use of locally maximum pixels as excitation for next layers [7]. The normalized values $b^i_{x,y}$ are given by the following formula:

$$b^i_{x,y} = a^i_{x,y} / \left( k + \alpha \sum_{j = \max(0, i-n/2)}^{\min(N-1, i+n/2)} (a^j_{x,y})^2 \right)^{\beta}$$

where $a^i_{x,y}$ indicates the input pixel value at the position $(x, y)$ by applying kernel $i$, $N$ the total number of channels and $k = 2, \alpha = 10^{-4}, \beta = 0.75, n = 5$ the hyper parameters.
After the second convolutional layer, a max-pooling without overlapping layer (g) is applied to divide the size of the image by two. Only one max-pooling is used in the network since the input image processed has already a small size of 24 x 24 pixels.

Before the last learned layer of the network, a dropout layer (l) is used to set to 0 the output of each hidden neuron with value 0.5, so they do not contribute to the forward pass and do not participate in back propagation. This dropout layer overcomes the aim of overfitting.

At the end of the network, a Softmax function (o) produces a distribution over the two class labels.

5.3.3 Classifier Training

Concerning the loss function, the network uses the Softmax with loss layer (p) implement in Caffe that computes the multinomial logistic loss of the Softmax of its input as follows:

\[ E = -\frac{1}{N} \sum_{n=1}^{N} \log(\hat{p}_{n,l_n}) \]

for Softmax output class probabilities \( p \), where \( l_n \) indicates the correct class label.

In addition, an accuracy layer (q) is used exclusively for the validation. This layer computes the accuracy of the output as follows:

\[ A = \frac{1}{N} \sum_{n=1}^{N} \delta \left\{ \hat{l}_n = l_n \right\} \]

where \( \hat{l}_n \) is the label estimated by the classifier, \( l_n \) the ground truth label and
\[ \delta \{ \text{condition} \} = \begin{cases} 1 & \text{if condition is true} \\ 0 & \text{otherwise} \end{cases} \]. However, since the accuracy layer has no backward step, it is actually not a loss layer.

The classifier uses a Stochastic gradient descent technique. The initial learning rate is set to 0.01 and decreases according to an exponential decay polity where \( \gamma = 0.975 \). The dataset is split to have 75% for the training and 25% for the validation.

Before the process of the image, a pixel means subtraction per channel is computed. This is used to accelerate the training by centering the data around zero for each channel R,G,B of the image.
The network is trained over 100 epochs, where the validation step is made every 10 epochs. Figure 19 made by DIGITS illustrates the values of the loss throughout training.

Figure 19: The value of the loss functions of the classifier throughout the training

5.3.4 Classifier Evaluation

The evaluation was made on 1,866 images from soccer sequences of three distinct games. Since the class “ball” is considered as the positive class, the confusion matrix of the results is resumed in table 4. Concerning the “ball” class images, the network achieves a true positive rate of 0.91 on the testing set. This measure computes the proportion of actual positives that are correctly identified. Some examples of images where the ball was not identified are illustrated in figure 20a. The reasons for the missed ball identification is often due to large blurry aspect of the ball, or the position of the ball close to a player. Concerning the “no ball” class images, the sensitivity, that measures the proportion of actual negatives that are correctly identified, reached a value of 0.98 on the testing set. Most of the false positive are images containing a goalkeeper’s glove or a white player shoe or sock. Some examples of images that are wrongly labeled in “ball” class are illustrated in figure 20b.
### Table 4: Confusion matrix of the classifier on the test set with the class “ball” considered as positive

<table>
<thead>
<tr>
<th>Ground truth classification</th>
<th>Positive</th>
<th>Negative</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positive</td>
<td>378</td>
<td>39</td>
<td>417</td>
</tr>
<tr>
<td>Negative</td>
<td>28</td>
<td>1421</td>
<td>1449</td>
</tr>
<tr>
<td>Total</td>
<td>406</td>
<td>1460</td>
<td>1866</td>
</tr>
</tbody>
</table>

(a) Examples of “ball” class images that are wrongly classified
(b) Examples of “no ball” class images that are wrongly classified

**Figure 20**

With the help of the classifier, a score is attributed to each ball candidate. Therefore, the score of a trajectory is computed by simply mean the score of the candidates that constitute the trajectory. The score assignment for each trajectory is implemented in the function `get_weight_traj` of the main code in the external appendix.

#### 5.4 Trajectory Selection

Once that a score is assigned to each candidate trajectory, a method that selects the ball trajectories is implemented.

First, a threshold value is chosen to select which trajectories are considered. Indeed, the trajectories with a score below the threshold value are not taken into account. The threshold value is set empirically to 0.2. This low value allows to consider ball trajectories where some elements did not well classified, inducing a lower trajectory score. In addition, it is noticeable that the score of the “non ball” elements is often 0, meaning that the low threshold value does not select “non ball” trajectories in practice.

The trajectory selection method considers each trajectory iteratively from the trajectory with the highest score to the one with the lowest score above the threshold value. The trajectory considered is selected if it does not have a temporal overlapping with a trajectory previously selected. In other words, a frame cannot contain candidates from different trajectories. This principle is based on the assumption that only one ball is present on the sequence. The candidate for the trajectory is therefore considered not to be a ball and the
entire trajectory is not selected.

The trajectory selection is implemented in the function $traj\_selection$ of the main code in the external appendix.

## 5.5 Trajectory Expansion

After the trajectory selection, all the remaining candidates are considered to be the final ball position along the soccer sequence. Indeed, sequence frames are labeled by the detected ball bounding box. However, some frames remain unlabeled of ball position. This lack of labels could be due to a difficult situation to handle for the detector, such as a large blurry aspect of the ball or the location on the front of a line, as well as an occlusion of the ball by a player. To estimate the ball position in these situations, a trajectory expansion method is implemented.

The method consists of interpolating the ball position between two trajectories. Indeed, the interpolation allows to create a curve which represents an estimated trajectory of the ball passing through the detected ball positions. The position of the ball estimated by the curve is used to label the frames where the ball was not detected.

The interpolation method used is the cubic spline interpolation because of its numerical stability, simplicity of calculation and smoothness of the interpolated curve [27]. Given a set of inputs $\{(x_1, y_1), ..., (x_n, y_n)\}$ where $x_1 > x_2 > ... > x_n$, the curve is defined by $n$ functions $f_1, ..., f_n$ where the respective domains are $f_i : [x_i, x_{i+1}] \rightarrow \mathbb{R}$ [17], such that:

$$f_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i - i(x - x_i) + y_i, \quad x \in [x_i, x_{i+1}]$$

The parameters $a_i$, $b_i$ and $c_i$ can be found by the properties that, the cubic spline, the first and the second derivatives are continuous at the merging points $x_i$, such that:

$$f_i(x_{i+1}) = y_{i+1}$$

$$f'_{i-1}(x_i) = f'_i(x_i)$$

$$f''_{i-1}(x_i) = f''_i(x_i)$$

The cubic spline implementation used comes from the spline library created by Tino Kludge [17] and exhibited in the external appendix. This library was preferred because the implementation is designed to extrapolate linearly out of the handle point interval. Indeed, the method uses second order polynomials for extrapolations. Therefore, the functions used are:
The figure made by Tino Kludge [17] illustrates the difference of extrapolation with the ALGLIB library [1], a well-known library for numerical analysis and data processing.

\[ f_o(x) := b_1(x - x_1)^2 + c_1(x - x_1) + y_1, \quad x \leq x_1 \]

\[ f_n(x) := b_n(x - x_1)^2 + c_n(x - x_1) + y_n, \quad x \geq x_n \]

Thenceforward, a cubic spline is also used for the extrapolation of the trajectories in the beginning and the end of the sequence.

In this work, two curves are created to estimate the ball position between two trajectories. For the first curve, the set of input refers to the ball position of the x-axis, where the \( y_i \) specifies the horizontal position of the center of ball bounding boxes and \( x_i \) specifies the corresponding frames. This curve is used to estimate the horizontal ball position on the frames between the two trajectories. The second curve follows the same method but the vertical position of the center of the bounding boxes is used instead of the horizontal position. Now that the global position of the ball is estimated, the frames are labeled by bounding boxes centered on the computed positions. The size of the created bounding boxes is equal to the average size of the bounding boxes constituting the trajectories.

Concerning the extrapolation at the beginning and the end of the sequence, the set of input is fulfilled by the ball positions of the first and the last ball trajectory, respectively.

However, the interpolation is based on the trajectories positions. Therefore, the more distant the trajectories, the less certain it is that the estimated positions correspond to the real trajectory. To limit this uncertainty, the interpolations are made when the number
of frames between the trajectories does not exceed 5. Likewise, the extrapolation is made when the number of frames between the trajectory and the limit of the sequence does not exceed 5. Figure 22 illustrates the outcome of the trajectory expansion on a sequence.

The interpolation and extrapolation of the trajectories is implemented in the function \textit{extrapolation} of the main code in the external appendix.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{trajectory_expansion}
\caption{(a) Horizontal positions of the ball throughout the sequence}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{trajectory_expansion}
\caption{(b) Vertical positions of the ball throughout the sequence}
\end{figure}

Figure 22: Illustration of the outcome of the trajectory expansion on a sequence
6 Discussion

6.1 Structure Summary

In this work, a ball tracking system for football sequences is presented. The targeted sequences are the long shot view filmed with the principal camera. This structure of the system, illustrated in figure 23, consists of two distinct modules: the Ball Detector followed by the ball Tracking Module.

The soccer sequence (a) contains each frame constituting the sequence. First, each frame (b) of the sequence is processed by the convolutional neural network for image segmentation (c). The network outputs a ball confidence map (d) of the same size as the input image. Each pixel of the confidence map has a value between 0 and 1 that represents the confidence of the pixel to be a ball pixel. Then, a threshold function (e) is applied on the confidence map to keep only the pixels above the threshold value, the others are set to 0. The remaining sets of pixels above the threshold value are represented with a corresponding bounding box to constitute the ball candidates (f) of the considered frame. These candidates are added to the candidates of the other frames to generate a pool containing all candidates of the sequence (g). The tracking module starts by using the Kalman filter (h) to create candidate trajectories when a candidate is detected on successive frames. Each candidate trajectory of the trajectory pool (i) is assigned a score with the use of a neural network for image classification (j). The network inputs each candidate of a frame and outputs the confidence of the candidate to belong to the “ball” class. The trajectory score assigned is the average confidence of each of its candidates. A ranked candidate trajectory pool (k) is therefore obtained. The trajectories with the highest scores are selected (l) and are considered as ball trajectories (m). Finally, a trajectory interpolation method (n) based on a cubic spline estimates the ball positions between the trajectories to merge them and constitute the final ball trajectory (o).
The implementation of the ball tracking system written in C++ is available in the external appendix. However, due to privacy policy, the code is only available for the jury of the Master thesis. To execute the code, a directory containing all the images of the sequence, as well as a file named “markers.txt” containing all the names of the sequence images must be referenced in the code. The code draws the obtained bounding boxes on the images and saves their in another directory. The code saves also two files containing respectively the horizontal and vertical positions of the center of the obtained bounding boxes with the corresponding frame numbers.

6.2 Result Evaluation

The evaluation of the ball tracking system is made on 6 soccer sequences from distinct matches of professional competitions, provided by Deltatec. The sequences contain between 145 and 530 images, filmed by the principal camera with a resolution of 960 x 540 pixels and a frequency of 25 images per second. The sequences include situations where the ball has a blurry aspect, is in front of the grass, a player, or a line, or is partially or totally occluded.
Table 5: Results of the ball tracking system on the test set

Table 5 resumes the number of frames, positives, true positives and false positives and the true positive rates of each sequence. The number of positives represents the number of frames that contain a ball. A frame is said to contain a ball if at minimum the half of the ball is visible on the frame. This measure is chosen for several reasons. First, there is no common agreement to determine the frames including a ball. The authors of the paper [36] consider that the frame includes a ball if “people can conclude the existence of the ball in the frame based on the previous and posterior frames even though the ball may be occluded but it must be in the field” [36], while the authors of the paper [22] consider that a frame has the ball if it can be found independently of the adjacent frame. Unlike these subjective measurements, a measure that was verifiable and easily estimated by the human eye was preferred: the visibility of half the ball. In addition, the ball detector is trained on images where the entire ball is visible. However, in practice, the ball is often recovered by the tip of a player foot, hugely decreasing the number of entire balls in the test set. Therefore, it is considered that the ball detector should identify the ball positions on the frames where 50% of the ball is visible at least. The true positives are the frames where the IoU between the detected and the ground truth bounding boxes of the ball is higher than 0.4, while the false positives represent the frames where this IoU is below or equal to 0.4. The numbers of true and false positives are computed only on the frames that contain a ground truth label. The number of detected and estimated ball positions adds the number of true positives with the number of positions that are estimated on frames that do not contain the ball. The last measure of table 5 divides this number with the number of frames on the sequence.

Note that no false positive appears on the results, meaning that each detection determined by the system identifies the ball correctly. Concerning the true positive rate, the least is made on sequence 3, which achieves a rate of 0.58. The most probable cause of the huge number of missed detection frames is the large ball size visible on the frames. Indeed, a part of the sequence exhibits the ball close to the camera, inducing a ball size around 12 x 12 pixels on a set of frames. In addition, a large black line is drawn on the ball, which makes detection even more difficult. A frame containing this ball is represented in figure 24a. Concerning the 5 other sequences, the system achieves a true positive rate between 0.85 and 0.9. The frames where the ball position is not found are mainly characterized
by two situations: when the ball is shot with a high speed inducing a blurry aspect or when the ball is close to a player for a long time inducing recurrent partial occlusions. An example of these situations is illustrated in figures 24b and 24c respectively. However, the ball tracking system is capable of handling situations where the ball is moving with a low speed, or when it is occluded in a few frames.

However, the results are specific to the test set used. Knowing that the sequences of the test set come from the company, it is not possible to compare the results with methods where the results have been evaluated on another test set. The publicly available test sets for ball detections observed during the research can not be used for a commercial activity. Since this work is coupled with an internship within the Deltatec company, these publicly available test sets were not used and a specific one was created.

6.3 Additional Processing

Although the ball tracking achieves convincing results on the test set, additional processing has been investigated in order to tweak the system.

The first limitation can be perceived in figure 25. It illustrates the vertical and horizontal ball positions throughout the sequence 6 of the test set. From the frame 90 onwards, the ball is visible with a high speed but the ball tracking outputs no ball position, even if a correct ball candidate is detected on each frame. Due to the high speed, the Kalman filter is not able to predict the ball position with enough precision to intersect the ball candidate on the next frame.

To improve the handling of such a situation, a method was considered. This method considers taking into account the movement of the camera. Indeed, the soccer sequences evaluated are filmed with a moving camera, inducing changes of direction of the ball on the images.

A software developed by the company Deltatec allows to estimate the camera settings such as the pan, tilt, zoom, and the location in space of the camera. Knowing this information, a function implemented by Deltatec converts the ball position on the image to a position in space, assuming that the ball is on the ground. Finally, the 3D position of the ball is reconverted in a 2D position on a hypothetical image obtained if the camera did not move. Therefore, the ball trajectory on the theoretical frames is not disturbed by unpredictable camera movements and the Kalman filter will be able to predict the position on the next image more effectively.

These functions can also improve another limitation of the ball tracking. Indeed, during the trajectory selection, the considered trajectory is selected if it does not overlap with a previously selected trajectory. However, no constraint is applied concerning the distance between the trajectories. Without space constraint between trajectories, if two trajectories follow one another, the ball position could move an unfeasible distance between the last
(a) Missed ball tracking due to a large size of the ball

(b) Missed ball tracking due to high speed of the ball

(c) Missed ball tracking due to partial occlusion of the ball

Figure 24: Examples of missed ball tracking
(a) Horizontal positions of the ball throughout the sequence

(b) Vertical positions of the ball throughout the sequence

Figure 25: Ball position throughout the sequence 6
position of a trajectory and the first position of the next one. To address this issue, the functions implemented by Deltatec, previously introduced, can be used again. Indeed, by converting the 2D ball position on the image into a 3D position in space, the maximum distance that the ball is able to move between two successive frames can be computed. By considering that the maximum ball speed is about 36 m/s \cite{13} and knowing that the sequence is filmed with a frequency of 25 fps, the ball can move at a maximum of 1.44 meter between two successive frames. The maximum distance between the ball positions of the two trajectories is determined by multiplying this number with the number of frames that separates the two trajectories. Finally, the considered trajectory must be selected only if the distance that separates this trajectory with the previously selected trajectories is under the maximum distance.

Note that this technique can be useful during the trajectory creation. Indeed, since the Kalman filter is initialized with a null velocity, the object on the second frame is searched around the object position of the first frame. During the initialization of the trajectory with the Kalman filter, the velocity is set to 0 since the movement of the ball is not known. The second position is therefore researched around the first position. The search range can, therefore, be determined by taking the maximum distance that the ball can move between two successive frames, previously computed.

This technique is nevertheless based on the assumption that the ball is on the ground. However, it is recurrent that the ball is in the air during a football match, which can lead to the unreliability of the computed distances.

Unfortunately, the implementation of this processing has been aborted due to the restrictive schedule of the internship and the quantitative limitations of this work.

### 6.4 Future Works

It should be recalled that the ball tracking system presented in this work uses two neural networks, one for image segmentation and the second for image classification, which have a huge impact on the performances of the ball tracking system. Therefore, the training of the networks is a key point of the system and can be refined. In fact, the more significant the training set, the more the network will be able to handle different situations. Thus, throughout the future use of the ball tracking, specific circumstances where the ball has not been tracked correctly can be used to fulfill the training set and to improve the results in such situations. This optimization is specifically interesting in this work since the network for image segmentation is trained on images with synthetic balls. Thus, some features that are difficult to model can be handled by the network simply by adding some images where these particulars features are encountered on the training set.

Another future work concerns the improvement of the trajectory expansion. Now, the trajectories are extrapolated using a cubic spline. However, these estimated positions do
not have any visual validation. A possible improvement of the extrapolation could be the use of the classifier to validate if the ball position estimated by the spline has a visible ball. In this case, the extrapolation could be made on a larger number of frames.

As described in the previous sequence, it is possible to estimate the 3D ball position in space. However, the estimation is based on the assumption that the ball is on the ground. This assumption can generate significant inaccuracy since it is common that the ball is in the air in a football match. To address this issue, another method to evaluate the 3D ball position when it is off the ground must be considered. The paper of Y. Li et al. [21] proposes a method based on a parabolic model to estimate the 3D positions of the ball when it is in the air. The method assumes that the ball follows a parabolic curve with a negligible friction between the time \( t_1 \) and \( t_2 \), at which it bounces against the ground or a player. From knowledge of the 3D positions \((x_{t_1}, y_{t_1}, z_{t_1})\) and \((x_{t_2}, y_{t_2}, z_{t_2})\) as well as the time interval \( t_2 - t_1 \), the 3D trajectory is determined in 3 steps. First, the time \( t' \) that corresponds to the highest point of the parabola is computed by:

\[
t' = \frac{p_{t_1} - p_{t_2}}{g(t_1 - t_2)} + \frac{t_1 + t_2}{2}
\]

where \( g \) is the gravitational acceleration. Then, the initial vertical velocity \( v_{1z} \) is computed. Finally, the iterative computation of the trajectory in finite time intervals allows the estimation of a 3D position at each time step, using the determined initial conditions.

Finally, the 3D ball positions are hugely beneficial for game analysis. Coupled with the players’ positions, the ball positions are used in the paper of X. Yu et al. [36] to analyze the ball possession and identify the passes and the shots. The paper of W. Naidoo et al. [26] uses the ball and players’ positions to proposes an automatic offside event detection method.

### 6.5 Conclusion

In this work, an offline ball tracking method in long-shot soccer sequences has been proposed. The method uses a ball detector based on a deep convolutional neural network for image segmentation to detect candidates on each frame. The results of the detector on an independent test set achieves a true positive rate between 0.82 and 0.94, depending on the sequence. This detector is able to handle several situations such as a blurry aspect of the ball due to high speed, or situations where the ball is touched by a player.

The ball detector is followed by a tracking module that uses the inter-frame information to identify the ball trajectory. To do so, a method based on the Kalman filter is implemented to create a pool containing candidate trajectories. The trajectories of the real ball are selected by a method based on a neural network for image segmentation and the trajectories are extrapolated with the help of a cubic spline interpolation.
This method has been evaluated on a test set containing 6 sequences of professional soccer matches. The sequences filmed with the principal camera at a frequency of 25 fps. The results of the implemented method do not exhibit false positive on any sequence, meaning that the system does not output any wrong ball position on the test set. Concerning the predicted ball positions, the ball tracking found the ball position on more than 80% of the images on most of the test set sequences. The method is capable of handling situations such as the ball movement with a modest speed or occlusion in a court period.

However, the poor ball position of 58% obtained on a precise sequence motivates the aspiration of training the neural networks with more data in order for them to be more robust on the ball and scene variance. In addition, the system expresses some difficulties to track the ball when it has a high speed. To address this issue, a method consisting of taking into account the movement of the camera in order to facilitate the ball prediction of the Kalman filter is considered. This is achievable by converting the 2D ball position on the image to a 3D ball position in space when the ball is on the ground. This method allows also validation of the consistency of the space-time distance between the ball positions, considering the maximal ball speed. Unfortunately, the method implementation has been aborted due to the restrictive schedule of the internship.

In future works, when it is off the ground, the 3D positions of the ball can be determined using a parabolic model to estimate the location. In addition, coupled with the players positions and identifications, several analyzes can be generated such as the ball possession or an automatic offside event detection method.

To conclude, this work has presented a ball tracking method for football sequences. This system is able to identify the ball position in most of the encountered situations and different methods have been introduced to improve the accuracy of the ball tracking.
References


A Appendix A

Strategic Analysis of the Company

This Master thesis was done during an internship at Deltatec Company. In this appendix, a description of the company Deltatec is introduced, with an analysis of the strategies adopted by the company to achieve its objectives. The purpose of this appendix is to present the key points of the company Deltatec, allowing it to gain an advantage over its competitors. The informations used in this appendix come from the websites of the company [3, 2, 4].

A.1 Description of the Company

Deltatec is a Belgian company founded in 1986, now located in Ans. First established as a hardware design house, the company activity has diversified to specialize in several advanced software and hardware technologies. As of today, Deltatec is active in several sectors such as aeronautics, space, TV broadcast, and industry. The company is constantly developing by acquiring new skills, by recruiting new staff, by participating in the realization of cutting-edge projects. The company is composed of about 80 employees who are distributed into specific three Business Units:

Deltatec: This Unit is responsible for custom design for several types of customers. The activities of Deltatec affect the space and the ground area. In the space segment, the activities of Deltatec consist in designing data processing subsystems, especially on the electronics of the camera that the satellites use for Earth and Sun observation. This type of use requests very high reliability to withstand extreme situations such as mechanical vibration and extreme temperature. Concerning the ground segment, the role of Deltatec consists of designing specific test beds and video processing and video processing applications.

Deltacast.tv: This Unit provides a range of cost-effective video cards used in products of professional broadcast. These cards are capable of ensuring the transmission of high quality video streams for professional video developers.

Deltacast: This Unit provides virtual graphics for live events of several sports. Deltacast uses the high performances of the artificial intelligence coupled with algorithms of images processing adaptive chroma-keying and a 3D engine to propose a catalogue of graphical products. Among the catalogue, the main products are DELTA-highlight, a virtual graphic tool for sport games analysis, or the virtual view that allows the High realistic 3D recreation of sport scenes. My internship took place in this unit.
All these units are located at the same building to promote knowledge exchanges and share everyone’s competences.

A.2 Marketing Strategy

The first objective of Deltatec is to provide and develop state-of-the-art products of high technologies. These advanced products allows to the customer to get a competitive advantage over its competitors. Deltatec is able to adapt in order to meet customer needs. This can range from a pure hardware design to a complete design (hardware, mechanical, software), including management. The company attempt to reduces the development cycle and in order promotes marketing in a shortest time, by always keeping an excellent design and high quality products. Deltatec offers to the customers a client support on the developed product throughout its lifetime. Generally, Deltatec retains no copyrights on the sold products.

The second marketing strategy of Deltatec concerns the Development of competence centers within the company. Considering the large complexity of the developed products, the technology can not be mastered by one person. This is why the company operates as a competence center in the Business Units. Each competence center, made up of one or more people, is an expert in its field. It keeps itself up to date with regard to the novelty, it can be consulted for assistance or for an expertise in a company project, which is very important in the realization of complex technical and technological projects in which the company participates. Unfortunately, Deltatec was unable to attend the event this year due to the Covid-19 health crisis.

A.3 Human Resources Strategy

Within the company, several committees have been set up for the organization of different aspects external to the work within the company. Each committee is made up of around 4 people. These are resumed below:

- Social works: This committee takes the initiative to support causes throughout the year. Recently, the committee made a donation to the Léon Frédéricq foundation in favor of the CHU de Liège to fight against Coronavirus.
- Sports: This committee organizes sports activities, such as football matches or running races, during lunchtime or after work.
- Kids: They research activities for staff who have children.
- Celebration: They organize team building activities and parties after work. A film evening was planned this year, which unfortunately was canceled due to social distance measures.
• Foot and drink: This committee ensures that there is always enough drinks and snacks for all the staff of the company. Many fruits are also available every day. In addition, an aperitif is scheduled every Friday afternoon by this committee, which allows to chat with the other employees in pleasant conditions.

• Ecoteam: The committee makes daily researches to decrease the environmental footprint of the Company.

To improve friendliness within the company, different entertainments have been set up within the cafeteria to maintain good hearing in the workplace. The entertainments include video games, table soccer, as well as a dart game.

Well-being at work remains a value of the company. Despite the health crisis linked to the pandemic, all employees were able to benefit from teleworking by installing adequate tools, including for me as intern. I was able to continue my internship thanks to the supply of adequate equipment and by setting up meetings and regular monitoring of my work.

A.4 Recruitment Strategy

Considering the recruitment, the company has the aspiration to expand the business and the general knowledge. To do so, the company is looking for a number of engineers and computer scientists to work on new projects. For example, the company came to the Job Fair at the University of Liège for several years. This event allows companies to exchange with applied sciences students who will soon graduate. In addition, Deltatec has the opportunity to present a conference in order to introduce in more details the field of activity and the different profiles sought. The company also presents a booklet containing Master thesis available in partnership with the company. This Master thesis took place following discussions that were undertaken at the Job fair.

Each year, Deltatec organizes an “immersion Deltatec job” day which allows interested people to discover the researches carried out within the company. This day permits to show the completed projects and to explain how the company is structured.
B Code labeling

```python
import cv2
import glob

for image_path in glob.glob('labels/*.png'):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    thresh = 100
    bin_img = cv2.threshold(img, thresh, 255, cv2.THRESH_BINARY)[1]
    name = image_path.split('/')[−1]
    label_path = "labels/" + name
    cv2.imwrite(label_path, bin_img)
```
import glob
from PIL import Image
import cv2
import random
i = 0
for filename in glob.glob('../Images/*.png'):
    img = Image.open(filename)
    width, height = img.size
    fname = filename.split('/')[-1]
    fname = fname[:-4]
    a = 0
    while a < 3:
        new_width = width * 0.3
        new_height = height * 0.3
        hor_start = random.random() * (width - new_width)
        ver_start = random.random() * (height - new_height)
        crop_img = img.crop((hor_start, ver_start, hor_start + new_width, ver_start + new_height))
        name = '../Cropped2/' + fname + str(a) + '.png'
        crop_img.save(name)
        contr_img = cv2.imread(name)
        alpha = random.random() * 0.6 + 0.7
        beta = random.random() * 100
        crop_img2 = cv2.convertScaleAbs(contr_img, alpha=alpha)
        name2 = '../Cropped2con/' + fname + str(a) + '.png'
        cv2.imwrite(name2, crop_img2)
        a += 1
    if i % 100 == 0:
        print('{} images processed'.format(i - 1000))
    i += 1
### D Tables of non-optimal results

<table>
<thead>
<tr>
<th>Sequence</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of frames</td>
<td>282</td>
<td>270</td>
<td>211</td>
<td>188</td>
<td>311</td>
<td>212</td>
<td>59</td>
</tr>
<tr>
<td>Number of true positives</td>
<td>243</td>
<td>187</td>
<td>84</td>
<td>158</td>
<td>219</td>
<td>170</td>
<td>41</td>
</tr>
<tr>
<td>True positive rate</td>
<td>0.86</td>
<td>0.69</td>
<td>0.4</td>
<td>0.84</td>
<td>0.7</td>
<td>0.8</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Table 6: Results on the first test set of the ball detector network trained on entire images.

<table>
<thead>
<tr>
<th>Sequence</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of frames</td>
<td>282</td>
<td>270</td>
<td>211</td>
<td>188</td>
<td>311</td>
<td>212</td>
<td>59</td>
</tr>
<tr>
<td>Number of true positives</td>
<td>260</td>
<td>237</td>
<td>195</td>
<td>172</td>
<td>266</td>
<td>204</td>
<td>58</td>
</tr>
<tr>
<td>True positive rate</td>
<td>0.92</td>
<td>0.88</td>
<td>0.92</td>
<td>0.91</td>
<td>0.86</td>
<td>0.96</td>
<td>0.98</td>
</tr>
</tbody>
</table>

Table 7: Results on the first test set.