Comparison of Player Tracking-by-Detection Algorithms in Football Videos

SUPING SHI
Comparison of Player Tracking-by-Detection Algorithms in Football Videos

SUPING SHI
Abstract

In recent years, increasing demands on sports analytics have triggered growing research interest in automatic player tracking-by-detection approaches. Two prominent branches in this area are Convolutional Neural Network (CNN)-based visual object detectors and histogram-based detectors.

In this thesis, we focus on a particular sub-domain: player tracking by detection in broadcast football games. To tackle challenges in this domain, such as motion blur and varied image quality, two different systems are proposed based on histogram and CNN respectively. With the help of transfer learning, the CNN-based system is fine-tuned from a pre-trained Tiny-You Only Look Once (YOLO)-V2 model. Experiments are conducted to evaluate the CNN-based system against the histogram-based system and off-the-shelf benchmarks, such as Faster Region-based convolutional Neural Networks (R-CNN). Results indicate that the CNN-based system outperforms the others in terms of mean Intersection Over Union (IOU) and Mean Average Precision (mAP).

Furthermore, we combine the CNN-based system with a histogram-based post-processor to take advantage of the player’s visual appearance characteristic. The combined system is evaluated against the pure CNN-based system and CNN-Simple Online and Realtime Tracking (SORT) system. Results reveal that the combined system manages to achieve better detection accuracy in terms of F1 and ITP scores.
Sammanfattning

Under de senaste åren har ökande krav på sportanalyser resulterat i ett växande forskningsintresse för automatisk spelarspåning. Två viktiga metoder inom detta område är CNN-baserade visuella objektdetektorer och histogrambaserade detektorer.

I rapporten fokuserar vi på ett visst underområde, nämligen spelarspåning genom detektion i direktsändning av fotbollsmatcher. För att hantera utmaningar som rörelseoskärpa och varierande bildkvalitet, föreslås två olika system baserade på histogram respektive CNN. Med hjälp av överföringsinlärning finjusteras det CNN-baserade systemet med utgångspunkt i en förtränad TinyYOLO-V2 modell. Experiment genomförs för att utvärdera det CNN-baserade systemet mot det histogrambaserade systemet och standardlösningar som R-CNN. Resultaten indikerar att det CNN-baserade systemet ger bättre resultat vad gäller medelvärden som IOU och mAP.

Dessutom kombinerar vi det CNN-baserade systemet med en histogrambaserad postprocessor för att också använda oss av spelarens visuella karakteristika. Det kombinerade systemet utvärderas mot det rena CNN-baserade systemet och CNN-SORT systemet. Resultaten visar att det kombinerade systemet lyckas uppnå bättre detektionsnoggrannhet när det gäller F1 och ITP poäng.
Acknowledgment

Firstly, I would like to thank my supervisors Volodya Grancharovat in Ericsson Research for his great advices and prompt feedback on this project. I would also appreciate Sigurdur Sverrisson in Ericsson research for knowledge sharing and Harald Plboth for his line support and keen interest in this topic.

Secondly, I want to express my gratitude to my academic supervisor Märten Björkman from KTH for all the good technical discussions and suggestions on the writing of this thesis.

Lastly, I would like to appreciate my beloved husband and parents, who give me consistent supports during my study.
## Contents

1 Introduction .................................................. 6  
   1.1 Background ............................................. 6  
   1.2 Ethical Problems and Social Implications ................. 7  
   1.3 Research Questions ..................................... 8  
   1.4 Thesis Organization .................................. 8  

2 Methodology and Related Works .......................... 9  
   2.1 Multi-Object Tracking ................................ 9  
      2.1.1 Tracking by Detection ........................... 9  
      2.1.2 Pioneering Work on Player Detection and Tracking .... 10  
   2.2 Visual Object Detectors in Sports Videos ............... 11  
   2.3 CNN-based Visual Object Detectors .................... 11  
      2.3.1 Overview of the Field .......................... 11  
      2.3.2 Faster R-CNN ................................ 12  
      2.3.3 YOLO ........................................... 14  
      2.3.4 Single Shot MultiBox Detector .................. 16  
      2.3.5 Transfer Learning ............................. 16  
   2.4 Histogram-based Detectors ............................ 17  
      2.4.1 Color Space .................................. 17  
      2.4.2 Color Histogram Distribution .................... 18  
      2.4.3 Metric for Histogram Distance ................. 18  

3 Methods ....................................................... 20  
   3.1 Dataset ............................................... 20  
      3.1.1 Training Sequences ............................ 20  
      3.1.2 Testing Sequences ............................. 22  
   3.2 Color Histogram-based System ........................ 24  
      3.2.1 Color Histogram Distribution ................... 25  
      3.2.2 Evaluation of Similarity ....................... 25  
   3.3 CNN-based System ................................... 27  
      3.3.1 Network Architecture of CNN .................... 27  
      3.3.2 Training ..................................... 29  
   3.4 Combined System ..................................... 29  
      3.4.1 Histogram-based Post Processor ................. 29  
      3.4.2 Workflow .................................... 32  
   3.5 Evaluation .......................................... 34  
      3.5.1 Evaluation Metrics ............................ 35
## 4 Experiment Results

4.1 Histogram-based System ........................................... 36
  4.1.1 Positive Detections ........................................... 36
  4.1.2 Negative Detections ......................................... 37

4.2 CNN-based System .................................................. 37
  4.2.1 Comparison between CNN-based System and Histogram-based System ........................................ 39
  4.2.2 Comparison between CNN-based System and Off-the-shelf Systems ........................................... 40

4.3 Combined System .................................................... 43
  4.3.1 Evaluation of Tracking Performance ........................ 43

## 5 Discussion and Conclusions

5.1 Limitations ........................................................... 44
  5.1.1 Dataset .......................................................... 44
  5.1.2 YOLO .............................................................. 45
  5.1.3 Histogram-based System ....................................... 45

5.2 Conclusions ........................................................... 46
  5.2.1 Future Work ...................................................... 47
Glossary

ANN  Artificial Neural Network. 11

CNN  Convolutional Neural Network. 1, 2, 3, 7, 8, 10, 11, 12, 17, 20, 27, 28, 29, 32, 33, 34, 37, 39, 40, 41, 42, 43, 45, 46, 47

COCO  Common Objects in Context. 16

DPM  Deformable Parts Model. 11, 14

GPU  Graphics Processing Unit. 29

HOG  Histogram of Oriented Gradient. 11, 18

HSI  Hue-Saturation-Intensity. 17

HSV  Hue-Saturation-Value model. 17, 18

ILSVRC  ImageNet Large Scale Visual Recognition Challenge. 16

IOU  Intersection Over Union. 1, 15, 35, 39, 46

ITP  Identity Tracking Performance. 35

LAB  LAB color space. 17

mAP  Mean Average Precision. 1, 42, 46

PASCAL  Pattern Analysis, Statical Modeling and Computational Learning. 13, 16

R-CNN  Region-based convolutional Neural Networks. 1, 2, 8, 11, 12, 13, 14, 16, 22, 34, 40, 42, 43, 46

RGB  Red-Green-Blue model. 17, 18, 25

RPN  Region Proposal Network. 13, 14

SIFT  Scale Invariant Feature Transform. 11

SORT  Simple Online and Realtime Tracking. 1, 7, 34, 43, 46
**SSD** Single Shot MultiBox Detector. 16

**SVM** Support Vector Machine. 12

**VOC** Visual Object Classes. 13, 16

**YOLO** You Only Look Once. 1, 2, 3, 11, 14, 15, 16, 17, 22, 34, 40, 41, 42, 45, 46

**YUV** YUV color space. 17
Chapter 1

Introduction

1.1 Background

In the past few decades, football has become one of the most popular sports that millions of people enjoy. With the expansion of internet and new media technologies, football analytics has gained growing attentions and demands from football clubs and players. One of the top requirements for football analytics is to get tracking information of players in football games, which will support coaches and sports scientists to prepare attack and defense strategy for future games. Also, players may have a need for such information to review and improve their performance during training or official games. To serve this demand, new methods for football analytics are proposed and developed by researchers on a daily basis. Detection and tracking of players in a football game are two essential parts within this area.

It has been a challenge for a long time to automatically detect and track players across a scene in video streams. Requirements on the quality of football analytics demand that the accuracy of player detection and tracking approaches needs to be guaranteed. There are several factors that may pose a negative influence on the performance of player detection and tracking in football games:

- Motion blur: Players in the videos of football games usually move with high speed and may appear blurry in the image.
- Complex motion pattern: Players usually have much more human posture patterns than normal pedestrians.
- Severe occlusion between the multiple players of interest.

Due to these reasons, player detection and tracking in broadcast football videos is a particularly challenging task in computer vision area. One of the prominent methods to tackle this task is background subtraction. The modelling of background subtraction contains two steps:

- The initialization of the background.
- The update of the background.
Background subtraction normally performs better with a static camera than a moving one. The changing of the background may have a negative impact on the detection performance. Template matching is also widely used in sports graphics systems. In the paper [1], the American football players are tracked by template matching and Kalman filtering. Tracking-by-detection has come into our view in recent years. The detection algorithm is continuously applied on each frame of a video sequence to generate region proposals of players. Then the association of detections between consecutive frames are involved in the tracking-by-detection method.

Nowadays, the attention is rising rapidly on applying CNN as a detector in the tracking-by-detection method, since it can achieve a significantly fast speed and high accuracy when dealing with visual recognition tasks. In the paper [2], the authors proposed a CNN-based detector, namely SORT, for multiple object tracking. It mainly focuses on frame-to-frame associations of objects for online and real time applications. The authors approximated the inter-frame displacements of each object with a linear constant velocity model.

The tracking-by-detection area has recently attracted interest from the research community and sports analytic companies. As one of such organizations, the Piero sports team aims to build an autonomous system to produce player detections in broadcast football videos. The objective of this thesis is to develop different tracking-by-detecting algorithms in football games and compare them with benchmarks.

1.2 Ethical Problems and Social Implications

This thesis proposes several systems for player tracking-by-detection in football videos and strives to benchmark the performances of the proposed systems on broadcast football video sequences. The outcome of this work would be comparative results of the proposed systems for player tracking-by-detection in football games. Companies with relevant interests can reference to this work and expect valuable information depending on their requirements.

The research of this topic requires a certain amount of training data to study players’ characteristics in football games. As we have entered the age of data, ethical considerations on data collection and usage should be well taken. Our work requires a large scale of data analysis in football videos, which may expose us to an ethical problem regarding data privacy. As Jules and Tene mentioned in [3]: Big data poses big privacy risks. The harvesting of large sets of personal data and the use of state-of-the-art analysis implicates growing privacy concerns. The data used for training and testing our neural networks is public football game videos provided and authorized by the Piero team. Before network training and testing, we manually draw bounding boxes and overlay them on top of players. Though we do not get the authorization from each player whose image is used in this experiment, the authorization for using the video sequences of the games is granted. Nowadays, videos of football games are widely used for analyzing a team or a player. As is noted in [4]: Researchers are legally obliged to conform with legal regulation relating to their research. Though ethics problems are not equivalent to legal problems, we still should take them carefully and seriously. To avoid ethical issues and respect people’s privacy, our data would not be transferred to any other parties for other usages.
1.3 Research Questions

This work is conducted under the supervision of Ericsson Research team in Ericsson AB. The video clips are provided by the Piero team [5], and the collection of football videos should not be managed by this thesis. This paper first presents two base algorithms for tracking-by-detection of players in real-time football videos, and then compares them performance-wise with conventional benchmarks. The idea of combining the two base systems to take advantage of both algorithms is also investigated and evaluated. The major research questions in this thesis can thus be formulated as follows:

1. How are the performances of the histogram-based system and CNN-based system? Which one performs better on our real-time testing sequences?

2. How is the performance of the CNN-based system compared to the off-the-shelf systems, such as Faster R-CNN? What contributes to the success or failure of this proposed system?

3. How is the performance of the combined system? What contributes to the success or failure of this proposed system?

1.4 Thesis Organization

This thesis is divided into individual chapters and organized as follows.

Chapter 1 provides a brief introduction of the background. Ethical problems and social implications are discussed in this chapter and followed by research questions.

The methodology and related research works are discussed and illustrated in Chapter 2. Previous research works for multi-object tracking, such as CNN and histogram-based detectors, are walked through in this chapter.

In Chapter 3, detailed explanations of the proposed systems are demonstrated, as well as the preparation of dataset for training and testing. The evaluation of the proposed systems are also included in this chapter.

Chapter 4 demonstrates and interprets the experiment results for each proposed system.

Chapter 5 structures a discussion about potential limitations of the results. The thesis is concluded with answers to the research questions and future work.
Chapter 2
Methodology and Related Works

2.1 Multi-Object Tracking
When it comes to computer vision, the first thing that comes into our minds would be image classification. Classification is one of the fundamental tasks in computer vision area. Though we can use a classification method to recognize an object, it fails to provide us with the position information. Due to the academic and commercial potentials, methods for object detection has gained increasing popularity. In 2001, an efficient algorithm for face detection was developed by Paul Viola and Michael Jones in [6]. The algorithm is fast enough to perform face detection in real-time videos. New methods for object detection have been consistently introduced since then and triggered numerous innovative ideas in related areas.

Multi-object tracking is one of the most popular topics nowadays within the tracking area. It is hugely required in diverse areas and scenarios, such as sports analytics and self-driving cars. Sports players, pedestrians or vehicles can all be regarded as the objects to be tracked. The objective of multi-object tracking is to locate multiple targets in consecutive video frames and label their identities. When narrowed down to sports video scenarios, multi-object tracking is faced with several challenges. As presented in the paper [7], though people have proposed several methods to tackle this problem, these methods still suffer from difficulties, such as object occlusions and close similarity of multiple objects. One of the key publications in sports tracking [8] proposed an approach to solve the problem of labeling the identities by using track graphs to track isolated objects. In this way, the identity in each track graph can be well maintained. The paper defined a similarity metric for each isolated track graph. By doing so, the identities of the isolated tracks can be associated easily.

2.1.1 Tracking by Detection
As the name indicates, tracking by detection aims to achieve object tracking by continuously applying a detection algorithm to consecutive frames of a video sequence. The generated detections in a given frame are then associated with
those in the previous frame. Methods for association of detections across frames is studied in these papers [9–12].

**Tracking**

In the simplest form, tracking is mainly defined as estimating the trajectory of an object as it moves in a video or a moving scene. In object tracking, the detection model and the tracking strategy are two key components. There are plenty of tracking strategies that can be applied for object tracking, such as Kalman filter [13] and Particle filter [14].

**Visual Object Detectors**

The detection model plays an important role in the detection-based tracking. With the fast development of object tracking research, various types of detection models have been proposed for object tracking in the previous works, such as feature points [15], color [15–20], templates [15, 21–23], and moving areas [24]. The best-known detectors before the emergence of CNN model are Viola and Jones’s algorithm proposed in [25] and the deformable part models proposed in [26]. The paper [25] described a machine learning approach for visual object detection. It introduced a new image representation called ‘integral image’ to accelerate the computation process. A learning algorithm based on Adaboost was then applied in the paper. [26] proposed a detection system using a mixture of multiscale deformable part models. Instead of using one model for object tracking, the detection model in [26] utilized a root filter together with part filters so that it could represent highly variable object classes.

Recently, the development in the area of object detection has contributed an increasing amount of promising detection models. Various object detectors provide a diverse source of detection models for tracking-by-detection tasks. For example, face detectors were applied for player tracking and evaluated positively, as shown in the paper [19].

### 2.1.2 Pioneering Work on Player Detection and Tracking

Even though multi-target tracking is widely developed and utilized for sports game analysis, it is still faced with challenges and has to cope with the fact that players typically move fast and adopt unusual postures when competing in sports games. In [27] and [1], a pipeline was proposed to solve this task: Players were first segmented and filtered out from each video frame. The background was assumed to have a uniform color. Then the filtered players were tracked by template matching and Kalman filter. In [27], teams of players were identified on the basis of color distributions. In this way, they first created a binary image representation based on the distance to the mean color of the background. Then they applied a threshold to the distance based on 3 standard deviations. In order to remove small errors, erosion and dilation were applied to the binary image representation.
2.2 Visual Object Detectors in Sports Videos

As mentioned above, tracking players in a sports game is challenging due to the fact that players usually have more postures than regular pedestrians. The speed of players is usually faster than that of pedestrians as well. Besides, occlusions of players happen frequently during sports games. All these issues add up to the difficulties in applying visual object detectors for player tracking.

As we mentioned before, Viola and Jones’s algorithm in [25] and Deformable Parts Model (DPM) proposed in [28] are the best-known detectors before the emergence of CNN model. In the paper [16], the authors implemented a pipeline to detect and track players in hockey games based on Viola and Jones’s algorithm. In order to adapt the detection model to track players, the paper [28] proposed a deformable parts model instead of treating the detection model as a whole. Since players usually have more complicated postures than pedestrians, the deformable parts model is adequate to capture each part of these postures and model players when putting the posture detections together.

The visual object detectors are usually far from perfection. The main problems of detection-based object tracking algorithms are poor detection precision and false positives. As a solution to mitigate these problems, methods in [29] and [17] fused multiple cues concurrently. In the paper [17], a weighted mask was applied to focus the descriptor vector on the team uniform’s area in a bounding box. This practice guarantees that the upper-middle region of the bounding box is mainly considered, around which the uniform is expected to appear with a higher possibility.

Recently, CNN-based detectors has achieved extraordinary success in the sports analytic area. The paper [2] proposed an efficient algorithm to achieve online and real-time applications for object tracking. One similar application of CNN-based detectors in tracking-by-detection tasks in sports videos can be found in [30]. It followed the scheme proposed in [2] but replaced the Faster R-CNN with a YOLO-based detector. They proposed an online multi-detection and tracking framework and performed experiments on a basketball dataset for evaluation.

2.3 CNN-based Visual Object Detectors

For a number of years, object recognition and detection in computer vision have been relying on color histograms or hand-designed features, such as Scale Invariant Feature Transform (SIFT) and Histogram of Oriented Gradient (HOG). However, these approaches only work well with low-level image details but fall short in high-level information. Motivated by the recent success of deep learning on object detection and recognition, we propose a CNN-based system in this thesis and compare it with a variety of mature CNN architectures. In this section, a brief introduction of relevant CNN-based visual object detectors are covered as follows.

2.3.1 Overview of the Field

Artificial Neural Network (ANN) is a family of models inspired by human brains. It can learn how to represent complex non-linear relations between inputs and
outputs. The structure of a neural network is constructed with interconnected neurons. The neurons are connected with links, which have associated weights. Typically, a neural network is composed of an input layer, several hidden layers, and an output layer. The hidden layers can be more than one. It is typically considered that the more hidden layers a neural network is composed of, the better it is capable of performing complex tasks. Similar findings can be found in neuroscience that when a brain is processing information, it will go through a stacked architecture of hierarchically organized layers. Each layer can contain numerous neurons. The inputs of a neuron in a hidden layer come from all the neurons in the previous layer. The weighted sum of these inputs are calculated and added with a bias term. Then this output value is passed to the next hidden layer after a non-linear activation function. When it has gone through all the hidden layers, the derived output is eventually returned to the output layer.

The application of neural networks as visual object detectors has enabled computers to detect multiple classes of objects with high accuracy, such as faces, pedestrians, and vehicles. However, neural networks usually suffer from a major drawback: they require a large amount of labeled data for supervised learning. In order to solve this problem, [31] proposed a semi-supervised machine learning method to exploit both labeled and unlabeled data to train a classifier in an efficient way.

As is said in [32], CNN is a specific type of artificial neural networks that utilizes convolution instead of vector operation. In the structure of CNN, convolutional layers play a crucial role in extracting and detecting features from images. LeNet is known as the first prototype of CNN, which includes 3 convolutional layers and one fully-connected layer. This network was proposed in the paper [33] for dealing with the handwriting recognition task. Experiment results demonstrated that LeNet outperformed all the other benchmarks back in the day.

2.3.2 Faster R-CNN

Recently, the class of R-CNN approaches have become fairly popular when dealing with object detection problems. One famous example of such type of methods is R-CNN in [34].

![Regions with CNN features](image)

**Figure 2.1: Regions with CNN features [34].**

This paper combined region proposals with a CNN model. As illustrated in Figure 2.1, the authors utilized a selective search algorithm to generate around 2000 bottom-up region proposals. Each region proposal was fed into the CNN model to compute features. After that, they used a Support Vector Machine
(SVM) to classify each proposal. Experiment results have shown that through R-CNN, the mean average precision can reach 53.7% on the test data set Pattern Analysis, Statical Modeling and Computational Learning (PASCAL) Visual Object Classes (VOC) 2010. However, the computational cost is hugely expensive for R-CNNs, as mentioned in this paper [34]. One year later, Ross Girshick employed several innovations for the previous work R-CNN to improve the speed of training and testing. The improved framework was referred to as Fast R-CNN in [35].

In [35], the selective search was still in use for generating bottom-up region proposals. As is shown in Figure 2.2, the input image and multiple regions of interest were fed into a fully convolutional network to generate the feature map for each region of interest. Then fully connected layers was applied to convert the feature map into a feature vector. During this process, the output for each region of interest should contain two elements, which are the class probabilities and the bounding-box regression offsets for each class.

Figure 2.2: The description of the architecture for the Fast R-CNN network [35].

In the same year, Faster R-CNN was introduced in the paper [36]. The running time of the detection network has been reduced by his previous work Fast R-CNN. With this new method, both speed and accuracy were largely improved by the introduction of the Region Proposal Network (RPN). Due to the fact that the RPN and the detection network shared the same convolutional

Figure 2.3: The description of the region proposal network [36]
features, the computational cost for generating region proposals was significantly reduced. For the RPN, the inputs are a collection of full images regardless of their size, and the outputs are a group of anchor boxes, which are a set of rectangular object proposals as described in the paper [36]. As is illustrated in the Figure 2.3, the region proposals are generated when a small network is slid over the convolutional feature map output, which happens in the last shared convolutional layer. The outputs will pass through two fully connected layers. One of the fully connected layers is for the regression of the anchor boxes, and the other one is for the classification of these boxes. The final output of the RPN are the objects’ bounding boxes and the predicted probability scores of being an object.

Compared to the R-CNN and Fast R-CNN, Faster R-CNN employs an RPN instead of the selective search to generate region proposals efficiently. Since the RPN and the detection network share the same convolutional features, the computational cost for generating region proposals is dramatically reduced.

2.3.3 YOLO

The above prior detection systems, such as R-CNN and Fast R-CNN, would first use region proposal methods to generate potential bounding boxes. Then they would run classifiers on each region proposal to generate a probability score of being an object. The regions with high scores will be filtered out and considered as detections.

The algorithm YOLO proposed in the paper [37] performed detection based on a different strategy. Instead of using complex pipelines, YOLO strives to achieve object detection by solving a regression problem. YOLO takes full images as inputs and could derive final region proposals and object classes with predicted probabilities through solely one stage. Therefore, it does not require the sliding window in DPM or the region proposal-based techniques in R-CNNs to separate the background and players first. Since YOLO can access the entire images no matter during training or testing, it is capable of utterly understanding and capturing the contextual information for each class. YOLO is developed based on the GoogLeNet [38]. As is shown in Figure 2.4, YOLO network contains 24 convolutional layers and 2 fully connected layers [37] in total.

Figure 2.4: The architecture of YOLO [37]
As shown in Figure 2.5, an input image is first divided into $S \times S$ grid cells in YOLO. Then bounding boxes are predicted for each grid cell. The paper [37] sets the number of bounding boxes for each grid cell to be $B$ and the number for class probabilities to be $C$. These predictions are eventually encoded and returned as $S \times S \times (B \times 5 + C)$ tensors.

Figure 2.5: YOLO models detection as a regression problem. It divides an input image into $S \times S$ grid cells and for each grid cell predicts $B$ bounding boxes [37].

Each bounding box in outputs contains not only its position, height and width but also the class information and the objectness confidence value. As stated in [37], the objectness confidence value represents the probability of containing an object of any class, and the method for calculating such a value is expressed in equation 2.1. If an object falls into the grid cell, then $Pr(\text{object})$ will be set to 1. Otherwise, it will take 0 as its value. $\text{IOU}^{\text{truth}}_{\text{pred}}$ denotes the IOU value between the predicted bounding box and the ground truth.

\[
\text{Confidence} = Pr(\text{object}) \cdot \text{IOU}^{\text{truth}}_{\text{pred}} \tag{2.1}
\]

Each grid cell also predicts a conditional class probability $Pr(\text{Class}_i|\text{Object})$, which indicates the likelihood of the detected object belongs to $\text{Class}_i$ given that the grid cell contains an object. Then the class-specific confidence score for each bounding box can be calculated based on equation 2.2.

\[
Pr(\text{Class}_i, \text{Object}) \cdot \text{IOU}^{\text{truth}}_{\text{pred}} = Pr(\text{Class}_i|\text{Object}) \cdot Pr(\text{Object}) \cdot \text{IOU}^{\text{truth}}_{\text{pred}} \tag{2.2}
\]

The class-specific confidence score directly reflects the likelihood of a bounding box containing an object from a given class. After obtaining this confidence
score for each bounding box, we can set a threshold to filter these boxes and apply non-max suppression to reduce repetitive detections. The major steps of YOLO can be concluded as firstly resizing input images, secondly feeding images into a single convolutional network, and thirdly thresholding the derived detections, as shown in Figure 2.6.

![Figure 2.6: The main steps of the YOLO][37].

As is mentioned in [37], the localization errors made by YOLO are more comparable to many other cutting-edge detection systems, but the false positives on the background are significantly reduced. Since YOLO utilizes an unified single neural network to perform one-stage detection, it is generally faster than plenty of existing detection approaches.

### 2.3.4 Single Shot MultiBox Detector

Similarly as YOLO, Single Shot MultiBox Detector (SSD) is another one-stage approach for object detection. Two-stage algorithms, such as R-CNN and Fast R-CNN, have to go through a stepwise pipeline to calculate region proposals first and then classify each proposal in the second stage. Compared to these methods, SSD [39] simplifies the pipeline and uses a single deep neural network to detect objects in an image, which completely avoids the proposal generation. The simplified model allows a painless training process for SSD. Different from YOLO, the output space of SSD is discretized into multiple default bounding boxes with different ratios and sizes. The concept of these bounding boxes are comparatively similar to the anchors in R-CNNs. The authors evaluated SSD on various datasets, such as PASCAL VOC, Common Objects in COntext (COCO), and ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Results have shown that compared to the two-stage approaches, SSD has competitive accuracy and faster training speed. In addition, compared to other one-stage algorithms, such as YOLO, SSD delivers a competitive training speed and superior detection accuracy.

### 2.3.5 Transfer Learning

Transfer learning is a machine learning technique to reuse a task-specific pre-trained model and adapt it to a different task. One of the commonly-used ways of transfer learning, as mentioned in [40], is known as fine-tuning. By fine-tuning, weight parameters in a pre-trained model are stored and reused as an initialization of the new network model. Then weights can be fine-tuned by
re-training the network on a different dataset to suit other use cases. As one alternative of fine-tuning, weight parameters in the first several layers may be frozen and only those from the last few layers are allowed to change. Another way is to adjust all the parameters in the pre-trained network based on the new training data.

In this thesis, we implement a CNN-based detection model to tackle the tracking-by-detection of football players. The first 8 layers of our CNN model takes the parameters from Tiny-YOLO-V2 directly, and the last 4 layers are newly added and randomly initialized. The whole network architecture is fine-tuned on a football video dataset provided by the Piero team [5] based on the second alternative as mentioned above.

\section*{2.4 Histogram-based Detectors}

Color is an essential feature in the sports video analytics domain. The playing field in most sports can be characterized by a single dominant color, and players usually wear uniforms in distinguishable colors. In some papers that concentrate on background detection, the dominant color of the background, such as green for the grass on a football pitch, is detected to determine the background. While in the papers that focus on player tracking, players are usually identified by comparing their color distributions with the ground truths pre-stored in a database. During training, the color histograms of the detected players are segmented from the background and stored in the database. In our histogram-based method, we apply the color histogram as the representation of color distributions in an image. The spatial distribution of colors is not considered in this thesis.

\subsection*{2.4.1 Color Space}

The color histogram can be built in different color spaces. Three-dimensional spaces, such as Red-Green-Blue model (RGB) or Hue-Saturation-Value model (HSV), are widely used. A color space can be defined by multiple color axes.

A color histogram can be represented in different color spaces such as RGB, HSV, Hue-Saturation-Intensity (HSI) and LAB color space (LAB). Different color spaces have their own advantages and limitations. Also, it is possible to represent a color distribution in a combined color space for a better performance. The RGB space is a combination of red, green and blue color channels, while HSV stands for hue, saturation, and value.

In the player tracking area, the HSV color space is frequently used. The reason is that the effects of weather, lighting and color variations may much impact the tracking performance, and the HSV color space could better represent these variations, even though a football pitch has one distinct dominant background color. It is a common practice to convert RGB frames into the HSV space to handle these potential variations. In the paper [41], the authors first converted RGB frames into the HSV space, then calculated a color histogram for the hue channel, and located the highest peak of the hue histogram. This series of steps permitted to detect the dominant hue in the background. In [42], the RGB color space was used and the target histogram was derived in the RGB space with \(32 \times 32 \times 32\) bins. In the paper [43], the authors adopted the YUV color
space (YUV) space since they didn’t manage to group some similar colors with minor intensity contrast in the RGB color space. Sometimes we may combine methods in different color spaces. In [44], the authors conducted experiments in different color spaces and concluded that the optimal results came from a combination of color space pairs.

2.4.2 Color Histogram Distribution

The color histogram model can also differ depending on how we build them. Usually, the color histogram is $N$-dimensional. The number $N$ is determined by the measurements taken. Taking HSV as an example, the dimensionality of a HSV histogram is usually jointly distributed. However, in order to decrease the computational complexity, we can use different ways to reduce the dimensionality of a color histogram. In [19], color models were created by extracting 1-D histograms from the H (hue) channel in the HSV space. In [17], the descriptor vectors were constructed by histogramming the pixel intensities into 64 bins for each color channel. By this means, each color channel was treated independently and this feature vector would thus have a dimensionality of 192 ($64 \times 3$) instead of 4096 ($16 \times 16 \times 16$). In [45], a color histogram was constructed in the HSV color space with 16 bins for each color channel. Since the HSV color space decouples the intensity from color, the feature vector can have a dimensionality of 272 ($16 \times 16 + 16$) bins in total. After the color space and the way of construction are determined, each pixel from the selected bounding box can be allocated into different bins based on its color features.

Inevitably, a bounding box may contain a certain background region, and the targeted player region is more likely to appear in the middle of a bounding box. In order to favor the player area over the background, we can assign higher weights to those pixels that approach the central region of a bounding box. In the paper [10], the authors applied a Gaussian weighting function centered in the patch to emphasize the central region.

2.4.3 Metric for Histogram Distance

When it comes to the filtering of region proposals, the histogram distance between a target and a detected bounding box can be calculated as a metric. The manually-drawn bounding boxes during initialization are regarded as the ground truths in this case, and their color histograms are stored in a look-up table as the target models. A one-to-one mapping is guaranteed in this thesis context between the detected bounding box and the target model.

There are several ways to define the distance of color histograms between the detected bounding boxes and the target models. In [46], the Bhattacharyya similarity coefficient was applied to define the distance between HSV and HOG histograms respectively, as in equation 2.3. The Bhattacharyya coefficient measures the overlap level between two statistical samples. The similarity between these two samples can be evaluated based on this metric.

$$BC(p, q) = \sum_{x \in X} \sqrt{p(x)q(x)}$$  \hspace{1cm} (2.3)

Different from [46], the Euclidean distances between the centers of bounding boxes and the predicted locations of players were selected as the matching
scores in the paper [10]. Color histogram intersection, proposed in [47], is another alternative for matching a detected histogram with a model histogram. In this thesis, after testing different metrics of histogram distance, we select the square-root distance as the metric for evaluating the similarity between two color histogram samples.
Chapter 3

Methods

In order to respond to the research questions stated above, we develop three systems for the players tracking task on video sequences from football games. The first system is a sequence-adaptive color histogram-based tracking system, which is capable of capturing the color distribution of players’ uniforms in a quick fashion. The second system is a CNN-based system. We develop a CNN-based detector with its architecture and weights optimized for the player tracking task. The third system is a combined system which fuses a CNN-based system with a histogram-based inter-frame-connection post processor.

3.1 Dataset

3.1.1 Training Sequences

One crucial step before diving into the methods is data collection and preprocessing. We collected all the needed data for training and evaluation of the proposed systems. Overall, 38 football video sequences are provided by Piero beforehand and used for training in this thesis. These video sequences contain in total 9223 frames and 115921 bounding boxes. Each bounding box is supposed to include a football player in it.

<table>
<thead>
<tr>
<th>Table 3.1: Training dataset information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Labeled video frames</td>
</tr>
<tr>
<td>Labeled players</td>
</tr>
<tr>
<td>Video formats</td>
</tr>
</tbody>
</table>

The video sequences are selected carefully, considering the potential variety of properties that may have an influence on the tracking performance. Thus, 38 video sequences with different quality (high image quality and low image quality), illumination (dark and light), camera angle (upper and horizontal), and aspect ratio (16:9 and 4:3) are applied in our experiments. The variety of the team uniform colors is considered as well when choosing the samples. Details of the dataset can be seen in Table 3.1.
In each frame-set, we manually draw bounding boxes over players in a frame-by-frame manner. The goal here is to create a parameter matrix for each player and parameterize it by the bounding box’s upper-left corner coordinate \((x, y)\), width \(w\), and height \(h\). By this means, each player can be represented in the form of a matrix \((x, y, w, h)\). The format of the labeled data is shown in Figure 3.1. The process of data labeling is illustrated in Figure 3.2. As we can see from Figure 3.2, the inputs are continuous frames sampled from football video sequences, and the annotated players serve as the output of this process. Examples of positive and negative samples are shown in 3.3.

![Figure 3.1: Labeled data format](image)

![Figure 3.2: Process of data labeling step](image)
The details of the training dataset are shown in Table 3.2. Here, BBs Number refers to the number of bounding boxes. AvgW means the average width of all the bounding boxes, and AvgH represents the average height of all the bounding boxes.

3.1.2 Testing Sequences

For the evaluation process, we prepare and use two different sequences in this thesis, which vary in certain characteristics. Frame examples from these two sequences are shown here as in Figure 3.4 and 3.5. As is shown in Figure 3.4, the first testing sequence has a brighter background than the second one in Figure 3.5. The size of players in the first sequence is generally larger than that in the second sequence. We can also notice that the first sequence has a much better image resolution than the second one.

In this thesis, the proposed three systems are evaluated against benchmarks, such as YOLO and Faster R-CNN, on these two testing sequences. The obtained results should indicate how well the proposed systems generalize on different football videos.
<table>
<thead>
<tr>
<th>Sequence No</th>
<th>Frame number</th>
<th>BBs Number</th>
<th>AvgW</th>
<th>AvgH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>355</td>
<td>4426</td>
<td>73(0.04)</td>
<td>139(0.13)</td>
</tr>
<tr>
<td>2</td>
<td>245</td>
<td>3737</td>
<td>40(0.03)</td>
<td>96(0.13)</td>
</tr>
<tr>
<td>3</td>
<td>331</td>
<td>5678</td>
<td>49(0.04)</td>
<td>94(0.13)</td>
</tr>
<tr>
<td>4</td>
<td>323</td>
<td>4036</td>
<td>45(0.03)</td>
<td>82(0.11)</td>
</tr>
<tr>
<td>5</td>
<td>330</td>
<td>4704</td>
<td>48(0.04)</td>
<td>100(0.14)</td>
</tr>
<tr>
<td>6</td>
<td>439</td>
<td>5927</td>
<td>40(0.03)</td>
<td>85(0.12)</td>
</tr>
<tr>
<td>7</td>
<td>227</td>
<td>4037</td>
<td>35(0.03)</td>
<td>58(0.08)</td>
</tr>
<tr>
<td>8</td>
<td>396</td>
<td>4108</td>
<td>56(0.04)</td>
<td>99(0.14)</td>
</tr>
<tr>
<td>9</td>
<td>351</td>
<td>4288</td>
<td>47(0.04)</td>
<td>79(0.11)</td>
</tr>
<tr>
<td>10</td>
<td>477</td>
<td>3824</td>
<td>63(0.05)</td>
<td>118(0.16)</td>
</tr>
<tr>
<td>11</td>
<td>341</td>
<td>5472</td>
<td>45(0.03)</td>
<td>82(0.11)</td>
</tr>
<tr>
<td>12</td>
<td>405</td>
<td>4499</td>
<td>45(0.03)</td>
<td>75(0.10)</td>
</tr>
<tr>
<td>13</td>
<td>500</td>
<td>5829</td>
<td>31(0.03)</td>
<td>79(0.11)</td>
</tr>
<tr>
<td>14</td>
<td>252</td>
<td>4334</td>
<td>36(0.03)</td>
<td>71(0.10)</td>
</tr>
<tr>
<td>15</td>
<td>129</td>
<td>1949</td>
<td>37(0.03)</td>
<td>71(0.10)</td>
</tr>
<tr>
<td>16</td>
<td>262</td>
<td>4139</td>
<td>38(0.03)</td>
<td>85(0.12)</td>
</tr>
<tr>
<td>17</td>
<td>60</td>
<td>780</td>
<td>52(0.04)</td>
<td>89(0.12)</td>
</tr>
<tr>
<td>18</td>
<td>136</td>
<td>1729</td>
<td>43(0.03)</td>
<td>81(0.11)</td>
</tr>
<tr>
<td>19</td>
<td>269</td>
<td>3607</td>
<td>48(0.04)</td>
<td>95(0.13)</td>
</tr>
<tr>
<td>20</td>
<td>126</td>
<td>1174</td>
<td>44(0.03)</td>
<td>88(0.12)</td>
</tr>
<tr>
<td>21</td>
<td>83</td>
<td>909</td>
<td>47(0.04)</td>
<td>87(0.12)</td>
</tr>
<tr>
<td>22</td>
<td>104</td>
<td>815</td>
<td>59(0.05)</td>
<td>99(0.14)</td>
</tr>
<tr>
<td>23</td>
<td>169</td>
<td>1192</td>
<td>59(0.05)</td>
<td>90(0.12)</td>
</tr>
<tr>
<td>24</td>
<td>151</td>
<td>1376</td>
<td>48(0.04)</td>
<td>96(0.13)</td>
</tr>
<tr>
<td>25</td>
<td>135</td>
<td>1266</td>
<td>57(0.04)</td>
<td>102(0.14)</td>
</tr>
<tr>
<td>26</td>
<td>89</td>
<td>1590</td>
<td>40(0.03)</td>
<td>66(0.09)</td>
</tr>
<tr>
<td>27</td>
<td>123</td>
<td>1546</td>
<td>55(0.04)</td>
<td>101(0.14)</td>
</tr>
<tr>
<td>28</td>
<td>275</td>
<td>3566</td>
<td>49(0.04)</td>
<td>85(0.12)</td>
</tr>
<tr>
<td>29</td>
<td>105</td>
<td>1440</td>
<td>48(0.04)</td>
<td>82(0.11)</td>
</tr>
<tr>
<td>30</td>
<td>49</td>
<td>452</td>
<td>63(0.05)</td>
<td>110(0.15)</td>
</tr>
<tr>
<td>31</td>
<td>247</td>
<td>2535</td>
<td>60(0.05)</td>
<td>100(0.14)</td>
</tr>
<tr>
<td>32</td>
<td>67</td>
<td>236</td>
<td>48(0.04)</td>
<td>86(0.12)</td>
</tr>
<tr>
<td>33</td>
<td>131</td>
<td>1514</td>
<td>49(0.04)</td>
<td>80(0.11)</td>
</tr>
<tr>
<td>34</td>
<td>251</td>
<td>2789</td>
<td>40(0.03)</td>
<td>89(0.12)</td>
</tr>
<tr>
<td>35</td>
<td>73</td>
<td>1095</td>
<td>47(0.04)</td>
<td>70(0.10)</td>
</tr>
<tr>
<td>36</td>
<td>175</td>
<td>1687</td>
<td>53(0.04)</td>
<td>92(0.13)</td>
</tr>
<tr>
<td>37</td>
<td>365</td>
<td>3384</td>
<td>51(0.04)</td>
<td>91(0.13)</td>
</tr>
<tr>
<td>38</td>
<td>677</td>
<td>10252</td>
<td>39(0.03)</td>
<td>70(0.10)</td>
</tr>
</tbody>
</table>
3.2 Color Histogram-based System

The color histogram-based system aims to capture the color distribution of a player region in a particular football game scenario. The color histogram of each player is extracted from its corresponding bounding box’s region. Each bounding box has already been parameterized as $bb = (x, y, w, h)$ and normalized by the width and height of the image during the data labeling step. The histogram of each bounding box in the training dataset is computed and stored.
in a player histogram matrix $H_{PL}$, which is involved in an improvement step for robust player tracking.

### 3.2.1 Color Histogram Distribution

To calculating the color histogram, we select the RGB feature of images to be the detection feature. In order to decrease the computational complexity, efforts are made to reduce the dimension of the color histogram representation. Instead of using a jointly distributed color space, we create a color model by merging the RGB histograms into a single vector and flattening it as $h = [h_R, h_G, h_B]$. By this means, we divide each color channel into 16 bins and build the RGB histogram into a one-dimensional flat vector, which contains in total 48 bins.

### 3.2.2 Evaluation of Similarity

The histogram-based system relies on the histogram similarity of detections between consecutive frames to track players. Initially, the system is provided with a set of manually-drawn bounding boxes in the first frame. Then for the next frame, each bounding box will shift around the previous position within a region and calculate the histogram similarity between the newly-shifted bounding box and the previously-stored one. The one with the highest similarity will be chosen as the new position of the bounding box. This process can then be iterated and player tracking across frames can therefore be achieved.

There exist different methods to evaluate the histogram similarity between predictions in the current frame and the target in the previous frame. In this sections, we mainly focus on two methods for this purpose, namely the histogram intersection and the square root of histogram difference.

#### Histogram Intersection

The paper [47] proposed a technique called Histogram Intersection. As stated in this paper, the histogram has its own advantage in dealing with real-time indexing problems with the sizeable pre-stored database. The histogram intersection is considered robust since accurate separations of objects from the background are not required by this method.

Figure 3.6 and 3.7 demonstrate two examples of how histograms may be intersected between background and player, and between different players. The red bars in Figure 3.6 represent the histogram of a football player, while the blue bars represent that of a background region. The intersection between the two histogram bars indicates the similarity between these two regions. Similarly, Figure 3.7 shows the histogram intersection between two different players. If the two histogram bars are similar as in Figure 3.7, then the intersection area should be larger since their histogram distributions are much more comparable. On the contrary, if the two histogram bars differentiate much, the intersection area will be considerably small as in Figure 3.6.
One notable difference between the player’s histogram and the background’s, as shown in the above figures, is the variance of bins. The background usually has a lower variance over different bins in the histogram (mostly lower than 0.15). However, the player’s histogram usually has a larger variance over color bins than the background. Based on this observation, players and the background may also be separated by comparing the variance of their histogram bins. This idea is not covered in this thesis scope and can be experimented as a future work.
Square Root of Histogram Difference

Another way to express the histogram similarity is to calculate the square root of the histogram difference, or in other words the distance between two histograms, as shown in equation 3.1. \( H_{\text{current}}\{p,t\} \) represents the histogram of a derived bounding box \( p \) in the current frame \( t \), while \( H_{\text{previous}}\{q,t-1\} \) refers to that of a bounding box \( q \) in the previous frame \( t-1 \).

\[
D_{\text{previous}}\{p,q,t\} = \sqrt{(H_{\text{current}}\{p,t\} - H_{\text{previous}}\{q,t-1\})^2} \quad (3.1)
\]

In order to improve the robustness of the histogram-based system, we also calculate the distance \( D_{\text{stored}} \) between the histogram of currently derived bounding boxes and that of the stored player annotations \( H_{PL} \) from the training data, as shown in equation 3.2. A weight parameter \( \alpha \) is applied here to take both \( D_{\text{previous}} \) and \( D_{\text{stored}} \) into consideration when deciding the final detection. As is shown in equation 3.3, the larger value the parameter \( \alpha \) takes, the higher weight is assigned to the previous histogram distance than the stored histogram distance.

\[
D_{\text{stored}}\{i,p,t\} = \sqrt{(H_{\text{current}}\{p,t\} - H_{PL}\{i\})^2} \quad (3.2)
\]

\[
D_{\text{adjust}} = (1 - \alpha) * D_{\text{stored}} + \alpha * D_{\text{previous}} \quad (3.3)
\]

Here \( D_{\text{adjust}} \) is the adjusted distance. \( D_{\text{stored}} \) represents the least histogram distance between the current bounding boxes and the stored ones, and \( D_{\text{previous}} \) means the least histogram distance between bounding boxes in the current frame and those in the last frame.

In this thesis, we choose the above histogram distance over histogram intersection as our similarity evaluation metric due to the relative simplicity of its expression and the robustness.

3.3 CNN-based System

3.3.1 Network Architecture of CNN

Another proposed system in this thesis is the CNN-based system, in which a CNN architecture is designed and fine-tuned with the training video sequences. The network architecture of the CNN-based system is shown in Table 3.3.

As we can see from Table 3.3, the CNN network is designed to contain 8 convolutional layers and 4 max-pooling layers for football player detection. The depth of the network is optimized to achieve a desired detection accuracy. Figure 3.8 briefly illustrates the architecture of the CNN-based system.
Table 3.3: CNN-based system network architecture

<table>
<thead>
<tr>
<th>#</th>
<th>Layer</th>
<th>Filters</th>
<th>Size/Stride</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>conv</td>
<td>16</td>
<td>3 x 3 / 1</td>
<td>640 x 368 x 3</td>
<td>640 x 368 x 16</td>
</tr>
<tr>
<td>1</td>
<td>max</td>
<td>2 x 2 / 2</td>
<td></td>
<td>640 x 368 x 16</td>
<td>320 x 184 x 16</td>
</tr>
<tr>
<td>2</td>
<td>conv</td>
<td>32</td>
<td>3 x 3 / 1</td>
<td>320 x 184 x 16</td>
<td>320 x 184 x 32</td>
</tr>
<tr>
<td>3</td>
<td>max</td>
<td>2 x 2 / 2</td>
<td></td>
<td>320 x 184 x 32</td>
<td>160 x 92 x 32</td>
</tr>
<tr>
<td>4</td>
<td>conv</td>
<td>64</td>
<td>3 x 3 / 1</td>
<td>160 x 92 x 32</td>
<td>160 x 92 x 64</td>
</tr>
<tr>
<td>5</td>
<td>max</td>
<td>2 x 2 / 2</td>
<td></td>
<td>160 x 92 x 64</td>
<td>80 x 46 x 64</td>
</tr>
<tr>
<td>6</td>
<td>conv</td>
<td>128</td>
<td>3 x 3 / 1</td>
<td>80 x 46 x 64</td>
<td>80 x 46 x 128</td>
</tr>
<tr>
<td>7</td>
<td>max</td>
<td>2 x 2 / 2</td>
<td></td>
<td>80 x 46 x 128</td>
<td>40 x 23 x 128</td>
</tr>
<tr>
<td>8</td>
<td>conv</td>
<td>256</td>
<td>3 x 3 / 1</td>
<td>40 x 23 x 128</td>
<td>40 x 23 x 256</td>
</tr>
<tr>
<td>9</td>
<td>conv</td>
<td>512</td>
<td>3 x 3 / 1</td>
<td>40 x 23 x 256</td>
<td>40 x 23 x 512</td>
</tr>
<tr>
<td>10</td>
<td>conv</td>
<td>512</td>
<td>3 x 3 / 1</td>
<td>40 x 23 x 512</td>
<td>40 x 23 x 512</td>
</tr>
<tr>
<td>11</td>
<td>conv</td>
<td>5</td>
<td>1 x 1 / 1</td>
<td>40 x 23 x 512</td>
<td>40 x 23 x 5</td>
</tr>
</tbody>
</table>

Figure 3.8: Illustration of network architecture of CNN-based system.
3.3.2 Training

Transfer learning is used for training our CNN-based system. The first 8 layers of our CNN-based system in Table 3.3 are directly initialized with weight parameters from the pre-trained Tiny-Yolo-V2 model. For the last four layers, weights are randomly initialized. During training, weight parameters from all the layers are fine-tuned and adjusted based on our football training data.

The training of the CNN-based system is achieved using a Graphics Processing Unit (GPU). The memory requirement for the training is 4 Gigabytes.

3.4 Combined System

In order to use the CNN-based system while also taking advantage of players’ visual appearance information, we combine the above two systems. This combined system CNN-Inter Frame Connection is related to the algorithms presented in [30] and [48], which combine a CNN-based detector with a tracklet handling post-processor [49]. To improve the readability, “combined system” or “CNN-IFC” is used to refer to this system in the rest of the presentation. This system is proposed and developed as a potential way to boost the performance of the CNN-based system and the histogram-based system. The histogram-based sequence-adaptive algorithm is updated and applied behind the CNN-based system as a post-processing module.

The two main parts in this combined system are the CNN detector and the histogram-based post-processing module. The CNN detector simply shares the same CNN architecture as in Figure 3.8, and it has been well covered in Section 3.3. Therefore, this section is started with an introduction of the histogram-based post-processing module, and then follows a walk-through of the workflow of the combined system.

3.4.1 Histogram-based Post Processor

There are three major steps conducted by the histogram-based post processor, namely data association, adaptation of probability, and tracklet ID handling. Note that these steps are performed right after the CNN detector outputs initial region proposals of players. The common practice used in CNN that proposals are filtered with a confidence threshold is delayed to the third step.

In order to associate detections in the previous frame $\Phi^{n-1}$ with the proposals in the current frame, the data association step is applied to calculate a score matrix $D$ between detection pairs. It measures the distance between a detected bounding box in frame $n-1$ and that in frame $n$. The distance here is defined as $D_{ij} = 0.25 \|X_i - X_j\| + 0.50 \|H_i - H_j\| + 0.25 |P_i - P_j|$. $X = [x, y, w, h]$ is the parameterized vector of a bounding box, and $H$ represents the color histogram of a bounding box. $P$ indicates the inferred probability that a bounding box contains a player. A higher weight is specifically assigned to the color histogram term to reflect the fact that color information plays a crucial role in player tracking in football videos. The pseudo code of the data association step is attached as below. $\Psi^n$ represents the proposals in the current frame.
Algorithm 1: Data Association

Data: $\Phi^{n-1}, \Psi^n$
Result: Accepted mappings set $\Omega$

begin
  // Initialization
  Initialize connected pairs set $\Upsilon$ and accepted mappings set $\Omega$;
  for $i \in \Phi^{n-1}$ do
    for $j \in \Psi^n$ do
      Calculate the matching score $D_{ij}$ and store it in a score matrix $D$.
    end
  end
  while $\Phi^{n-1} \neq \emptyset$ and $\Psi^n \neq \emptyset$ do
    Find the pair $(i_{\text{min}}, j_{\text{min}})$ with the minimum score in $D$.
    Append the selected pair into $\Upsilon$ as $\Upsilon = \Upsilon \cup (i_{\text{min}}, j_{\text{min}})$.
    Remove $i_{\text{min}}$ and $j_{\text{min}}$ from $\Phi^{n-1}$ and $\Psi^n$ respectively.
  end
  for $(i, j) \in \Upsilon$ do
    if $(|x_i - x_j| + |y_i - y_j|) \leq \frac{1}{2} \left( \frac{\omega_i + \omega_j}{2} + \frac{h_i + h_j}{2} \right)$ then
      Append pair $(i, j)$ into the final accepted mappings set $\Omega$.
    end
  end
end

The adaptation of probability aims to compensate the probability $P$ when an unexpected probability drop takes place. As is demonstrated in the pseudo code below, a threshold $\Theta_{\text{min}}$ is determined based on the color histogram distance between detected bounding boxes in two consecutive frames. If $\|H_i - H_j\|$ is smaller than 0.1, it means the color histogram distributions of these bounding boxes are considered similar with a high confidence. We should then set $\Theta_{\text{min}}$ aggressively to 0.2 so that the probability adaptation can be triggered easily. If $\|H_i - H_j\|$ is larger than 0.1, we should set $\Theta_{\text{min}}$ conservatively to 0.4 to raise the bar for probability adaptation in such case. After $\Theta_{\text{min}}$ is determined, $P_j^n$ is updated based on the equation $P_j^n = (1 - \alpha) P_j^n + \alpha P_j^{n-1}$. $\alpha$ is selected as 0.98 in our implementation. This step is designed to mitigate unexpected probability drops for region proposals in the current frame, and it is expected to improve the system performance in terms of a better recall rate.
Algorithm 2: Probability Adaptation

Data: Accepted mappings set $\Omega$, $\Phi^{n-1}$, $\Phi^n$

Result: Mappings set with adjusted probabilities

begin
  // Probability Adaptation
  for $(i,j) \in \Omega^{n,n-1}$ do
    Determine $\Theta_{\text{min}}$ based on the histogram distance $\|H_i - H_j\|$.
    if $(P^{n-1}_i > P^n_j)$ and $(P^n_j > \Theta_{\text{min}})$ then
      Update $P^n_j$: $P^n_j = (1 - \alpha) P^n_j + \alpha P^{n-1}_i$.
    end
  end
end

The third step mainly deals with ID handling. The ID information will be inherited from the previous frame $n-1$ if a current proposal has a matching pair in the previous frame. Otherwise, a new ID will be initiated for this proposal. Besides, if a player is detected in frame $n-1$ but not detected in frame $n$, the ID information of this player will be released. The final detection threshold $\Theta = 0.5$ is applied in this step as well.

The three steps could be briefly illustrated in Figures 3.9, 3.10 and 3.11.

Algorithm 3: Object Acceptance and Tracklet ID Handling

Data: Accepted mappings set $\Omega$, $\Psi^n$

Result: Accepted detections $\Phi^n$ with ID information

begin
  // Object Acceptance and Tracklet ID Handling
  Initialize $\Phi^n = \{\}$, probability threshold $\Theta = 0.5$.
  for $j \in \Psi^n$ do
    if $P^n_j \geq \Theta$ then
      if $(i,j) \in \Omega^{n,n-1}$ then
        Update $\Phi^n$ with ID transferred from previous frame:
        $\Phi^n = \Phi^n \cup \{(X^n_j, H^n_j, P^n_j, ID^n_i)\}$.
      else
        Update $\Phi^n$ with new ID incremented from $ID_{\text{max}}$:
        $\Phi^n = \Phi^n \cup \{(X^n_j, H^n_j, P^n_j, ID_{\text{max}} + 1)\}$.
      end
    end
  end
end
3.4.2 Workflow

The workflow of the combined system can be summarized from a black-box model perspective. Consecutive frames with resolution $640 \times 368$ in RGB are fed into the combined system as inputs, in which the CNN model generates accordingly $40 \times 23$ region proposals. These proposals would be further filtered by a non-maximum suppression [50] algorithm. $J_{NMS}$ is used here to denote the number of region proposals after the non-maximum suppression algorithm.
After getting the initial proposals from the CNN, the combined system will go through the three steps as mentioned above and generate final detections. The obtained detections in the current frame will be determined not only based on the available region proposals $\Psi_n$ in the current frame but also the relevant detections $\Phi_{n-1}$ from the previous frame. The whole process is illustrated in Figure 3.12.

Figure 3.12: The workflow of the combined system

The process can also be briefly outlined in the pseudo code below as in Algorithm 4. The combined system takes frame sequences sampled from the football video clips as inputs and returns a set of detected proposals in each frame as outputs. The input data is firstly processed by a fine-tuned CNN model, and then it will be processed by the histogram-based post-processor.
Algorithm 4: Customized CNN with Post Processing Algorithm

Data: Frame $n$ in video sequence, detected players $\Phi^{n-1}$ in frame $n - 1$

Result: Detected players $\Phi^n$ in frame $n$

begin
    // Initialization
    1 Initialize: $\Omega = \{\}, \Phi^n = \{\}, \Psi^n = \{\}$.

    // Apply customized CNN
    3 Apply customized CNN on the input frames and generate bounding
         boxes in frame $n$: $\Psi^n = \{(X^n_j, P^n_j)\}_{j=1}^N$.

    // Assemble Histogram metrics
    4 for $k$ in $N$ iterations do
        5 Calculate the histogram $H^n_k$ for the $k$th proposal $\Psi^n_k = (X^n_k, P^n_k)$.
        6 Add the histogram metric $H^n_k$ into $\Psi^n_k$: $\Psi^n_k = (X^n_k, H^n_k, P^n_k)$.
    end

    // Data Association
    8 Associate bounding boxes in frame $n - 1$ with those in frame $n$, as
       mentioned in algorithm 1.

    // Probability Adaptation
    9 Perform the probability adaptation step as mentioned in algorithm 2.

    // Object Acceptance and Tracklet ID Handling
   10 Perform the Object Acceptance and Tracklet ID Handling step as
        mentioned in algorithm 3.
end

3.5 Evaluation

After training, we use the two testing sequences to evaluate the performance of the proposed systems. In order to collect supporting facts for answering the research questions in Section 1.3, three groups of comparisons are planned during evaluation as below:

- For research question 1, the histogram-based system is evaluated against the CNN-based system. The comparative results indicate the performance differences between player tracking systems with different mechanisms.

- For research question 2, the CNN-based system is compared with several off-the-shelf systems, such as YOLO, Tiny-YOLO-V2, and Faster R-CNN. Such comparisons reflect how well the CNN-based system performs versus state-of-the-art methods in this domain.

- For research question 3, the combined system is evaluated against the pure CNN-based system and the CNN-SORT system. This comparison could reveal the impacts of adding a histogram-based post processor for football player tracking.
3.5.1 Evaluation Metrics

In order to assess the performances of the proposed systems in a quantitative way, the following evaluation metrics are measured during experiments:

Precision evaluates the fraction of the true positive detected bounding boxes amongst the retrieved predictions, or in other words the ratio of the detected positive players to all the detected players. It is calculated as equation 3.4.

\[
\text{Precision} = \frac{\text{True positive predictions}}{\text{All predictions}} \quad (3.4)
\]

Another important metric in this thesis is referred to as recall, which represents the ratio of the positive detections to all the ground truth detections. It indicates how much proportion of the ground truths can be detected as positive. The calculation of recall is shown in equation 3.5.

\[
\text{Recall} = \frac{\text{True positive predictions}}{\text{Ground truths}} \quad (3.5)
\]

In this thesis, the correctness of detection is defined by the area of intersection between a prediction and a ground truth. The parameter IOU is used to represent this metric, which can be calculated as equation 3.6:

\[
\text{IOU} = \frac{\text{area}(B_p \cap B_t)}{\text{area}(B_p \cup B_t)} \quad (3.6)
\]

Here \( B_p \) denotes the predicted bounding box and \( B_t \) denotes the ground truth. In the histogram-based system, only those detected bounding boxes that have a larger IOU value than 0.6 are regarded as positive detections. One ground truth can then only be assigned to one detected box according to the maximum IOU. If more than one predictions are assigned to the same ground truth, only one of the bounding boxes is taken as a valid detection. Other detected boxes are manually reset to have zero IOU and will not be considered when calculating the average IOU of all the detections. On the other hand, one ground truth can also be only assigned to one detected box according to the maximum IOU. The system will be penalized in the same manner if the rule above is conflicted.

F1 score is selected when comparing the combined system with benchmarks. The metric F1 is calculated based on Equation 3.7. It conveys the harmonic mean of precision and recall.

\[
\text{F1} = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \quad (3.7)
\]

Another evaluation metric is Identity Tracking Performance (ITP) [51]. It calculates the ratio of the number of correct ID to the number of ground truths [51] as shown in equation 3.8.

\[
\text{ITP} = \frac{\text{#Correct ID}}{\text{#Ground truth}} \quad (3.8)
\]
Chapter 4

Experiment Results

4.1 Histogram-based System

4.1.1 Positive Detections

Positive detection results of the histogram-based system are shown in Figure 4.1. The two pictures on the first row are from the high-quality video sequence, while the second row comes from the low-quality one. Results on the high-quality testing sequence have shown that the system is capable of detecting a player or part of a player on the edge of a video frame. It demonstrates a decent tracking performance, especially when players have contrasting color distributions compared to the background. On the other hand, the system has proven to be well tolerant of poor image quality and dark background based on the results on the low-quality testing sequence.

Figure 4.1: Positive detections made by the histogram-based system.
4.1.2 Negative Detections

In spite of the positive examples above, there also exist some negative detections, as shown in Figure 4.2. At least one negative detection can be observed in each of the four subfigures. In these negative examples, the system fails to overlay players with bounding boxes correctly. Instead, the background is misidentified as a player. In the top two pictures, negative detections appear when two players stay a fairly close distance to each other. With the players moving continuously, one of the bounding boxes covers the two players, and the other one moves away from the player’s region. Since negative detections usually happen when players are moving fast, we could associate the failure of player tracking with the motion blur in football videos. The reason may be that the color histogram of a motion blur region approaches that of a background region better than a player region. Therefore, such false-positive detections are naturally more likely to happen in this case.

![Image of negative detections](image)

Figure 4.2: Negative detections made by the histogram-based system.

4.2 CNN-based System

We evaluate the CNN-based system by injecting the testing sequences into the fine-tuned system. After thresholding the initial region proposals, we can get the selected proposal vectors, each of which contains the information of its position, size and the matching probabilities for a particular class. The precision-recall curves can be obtained based on these filtered proposal vectors and the ground truth at different thresholds, as in Figure 4.3 to 4.5. A trade-off relationship between the precision and recall rate can be clearly observed from the curves. The threshold in each figure demonstrates above which confidence level a proposal is regarded as a ground truth. Only those proposals with higher probabilities than the threshold will be selected as final predictions. When the threshold is
increased from 0.1 to 0.9, the precision rate rises simultaneously from nearly 0 to approximately 0.9, while the recall rate of the predictions decays from 1 to around 0.1. The larger threshold we select, the fewer bounding boxes will remain. Based on the precision-recall figure, a proper threshold can be selected so that both the precision and recall rates stay in decent values.

![Precision & recall at different threshold levels](image.png)

**Figure 4.3**: Precision and recall rates at different threshold levels for the high-quality testing sequence 1.
4.2.1 Comparison between CNN-based System and Histogram-based System

For the proposed CNN-based system and histogram-based system, experiments are conducted to compare their performances in terms of average IOU and precision rate. Based on the same training and testing sequences, measures of these metrics in the two systems are listed in the below tables.

**Table 4.1:** Average Intersection over Union achieved by CNN-based system and histogram-based system on two testing sequences.

<table>
<thead>
<tr>
<th></th>
<th>Testing sequence 1</th>
<th>Testing sequence 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN-based system</td>
<td>0.87</td>
<td>0.85</td>
</tr>
<tr>
<td>Histogram-based system</td>
<td>0.76</td>
<td>0.73</td>
</tr>
</tbody>
</table>

As is shown in Table 4.1, the CNN-based system clearly outperforms the histogram-based system with a higher average IOU for both testing sequences.
As a complement of the above measures, the precision-recall curves of the two systems are demonstrated in Figure 4.5 as well. Since there is no thresholding in the histogram-based system, only one precision-recall sample is plotted for each testing sequence on the figure. Also, the CNN-based system outperforms the histogram-based system in terms of both precision and recall rate, regardless of what threshold is selected. As illustrated in Figure 4.5, a superior precision rate is achieved by the CNN-based system on both testing sequences, which proves that the CNN-based system delivers a more accurate detecting performance. The detections provided by the CNN-based system should contain more true positive region proposals than the histogram-based system.

4.2.2 Comparison between CNN-based System and Off-the-shelf Systems

The performance of CNN-based system is also compared with that of three different off-the-shelf systems, namely Faster R-CNN, YOLO, and Tiny-YOLO-V2. Same testing sequences are fed into the off-the-shelf systems and our CNN-based system to generate final player detections and measures of the evaluation metrics. Figure 4.6, 4.7, and 4.8 demonstrate respectively the detection results from Faster YOLO, Tiny-YOLO-V2, and Faster R-CNN alongside that from the proposed CNN-based system.

As is shown in Figure 4.6, our CNN-based system manages to recognize more players than the YOLO system does. Only two players are detected in this frame by YOLO in the left subfigure, while all the players are recognized.
correctly by our CNN-based system in the right subfigure.

Figure 4.6: Detection results through YOLO system with pre-trained model and our in-house CNN-based sports analytic system.

As for Tiny-YOLO-V2, it makes plenty of false positive predictions in Figure 4.7. For examples, some audiences and background regions are incorrectly detected as players. Besides, detection of the referee and certain players are missing in the left subfigure.

Figure 4.7: Detection results through Tiny-YOLO-V2 system with pre-trained model and our in-house CNN sports analytic system.
Figure 4.8: Detection results through Faster R-CNN system with pre-trained model and our in-house CNN sports analytic system.

Faster R-CNN usually employs 9 anchor boxes, with 3 different sizes and 3 width-height ratios, to generate region proposals. It is proved to normally show a superior performance than YOLO when detecting smaller objects. As is shown in Figure 4.8, the Faster R-CNN detector fails in recognizing all the players, but it generates no false positive detections. The CNN-based system manages to recognize more players than the Faster R-CNN detector since players usually make unique human postures during a football game compared to pedestrians. Training a system solely with regular human data resources is not adequate for the detection of players in the football analytics field.

Evaluation of Detection Performance in terms of mAP

Measures of the evaluation metric mAP are collected here to benchmark the detection performance of the CNN-based system against Faster R-CNN with VGG16 on 2 testing sequences of different image resolutions. YOLO and Tiny-YOLO-V2 are not compared since they have shown to deliver significantly worse performances than the Faster R-CNN system as mentioned above.

Table 4.2: Evaluation of detection performance in terms of mAP.

<table>
<thead>
<tr>
<th>System</th>
<th>Testing sequence 1</th>
<th>Testing sequence 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Faster R-CNN</td>
<td>0.60</td>
<td>0.19</td>
</tr>
<tr>
<td>CNN-based system</td>
<td><strong>0.86</strong></td>
<td><strong>0.85</strong></td>
</tr>
</tbody>
</table>

As table 4.2 indicates, Faster R-CNN falls short in the mAP measures in both testing sequences since it is not fine-tuned for the football players detection scenario, even though Faster R-CNN advances in the object detection domain. The result clearly demonstrates a superior detection performance of the CNN-based system over one of the best benchmarks. As the results reveal, the largest performance difference happens at the low resolution sequence. The reason for such a significant performance delta is that the Faster R-CNN is typically trained on common datasets which contain very limited blurs. As testing sequences 2 is a low image resolution sequence with a large amount of blurs, it is poorly handled.
by the Faster R-CNN system but well managed by the fine-tuned CNN-based system.

4.3 Combined System

4.3.1 Evaluation of Tracking Performance

In order to evaluate the tracking performance of the combined system, a new CNN system CNN-SORT is created as the benchmark. SORT is a simple realtime tracking algorithm proposed in [52], which relies on the Kalman filter framework as its tracking strategy. The CNN-SORT system shares the same core CNN detector as the combined system but utilizes a different strategy to track players across consecutive frames. Therefore, the detection module should return the same predictions, and only the tracking module contributes to the performance difference between these two systems.

Table 4.3: F1 averaged over consecutive video frames.

<table>
<thead>
<tr>
<th>System</th>
<th>Testing sequence 1</th>
<th>Testing sequence 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN-based system</td>
<td>0.85</td>
<td>0.81</td>
</tr>
<tr>
<td>CNN-SORT system</td>
<td>0.93</td>
<td>0.89</td>
</tr>
<tr>
<td>CNN-IFC system</td>
<td><strong>0.94</strong></td>
<td><strong>0.92</strong></td>
</tr>
</tbody>
</table>

The results in table 4.3 indicate that the combined system CNN-IFC has a more stable tracking performance than CNN-SORT in terms of F1 score. It also reveals that adding a post-processing module for tracking contributes to a performance boost compared to the CNN-based system.

Table 4.4: Average ITP per system and testing sequence.

<table>
<thead>
<tr>
<th>System</th>
<th>Testing sequence 1</th>
<th>Testing sequence 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN-SORT</td>
<td>0.99</td>
<td>0.97</td>
</tr>
<tr>
<td>CNN-IFC</td>
<td><strong>0.99</strong></td>
<td><strong>0.98</strong></td>
</tr>
</tbody>
</table>

As is shown in table 4.4, the combined system CNN-IFC has a slightly better performance than CNN-SORT, which reveals that the color distribution information of players contributes marginally to a more stable ID tracking in the combined system. Since the CNN-based system (without tracklet) does not output and track player ID information, it is naturally excluded from this comparison.
Chapter 5

Discussion and Conclusions

5.1 Limitations

The acquired results during experiments are limited to some extent and may much vary depending on the experiment settings. In this section, potential limitations and impact factors are discussed as follows.

5.1.1 Dataset

As the major input of model training, the dataset may impact the performance of the proposed systems. Important aspects of the dataset, such as quantity, quality, and variety, are briefly covered in this sub-section.

First of all, the quantity of the training data plays an essential role in influencing the systems’ performances. Training a network model with a limited amount of data may result in model overfitting and degraded generalization. In our experiments, we manually labeled 115921 players in 9223 selected video frames. It remains unknown whether such amount of data samples are substantial for network training in this case.

Besides, the quality of the prepared training data should be taken into consideration since carelessly-prepared frame images may end up “fooling” the trained networks. For instance, when labeling players with strong motion blurs, we may accidentally draw bounding boxes with a poor accuracy. Therefore, errors can be expected between our data annotations and the “real-world” ground truths, causing a potential decline in the quality of the training data.

The variety of the training data is another vital factor of neural network training. The data samples used in this thesis are collected from real-time videos of football games. These videos are carefully selected to guarantee a comprehensive coverage of diverse game scenarios, such as different image resolutions, player sizes, and light conditions. However, there still exists various situations that are not covered due to the limitation of our data source, and the generalization of the proposed systems may thus be impacted negatively. For example, weather conditions are not thoroughly considered in this thesis. When football games collide with extreme weather events (e.g. rain or snow), the proposed systems may not work as expected since they have insubstantial knowledge about such unseen scenarios.
5.1.2 YOLO

In both the CNN-based system and the combined system, YOLO is applied as a pre-trained model and fine-tuned with the selected football video sequences. Consequently, the performance of these systems may be affected by the settings of the YOLO model.

First of all, the number of grid cells configured in the YOLO model may impact the detection performance. When YOLO is used for detection tasks, an input image is typically divided into $S \times S$ grid cells. Each grid cell is assumed to contain only one object. This assumption may lose its validity when multiple objects or players in our case stay fairly close to each other. Some overlapped players may not be easily detected with a scarce number of configured grid cells. On the other hand, if an input image is densely distributed with grid cells, a single player may be segmented by multiple grid cells and thus overlaid with numerous bounding boxes. Such practice increases computational cost for training and may solely bring limited gain in detection accuracy. YOLO may fail in learning to separate closely-located players if such cases occur infrequently in the training dataset. The granularity of grid cells in our case is experimentally found based on how crowded the players are distributed in the ground truth data. The performance may be impacted if the choice of the granularity is driven by other means.

Another aspect of YOLO that may impact the performance is the network architecture. In this thesis, we trim a Tiny-YOLO-V2 model to preserve its first 8 layers and fuse them with additional 4 convolutional layers in our CNN-base system and the combined system. The architecture is determined and optimized based on the current provided dataset and may need further refinements to cope with various data inputs in the future. For example, if image samples of rainy or snowy football games are included in the dataset, the current network architecture may need a scale-up to handle such new scenarios. Then keeping up to the first 10 or 12 layers of the Tiny-YOLO-V2 model could be a feasible solution in this case.

5.1.3 Histogram-based System

The proposed histogram-based system in this thesis does not rely on neural networks and requires no network training. It is easily deployed but may not guarantee a matching quality of the tracking-by-detection results.

There are generally three major factors that may impact its performance. First of all, the initialization of the bounding boxes at the first frame is a crucial factor since it determines the reference color histogram of the targeted players. Drawing a bounding box on top of a player with an inappropriate size or position may pollute the reference histogram with irrelevant color information and cause a drop of detection accuracy.

Besides, each bounding box is moved around its current position within a region to derive proposals of the targeted player in the next frame. The region size is correlated with the sampling rate of frames. If two consecutive frames are sampled between a short interval from an video clip, then the position of a player should not differ much between two frames. A small region in this case should be adequate to capture the transition. On the contrary, a relative large region is required to cope with a low sampling rate since players may move far
away in the current frame from their previous positions. A lousy setting of the region size may result in an unexpected performance degradation.

Lastly, the performance of histogram-based system is negatively affected by motion blurs. When motion blurs happen, the region proposals may have a similar color histogram distribution as the background. Then using the histogram similarity as the sole criterion would make the histogram-based system suffer from this disturbance.

5.2 Conclusions

In this thesis, we first propose a sequence-adaptive histogram-based system and a customized CNN-based system to tackle the challenging problem of multi-players tracking in football video sequences. The sequence-adaptive histogram-based system associates player detections at the past frame with region proposals at the current frame. Based on the assumption that players move with linear velocity between short time intervals, the computational complexity of the histogram-based system is considerably reduced. The customized CNN-based system is fine-tuned from Tiny-YOLO-V2 with the architecture and weights optimized for the football player tracking-by-detection task. In addition, this thesis provides a recipe for the customization of a tracking-by-detection system. We combine the above two systems by concatenating a sequence-adaptive histogram-based post-processor to the CNN-based system. The combined system is expected to leverage the advantages of the CNN-based system while making full use of players’ visual appearance characteristics.

Based on the experiment results and related analysis, we can conclude this thesis by answering the research questions in Section 1.3:

• For question 1, both the histogram-based system and CNN-based system manage to track football players across consecutive video frames. Experiment results indicate that the CNN-based system generally outperforms the histogram-based system in terms of a higher mean IOU, precision and recall rate on the testing sequences.

• For question 2, experiments are conducted to benchmark the performance of the CNN-based system against off-the-shelf systems, such as Faster-RCNN. Results reveal that our CNN-based system delivers a superior performance than its benchmarks in terms of a higher mAP on the testing sequences. Fine-tuning the CNN-based system specifically with football video sequences prepares it with the knowledge about the characteristics of football players and motion blurs. This practice contributes to such a performance boost for players tracking-by-detection in football videos.

• For question 3, we evaluate the performance of the combined system against the pure CNN-based system and the CNN-SORT system. Results demonstrate that the combined system performs better than its benchmarks. Based on the result, it can be concluded that adding a histogram-based post-processor to the CNN-based system could significantly improve its tracking performance. Also, this fused pattern opens up the possibility to adjust the detection probability based on the dominant characteristics of a detection task.
5.2.1 Future Work

For the combined system, we can try other post-processor adaptive detectors based on the jersey color of each team in order to differentiate teams in the future. It is considered much tricky to accomplish this task if we use a pure CNN-based system. Also, now we only train our CNN-based system on football games. In the future, we can adapt it to other types of sports, such as basketball or baseball games. As mentioned in the limitation section, in the future we can also consider including scenarios of extreme weather when selecting training data.
Bibliography


