

Faculté des sciences

Multi-Player Tracking in field hockey

Framework analysis and visual cues leverage for
Referee, Goalkeeper and Field player detection

Author: Adrien Kinart

Supervisors : Robin Van Oirbeek, Andrea Pennisi

Reader : Christian Hafner

Academic year 2022-2023

ABSTRACT

With the advent of new technologies such as Artificial Intelligence (AI), the Internet of Things (IoT), and Cloud Computing, many industries are being revolutionized. These advances offer new low-cost solutions to complex problems. Despite this, amateur sports, such as field hockey, have seen limited research due to constraints such as budgets and a lack of public data for performance benchmarking. Throughout this study, the perspective of a field hockey coach wishing to make the most informed decisions related to in-game tactics and injury risk prevention is adopted. This study is two-fold. It first proposes a Multi-Player Tracking (MPT) framework that guides in selecting appropriate devices and techniques for insight extraction. To reflect the reality of the field, the following framework has been chosen: real-time and fully automatic insight extraction from a single stationary camera. Secondly, it paves the way for Computer Vision techniques by creating a first public annotated dataset of field hockey footage and applies current techniques before assessing their viability. Finally, YOLO detection algorithms were trained on the newly annotated dataset to classify between the goalkeepers, referees, and field players. This is possible in field hockey by exploiting the visual cues relative to the three classes. YOLOv5n, which can run almost in real-time on a CPU, was compared to YOLOv7-X and the newly released YOLOv8s that run in real-time on a GPU. Custom YOLOv5 rapidly converges, while the learning curves of Custom YOLOv7 and YOLOv8 still indicate a need for more training and data, although their performance is already satisfactory. As expected, YOLOv8 performance is better than YOLOv7 and should become the reference in MPT. Overall, this study shows that the visual cues of field hockey actors can be leveraged to turn the team classification task into a hard clustering task with two well-balanced groups, without having to deal with the noise introduced by the difference in appearance of the referee and goalkeeper.

DEDICATION AND ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my family for enabling me to undertake and successfully complete this Master's degree with this thesis. My heartfelt thanks also go to my friends who have consistently stood by me throughout this transformative journey. A special mention goes to Arthur and Simon, with whom I co-founded Troople, a Data Science consultancy firm. Their close involvement and unwavering support have been invaluable throughout this experience!

I would like to extend my heartfelt gratitude to my supervisor, Robin Van Oirbeek, and co-supervisor, Andrea Pennisi. Their invaluable expertise and complementary perspectives have greatly contributed to the success of this thesis.

AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programs and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: DATE:

TABLE OF CONTENTS

	Page
List of Tables	ix
List of Figures	xi
ACRONYMS	Page
INTRODUCTION	1
 I Framework Design for Multi-Player Tracking	 3
1 Framework establishing	5
1.1 Sport at hand and type of insights that are seeked	5
1.2 Budget and independence of degree	7
1.3 Camera's number and camera movement	8
1.4 Insights stream	9
1.5 Automation extent	10
 2 Summary and thesis framework choice	 11
 II Computer Vision pipeline techniques applied to field hockey	 13
3 Literature review and openings	15
3.1 Introduction	15
3.2 Court/Field detection	17
3.2.1 Segmentation techniques	17
3.2.2 Line and edge detection	18
3.3 Camera pose estimation and tracking	22
3.4 Player and ball detection and tracking	24
3.4.1 Introduction	24
3.4.2 Multi Players Detection	26
3.4.3 You Only Look Once (YOLO) - Theoretical background	29

TABLE OF CONTENTS

3.4.4	Multi Player Tracking	36
3.4.5	Ball detection and tracking	42
3.5	Player Clustering and team classification	43
3.5.1	Introduction	43
3.5.2	Traditional techniques	44
3.5.3	Deep learning methods	47
3.5.4	Opening : Build a new dataset and train custom detection algorithms based on it	48
3.6	Insights extraction and visualisation	49
4	Methodology	51
4.1	Dataset construction and sanity checks	51
4.2	Custom YOLO models for Field hockey	52
4.2.1	Training step procedure	52
4.2.2	Training assessment procedure	53
5	Results and discussions	55
5.1	Opening 1 - Annotated Dataset construction	55
5.2	Opening 2 - Custom training	56
	CONCLUSION	65
A	Appendix A	67
A.1	Tables	67
A.2	Figures	68
A.3	Supplementary notes	85
A.3.1	Annotation platform - Amazon Mechanical Turk	85
A.3.2	YOLO format structure	87
	Bibliography	89

LIST OF TABLES

TABLE	Page
3.1 Clustering techniques Evaluation on the initial frame	47
4.1 Speed and Performance comparison between YOLOv5n, YOLOv7-X and YOLOv8s . .	52
4.2 Main hyper parameters for YOLOs training	53
5.1 Best mAP@0.5 and mAP@0.95 results on Validation set for YOLOv5 (top) , YOLOv7 (middle) and YOLOv8 (bottom)	61
A.1 Hough Lines Parameters	67

LIST OF FIGURES

FIGURE	Page
1.1 Team sport process	6
1.2 Example of Local Positional Systems (LPS) infrastructure	7
1.3 Example when the camera is fixed without any zoom in/out	9
2.1 Framework of Multi-Players Tracking insights extraction	12
3.1 Low-cost methodology in CV. Example in football [Mavrogiannis and Maglogiannis, 2022]	15
3.2 [Ali et al., 2012] filter applied on field hockey pitch	18
3.3 Left: common line representation. Right: Hough space representation of this line for two points (x_1, y_1) and (x_2, y_2)	19
3.4 Left: common line representation. Right: Hough space Hesse normal form of this line for two points (x_1, y_1) and (x_2, y_2)	20
3.5 From two lines to Hough space Hesse normal form. Each line results in an intersection.	21
3.6 Hough lines detected in the initial frame	21
3.7 Projection on a board of players detected as points (in red) using Homography H . . .	23
3.8 YOLOv1 architecture	29
3.9 Grid cell partition of an input image in YOLO algorithm	30
3.10 YOLOv2 improvements compared to YOLOv1	33
3.11 FPN architecture in YOLOv3 algorithm	34
3.12 YOLOv5 detections on field hockey frame	35
3.13 YOLOv8 detection on field hockey frame	36
3.14 Object tracking procedure of sort [Bewley et al., 2016]	38
3.15 The Kalman filter: prediction and correction algorithm	39
3.16 Assignment problem representation as a Bipartite Graph. Cells numbers are the associated cost between the detections and trackers	41
3.17 Bipartite graph representation in MOT. LHS graph allows to account for additional detected objects. RHS graph accounts for the minimisation problem requirement . . .	42
3.18 RGB projection of image and segmentation using K-means with $K=5$	44
3.19 The resulting 5 sorted dominant colors	44

LIST OF FIGURES

3.20	RGB and HSI Color histogram applied to an object in initial frame (above) and field player in field player (below)	45
3.21	Field hockey actors lined up in a Belgian National 1 match	48
3.22	Example of a player heatmap in football [Foinea et al., 2010]	49
4.1	Difference in bounding boxes between a person and a hockey field player	52
4.2	PR and F1-Confidence curves	54
5.1	Class distribution of the field hockey dataset	56
5.2	Custom YOLOv5 detections	57
5.3	Custom YOLOv8 detections	57
5.4	YOLOv5 & YOLOv7 results training set	58
5.5	YOLOv5 & YOLOv7 results validation set	58
5.6	YOLOs mean Average Precision for two threshold ranges	59
5.7	YOLOs Precision and Recall curves	59
5.8	Confusion matrix of the custom YOLOv7 model on field hockey players (200 epochs)	62
5.9	YOLOv7	62
5.10		63
A.1	(1) Field segmentation and (2) edges segmentation procedure in [Chen and Little, 2018]	68
A.2	nms	68
A.3	YOLO versions comparison w.r.t speed and AP ([Wang et al., 2022])	69
A.4	DeepSort appearance feature space construction	69
A.5	DBSCAN Clustering with a greedy approach to find two groups and outliers with $\text{eps} = 5.0325$ and $\text{min}_{\text{samples}} = 6$	70
A.6	Confusion matrix from DBSCAN clustering with $K = 2$ groups	71
A.7	K-means clustering Confusion matrix on initial frame	72
A.8	K-means clustering on initial frame	72
A.9	RGB (left) and HSV (right) histograms example	73
A.10	Confusion matrix yoloV5 training	74
A.11	Confusion matrix yoloV8 training	75
A.12	YoloV8	76
A.13	YOLOv8 training losses	77
A.14	Illustration and comparison of F1 AUC and AP measures computations	78
A.15	YOLOv7 detections on initial frame	79
A.16		80
A.17		81
A.18	Sort and DeepSort algorithm architecture	82
A.19	Custom YOLOv7 results on initial frame	83

A.20 Classification example of the most common viewpoints that exist in field hockey . . .	84
A.21 A task of bounding box doing for objects detection	86

ACRONYMS

AI Artificial Intelligence. [1](#), [5](#)

CM Confusion matrix. [53](#)

CNN Convolutional Neural Networks. [27](#), [29](#), [43](#), [47](#)

CPU Central Processing Unit. [52](#)

CV Computer Vision. [1](#), [2](#), [7](#), [8](#), [11](#), [15](#), [24](#), [26](#), [65](#), [66](#)

DBSCAN Density-Based-Spatial Clustering of Application with Noise. [45–47](#), [65](#)

DL Deep Learning. [17](#), [27](#), [47](#), [66](#)

DNN Deep Neural Networks. [26](#)

Faster R-CNN Faster Region-based CNN. [27](#), [33](#)

FPN Feature Pyramid Network. [34](#)

GPS Global Positioning Systems. [7](#)

GPU Graphics Processing Unit. [52](#)

HOG Histogram of oriented gradients. [27](#)

IoT Internet of Things. [1](#)

IoU Intersection-over-Union. [30](#), [41](#), [42](#), [59](#)

LPS Local Positional Systems. [xi](#), [7](#)

mAP mean Average Precision. [53](#), [54](#), [59](#), [60](#), [66](#)

ML Machine learning. [6](#)

MOT Multi-Object Tracking. [24–27](#), [36](#), [41](#)

MPT Multi-Player Tracking. [1](#), [11](#), [15](#), [24](#), [26](#), [52](#)

NMS Non Maximum Suppression. [19](#)

OS Optical Systems. [7](#)

PR Precision-Recall. [53](#)

RANSAC RANdom SAmple Consensus. [22](#)

SA Segment Anything. [66](#)

SAOT Semi-Automated Offside Technology. [7](#)

SIFT Scale-Invariant feature transform. [27](#)

SORT Simple Online and Realtime Tracking. [26](#), [36](#), [38](#), [41](#), [42](#)

SSD Single-Shot Detector. [26](#), [27](#)

SSE Sum of Squared Errors. [31](#), [32](#)

VAR Video Assistant Referee. [7](#)

YOLO You-Only-Look-Once. [26–29](#), [31–34](#), [51](#), [53](#), [65](#), [66](#)

INTRODUCTION

Background and Context: The sports industry represents an substantial market, valued at over 1.4 trillion US dollars in 2019. The significance of technology within this industry is growing, with the global sports technology market expected to reach 48.7 billion dollars by 2028. This surge is driven by advancements in sports analytics, including [Artificial Intelligence \(AI\)](#), [Computer Vision \(CV\)](#), [Internet of Things \(IoT\)](#), and cloud computing. For instance, [IoT](#) enables the collection of real-time data on movement, biometrics, and environmental factors, while cloud computing facilitates the storage and management of large data volumes. The combination of these technologies, , along with the emergence of 5G connectivity, is revolutionizing various industries, including sports. However, in amateur sports like field hockey, which often operate with limited budgets for advanced technologies and data analysis, relatively little research as been conducted compared to more prominent sports.

Research Question: However, the need for real-time insights and strategic decision-making during matches remains crucial for both teams and coaches. The challenge lies in the lack of a definitive framework that justifies the selection of the most efficient techniques. Addressing this question is important for both the users of the insights and the developers of tracking algorithms. A clear framework enables users to anticipate the nature of the insights they can expect and understand the level of investment required. For developers, a well-defined framework aids in making informed decisions regarding the selection of appropriate devices, algorithms, preprocessing techniques, and other relevant factors. Another issue related to field hockey (and other amateur sports) is the lack of publicly available data. Most studies in the field have used private data, making the analysis more challenging, such as the absence of a benchmark for performance assessment.

Objectives: The global aim of this thesis is twofold. The first part encompasses a consultancy vision where a [Multi-Player Tracking \(MPT\)](#) framework is established in order to use the most suitable devices and techniques accordingly. Then, the second part encompasses a more technical vision where a [CV](#) pipeline is specifically designed for field hockey. Tracking the ball and players helps understand the behavior of both the own and adverse teams on the field, enabling more objective and tailored decisions. It also aids in injury prevention.

Research Significance: To achieve this, the thesis will first establish a well-defined process of inquiry to assess the use of the best available techniques according to needs and budget. Additionally, a newly annotated dataset has been created to improve tracking performance. A custom detection algorithm will be trained to separate goalkeepers, referees, and field players, removing noise that can interfere with algorithms clustering players into the first or second team. This transforms the problem into a hard clustering task with two fixed groups of the same size. Finally, since the goal is to extract insights from match footage, each chosen technique will be tested on simple examples to assess their viability within the defined framework.

Scope: The research primarily focuses on the real-time tracking of players in field hockey. It will assess existing techniques, drawing insights from football techniques as well, due to their similarities and more extensive research. Camera settings and the degree of human intervention play a central role in choosing the recording devices and optimal algorithms. The study aims to align with the reality of field hockey, where matches are often recorded using only one stationary camera. Techniques that require no intervention while running will be explored, with an emphasis on player detection and how it can enhance insights extraction.

Methodology: Part [I](#) of the thesis involves raising pertinent questions that contribute to establishing a clear framework. Subsequently, a carefully defined framework will be selected, facilitating the adoption of the most suitable techniques available while keeping the necessary budget low. From this clear framework, Part [II](#) is addressed. Chapter [3](#) reviews the current techniques corresponding to the framework, from detecting the field to the extraction of insights. Video footage has been collected from two YouTube sources and is now publicly available on the Kaggle website [[Kinart, 2023d](#)]. At the end of each subsection explaining the given [CV](#) step, the current technique(s) will be applied to a simple example to assess its viability within the field reality. Given the newly created data, Subsection [3.4](#) will focus longer on detections through the analysis of current algorithms. Each of them will be trained, assessed, and discussed before summarizing the main takeaways of this work.

Part I

Framework Design for Multi-Player Tracking

FRAMEWORK ESTABLISHING

1.1 Sport at hand and type of insights that are sought

Review

[Beal et al., 2019] identified four key areas where performance and/or financial benefits can be delivered:

1. **Match outcome prediction:** The task is usually considered as a classification problem where one tries to predict the probability P of a team t_i to win a match given a certain amount of information $\mathbf{X}$ [Horvat and Job, 2020]:

$$P(t_i|\mathbf{X})$$

Here, mostly used machine learning models are Neural Network.

2. **Strategic and tactical decisions making:** this includes player recruitment, tactics, team selection and the training of young players. The framework is proposed in Figure 1.1. This allows to realise the potential the data can offer. AI is useful for tailoring training sessions, depending on fitness, mood, weather and injury risk. It also allows to bring intelligence to team selection. Moreover, Deep learning methods exist in order to "imitate" team movements and take the best decision accordingly [Seidl et al., 2018]. In-game tactics also play a huge role in the pursuit of victory or risk of injury. There are many match examples where a tactical shift totally changed the game's output, in a way or another. Knowing opponent's team behaviour might help to take better decision accordingly. For example, knowing the covered distance, the number of long sprints, acceleration, etc. can help to know risk of injury. Details will be discussed in Section 3.6.

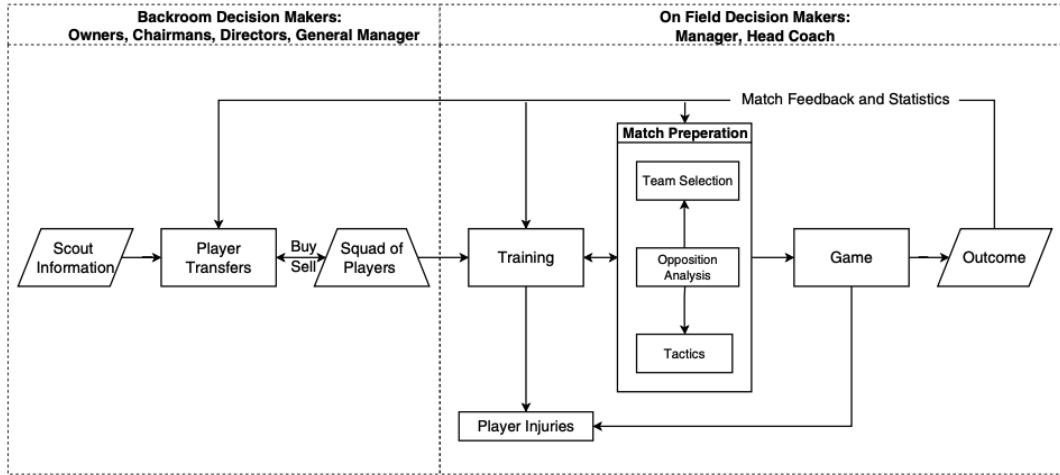


Figure 1.1: Team sport process

3. **Fantasy sport games:** This consists in predicting the performance of real-world sports teams and players. The most famous one is the Fantasy Premier League with over than 9 million daily users ¹. It consists in evaluating the optimal sport team based on players performance, predicting the player values and creating betting strategies. This is a problem relative to time series where features are related to each other [Beal et al., 2020].
4. **Managing injuries:** Once again, this problem is part of the traditional machine learning pipeline where there algorithms are trained to predict the probability/risk of injury given physiological information. Currently, Machine learning (ML) methods manage to correctly identify high risk injuries while mentoring important injury factors [Eetvelde et al., 2021].

This thesis framework: Field hockey & strategic decisions during the game

Here the focus is applied to the field hockey world with its typical view angles, actions, structure, etc. This means that the algorithms and techniques that will be presented are either already existing in the literature for hockey purposes or are thought (and justified) to be translated to field hockey. For example, it has many similarities with football. Indeed, it has similar field dimension, with both team a goal to defend. The team structure is also the same since teams are composed of 11 players including a goalkeeper. The latter tends to stand close to the goal. In this thesis, as it can be seen from Figure 1.1, one puts themselves in the shoes of someone who helps the coach to take better strategic decisions relative the in-game. It also allows to estimate the physical load associated to the players. [Akenhead and Nassis, 2016] affirms that total distance, distance covered above 5.5m/s, metabolic power and heart-rate exertion measures play a central role to assess injury risk.

¹<https://fantasy.premierleague.com/>

1.2 Budget and independence of degree

Review

There exist many devices dedicated to extract analytics from sport, especially in football, basketball, baseball, etc. Nowadays, there are 2 main technologies. The first one is related to positioning systems with [LPS](#) along with [Global Positioning Systems \(GPS\)](#), [Optical Systems \(OS\)](#). Using best known respective techniques, they allow to extract extremely precise insights with a very high quality [[Pons et al., 2019](#)]. However, it is highly labour and time intensive to collect such data. Moreover, those technologies can only be used in locations that have the adequate infrastructure. An example for [LPS](#) in football is shown in Figure 1.2 ².

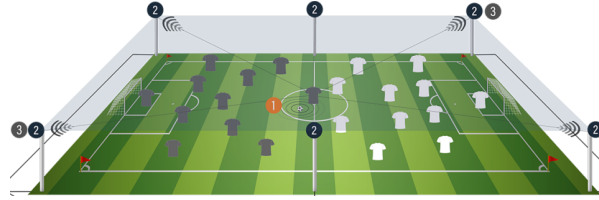


Figure 1.2: Example of [LPS](#) infrastructure

On top of that, maintenance must be continually carried on. This makes those technologies very costly [[Sharma et al., 2017](#)]. Moreover, one cannot ask opponents to wear sensors in order to retrieve their performance. Accordingly, if one wishes to get information on the opponents' behaviour, they need to turn to [CV](#) techniques. In that area, the cutting-edge technologies refer to the [Semi-Automated Offside Technology \(SAOT\)](#), which is proposing an augmentation of the [Video Assistant Referee \(VAR\)](#) with bespoke cameras installed under the roof of the stadium. The cameras offer real-time tracking for all actors. In amateur sport world, it is then unthinkable to have such infrastructure because of budget's lack. This means that the budget that can be allocated is extremely important when considering in-game analytics extraction. This does not mean that nothing can be done, one will see which techniques can be leverage with an extremely low budget at hand. As an example, [[Direkoglu et al., 2017](#)] and [[Bialkowski et al., 2013](#)] provide very interesting insights on field hockey players detection while recording on 8 fixed cameras at the same time. Moreover, [[Duarte Moura et al., 2022](#)] uses [CV](#) techniques from a single low-cost stationary camera.

This thesis framework: Extremely low budget and complete independence.

Given this framework, it is proposed to apply [CV](#) techniques where only one stationary camera is used to provide the frames. It is wished to represent to reality of amateur field hockey environment. As it will be seen, it has a large influence on the algorithms architecture to use. From the in-

²<https://inmotio.eu/football-performance-analysis>

puts feeding to insights extraction, there is a typical pipeline, see [Mavrogiannis and Maglogiannis, 2022]. It will be presented and depicted in the Section 3.1.

1.3 Camera's number and camera movement

Review

This question is crucial in the sense that it completely changes the inputs to be fed to the algorithms. Here below is a list of the pros and cons of having multiple cameras:

Advantages

1. **Robustness to occlusions** [Sternig et al., 2011]: Occlusion is when objects cross each other during their tracking. This leads to identification loss so loss of insights. This is more detailed in Section 3.4 challenges. Multiple cameras allows to decrease the probability of objects detection losses.
2. **Reduced distance to the targets**: Tracking algorithm performances decreases with the distance to the target. Having e.g. a second camera in opposite side would reduced average distance by two.
3. **Robustness to lightning changes**: Many CV algorithm relies on colors, see for example field detection in section 3.2, teams assignation in section 3.5, etc. Accordingly, having multiple cameras allows to smooth the aggregated inputs [Wang et al., 2014].

Disadvantages

1. **Systems complexification**: one needs hardware and software devices able to synchronise the cameras together. It also requires adapted maintenance .
2. **Data management**: the data flow becomes way more complex. One first needs to be able to synchronise the data inputs. Moreover, there are a lots of other difficulties that arise such as the fact that one needs to merge effectively the information. For example, one can think at merging detected objects identities.

Regarding the stationarity of the camera, the preprocessing steps are simpler and faster compared to moving camera where computations are needed in order to synchronise locations. On the other hand, it required that the whole field is covered by the camera, which often leads to bad resolution. Because of that, detection and tracking accuracy is often lower.

This thesis framework: 1 stationary camera

In practice, in Belgian amateur field hockey, there is a single fixed camera above one of the goals. Figure 1.3 shows an example in practice.



Figure 1.3: Example when the camera is fixed without any zoom in/out

Therefore, this will be the chosen framework. It is recognised that, in future work, if one manages to get two-cameras recorded hockey matches, it would be very interesting to dig that field of computer vision. Indeed, it is not too difficult to synchronise the cameras. [Csanalosi et al., 2020] proposes one, based on the start whistle of the match.

1.4 Insights stream

Review

The application of real-time versus offline insights is different. Real-time is related to decision making during the game (e.g. tactic, injury risk, etc.) while offline algorithms are used post-match in an improving global performance aim. Hence, a real-time algorithm is an algorithm that not only satisfies a logical correctness but also satisfies a time deadline. Moreover, it should be pointed out that the data availability is not the same since in offline settings, one can also use data retrospectively. Finally, since there is no computation efficiency constraint, one can use more complex algorithms. In that view, the computation power of the device are key elements.

This thesis framework: Real-time analytics.

One could imagine being a coach during the match and looking at what is currently going on in order to take the best advised decisions. At worst, one can tolerate what is called a *burning period*, which is a latency period at the beginning of the video, during which the algorithm runs before getting insights. Of course, one wishes it to be as short as possible.

1.5 Automation extent

Review

Throughout the insights extraction's pipeline, it is possible to add human intervention. Each time that happens, it provides additional inputs to enhance the algorithms that are used. It can provide a great deal of added value, such as in [Duarte Moura et al., 2022] where previously labeled data about the teams allows 98 % accuracy compared to 69.2 % for team classification when there is no human intervention at all.

This thesis framework: Fully automatic pipeline.

This means the algorithms can be pretrained but are essentially in an unsupervised environment. For example, an algorithm can be pretrained in order to detect field hockey players and the ball but the training must be independent from the video that is running. As it will be seen, this has big consequences on the team labelling task. Intuitively, this task seems easy if one can for instance prelabel the belonging of the players or give the jersey's color of the player. This problematic will be more deeply studied in this thesis.

SUMMARY AND THESIS FRAMEWORK CHOICE

The previous chapter outlines the necessary process to undertake when one wishes to extract insights from a match in the **MPT** context. As soon as one is on **low budget** and needs **insights independently** from the opponents, **CV** tools are the best alternative. Furthermore, once the framework has been established, it allows focus on the right techniques and scientific literature, with a clear view of their respective pros and cons. On the other hand, developers know which techniques they should concentrate on. This also enables them to provide arguments to users regarding the quality of extracted insights given the requested framework, as well as justify higher resource needs. In this thesis, the chosen study techniques are for extracting **real-time** and **fully automatic** insights from a **single** and **stationary** camera. The global framework, as well as the thesis framework, is shown in Figure 2.1.

This portion of the work leans more towards a consultancy perspective, rather than a technical one. However, it should help data scientists quickly navigate the appropriate framework according to their users' expectations and budget.

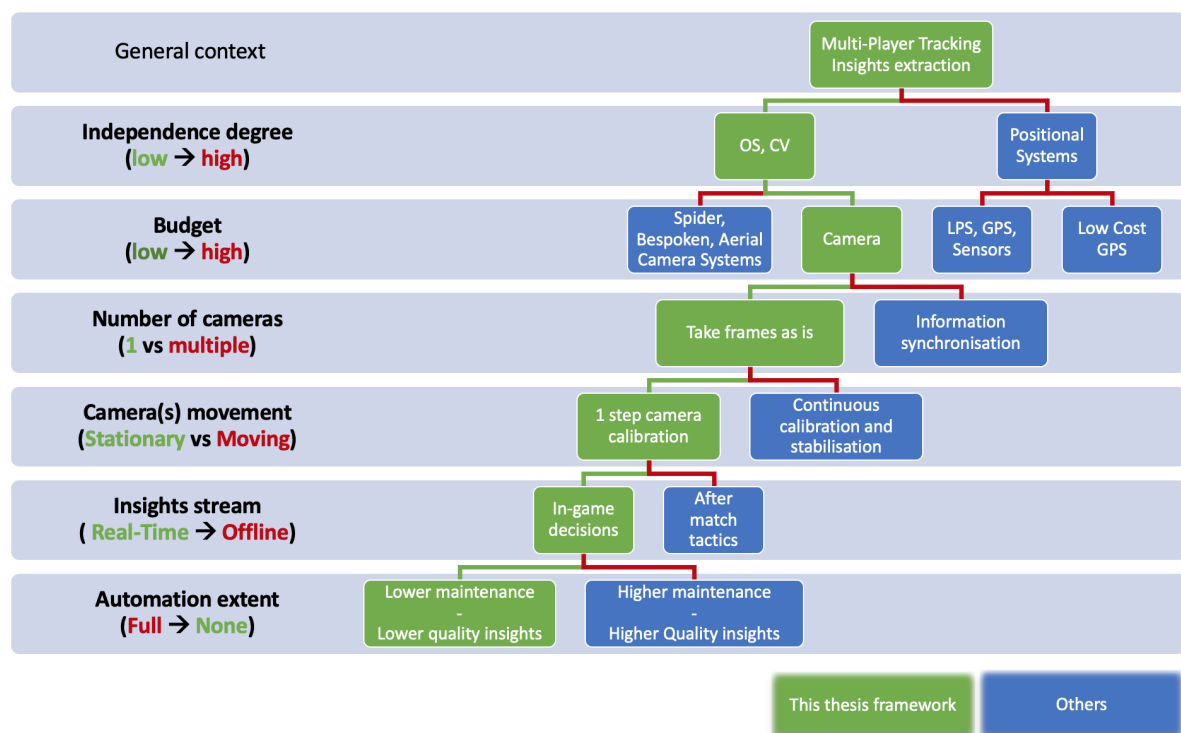


Figure 2.1: Framework of Multi-Players Tracking insights extraction

Part II

Computer Vision pipeline techniques applied to field hockey

LITERATURE REVIEW AND OPENINGS

3.1 Introduction

Part I demonstrated that **MPT** using **CV** is encapsulated in a pipeline divided into seven steps, as shown in Figure 3.1. The subsequent sections in this chapter will provide a literature review within the framework discussed earlier, apply current techniques to a simple example, and then focus on the Detection step.

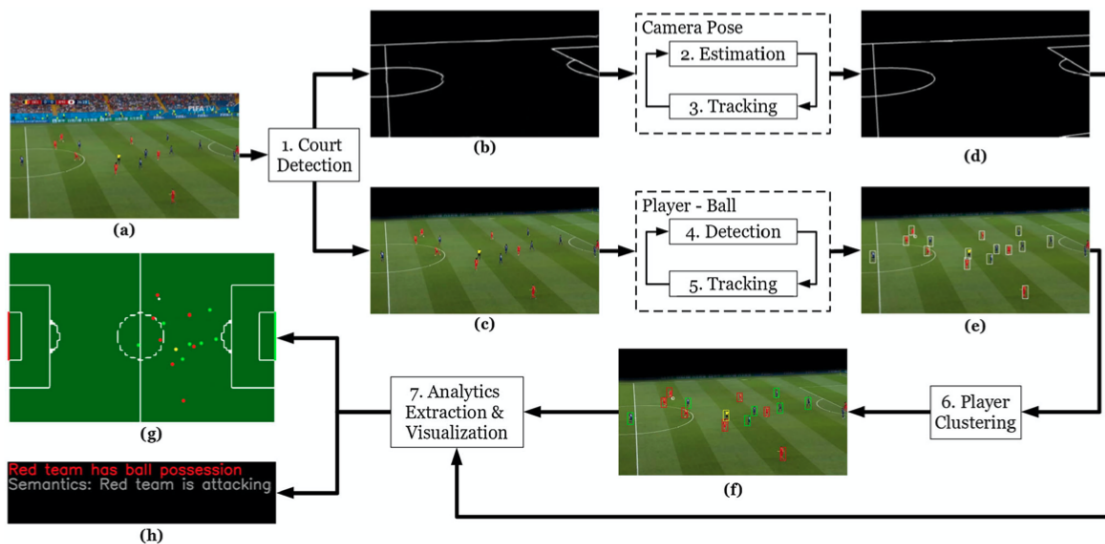


Figure 3.1: Low-cost methodology in CV. Example in football [Mavrogiannis and Maglogiannis, 2022]

Specifically, this review will concentrate on two aspects. Firstly, computer vision techniques applied to field hockey will be examined. Secondly, existing techniques that could be adapted for field hockey will be identified and justified based on their translatability. Among these, some will be tested for further discussion. Throughout these steps, footage of a field hockey match, as in Figure 1.3, will be used.

3.2 Court/Field detection

As previously stated, numerous parameters impact the challenge of effectively delimiting the field, depending on whether the camera is fixed or moving, and whether zoom-ins or zoom-outs occur. Expectedly, the simplest setting would be when the camera is fixed, recording the entire field without zooms, as illustrated in Figure 1.3. Generally, there are three primary potential approaches. One can either utilize the color of the field for segmentation, extract the lines of the pitch, or do both simultaneously.

3.2.1 Segmentation techniques

First, the colors of the field can be projected into RGB space. This method has the advantage of being fast, however, it suffers from changes in brightness. It's also important to remember that in this framework, there's no human intervention. Unlike football, a field hockey pitch can be either green or blue. The blue color actually enhances visibility in TV broadcasts, especially during low light conditions or under floodlights. This implies that assuming green color for the grass is an even tougher requirement for field hockey. However, in such a case, one can filter out non-ground pixels using the filter proposed by [Ali et al., 2012], which is shown in Equation (3.1).

$$(3.1) \quad \text{Ground}(x, y) = \begin{cases} 0 & (g(x, y) > r(x, y) > b(x, y)) \\ 0 & \text{otherwise} \end{cases}$$

This filter applied to initial image gives the result shown in Figure 3.2.

As can be seen, this simple filter effectively segments the field, allowing for the accurate extraction of field dimensions. The presence of shadows on the field leads to unsegmented regions, underscoring the sensitivity of color-based algorithms. In the context of this thesis, the filter shows promise as a viable candidate for field detection due to its performance trade-off between computational intensity and accuracy. Moreover, there are many potential improvements. For instance, since the camera is fixed, one can decrease the computational time by computing average or median values of pixels to estimate background pixel values. However, this requires a **period** during which this value is calculated. A burning period refers to the time elapsed before getting insights while the stream of inputs go. Another example is refining the detection of dominant color regions by filtering out small isolated areas, as seen in "Region-Growing Refinement" in [Ngo et al., 2010]. It is also possible to use convex polygon detectors [Hoernig et al., 2015]. Since a field hockey pitch is rectangular, this property could be leveraged in the same way it has been done in football

Additionally, a wide range of segmentation techniques based on **Deep Learning (DL)** has recently emerged. This has been accomplished in [Duarte Moura et al., 2022] for segmenting field hockey pitches. Another promising technique in already applied football is [Chen and Little, 2018]. Theoretically, it could be easily adapted to field hockey, provided there is enough training data,

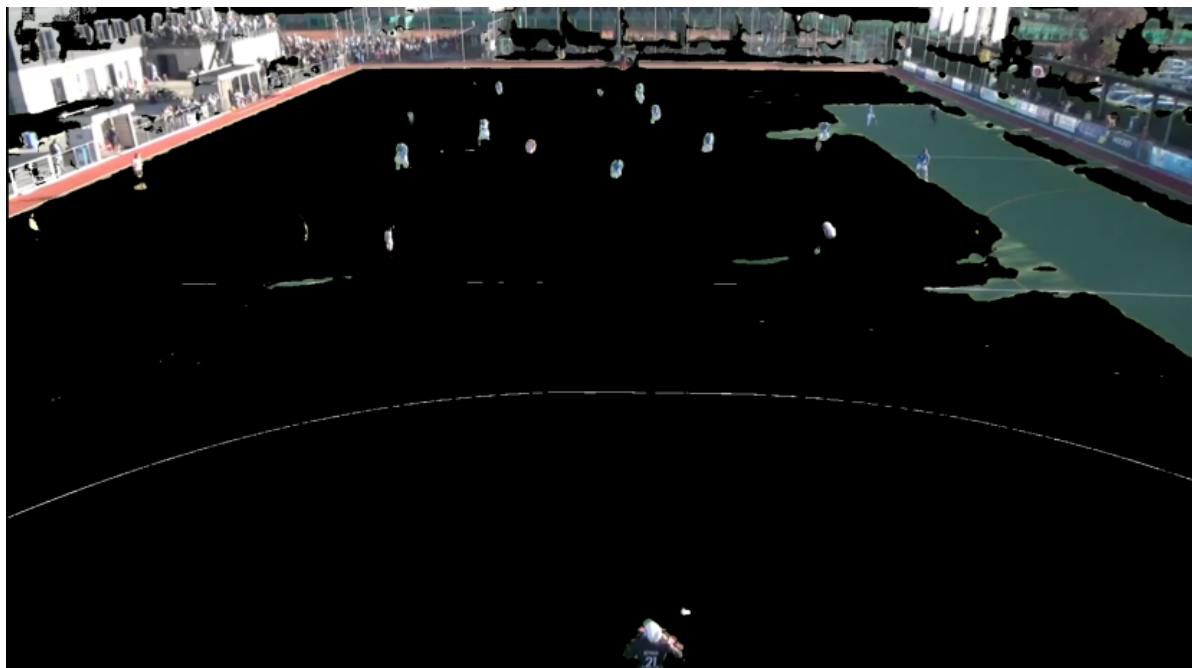


Figure 3.2: [Ali et al., 2012] filter applied on field hockey pitch

i.e., a large amount of manually segmented data. It is based on the so-called Pix2Pix algorithms that excel at translating images into new images [Zhu et al., 2017]. The first translation is made to segment the field and the other one is used to segment the pitch lines. Illustration shown in Appendix Figure A.1.

3.2.2 Line and edge detection

edges detection

The edges of the pitch must be determined. Before this, it is necessary to reduce the noise from the frames at hand. Typically, a Gaussian filter/kernel is chosen, which is:

$$(3.2) \quad G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

For each pixel, the surrounding pixels within the kernel's range are multiplied by the corresponding values in the kernel. Consequently, a larger σ requires larger convolution kernels to be accurately represented. This implies that this single parameter can have significant implications for the entire pipeline. Therefore, processing should be allocated to find the right parameters to control the extent of blurring.

Afterward, the edge filter can be utilized. The changes in color intensity are used to compute an approximation of the gradients of an image. For this purpose, the **Canny** or **Sobel** filters are typically used. The latter operator is a convolutional operator, while the former is a multi-stage

edge detection algorithm. Consequently, the Sobel operator is simpler and thus more time-efficient but with rougher edges than the Canny operator.

Given the real-time framework of this thesis, the **Sobel** operator should be preferred. It typically consists of two 3x3 convolutional kernels G_x^s and G_y^s , applied on the X and Y axes of the frame, respectively:

$$G_x^s = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad G_y^s = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

Both components are then combined to produce the final output. The gradient magnitude is given by $|G| = \sqrt{G_x^2 + G_y^2}$ and is approximated (for computational efficiency reasons) by $|G| = |G_x| + |G_y|$.

On the other hand, the **Canny** operator uses the output of the Sobel filter as an input before suppressing all the unwanted pixels that do not constitute an edge. This is done by **Non Maximum Suppression (NMS)**, i.e., checking if all the pixels are locally maximal in the gradient's direction. An illustration is shown in Appendix Figure A.2.

Finally, a determination must be made as to whether all detected edges are actually considered edges. Accordingly, range thresholds are proposed. Values below the lower threshold are considered as non-edges, values above the higher threshold are considered as edges. Those in between are classified based on their connectivity to other edges.

Lines detection

A line is represented by a two-parameter space $\Omega = (b, m)$ and is defined by the following equation:

$$(3.3) \quad y = ax + b$$

These two parameters can be represented in a so-called **Hough space**. See a line example in Figure 3.3.

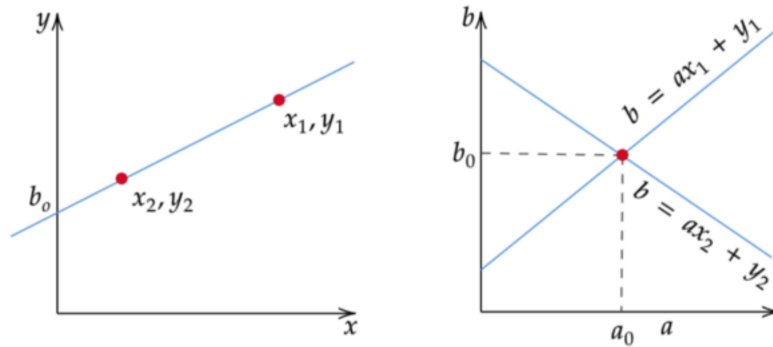


Figure 3.3: Left: common line representation. Right: Hough space representation of this line for two points (x_1, y_1) and (x_2, y_2)

One could solve the equations to find intersection points. However, for vertical lines, a would approach infinity. Accordingly, it is computationally convenient to use the so-called **Hesse normal form** that involves the following transformation:

$$(3.4) \quad \rho = x \cos \theta + y \sin \theta$$

where ρ is the distance from the origin to the closest point on the straight line, and θ is the angle between the x-axis and the line connecting the origin with that closest point. Therefore, one line results in one intersection. This concept is graphically illustrated in Figure 3.4.

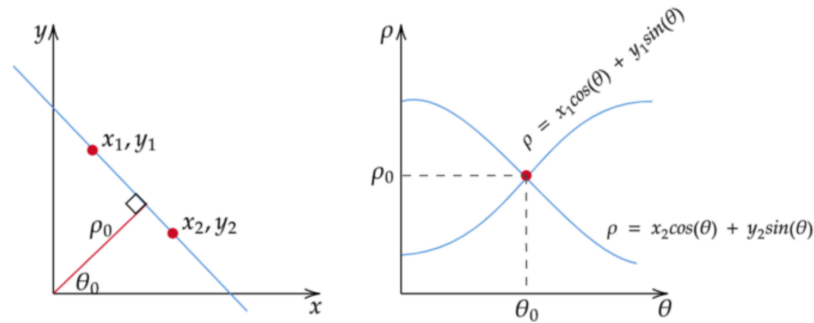


Figure 3.4: Left: common line representation. Right: Hough space Hesse normal form of this line for two points (x_1, y_1) and (x_2, y_2)

This transformation allows all points from a straight line to converge to a single point. From this, one can determine the angle and distance from the image center of the lines in the input image. An example with two lines is shown in Figure 3.5.

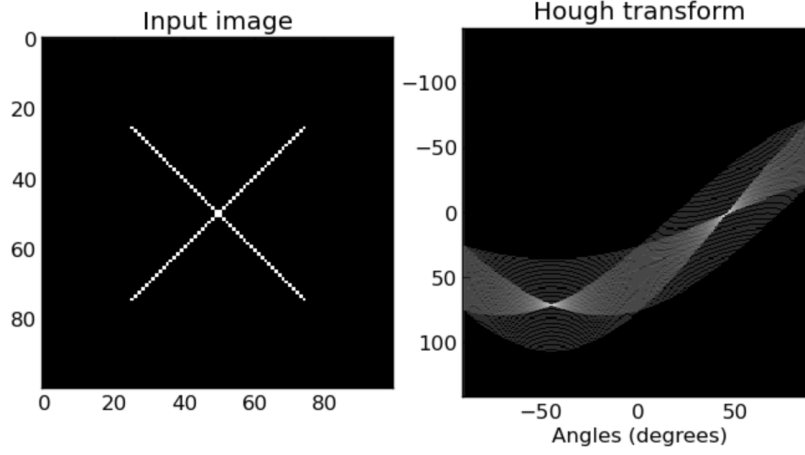
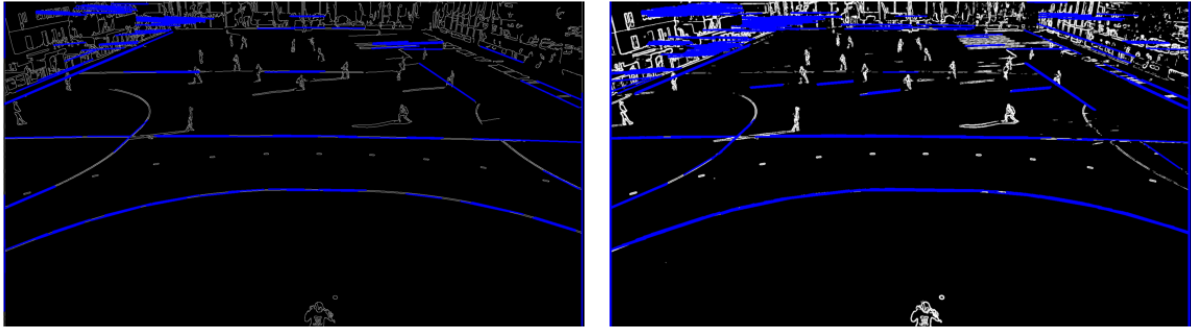


Figure 3.5: From two lines to Hough space Hesse normal form. Each line results in an intersection.

When applied to the original frame, the process yields the following results:



(a) Edges & lines detected from the Canny edge detector (b) Edges & lines detected from the Sobel edge detector

Figure 3.6: Hough lines detected in the initial frame

As observed, the Hough detector enables the detection of lines that are in close proximity to the camera. The Sobel edge detector exhibits similar performance to the Canny edge detector. Nevertheless, for both methods, a systematic approach was necessary to determine an optimal combination of parameters for obtaining satisfactory results (see parameters in Appendix Table A.1). This practical constraint makes it challenging to apply in a fully automated pipeline. This algorithm has been further optimized to make it usable in a real-time framework, for example, by using a kernel-based Hough transformation [Fernandes and Oliveira, 2008].

3.3 Camera pose estimation and tracking

This section explores techniques for determining the position and orientation of a camera in space, as well as tracking the camera's movement over time. Given the fixed camera setup in this thesis, the camera's pose relative to the scene remains constant. This allows the pose estimation step to be done only once. This step is crucial because it enables the connection between the 2D image coordinates and 3D real-world coordinates. This connection is mathematically established via a projection matrix, referred to as the **Homography** matrix. Once computed, it is possible to project the original image into a new coordinate system. Mathematically, this involves applying the homography matrix $\mathbf{H}$ to a point $\mathbf{P}$, to obtain its transformed position $\hat{\mathbf{P}}$ in the other image or coordinate system:

$$(3.5) \quad \mathbf{H} \cdot \mathbf{P} = \hat{\mathbf{P}}$$

When exactly 4 points are used to map between the world and image coordinates, the homography matrix is called the *Perspective Transform*, which requires *homogeneous matrix notation*. This notation means we note (x, y) as (kx, ky, k) with $k \neq 0$. Accordingly, Equation (3.5) can be rewritten as follows:

$$(3.6) \quad \begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ c_1 & c_2 & t \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = k_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}$$

with

- $\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix}$: the **rotation** and **scaling** vector.
- $\begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$: the **translation** vector
- $(c_1 \ c_2)$: the **projection** vector.
- $(x_i \ y_i \ 1)$: the **original point** coordinates in the first image.
- $\begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix}$: the **transformed point** coordinates in the second image.

Using 4 points, this system of equations is solvable¹. However, more than 4 points can be used to increase the robustness of the algorithm. For example, one can use detected lines from previous pipeline steps. This can be accomplished via various algorithms, among which the **RANdom SAmple Consensus (RANSAC)** algorithm is most commonly used [Sharma et al., 2017]. It is particularly useful when the image is noisy. Given a threshold T ², the **RANSAC** algorithm works as follows:

¹see the demonstration in <https://franklinta.com/2014/09/08/computing-css-matrix3d-transforms/>

²see ransacReprojThreshold in OpenCV https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga4abc2ece9fab9398f2e560d53c8c9780

1. Randomly select a subset of correspondence points, e.g., 4
2. Solve equation to find the homography matrix H
3. Compute the inliers, data points that are consistent with the estimated model within a predefined threshold or error tolerance, and outliers provided T , e.g., the distance.
4. Repeat steps 1-3 n times
5. Select the homography/perspective transform matrix with the highest number of inliers.

The technique above can also be optimized or expanded, but that is beyond the scope of this thesis. Applying the above methodology to the original image with four points yields the result shown in Figure 3.7.

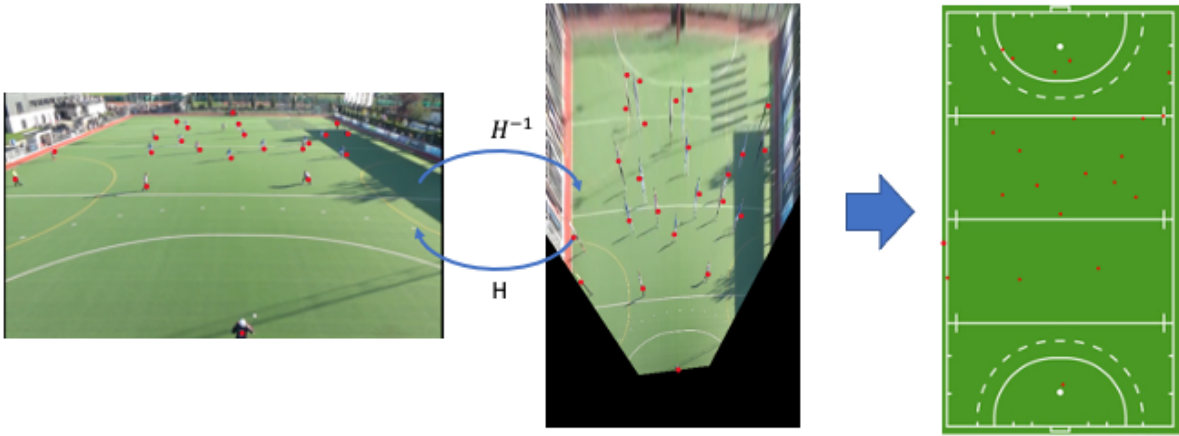


Figure 3.7: Projection on a board of players detected as points (in red) using Homography H

The projection quality of the players is significantly influenced by the filming angle. In this specific case, the angle is relatively small, leading to an inadequate representation of distant players. However, given the frame quality and the existing angle, the obtained results can be considered satisfactory. It is worth noting here that the implementation of a dual-camera system may enhance the quality of the projections, as discussed in [Eshel and Moses, 2008].

Finally, it is worth mentioning that [Chen and Little, 2018] manages to project images in bird's-eye view using previously generated edge maps (see deep learning discussion in Section 3.2.1) and then matches them to a "Feature-pose database" to perform the projection. The results are state-of-the-art in football but require substantial training data. Once again, if resources could be allocated to create similar data corresponding to field hockey footage, it would make an excellent methodology.

3.4 Player and ball detection and tracking

3.4.1 Introduction

Firstly, unlike image classification, which labels an entire image, object detection recognizes multiple objects in the image and provides bounding boxes indicating the objects' locations before classifying them. Secondly, tracking involves maintaining the identities of the detected objects over a sequence of frames. More precisely, in [MPT](#), there can be numerous targets in the scene:

- The spectators: in field hockey, they often are very close to the field.
- The players: These include the field players and the goalkeepers.
- The referees: in field hockey, there are always two referees on the field.
- The ball : Tracking it enables extraction of many key insights. An example is given in dedicated Section [3.6](#).
- The ball patrol: They are at the pitch's edge and are not always present.

Detection and tracking are quite challenging tasks in [CV](#), as conditions significantly and directly impact the algorithm's performance. These challenges can be divided into two categories: the systemic to the [Multi-Object Tracking \(MOT\)](#) step and those depending on the world environment.

Systemic challenges:

1. **Occlusion:** This happens when a target object is (at least partially) temporarily hidden. It is particularly recurrent in team sports. In [MPT](#), we typically refer to *fragmented occlusion*.
2. **Erratic motion:** Players in field sports often change their direction during the game, making it harder to predict their position for tracking tasks.
3. **Visual similarity:** When multiple objects in the scene share similar appearance or motion characteristics, it becomes more difficult to discriminate between the different tracked objects. In field hockey, players from the same team wear the same uniform color.
4. **Deformations:** Between frames, the tracking identity might be lost because the players' positions have changed. For instance, they could be crouched on the field, bend over to reach the ball, or stand up straight in other cases.

World environment related challenges:

The conditions in which the scene occurs also have a substantial impact on [MPT](#) accuracy. Here below are most environmental conditions that can affect the [MOT](#) accuracy:

1. **Motion blur:** Depending on the camera quality, the motion of the players and the camera itself can cause the objects to become blurred. It results in the loss of important details and features of moving objects,
2. **Viewpoint variation:** Proper procedures should be established in case of changes in the camera's views, which can lead to a decrease in detection and tracking performance.
3. **Scale change:** Since there is a variation in the size of the bounding boxes of detected players when zooming in/out occurs, it may cause discrepancies in the perceived object size.
4. **Background clutter:** When the background color is similar to the objects in question. Then, the algorithm is more likely to consider players as background, losing its detection and tracking capabilities.
5. **Illumination variation:** They can greatly influence the appearance of the objects. This is one of the biggest problems in MOT tasks.
6. **Changes in the resolution:** They may result in inconsistencies in object size, distort spatial relationships, and alter the quality and detail of detected features, thus impacting the accuracy of the object detection and tracking processes.
7. **Objects size:** small object detection is more challenging. Since less information is associated with small objects, they can be confused with the background, which is a problem of player detection.

All these factors introduce multiple problems that decrease detection and tracking performance. Especially in MOT, this often results in *identity switching*. Once two similar objects have crossed each other, which often occurs in field hockey, there is a risk of uncertainty about the object's ID. Moreover, there are challenges related to *trajectory initialization and termination*. This is the process of starting a new trajectory for an object that has not been previously detected or tracked, and the process of ending a trajectory for an object that is no longer visible in the scene, respectively.

In terms of algorithmic procedure, the [MOT](#) step can be approached from three angles:

1. **Tracking by detection:** Initially, the targets are detected in each frame of the video sequence, cut according to the bounding box, thereby obtaining all the targets in the image. This process is then transformed into a problem of target correlation between successive frames. A similarity matrix is constructed and solved for the most likely matches. As such, the tracking performance highly depends on the efficacy of its object detection. Most detection algorithms are based on architectures like [You-Only-Look-Once \(YOLO\)](#), [RetinaNet](#), and [Single-Shot Detector \(SSD\)](#), while tracking algorithms are often based on [Simple Online and Realtime Tracking \(SORT\)](#) and [DeepSORT](#) algorithms. Tracking by detection constitutes the majority of the current solutions in the industry.
2. **Joint detection and tracking:** These techniques show significant promise because they manage to provide a unified framework, which often reduces the complexity. This framework has been established in [[Feichtenhofer et al., 2018](#), [Chen et al., 2018](#), [Zhang et al., 2021](#), [Zhou et al., 2020](#)].
3. **Attention mechanisms:** While they offer high accuracy, their real-time usability is contingent upon the capabilities of the hardware in use. See [[Sun et al., 2021](#), [Meinhardt et al., 2022](#)] for a description of the framework.

At the time of writing, YOLO_DeepSort appears to be the most widely used framework for [MPT](#). As such, it is an appropriate candidate for this thesis.

3.4.2 Multi Players Detection

Many persons' detection algorithms are benchmarked on pedestrian videos. However, player detection is a more complex problem due to the challenges mentioned earlier [[Manafifard et al., 2017](#)]. Formally, object detection can be seen as a mapping function f that takes an image $\mathbf{I}$ as input and outputs a set of bounding boxes $\mathbf{B}$ and associated labels $\mathbf{L}$:

$$(3.7) \quad f : \mathbf{I} \rightarrow (\mathbf{B}, \mathbf{L})$$

Prior to the advent of [Deep Neural Networks \(DNN\)](#), numerous techniques have been used and remain vital in today's [CV](#) pipeline. Here are a few:

- **Viola-Jones** object detection framework [[Viola and Jones, 2001](#)]: This method extracts features relative to a specific class and trains a binary classifier to recognize them. This requires running as many classifiers as there are classes. Originally used for face recognition, its performance decreases drastically as the number of classes increases. Moreover, it assumes similar object positions and is highly sensitive to occlusion, rendering it unsuitable for complex scenes such as in field hockey.

- **Scale-Invariant feature transform (SIFT)** [Lowe, 1999]: The **SIFT** algorithm detects key-points or interest points in an image during training that are invariant to scale, orientation, and illumination changes. During testing, these points are compared to determine the closest matching object. While it's technically a feature descriptor rather than a detector as it doesn't output classes, it remains useful. For instance, see [Tsourounis et al., 2022] where **SIFT** is used with Deep **Convolutional Neural Networks (CNN)** for image classification.
- **Histogram of oriented gradients (HOG)** features [Dalal and Triggs, 2005]: This method divides an image into a grid of small cells and computes histograms of gradient orientations in each cell. The normalized histogram values from all cells are concatenated to form a feature vector that represents the image and can be used for object detection and tracking. However, it is highly sensitive to occlusions, complex backgrounds, and changes in shape.

Even though these techniques have been largely overtaken by **DL** methods, particularly **CNN**, they remain important as foundational principles and can even be combined. See for instance [Kalake et al., 2022] where the **HOG** descriptor simplifies the structural information of objects in video frames before the extracted features are passed into the **CNN** for further processing and classification.

In regard to **DL** techniques for **MOT**, there are two types of detector architectures: one-stage and two-stage detectors. In two-stage detection, a region proposal (i.e., coordinates) is generated before classification. Conversely, one-stage detectors apply to a regular, dense sampling of object locations, scales, and aspect ratios simultaneously. In terms of inference, one-stage methods prioritize speed. The most well-known one-stage algorithms include **YOLO**, **SSD**, and **RetinaNet**. In contrast, two-stage methods prioritize detection accuracy, with examples including **Faster Region-based CNN (Faster R-CNN)**, **Mask R-CNN**, and **Cascade R-CNN**. Since the goal is to extract real-time insights, as discussed in Part I of this thesis, one-stage detectors are a more natural choice. These algorithms have been extensively studied and improved over time. For example, **YOLO** has seen many iterations and enhancements since its first release in 2015 [Redmon et al., 2015].

The architecture of **YOLO**, facilitated by a single **CNN**, makes it highly efficient during inference time. In its earlier versions, **YOLO** made more location errors than **Faster R-CNN**, although it was better at avoiding background errors. It also had lower accuracy than **SSD** [Tan et al., 2021]. Subsequently, it has been able to nearly match the accuracy reported by **Faster R-CNN** and **SSD**, while maintaining its speed advantage. Overall, it often provides a better trade-off between accuracy and speed compared to **SSD** and **RetinaNet** [Tan et al., 2021]. It's worth noting that if objects occupy less than 10% of an image, **YOLO** performs better than **SSD** [Pham et al., 2017]. The **RetinaNet** architecture has a problem with small target detection in aerial images [Tian et al., 2020]. Furthermore, while **YOLO** is a large neural network with millions of parameters, **SSD** has two orders less parameters [Wang and Li, 2022]. This implies

that the supporting device must be powerful enough or utilize cloud-computing capabilities [Wang and Li, 2022]. Moreover, YOLO algorithms are preferable when there are significant movement changes in real-time, as in football [Han et al., 2022]. Under such conditions, it tends to offer the best speed-accuracy trade-off. Consequently, the YOLO architecture has been selected.

3.4.3 You Only Look Once (YOLO) - Theoretical background

YOLOv1

As shown in Figure 3.8, YOLO consists of a single deep CNN with fully connected layers at its ends, which performs detection and classification at the same time.

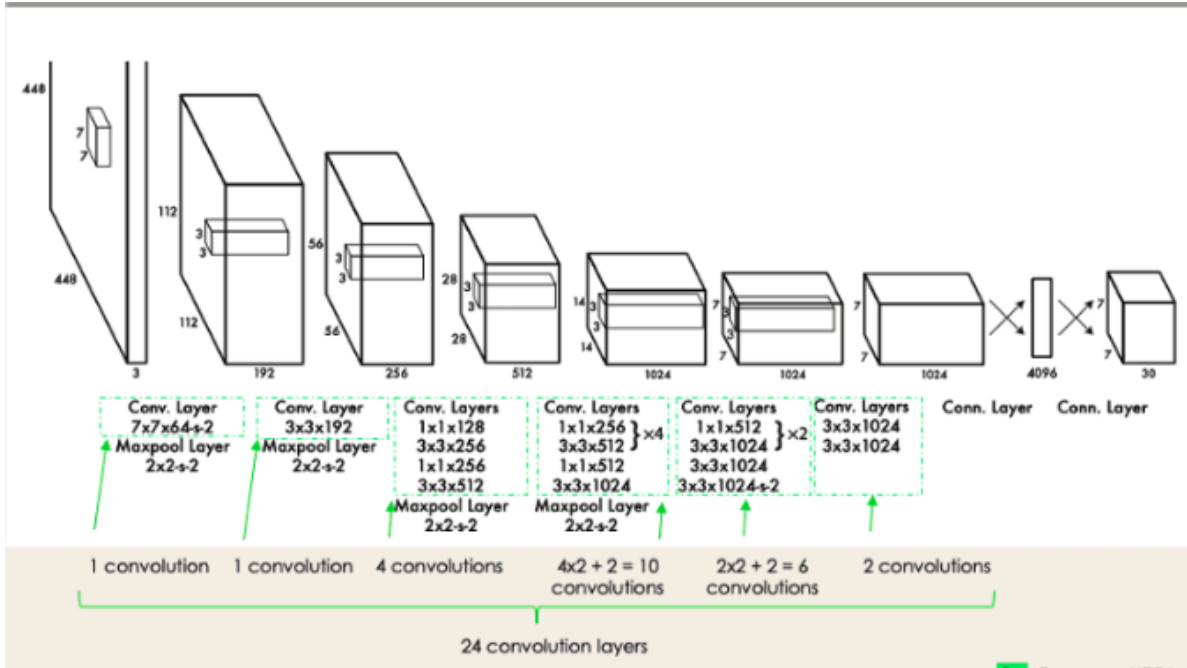


Figure 3.8: YOLOv1 architecture

The YOLO network accepts an image that has been resized to 448x448 as input, which it then divides into a $S \times S$ grid. If an object's center falls within a grid cell, that cell is responsible for detecting the object. Each grid cell predicts K bounding boxes, as well as confidence scores for those boxes for each class.

For example, Consider a example with $K = 5$ bounding-boxes and $C = 80$ classes:

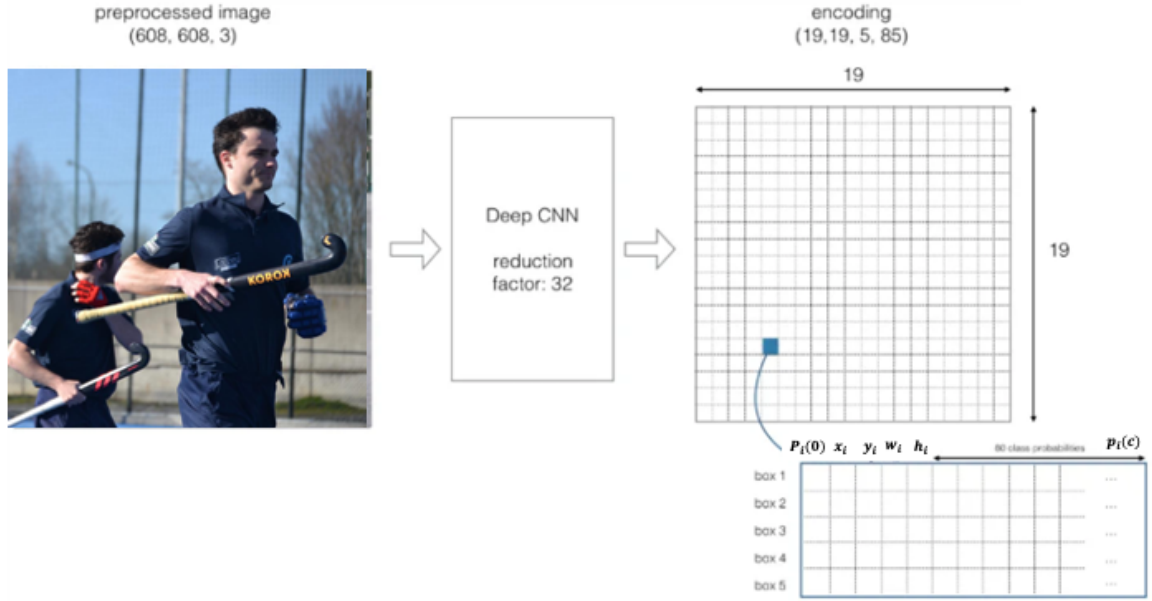


Figure 3.9: Grid cell partition of an input image in YOLO algorithm

As shown in bottom left of above example, the j^{th} bounding box in the i^{th} cell must have the following format:

- $P_i(0)$: Probability the box contains an object.
- (x_i, y_i) : The center of the box coordinates.
- w_i : The width of the box.
- h_i : the height of the box.
- $p_i(c)$: The confidence score of an object of class c_i being within the box.

During training, the confidence scores are computed in the following ways :

$$(3.8) \quad p_i(c) := P_i(0) \times \text{IoU}_{\text{pred}}^{\text{truth}}$$

Intersection-over-Union (IoU) is the overlapping area between ground truth bounding box and prediction bounding box divided by their union area. This means the confidence score is based on conditional probability. Hence, at inference time, the class specific confidence score for each bounding box can be given by:

$$(3.9) \quad \begin{aligned} & \Pr(c|\text{object})\Pr(\text{object}) \\ &= \Pr(\text{class}_i) \times \text{IoU} \end{aligned}$$

The output dimension, therefore, is $S \times S \times (K + 5 + C)$. From this, we can infer that images are divided into grid cells, and the model predicts bounding boxes for each of these, accompanied by an object confidence score. If this score is lower than a certain threshold, it is considered as background. As such, in the YOLO framework, there isn't an explicit Background class.

The Loss function is designed with ease of optimization in mind. As a result, it comprises multiple **Sum of Squared Errors (SSEs)**, given that its gradient is of order 1. It conveys very compact information, which is what makes it so fast at inference time. The loss function can be broken down into three components.

$$(3.10) \quad \mathcal{L}_{\text{YOLOv1}} = \mathcal{L}_{\text{Box}} + \mathcal{L}_{\text{Objectness}} + \mathcal{L}_{\text{Classification}}$$

1. **Coordinate Loss (Box Loss):** YOLO penalizes errors in the predicted locations (x, y) and sizes (width, height) of bounding boxes for objects, encouraging the network to predict better boxes.

$$(3.11) \quad \begin{aligned} \mathcal{L}_{\text{Box}} = & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\ & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \end{aligned}$$

λ_{coord} and λ_{noobj} are tunable parameters allowing importance weight to the Localization loss and if there are no-object loss, respectively. As there are more positive than negative examples in practice, it is necessary to allow more weight to λ_{coord} than λ_{noobj} : namely 5 and 0.5 [Redmon et al., 2015].

$\mathbb{1}_{ij}^{\text{obj}}$ is the indicator function that equals 1 if there is an object in the i^{th} cell and the j^{th} box, and 0 otherwise.

$\mathbb{1}_{ij}^{\text{noobj}}$ is the indicator function that equals 1 if there is no object (i.e., the background) in the i^{th} cell and the j^{th} box.

Note that the square root of w_i and h_i are taken because it is less sensitive to changes in larger boxes and helps achieve faster convergence.

2. **Objectness Loss:** This component measures the error related to the confidence of an object's presence in a predicted bounding box. It includes two parts: one penalizes incorrect confidence predictions when an object is actually present in the box, and the other penalizes incorrect confidence predictions when no object is present in the box.

$$\begin{aligned}
 \mathcal{L}_{\text{Objectness}} = & \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
 & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2
 \end{aligned}
 \tag{3.12}$$

$C_i - \hat{C}_i$ is the difference of the confidence. It is good practice to assign a [SSE](#) instead of a generic Cross-Entropy (CE).

3. **Classification Loss:** If an object is present in a cell, [YOLO](#) penalizes errors in the predicted class probabilities.

$$\mathcal{L}_{\text{Classification}} = \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
 \tag{3.13}$$

$(p_i(c) - \hat{p}_i(c))^2$ is the [SSE](#) of the probabilities. Cross-Entropy (CE) could have been used, but [SSE](#) was found to be easier to optimize.

YOLOv2:

The main novelties of the second version of **YOLO** along with their effects are depicted in Figure 3.10.

	Modification	Effect
Bounding Boxes	Anchor Boxes	7% recall increase
	Dimension clusters + new bounding box parameterization	4.8% mAP increase
Architecture	New Darknet-19 replaces GoogLeNet	33% computation decrease, 0.4% mAP increase
	Convolutional prediction layer	0.3% mAP increase
Training	Batch normalization	2% mAP increase
	High resolution fine-tuning of weights	4% mAP increase
	Multi-scale images	1.1% mAP increase
	Passthrough for fine-grained features	1% mAP increase

Figure 3.10: YOLOv2 improvements compared to YOLOv1

Darknet-19 serves as the feature extraction backbone, offering increased depth and computational efficiency. Another important improvement introduced in YOLOv2 is the use of anchors, which replace the previous approach of generating multiple bounding boxes. Anchors are predefined boxes assigned to specific feature points in the final layers, encompassing varying sizes and aspect ratios to aid in object localization. They have been introduced in the two-stage detector **Faster R-CNN** presented in [Khan et al., 2019]. Accordingly, the loss function becomes:

$$\begin{aligned}
 \mathcal{L}_{\text{YOLOv2}} = & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
 & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
 & + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} (C_i - \hat{C}_i)^2 \\
 & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
 & + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
 \end{aligned}
 \tag{3.14}$$

Note that the Loss function is not explicitly given in original paper from [Redmon and Farhadi, 2016]. Moreover, the double sum is absent because the K predictors per grid cell was replaced by the concept of anchor boxes.

YOLOv3

First, it utilizes Darknet-53 as a backbone. This architecture is larger and deeper compared to YOLOv2's Darknet-19, adding complexity and size to the model. This implies that it is computationally more demanding than its predecessor. Another novelty of YOLOv3 is the use of a **Feature Pyramid Network (FPN)** and residual connections to enhance feature extraction. Its architecture can be illustrated as follows:

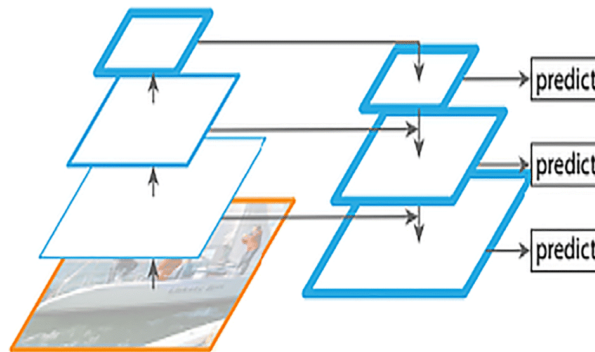


Figure 3.11: **FPN** architecture in YOLOv3 algorithm

The feature extraction part consists of a ResNet and, on top of that, a **FPN**. In the ResNet network, there is a bottom-up pathway where the resolution of the initial input decreases while higher-level structures are extracted. The **FPN** assists in reconstructing higher resolution layers through a top-down pathway. The network also includes lateral connections that efficiently construct a rich multi-scale feature. This architecture allows the entire pipeline to be more efficient as a single image input is necessary to capture multi-scale features. Moreover, YOLOv3 utilizes more anchor boxes, which are determined by running k-means clustering on the training set.

Overall, it further improves the detection of smaller objects and decreases localization error, while requiring more computational resources. Since then, incremental improvements have been proposed in subsequent **YOLO** versions (see Appendix Figure A.3 for details).

Moreover, many propositions are not official. At the time of writing, the most commonly used versions is YOLOv7, but YOLOv8 has just released.

YOLOv7 introduces several key enhancements compared to its predecessors. One of its advancements includes implementing a novel efficient layer aggregation technique which helps optimize convolutional layers for faster inference speeds. Furthermore, new model scaling techniques have been incorporated that consider network depth and width allowing optimization across varying sizes. Finally, YOLOv7 introduces new re-parameterization planning techniques incorporating using gradient flow propagation paths along with an auxiliary head for intermediate supervision and unique coarse-to-fine backpropagation methods.

On the other hand, YOLOv8 adopts a new anchor-free model with a distinct head for each task - objectness, classification, and regression, thereby boosting accuracy through focused task handling. The output layer uses sigmoid and softmax functions for scoring objectness and class

probabilities, respectively. Furthermore, it employs CIoU and DFL loss functions, combined with binary cross-entropy, significantly enhancing object detection performance, particularly for smaller objects.

These models are used on the initial footage example in Figures 3.12 and 3.13. One can see the big difference in quality between YOLOv5 and YOLOv8 in Figures As expected, YOLOv8 seems to be way more efficient at locating more distant objects. Yet, YOLOv5 remains relevant since it can run almost in real-time on CPU.

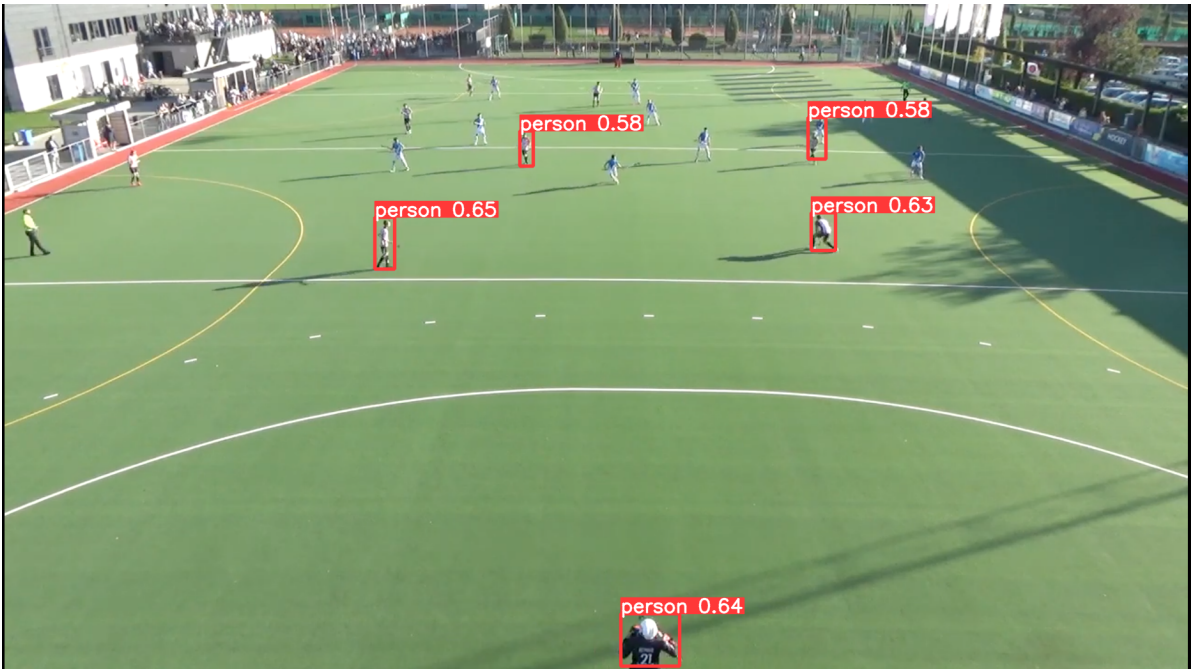


Figure 3.12: YOLOv5 detections on field hockey frame



Figure 3.13: YOLOV8 detection on field hockey frame

3.4.4 Multi Player Tracking

3.4.4.1 Introduction

One cannot rely solely on detection to track objects, as it is also necessary to maintain their identities across different frames. Many tracking algorithms exist, each with properties that make them more or less suited to multi-player tracking in field hockey. The most straightforward and well-known is the **Meanshift** algorithm [Cheng, 1995]. Essentially, it is a distribution density mode estimator that uses the color or shape histogram of the targets as input. Consequently, it is not well-suited for **MOT** in sports, as the targets have very similar shapes and colors. Another renowned algorithm is based on **Particle filters** [Okuma et al., 2004]. This algorithm utilizes the direction and velocity of the objects over time to predict the trajectory of the object in question, see e.g., the *Lucas Kanade* algorithm in [Mavrogiannis and Maglogiannis, 2022]. Yet another type of algorithm relies on the so-called **Kalman filter**, extensively used in **MOT** in sports and field hockey, see e.g., [Duarte Moura et al., 2022, Csanalosi et al., 2020]. This filter has been adapted to the tracking-by-detection framework with the **SORT** algorithm, where it becomes very effective.

The general timeline for tracking algorithms is as follows:

1. **Object detection:** Detection of objects in the new frame. This has been extensively discussed in Section 3.4.
2. **Data association:** Matching the currently detected objects with the previously detected

ones.

3. **Prediction:** Estimating the future positions of the tracked objects using motion models or historical information.
4. **Update:** Incorporating the current measurements to refine the current track estimation.

Meanwhile, the implementation of a robust track maintenance system is crucial. After detecting a new object, the algorithm initializes the corresponding target. This process involves creating a new track, with the initial position (x, y) derived from the detected bounding boxes. Furthermore, it becomes imperative to establish an effective framework for track deletion to handle cases where tracks fail to make associations.

3.4.4.2 Tracking using SORT

Figure 3.14 shows the procedure followed by the algorithm.

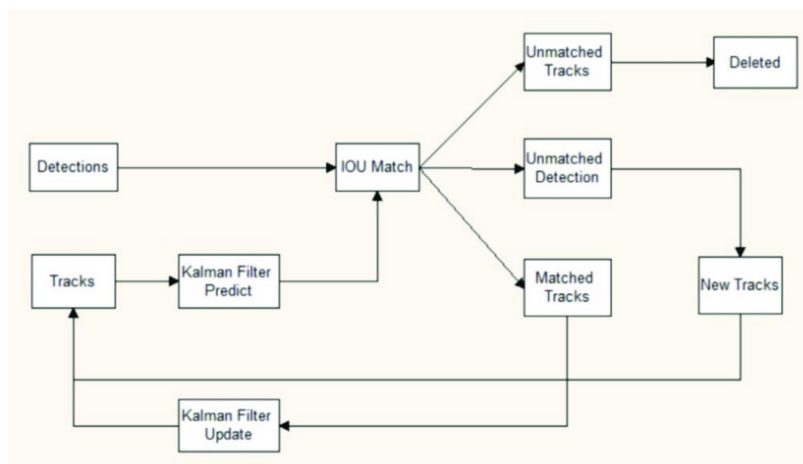


Figure 3.14: Object tracking procedure of **sort** [Bewley et al., 2016]

The different steps are depicted here below:

1) Detection: This is the detection part explained earlier, see Section 3.4.2

2) Estimation: This is where the Kalman filter is utilized. It propagates the detection coordinates from the current frame to the next one using a motion model. In practice, it estimates the next position and velocity of each object in the scene. The detected bounding boxes are used as measurements. Formally, as shown in Equation (3.15), the state vector $\mathbf{x}$ for each object consists of its position (x, y) and velocity (v_x, v_y) .

$$(3.15) \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}$$

The assumption here is that the state process depends solely on the previous state, in which the predictions (termed *a priori estimations*) and measurements (or *measurement estimates*) are both subject to Gaussian noise. The Kalman filter optimizes the object's position by determining the optimal gain that minimizes the a posteriori estimate variance. In other words, the Kalman filter is a two-step process that finds the best trade-off between process estimation and measurement estimation, while minimizing the covariance. Note that, if using YOLO format data, x vector becomes $x = [c_x, c_y, w, h, v_x, v_y, v_w, v_h]$ but the logic remains intact.

If we model a 1-dimensional example where we want to estimate the position x at time $t+1$ can be illustrated as follows:

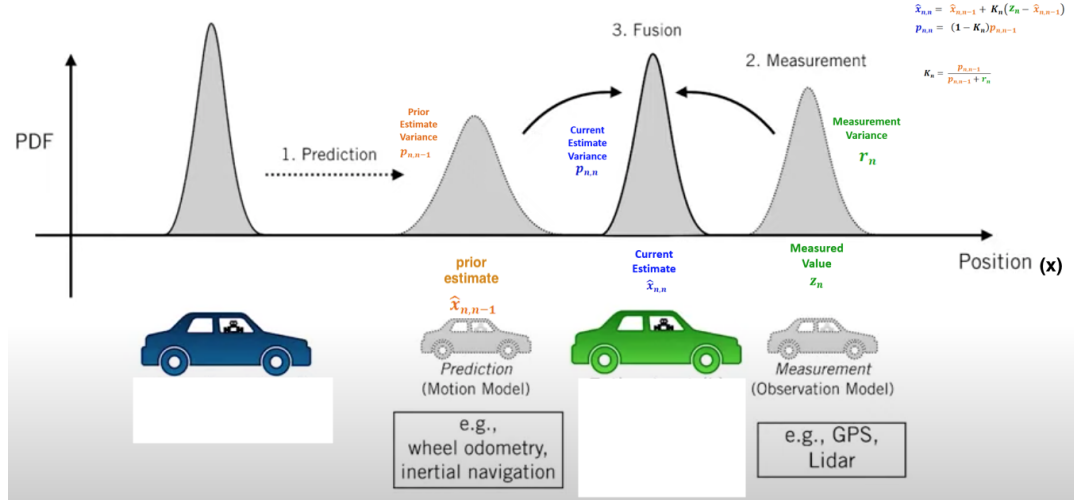


Figure 3.15: The Kalman filter: prediction and correction algorithm

On the other hand, it can be formalised in the following way:

(A) **Prediction Step:**

a) State prediction:

$$\mathbf{x}_{\text{pred}} = \mathbf{F} \cdot \mathbf{x}_{\text{prev}}$$

b) Covariance prediction: it is computed by linear transformation of a random variable and a noise diagonal matrix:

$$\mathbf{P}_{\text{pred}} = \mathbf{F} \cdot \mathbf{P}_{\text{prev}} \cdot \mathbf{F}^T + \mathbf{Q}$$

- $\mathbf{x}_{\text{pred}}$: Predicted state vector for the object
- $\mathbf{x}_{\text{prev}}$: Previous state vector for the object
- $\mathbf{F}$: State transition matrix, which models the system dynamics. In our settings,

this is a 4x4 matrix: $\mathbf{F} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ The dt terms allow to account for the velocity that cannot be directly measured.

- $\mathbf{P}_{\text{pred}}$: Predicted state covariance matrix for the object
- $\mathbf{P}_{\text{prev}}$: Previous state covariance matrix for the object
- $\mathbf{Q}$: Process noise covariance matrix

(B) **Update Step:** The Kalman filter takes as inputs two potential states, and produces a merged state that incorporates information from both of them, both subject to Gaussian noise.

a) Kalman gain:

$$\mathbf{K} = \mathbf{P}_{\text{pred}} \cdot \mathbf{H}^T \cdot (\mathbf{H} \cdot \mathbf{P}_{\text{pred}} \cdot \mathbf{H}^T + \mathbf{R})^{-1}$$

- **K:** Kalman gain
- **H:** Measurement matrix, which models the relationship between the state and the measurements. Here, it is a 2x4 matrix: $\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$
- **R:** Measurement noise covariance matrix

Since we have a prior distribution (prediction based on previous state) and a likelihood distribution (the measurement), the optimal gain can be obtained using Bayesian inference. Their product results in a Gaussian distribution, and so the posterior can be optimised so that the variance is minimised.

b) State update:

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{pred}} + \mathbf{K} \cdot (\mathbf{z} - \mathbf{H} \cdot \mathbf{x}_{\text{pred}})$$

c) Covariance update:

$$\mathbf{P}_{\text{new}} = (\mathbf{I} - \mathbf{K} \cdot \mathbf{H}) \cdot \mathbf{P}_{\text{pred}}$$

- **z:** Measurement vector (e.g. bounding box center coordinates)
- **x_{new}:** Updated state vector for the object
- **P_{new}:** Updated state covariance matrix for the object
- **I:** Identity matrix with the same dimensions as P_{pred}

3) Data association: It consists in associating the detections to existing trackers depending on an associating cost. Those can be represented in a **Bipartite Graph**. See here below an **MOT** example:

	Tracking 1	Tracking 2	Tracking 3
Detection 1	21	10	0
Detection 2	56	62	11
Detection 3	20	77	90
Detection 4	10	80	20

Figure 3.16: Assignment problem representation as a Bipartite Graph. Cells numbers are the associated cost between the detections and trackers

Various cost functions could be used, such as the Euclidean distance. If a detected and tracked object are close to each other, they are likely to be the same object. This might work in very simple cases but not in **MOT**, especially when numerous occlusions or shape changes occur. Therefore, **SORT** uses **IoU**. It constitutes the proportion of overlap between the detections and the tracks. Here, the intuition is that if the overlap is high, then it is likely to be the same object. If so, the new observation is associated with that specific track.

The **Hungarian algorithm** (also known as **Kuhn-Munkres** algorithm) is used to solve this problem with complexity $O(n^3)$. In **MOT**, one must take into account the fact that objects can disappear while others can appear. This leads to a squared matrix (as shown in Figure 3.16 where an additional object appeared). Hence, a dummy column is added, which includes the maximum overlapping score. Moreover, since the Hungarian approach is a minimization algorithm, the scores must be transformed by subtracting them from the maximum.

4) Deletion of Tracking Identities: When dealing with the monitoring of dynamic scenarios like the one described in this work, the appearance and disappearance of players in the scene can be quite frequent. Ideally, a system should be able to re-identify an object that had previously disappeared. However, the **SORT** algorithm does not support such behavior. Instead, it creates a track with null velocities and a large variance. On the other hand, the tracks are terminated when they are not detected for more than T_{lost} frames.

3.4.4.3 Tracking using DeepSort

By definition, **SORT** uses box locations to infer their identities. This approach leads to poor performance when there are a lot of occlusions and when, as in hockey, the shape of the bounding

	Tracking 1	Tracking 2	Tracking 3	Tracking 4
Detection 1	21	10	0	90
Detection 2	56	62	11	90
Detection 3	20	77	90	90
Detection 4	10	80	20	90

	Tracking 1	Tracking 2	Tracking 3	Tracking 4
Detection 1	69	80	90	0
Detection 2	34	28	79	0
Detection 3	70	13	0	0
Detection 4	80	10	70	0

$80 = 90 - 10$

Figure 3.17: Bipartite graph representation in MOT. LHS graph allows to account for additional detected objects. RHS graph accounts for the minimisation problem requirement

boxes frequently changes due to players' movements. Therefore, rather than using the **IoU**, one can consider the appearance cues of the detected objects. This approach somewhat addresses the identification switches problem, although it imposes a higher computation cost, which increases as the number of the tracked objects increases. For this reason, we define another distance metric based on the work presented by [Wojke et al., 2017], where the final classification layer is removed, and the features from the last fully connected layers (called appearance feature vector) are considered (an illustration is provided in appendix Figure A.4). The appearance vector provides an additional valuable source of information for tracking the objects. The associated cost necessitates the definition of an *appearance cost*, which is the cosine distance between the detected and tracked object in the feature appearance space. Details of the **SORT** and **DeepSORT** algorithms are illustrated in Appendix Figure A.18. In this work, **DeepSORT** is used.

3.4.5 Ball detection and tracking

Extensive literature addresses ball detection and tracking due to its associated challenges. For instance, [Komorowski et al., 2019] concentrates on detecting balls in long shot football videos. However, no significant advancements have been made in the context of field hockey. Undertaking such a task presents considerable difficulties due to the high velocity and size of the hockey ball (with a diameter ranging from 7.13 to 7.48 cm)³. Despite these challenges, it may be beneficial to draw insights from related fields like football, where the ball's movements exhibit similar patterns, or even from sports such as tennis and golf that share similarities in terms of ball size and speed.

³<https://www.networldsports.co.uk/buyers-guides/hockey-ball-guide>

3.5 Player Clustering and team classification

3.5.1 Introduction

This step is critical as most insights and statistics derived from video rely heavily on the assignment of players to their respective teams [Mazzeo et al., 2010]. In this thesis, the goal is to cluster players from one team against the other. However, in field hockey, the scene includes many participants. They can be broken down as follows:

- **TeamA and TeamB** players: The **GoalKeeper** and the **FieldPlayer** categories, despite being on the same team, wear different jerseys and have a distinct appearance (see the figure below).
- **Referees**: There are exactly two referees on the pitch in field hockey. They can be found on or just outside the field edges.
- **Others**: This category can include coaches, ball patrols, the public, etc.

Ideally, it should be possible to assign a unique identifier to each player, as every player is uniquely identified by their team's jersey and number. However, this is often difficult, if not impossible, due to the video quality. Furthermore, as discussed in Part II, insights are expected to be extracted in real-time without any human intervention. When human intervention is involved, prior available information could simplify the problem. For example, [Csanalosi et al., 2020] suggest using the dominant jersey color as an input to create a mask. Another possibility is supervised clustering. [Lu et al., 2013] propose training a logistic regression classifier among *TeamA*, *TeamB*, and *Others* based on RGB color histograms with a 97% accuracy rate. A third alternative, presented in [Duarte Moura et al., 2022], uses a CNN classifier on field hockey video frames with 98% accuracy.

In practice, one could look at the following information to determine team membership and referees:

- **Jersey's color**: This is traditionally the most used feature. It will be discussed further in the next section.
- **Players location**: The absolute position of the detected people with respect to the temporality. [Xu et al., 2005] propose a Bayesian methodology on football footage. Field players have uniform prior probability across the entire pitch while the keepers are heavily constrained to their rectangle. This could be translated to field hockey with goalkeepers confined to the semi-circle around their goal.
- **Visual cues**: The fact that players carry a hockey stick while the referee does not could be a factor. Another cue is that the goalkeepers wear completely different equipment than the

others and the referees hold a whistle in their hand. This information has yet to be utilized in the literature. This will be discussed in Subsections 3.5.4.

The next sections will present a literature review on player clustering and the specific cues that can be used to cluster the players on a field hockey pitch. It is proposed to cluster according to the following groups: **TeamA**, **TeamB**, **GoalKeepers** and **Referees**.

3.5.2 Traditional techniques

Intuitively, color seems to be a reasonable starting point for inferences. We expect to define five clusters: one for each team, one for both referees, and one for each goalkeeper. [Mavrogiannis and Maglogiannis] experimented with **color dominance** and **color histogram** for feature extraction. These two preprocessing techniques will be explained in this section, followed by an overview of traditional clustering methods.

Preprocessing

One significant source of color noise arises from the background present in the bounding boxes. Therefore, applying a mask to remove this effect is necessary. Although this sounds simple, in practice, changes in brightness, due to shifts in sun orientation, rain, snow, and so on, can have a substantive impact. Despite its significant influence and common occurrence, this effect hasn't been extensively researched. Ideally, a dynamic background subtraction technique that does not require prior color information would be beneficial. Another essential decision is whether to use RGB or HSV color space. Generally, HSV color space seems to perform better, as evidenced in [Mavrogiannis and Maglogiannis, 2022, Duarte Moura et al., 2022].

Color dominance: Let's define N as the number of dominant colors per bounding box ($N \geq 1$). After this, a K-means clustering algorithm (with k fixed to N) is applied. The N extracted dominant colors are then sorted according to their degree of dominance. Figure 3.5.2 illustrates an example of dominant color extraction.

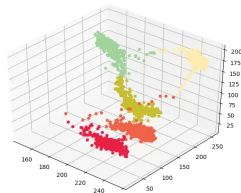


Figure 3.18: RGB projection of image and segmentation using K-means with K=5



Figure 3.19: The resulting 5 sorted dominant colors

Color histogram: Any frame can be represented using a color histogram, which shows the number of pixels corresponding to a specific level of RGB or HSV. Figure 3.20 provides an example.

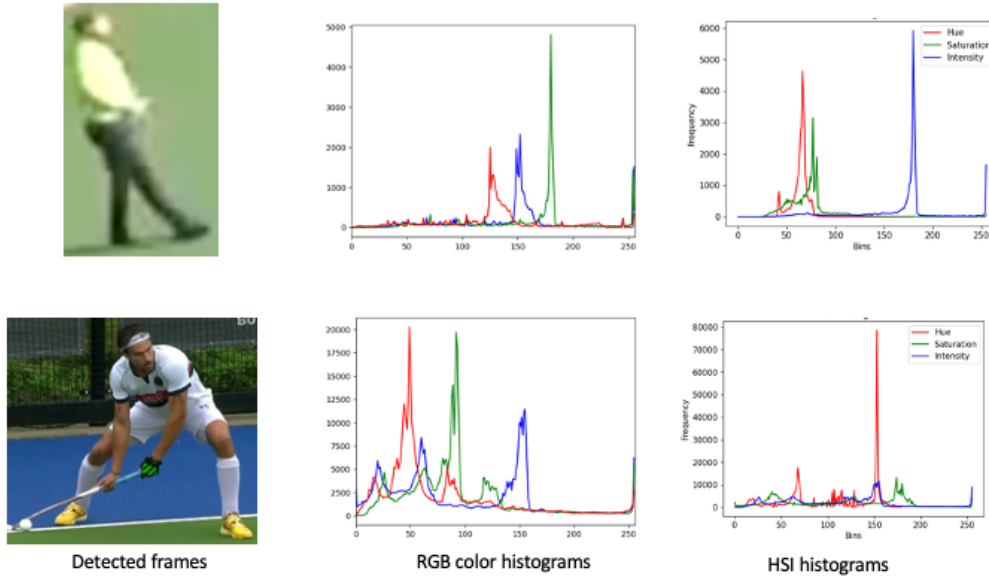


Figure 3.20: RGB and HSI Color histogram applied to an object in initial frame (above) and field player in field player (below)

Clustering techniques

It is first important to differentiate between **soft** and **hard clustering**. Soft clustering does not predetermine the number of clusters, while hard clustering defines the number of clusters in advance. In field hockey (and other sports such as football), there is a debate on how to best approach this issue. Theoretically, the number of clusters is fixed (refer to earlier proposed groups). Hence, **K-means** algorithm could be used, as done in [Mazzeo et al., 2010]. However, in practice, it doesn't make much sense to assign a group to only one or two individuals (one for each goalkeeper and one for the two referees since they wear different jerseys). To address this, [Duarte Moura et al., 2022] propose to consider it as a soft clustering task. Specifically, propose the use of the so-called **Density-Based-Spatial Clustering of Application with Noise (DBSCAN)** algorithm to distinguish between TeamA, TeamB, and non-players, with an accuracy of 69.2%. As such, these two techniques along with their respective preprocessing are studied in this work.

Hard clustering via K-means: To enhance clustering performance, feature engineering is vital. Ideally, features should be scale and rotation invariant, while also being executable in real-time. From this perspective, [Mazzeo et al., 2010] propose the following methodology:

1. Players detection

2. For each player:

- a) **Background subtraction:** Apply a mask to remove background (i.e., pitch) effect.
- b) **Transformed RGB histogram:** Each channel is standardized so that each channel has a mean $\mu = 0$ and standard deviation $\sigma = 1$.

3. K-means clustering: fix number of clusters K to 5.

Soft clustering via DBSCAN: More precisely, **DBSCAN** is a Density-based clustering technique. It requires fixing a distance function (e.g. Euclidean) and two parameters:

- ϵ : It defines a parameter specifying the radius of a neighborhood with respect to some point.
- *MinPoints* : It defines the minimum number of points to consider a point as a core point within its neighborhood. It is often defined as twice the dimension [Sander et al., 1998].

First, the DBSCAN algorithm checks each data point to determine if it is a **core point**. This is done by checking if there are at least *MinPts* points within a neighborhood determined by a radius of ϵ around the given point. Then, starting from a random core point, all directly reachable core points are added to the same cluster. This process is iterated until no more core points can be added to the cluster.

At this stage, all points in the cluster are core points. Subsequently, all non-core points, which are directly reachable from the core points in the cluster, are added to the nearest cluster. These non-core points, also known as **border points**, can join the cluster but do not have the capability to extend it.

After all the core points have been assigned to clusters, and all border points have been included in their nearest clusters, any remaining points are considered as **outliers**. These points do not belong to any cluster.

In practice, a burn-in period is necessary during which the optimal hyperparameters are sought to produce 3 clusters (TeamA, TeamB, and outliers (i.e., goalkeeper and referees)).

1. Players detection**2. For each player:**

- a) **Background subtraction:** Apply a mask to remove background (i.e., pitch) effect.
- b) **HSV Histogram Normalization:** Normalize each dimension of the HSV histograms.

3. DBSCAN clustering: Use a greedy approach to find the optimal parameters that create two clusters.

On the one hand, **DBSCAN** does not require any previous training. It is simple and runs quickly with a short burn-in period (i.e., processing time before outputs are computed), making it suitable for real-time applications. On the other hand, due to its nature, it only works when both teams are visible in the image, and each of them significantly outnumbers non-players (i.e., goalkeepers and referees). As seen in the results, it is highly sensitive to lighting variations, occlusion, and when teams wear similar jerseys. Ultimately, at its best, it only classifies TeamA, TeamB, and outliers (that include referees and goalkeepers). That explains why K-means clustering is still being investigated (see, e.g., [Mazzeo et al., 2010]) even though some clusters do not make much sense. This is where one could use visual cues of the field hockey players. Indeed, if the Referees and Goalkeepers are detected upstream, the clustering step would only be between two groups. This track is explored in more detail in Subsection 3.5.4.

The results are shown for Kmeans and **DBSCAN** in Table 3.1, and Appendices Figures A.6, A.7, A.8 . The performance is unsatisfactory, with a global accuracy of 56% and 59%. However, this could be due to the overall frame resolution and lighting conditions. Further exploration of this algorithm with more labeled data could provide a more robust performance assessment analysis.

Model	Accuracy	Precision	Recall	F1-score
DBSCAN	0.59	0.60	0.59	0.58
K-means	0.56	0.56	0.56	0.55

Table 3.1: Clustering techniques Evaluation on the initial frame

3.5.3 Deep learning methods

As often now, **DL** might be the solution to that kind of task. [Istasse et al., 2019] propose to build a **CNN** to make a latent space representing players of same team. [Koshkina et al., 2021] use the same idea but propose to use a **CNN** that has been previously trained by so-called *contrasting learning*. It uses a triplet loss, which means there is an anchor, a positive and a negative examples so the network learns to minimise the distance in the embedding space from the anchor to the positive image while maximising the distance to a negative example. This allows to build a latent space from which a k-means algorithm is used to discriminate between TeamA and TeamB. This gives an algorithm that requires a certain number of *burning images* in order to make the clustering. It should be noted that referees are not fed to the pipeline because, in their examples, all of them wear always the same jersey. The proposed technique seems very interesting given the fact that it exploits all the information one can think about when classifying teams. To go even further, it can be noted that a new architectural is getting ground: the so-called *Capsule Neural Networks*. They seem to be promising because their architecture have shown to be very efficient in embedding image spaces [Panwar and Patras, 2022]. If they could be merged with

contrasting learning, they would make a strong candidate for the team clustering task, especially if this clustering task is done only on FieldPlayers without outliers.

3.5.4 Opening : Build a new dataset and train custom detection algorithms based on it

Unlike football, goalkeepers, referees, and field players in field hockey are easily distinguishable based on their visual cues. Referees do not wear anything special, field players hold a hockey stick, and goalkeepers wear a complete set of equipment. This is shown in Figure 3.21.



Figure 3.21: Field hockey actors lined up in a Belgian National 1 match

Since no field hockey dataset exists, it was necessary to create one from scratch. The custom dataset consists of annotated images that capture multiple instances of field players, referees, and goalkeepers. The annotations define the bounding boxes associated with the regions of interest, ensuring that the detection algorithm learns their unique visual cues. After the annotation phase, it will be possible to train a custom detection algorithm. Although various pre-trained algorithms capable of detecting objects such as individuals, sports balls, or baseball bats are readily available, they can serve as a starting point from which the algorithm can be fine-tuned by adjusting the weights to facilitate the detection of the custom targets.

3.6 Insights extraction and visualisation

As a reminder, the framework has been defined in Section 1.1. The goal is to extract insights from hockey footage during the game.

A widely used tool is the **entropy map**, which provides a measure of the predictability of team behaviors across the field [Lucey et al., 2021]. It involves dividing the field into a grid of n cells. For each cell i , it records all the pass origins and destinations and calculates their respective proportions $p(i)$. This allows the estimation of a probability distribution in each cell, which represents the likelihood of a pass going to another cell. The entropy of each cell can then be computed using the following equation:

$$(3.16) \quad H(X) = - \sum_{i=0}^{n-1} p(i) \log p(i)$$

A higher entropy value indicates that the corresponding part of the field is more important for the team, as it serves as a hub. While entropy maps could be useful in field hockey like in football, the technical constraints are even higher due to the smaller ball size and faster game pace.

Another widely used tool is the **heatmap** of the players. Similarly, it involves dividing the pitch into a grid, but instead of pass data, it focuses on the proportion of time each player has spent within each grid cell. Heatmaps are designed on a per-player basis to understand their movement behavior. An example is shown in Figure 3.22.

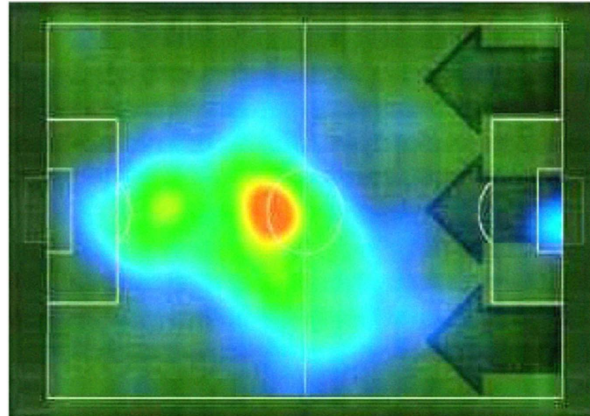


Figure 3.22: Example of a player heatmap in football [Foina et al., 2010]

Finally, as mentioned in Part I, a crucial aspect of the game is monitoring the fatigue levels of players throughout the match. This can be achieved through **real-time tracking metrics**, such as the number of short sprints, long sprints, direction changes, and covered distance. While these metrics are commonly obtained using localization devices, it is worth considering the pipeline outlined in Figure 3.1. It has been demonstrated that the steps of this pipeline can be applied to field hockey using a simple camera. However, it is important to note that the quality and

viewing angle of the recorded video significantly impact the accuracy and reliability of the derived insights.

4.1 Dataset construction and sanity checks

Since there is no publicly available dataset of annotated field hockey video footage, it was necessary to pool together many frames from different sources and then annotate them with respect to the task at hand. In the pooling phase, images were retrieved from 2 channels on YouTube¹. This allowed for some diversity in the data quality and view angles. Accordingly, 450 frames were randomly extracted. Regarding the annotation task, as a reminder, there are 3 categories: FieldPlayer, Referee, and GoalKeeper. It must also be noted that, although many such task facilitation tools exist, it remains a labor-intensive and tedious task. A first attempt is described in Appendix A.3. Then, the **LabelImg**² Python library was used as an annotation tool. In order to use the visual cues, the bounding boxes need to include the players as well as their equipment (the stick, the helmet, the whistle, etc.). See an example in Figure 4.1.

After labeling, the dataset was divided into training and validation sets, with proportions of 80% and 20%, respectively. Each dataset was organized into two folders: "images" and "labels." The "images" folder contained the actual images (in *.png* format) while the "labels" folder contained text files (*.txt* format) that included the coordinates of the ground truth bounding boxes in YOLO format (details in Appendix A.3). By definition, there is class imbalance, with field players representing the vast majority of the detected objects. This means that, during assessment time, it will be necessary to use adequate measures and not, for example, blindly use average accuracy. This is discussed in the next sections along with health checks to assess possible introduced biases. This will be done via class distribution and bounding boxes location distributions.

¹See appendices for granted rights

²<https://github.com/heartexlabs/labelImg.git>

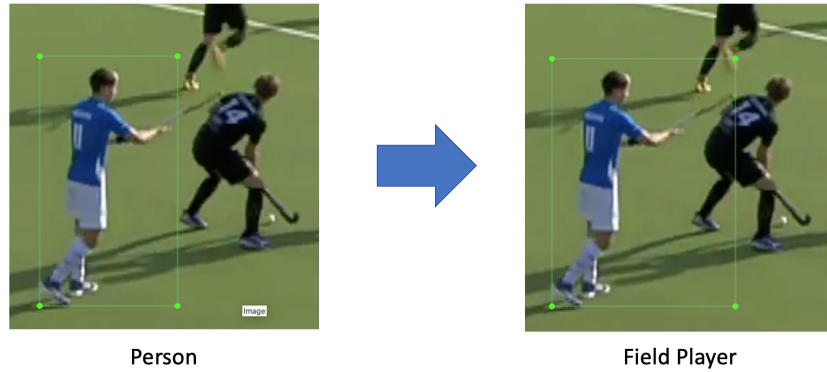


Figure 4.1: Difference in bounding boxes between a person and a hockey field player

4.2 Custom YOLO models for Field hockey

4.2.1 Training step procedure

YOLOs are fine-tunable models for specific applications. This means that we can use these models as a starting point rather than training a model from scratch. In this thesis, YOLOv5 [Jocher, 2020], YOLOv7 [Wang et al., 2022], and YOLOv8 [Jocher et al., 2023] implementations are assessed. They all have different versions which indicate their size and complexity. Greater size/complexity translates to higher performance, but at the cost of reduced speed. These specific models have been chosen because the *nona* version of YOLOv5 can run on a [Central Processing Unit \(CPU\)](#) almost in real-time, while YOLOv7 and YOLOv8 require high [Graphics Processing Unit \(GPU\)](#)-supporting devices. If the former performs well, it becomes a good candidate in practice without necessitating a specific device or cloud computing technologies. Thus, the different versions of YOLO that will be trained and assessed are presented in Table 4.2.1.

Model	Size (pxl)	mAPval 50-95 (%)	mAPval 50 (%)	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs (G)
YOLOv5n	640	28.0	45.7	45	6.3	0.6	1.9	4.5
YOLOv7-X	640	52	71.2	410.5	8.8	4.3	71.3	189.9
YOLOv8s	640	44.9	61.8	128.4	1.2	-	11.2	28.6

Table 4.1: Speed and Performance comparison between YOLOv5n, YOLOv7-X and YOLOv8s

As can be seen, once GPU power is unleashed, YOLOv7 and YOLOv8 are expected to be highly effective. Both of them will be tested because YOLOv7 is the current method, while YOLOv8 has just been released. This will allow a comparison of them in the context of [MPT](#) with the common challenges. The training is carried out using Google Colab Pro in order to utilize the necessary (A100) GPU power and the required time without being timed out. The starting hyperparameters

Batch Size	Epochs	Learning Rate
16	200	0.01

Table 4.2: Main hyper parameters for YOLOs training

were the same for all the models and are summarized in Table 4.2.1.

4.2.2 Training assessment procedure

Since the choice of training has been made with YOLO algorithms, one can track the different loss parts, as explained in Equation (3.10) during training. As a reminder, one will have box, objectness and classification losses. Besides that, one can also track the so-called **Precision**, **Recall** and **mean Average Precision (mAP)** values in the training and validation set. These concepts are explained here below along other essential assessment tools after the training step:

1. **Confusion matrix (CM)** : It compares the actual target values with those predicted by the model, revealing the counts of true positive, true negative, false positive, and false negative predictions.
2. **Precision-Confidence curve** and **Recall-Confidence curve**: The Precision the proportion of correct predictions among all predictions for a particular class and Recall is the proportion of examples of a particular class that has been predicted by the model as belonging to that class:

$$(4.1) \quad \text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}} \quad (4.2) \quad \text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}$$

3. **Precision-Recall (PR) curve**: This is the balance between the two previous metrics and also computed for different threshold values. Note that since there is classes imbalance, it is a useful tool because it allows to assess all classes independently. One wishes to have high precision and recall at the same time. However, depending on the application, one could prefer to have one or another to be higher. A PR curve illustration is illustrated in Figure 4.2a. Accordingly, the closer the curve follows the right-hand and top borders of the PR space, the more accurate the model is believed.
4. **F1-Confidence curve**: The F1-score is a harmonic mean of precision and recall.

$$(4.3) \quad F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

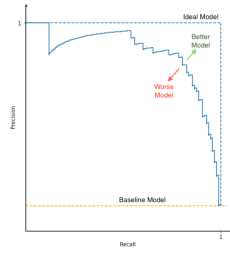
This score can be plotted with respect to multiple confidence thresholds. As illustrated in Figure 4.2b, the higher the curve (i.e., the higher the F1-scores at all confidence levels), the better the model is considered to be. It implies that the algorithm achieves a high level of

accuracy in correctly identifying and tracking objects while minimizing both false positives (incorrectly identifying an object) and false negatives (missing an object). See below an illustration:

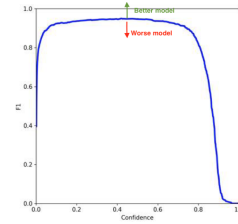
5. **mAP** : This constitutes a really good one-number metric. It takes both Precision and Recall into account in a balanced way. Moreover, it considers a range of thresholds, which allows a more compact assessment view. It can also be computed for each class, which is particularly useful in this case provided current class imbalance. All of those properties makes it the standard benchmarking metric.

$$(4.4) \quad \text{MaP} = \frac{1}{C} \sum_{c=1}^C \text{AP}_c$$

where n is the number of classes. AP is the Average Precision and be viewed a "discretized" Area Under the Curve (AUC) (see illustration in Appendix Figure A.14) with each points corresponding to a threshold. Then, the average is made over the C classes.



(a) Precision-Recall curve illustration



(b) F1-confidence curve illustration

Figure 4.2: PR and F1-Confidence curves

RESULTS AND DISCUSSIONS

5.1 Opening 1 - Annotated Dataset construction

The field hockey dataset consists of 450 images and 5966 annotations. Figure 5.1 shows the class distribution as well as the bounding box distributions. Among these, there are 5091 field players, 396 referees, and 479 goalkeepers.

The field players are evenly distributed over the scenes. However, there is a strong concentration of goalkeepers at the top and bottom centers. This is explained by the fact that goalkeepers are primarily located in these areas when filming the entire pitch (recall, for example Figure 1.3). It is somewhat less visible, but one can observe that the referees are often located at the edges of the frames, which correspond to the edges of the field. This introduces a location bias, which can lead to issues at inference time when the objects are located in other places. Similarly, many GoalKeeper bounding boxes have the same size, which introduces a scale bias. Regarding the x and y average locations, i.e., the bounding box centers, there is clearly a concentration at the top and bottom, which is caused by the sources of the data. Indeed, many of the frames come from images filmed from the top of the goal. Ideally, this distribution should be uniform over the scene. This will lead to another location bias. As for the *width* against the *height*, the aspect ratios generally follow that of a standing person. However, in field hockey, as discussed in section 3.4, aspect ratios are highly variable.

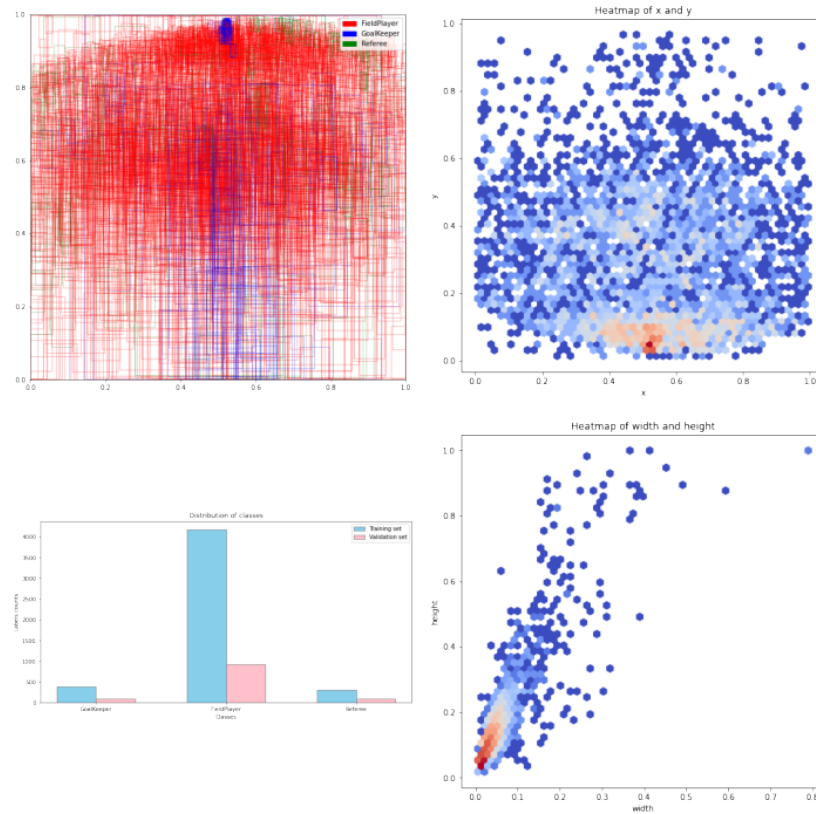


Figure 5.1: Class distribution of the field hockey dataset

5.2 Opening 2 - Custom training

First, the custom YOLOv5 and YOLOv8 detections on the initial frames are shown in Figures 5.2, 5.3, and A.19.

A first striking observation is that, now, all the actors on the field are detected, in contrast to the YOLOv5 (see Figure 3.12) and YOLOv8 (see Figure 3.13) initial models. Moreover, they have been tested on real footage from the Belgian Honnor Division and posted on YouTube. see [Kinart, 2023a, Kinart, 2023b, Kinart, 2023c]



Figure 5.2: Custom YOLOv5 detections



Figure 5.3: Custom YOLOv8 detections

Then, to assess the model's convergence and performance, one can plot the different Loss functions over the epochs, as shown in Figure 5.4 and 5.5 for YOLOv5 and YOLOv7. Note that the custom YOLOv8 training losses are shown in a separate graph in Appendix A.13 because of the difference

in unit losses. The columns refer to the three parts of the loss explained in Section 3.4.3.



Figure 5.4: YOLOv5 & YOLOv7 results training set

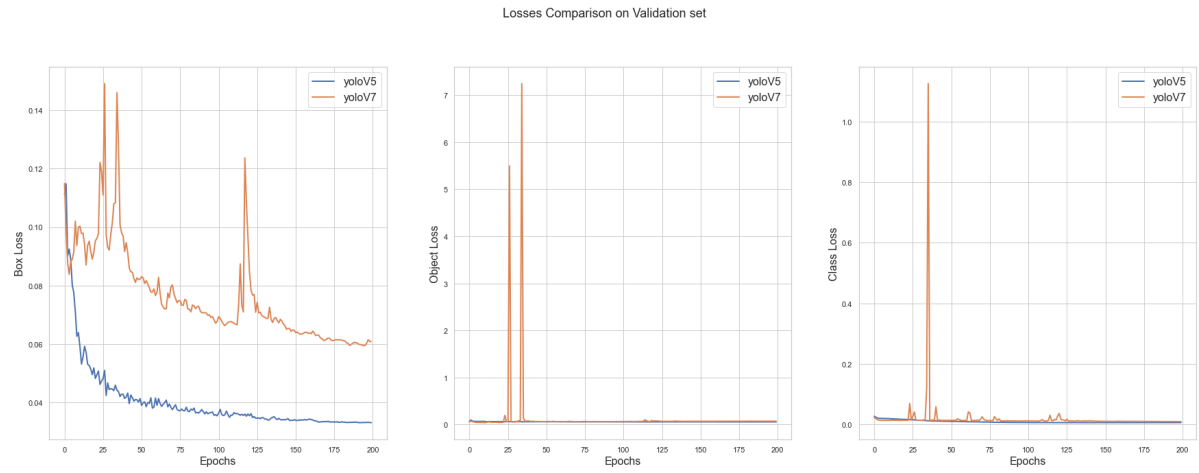


Figure 5.5: YOLOv5 & YOLOv7 results validation set

Then, the **mAP** when **IoU** threshold is larger than 50% (**mAP@0.5**) and the **mAP** when the threshold is between 50% and 95% (**mAP@0.5:0.95**) along with Precision and Recall are plotted in Figures 5.6 and 5.7.

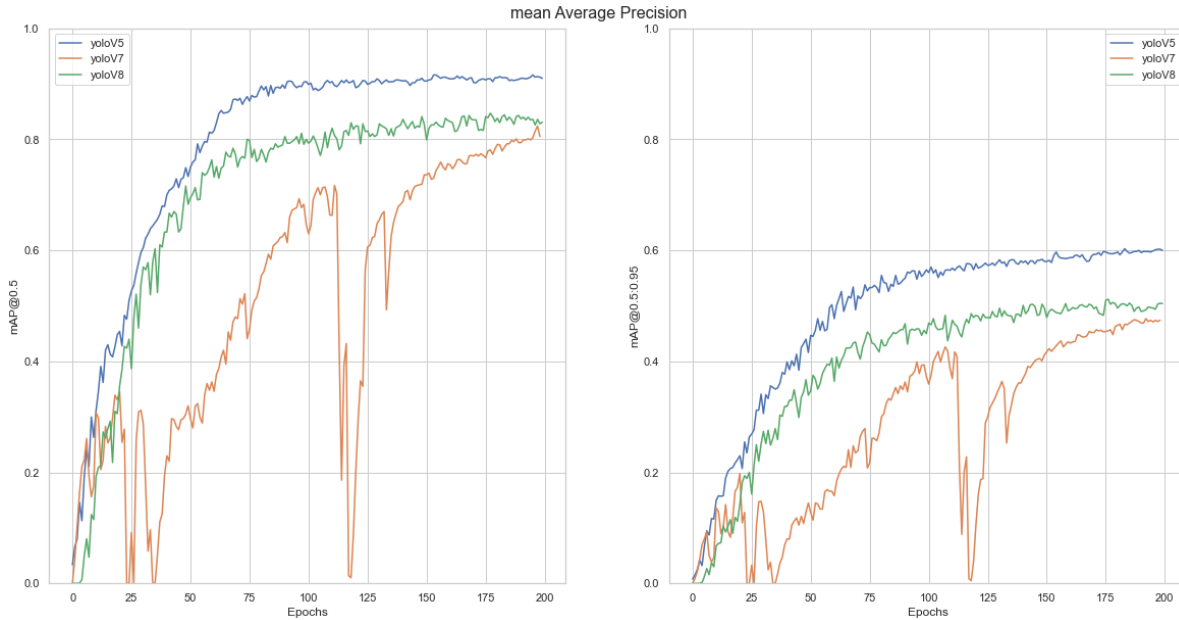


Figure 5.6: YOLOs mean Average Precision for two threshold ranges

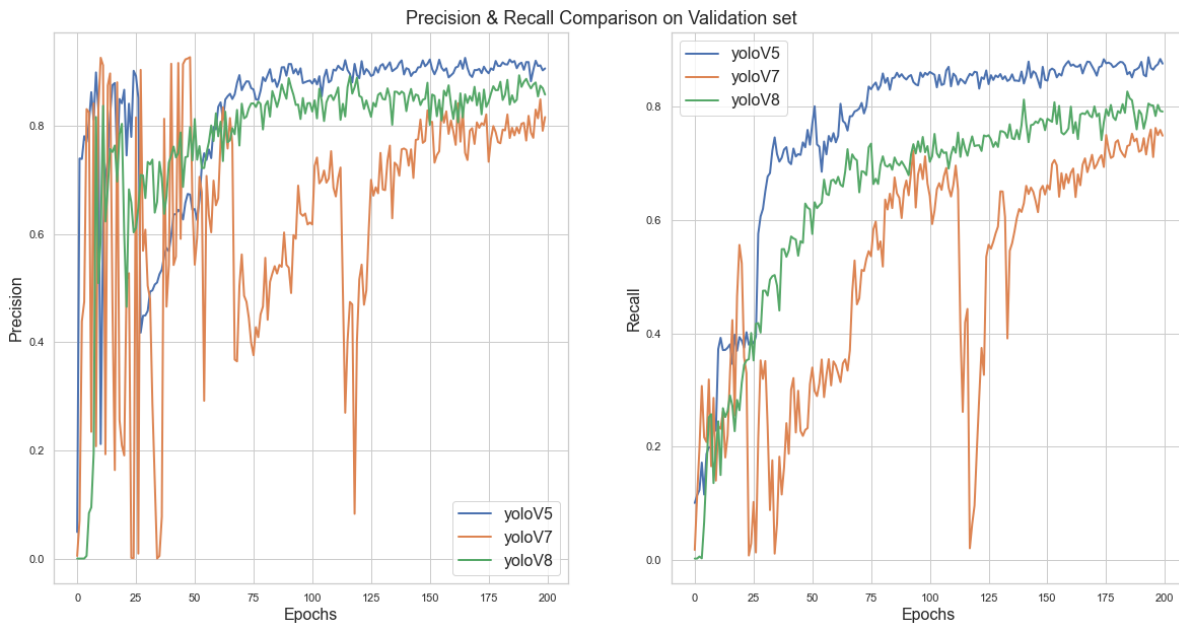


Figure 5.7: YOLOs Precision and Recall curves

The best performing model is YOLOv5, but with a learning curve that rapidly approaches a plateau (more or less after 100 epochs), while its counterpart models continue increasing

their performance. This leads to the conclusion that it would be relevant to train those two models for more iterations and add more data examples. This is probably due to their respective complexity (recall 4.2.1). Another striking observation is the fact that, while YOLOv5 and YOLOv8 performance increase, for YOLOv7, it is clear that something happens around the 30th and 110th epochs where the performance drastically decreases for a very few epochs before stabilising and then increasing again. Sudden jumps in error during the training can suggest a few potential issues:

- Class imbalance: this could be the case since the dataset is highly imbalanced.
- Complex samples: sudden drops can happen where there are a high number of objects, occlusions, or various scales and rotations.
- High learning rate: This can lead to overfitting. This is when the model lacks generalisation power because it adheres too closely to the training data, resulting in low performance with regard to the validation data. In this case, it is not believed to be the issue because there is also a significant dip in the training data performance, while overfitting is related to validation data. On the other hand, it is possible that overfitting occurred around the 30th epoch, since there is a big spike in the validation objectness and classification losses.

It can also be noted that, still for YOLOv7, there are large variations in precision, recall, and **mAP** in the earlier epochs. This could first be explained by the batch size (16 here). At each epoch, 16 images from the training set are used to estimate the gradient error. Smaller values lead to higher variability. Another possibility is again the class imbalance issue because the model might be predicting on the majority class or minority class. On the other hand, YOLOv5 and YOLOv8 are much more stable for those metrics.

The performance metrics from the best models are shown in Table 5.1. Given the Precision, the models correctly identify an object between 81.6% (YOLOv7) and 90.3% (YOLOv5) of the time. Moreover, the models accurately detect between 74.8% (YOLOv7) and 77.7% (YOLOv5) of all the true class instances. On the other hand, **mAP** values reflect a reasonably well-performing model, albeit not outstanding. This suggests there is room for global improvement. YOLOv5 seems to have reached a plateau, while there is still room for improvements for the two other models just by increasing the number of epochs. Another possibility would be to feed more data to the models.

In the context of this thesis, it is also essential to evaluate the model's performance concerning the different classes, since the ultimate goal is to separate them for the clustering step (recall section 3.5). The FieldPlayer class generally shows very high precision, recall, and **mAP** scores, indicating good performance in identifying and distinguishing instances. Conversely, the GoalKeeper class scores lower, suggesting difficulties in differentiating goalkeepers from other players in the scene. Finally, the performance of the Referee class for YOLOv7 is mixed. There is high precision (0.844) but lower recall (0.644), indicating missed detections for many actual

Class	Images	Instances	P	R	mAP@0.5	mAP@0.5:0.95
all	90	1101	0.903 0.816 0.861	0.777 0.748 0.773	0.873 0.809 0.84	0.551 0.478 0.501
FieldPlayer	90	922	0.937 0.842 0.864	0.86 0.873 0.866	0.924 0.902 0.907	0.56 0.517 0.537
GoalKeeper	90	92	0.884 0.761 0.811	0.75 0.728 0.717	0.863 0.754 0.8	0.563 0.465 0.503
Referee	90	87	0.887 0.844 0.91	0.72 0.644 0.736	0.833 0.771 0.812	0.529 0.451 0.463

Table 5.1: Best mAP@0.5 and mAP@0.95 results on Validation set for YOLOv5 (top) , YOLOv7 (middle) and YOLOv8 (bottom)

referee instances. The performance can also be represented in a Confusion Matrix (CM). As they are similar across all YOLO models, only YOLOv7 is presented here (see Appendix Figures A.10 and A.11 for the two others) in Figure 5.8.

This confusion matrix exhibits a unique characteristic compared to traditional classification tasks, as it includes a distinct class known as the Background. Since there are no predicted bounding boxes for the background, only the FN and FP instances are relevant. That's why the diagonal cell relative to them is empty. The majority of the error comes from errors associated with the background. Indeed, all of the FP detections where the model wrongly identified something in the background as an object of interest (when there was none), 76% of these errors occurred with the model believing that these were instances of the FieldPlayer class. This indicates that there are features in the scenes that lead the model to mistakenly associate FieldPlayer. This is quite intuitive since, in the dataset, there are many visible supporters. In fact, many of them are considered field players. This shows the need for additional information when detecting players. To overcome this, two techniques could be used. The first one is adding a class representing supporters in the training set. This would allow for "harder" training. Another one is using the homography of the field. After the field has been detected and modeled, one could perform the detection algorithms within those boundaries.

Since there is class imbalance, it is important to monitor the performance with respect to the groups as well. This is shown in Figure 5.9 for YOLOv5 and YOLOv7 (see Appendix Figures A.12 for YOLOv5, which is similar to YOLOv8).

As you can see, the PR and F1-confidence curves significantly differ among the classes. The global performance for YOLOv7 is greatly driven by the FieldPlayer class, while it is more centered around the mean (thick blue curve) for YOLOv8. The FieldPlayer is best represented, providing satisfactory results while the other two show poorer performance. Regarding the F1-

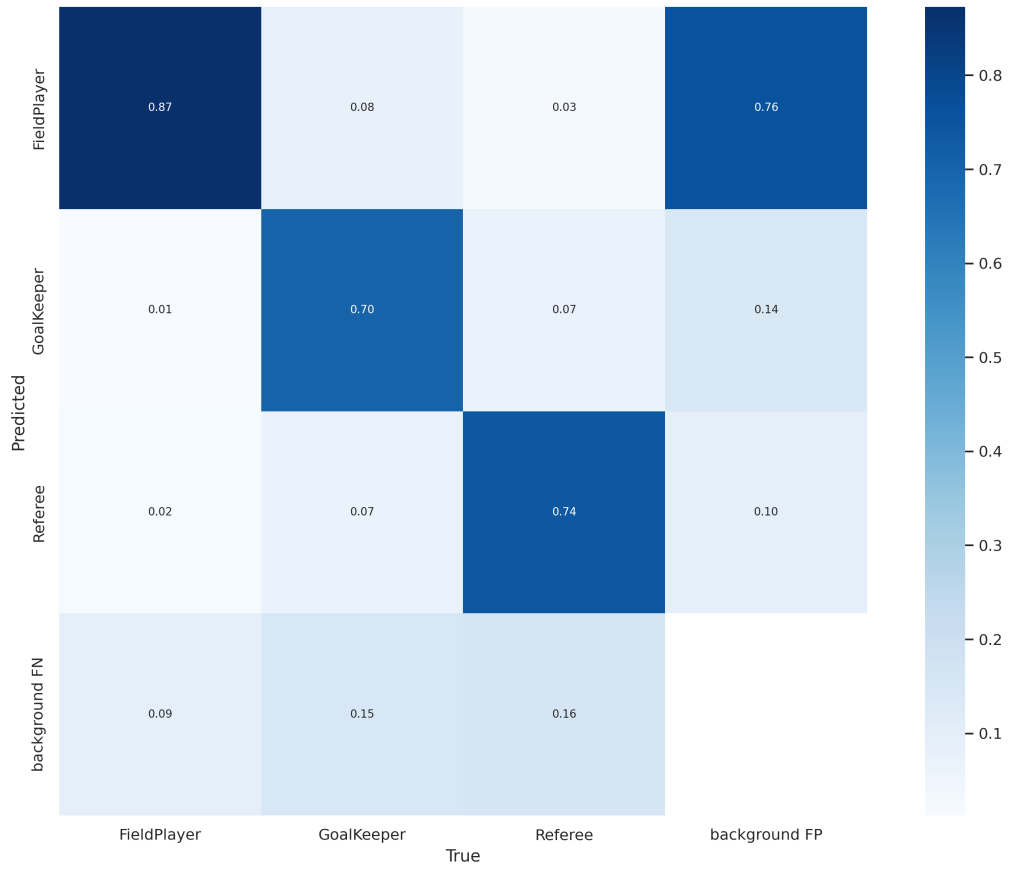
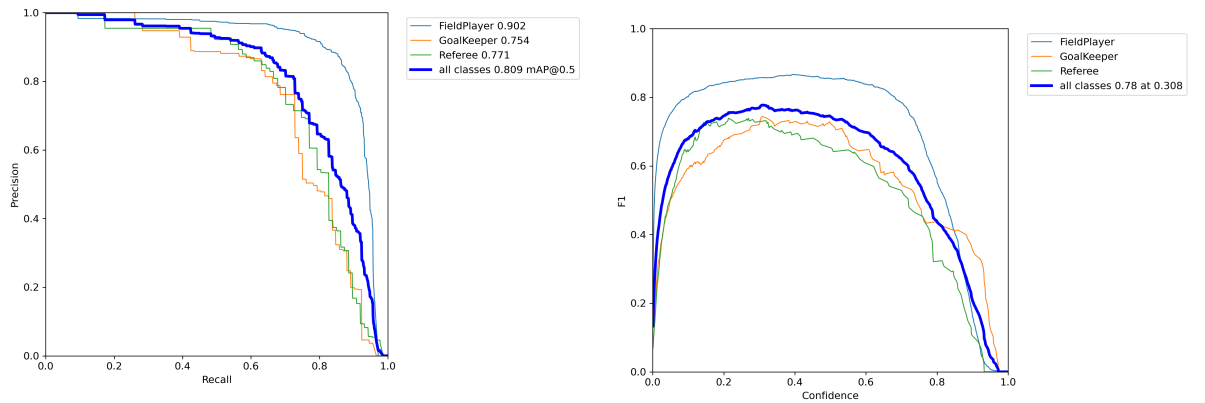


Figure 5.8: Confusion matrix of the custom YOLOv7 model on field hockey players (200 epochs)

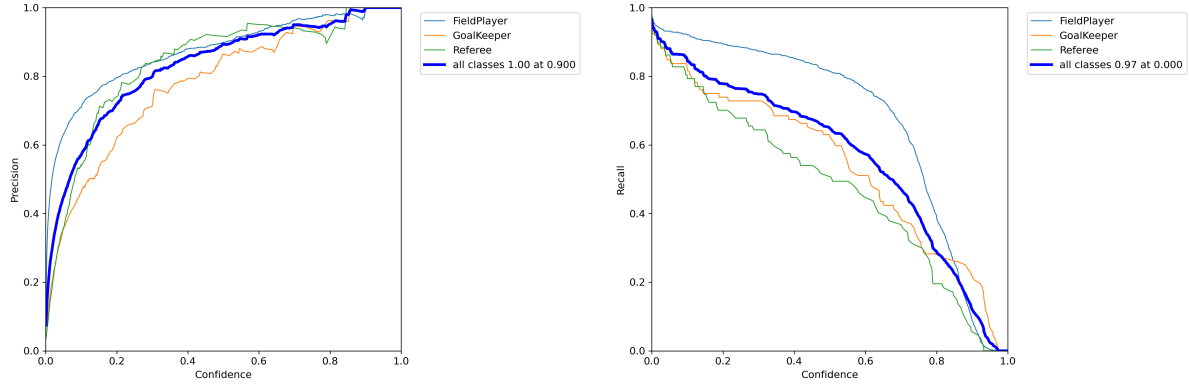


(a) Precision-Recall with 0.81 mAP@0.5 curve for field hockey custom model (b) F1-Confidence curve for field hockey custom model

Figure 5.9: YOLOv7

Confidence curve, the highest scores appear to occur at 0.5 for Fieldplayer, between 0.3 and 0.6 for GoalKeeper and 0.2 for referees in both models. Moreover, Figure 5.2 shows that the precision

rapidly grows with respect to the confidence threshold, indicating that many of the predicted positives for this class are truly positive. However, the Recall confidence curve exposes the model's weaknesses, especially for the GoalKeeper and Referee classes, since it is desired that the recall remains as high as possible while the confidence threshold gets higher.



(a) Precision-Confidence curve custom YOLOv7 for field hockey (b) Recall-Confidence curve custom YOLOv7 for field hockey

Figure 5.10

CONCLUSION

Objectives and research questions: The main objective of this thesis was first determine a clear framework one needs in order to extract insights in a multi player context. Then, the idea was to go through all the classical CV techniques and apply them to real field hockey footage in order to assess their real viability. Finally, a focus was made on detection algorithms in order to discriminate between the referees, the goalkeeper and the field players based on their visual cues. This allows to turn team classification task from a soft clustering task or bad hard clustering task to a hard clustering task with two well-balanced groups.

Methodology and techniques: First part consisted in reviewing the literature related to multi player insights extraction in order to establish a framework that allows to choose the best devices and techniques with respect to the situation. Once it has been done for this thesis scope, it was possible to review current techniques in computer vision and emphasize the total absence of public data in that field for machine learning. Accordingly, each CV step techniques were applied on real footage in order to see their viability. At that point, it was pointed out that visual cues could be used in field hockey to discriminate between field players, referees and goalkeepers to make the Team clustering task a hard clustering problem. DBSCAN (soft clustering) and K-means (hard clustering) were compared within this framework. Accordingly, a proposition was made by creating a dataset of annotated images allowing to train algorithms that detect the referees, goal keeper and field players. They have been thoroughly assessed and discussed in terms of their applicability in practice.

Effectiveness and performance : In a theoretical perspective, establishing the network allowed to look at the literature in a more efficient way, without losing time to non-useful sources. In a practical perspective, because of the simple framework that has been chosen (a single fixed camera, with real-time fully automatic insights), it was possible to implement all the CV steps. Besides that, building a dataset of images related to field hockey allows to open the gate for further studies related to that field. However, all the noise inserted by brightness, occlusions, etc., make it very difficult to make a well diversified data with enough observations. The current dataset is imbalanced and is made of 450 images with about 6000 annotations including 85.3% of field players. One should also pay attention to the typical location of the referees and goalkeepers since there often located at the same areas. The three YOLO models that have been trained

on this dataset allow to discriminate between the three classes although the training epochs show high instability for YOLOv7 in comparison to its counterparts. On the other hand, the custom YOLOv5 model, which demonstrates efficient real-time performance using CPU, makes it suitable for practical implementation. The other two custom YOLOv7 and YOLOv8 models, which require GPU, seem to require more data to be trained. Training losses showed that their learning curve has not plateaued yet, which means they could be trained longer and would require more data in order to make the model more robust to unseen situations, e.g. with rain, new camera angles, etc. Overall, the custom models demonstrate a fair performance. The ability to detect "FieldPlayer" instances is particularly strong with $mAP@0.5$ higher than 90%, but there are noticeable weaknesses, especially in the detection of GoalKeeper with $mAP@0.5$ between 75% and 86%, and "Referee" between 77% and 83%.

Limitations and challenges: In that context, the most difficult part was dealing with the complete absence of data. Each CV steps require associated data in order to be able to assess effectively the results. The work allows to know which techniques could be tested when data are available. Moreover, it is known that many state of the art techniques use DL techniques require many data, which made difficult the study. This has introduced biases that have been discussed in previous chapter. Another challenge inherent to this field is the fact that current CV advancements using DL are booming. This gives a high risk of many techniques discussed here to become obsolete.

Possible future work : As such, it would be relevant to add more data to the newly made dataset to make it more diversified and robust. Moreover, CV pipeline steps techniques have been implemented to simplistic examples. As soon as annotated data will be available, it will be possible to assess these techniques more thoroughly. A main improvement step in order to overcome the issue related to the distance and angle of the camera is using multiple fixed cameras. This is expected to hugely increase the performance of the Teams clustering as well as the precision of the field projection on board using homography. Aside of that, as in many application of DL, many new techniques are coming. For example, while working on this thesis, a new YOLO version came out, called YOLO-NAS¹ that claims itself state-of-the-art in object detection. More broadly, it can also be added that a new extremely efficient segmentation model just released, which has not been applied to field hockey, see [Segment Anything \(SA\)](#) [Kirillov et al., 2023], with incredible performance since, it can literally segment anything.

In a nutshell: This work showed that it is possible to implement simple solutions tailor-made to field hockey world and help them to take better in-game decisions in order to take better tactical decisions and prevent possible injuries.

¹<https://dec.ai/blog/yolo-nas-object-detection-foundation-model/>



APPENDIX A

A.1 Tables

Model	Kernel Size	Minimum Line Length	Maximum Line Gap
Canny	5	100	10
Sobel	5	200	10

Table A.1: Hough Lines Parameters

A.2 Figures

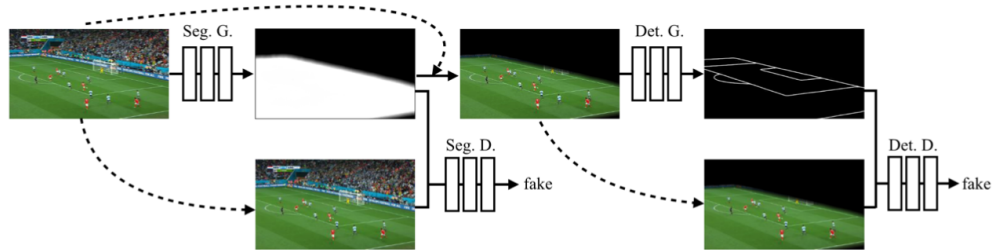
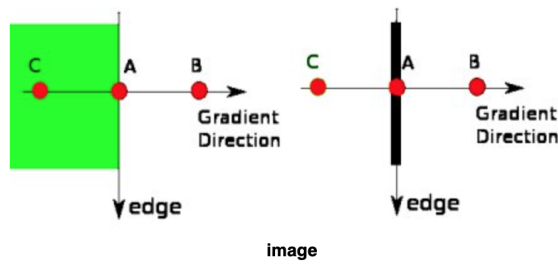


Fig. 4: The two-GAN model. We use a two chained conditional GAN to detect field markings. “Seg.” and “Det.” are short for segmentation and detection, respectively. “G.” and “D.” are short for generative network and discriminative network, respectively.

Figure A.1: (1) Field segmentation and (2) edges segmentation procedure in [Chen and Little, 2018]



Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero).

In short, the result you get is a binary image with "thin edges".

Figure A.2: nms

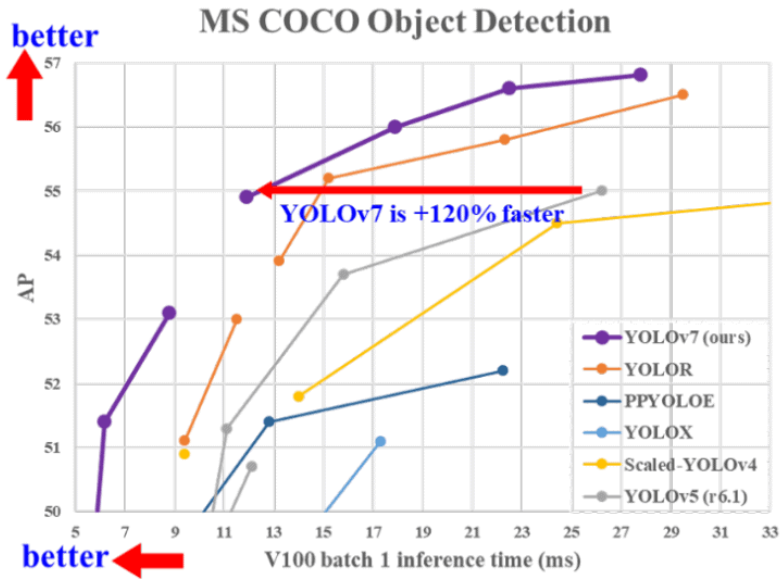


Figure A.3: YOLO versions comparison w.r.t speed and AP ([Wang et al., 2022]

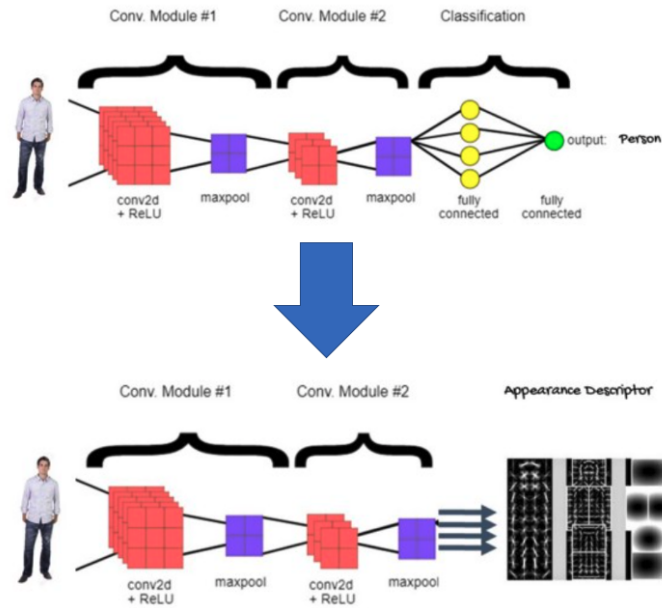


Figure A.4: DeepSort appearance feature space construction

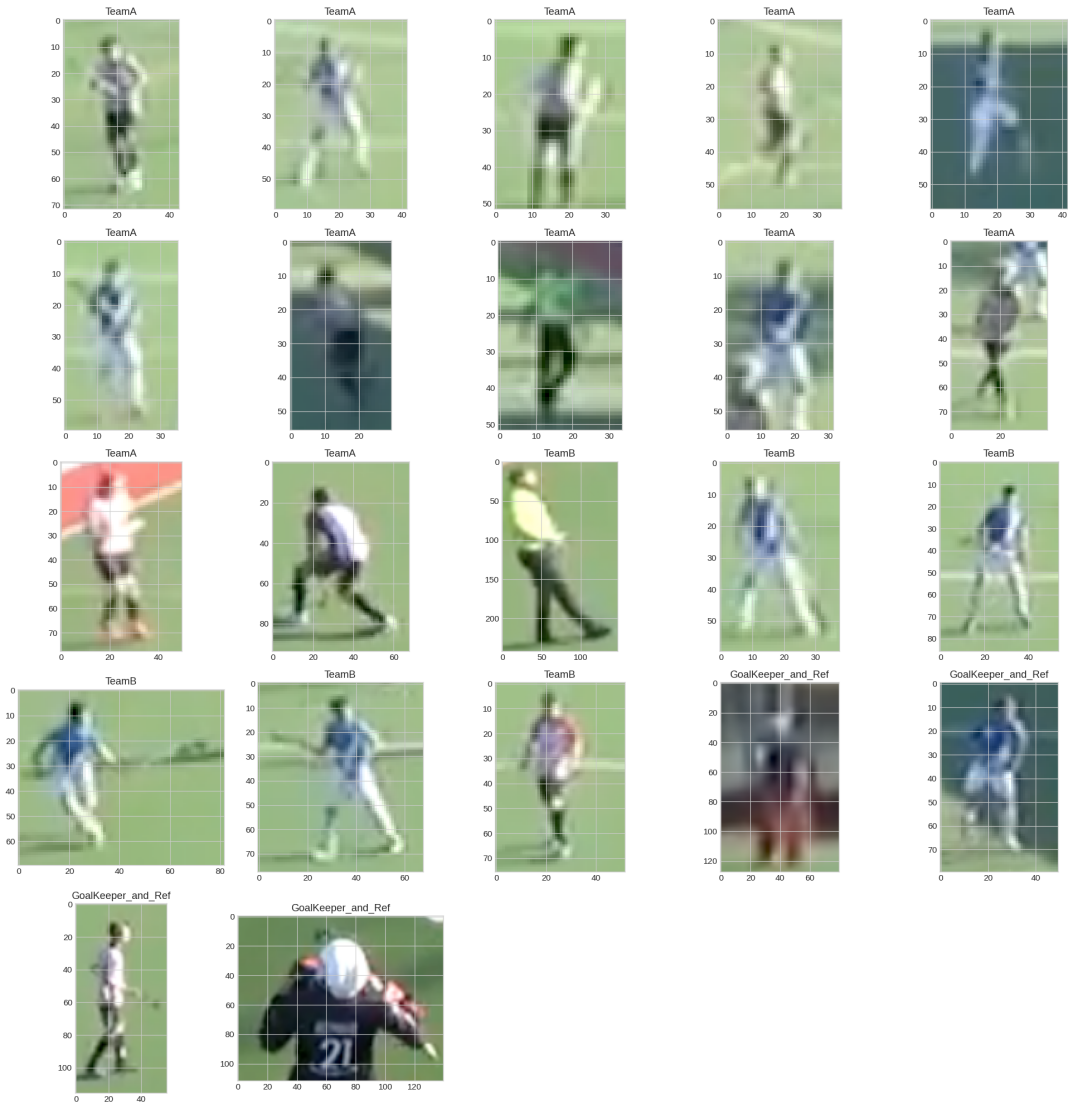


Figure A.5: DBSCAN Clustering with a greedy approach to find two groups and outliers with $\text{eps} = 5.0325$ and $\text{min}_{\text{samples}} = 6$

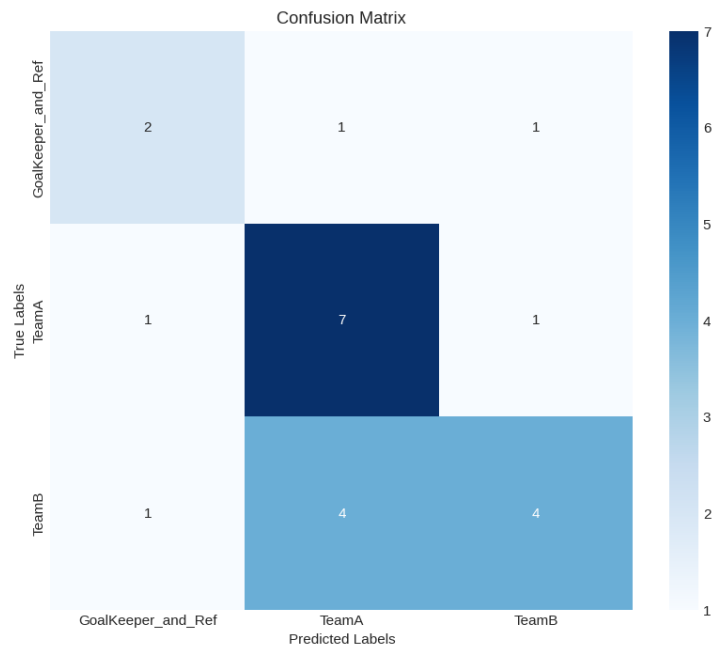


Figure A.6: Confusion matrix from DBSCAN clustering with $K = 2$ groups



Figure A.7: K-means clustering Confusion matrix on initial frame

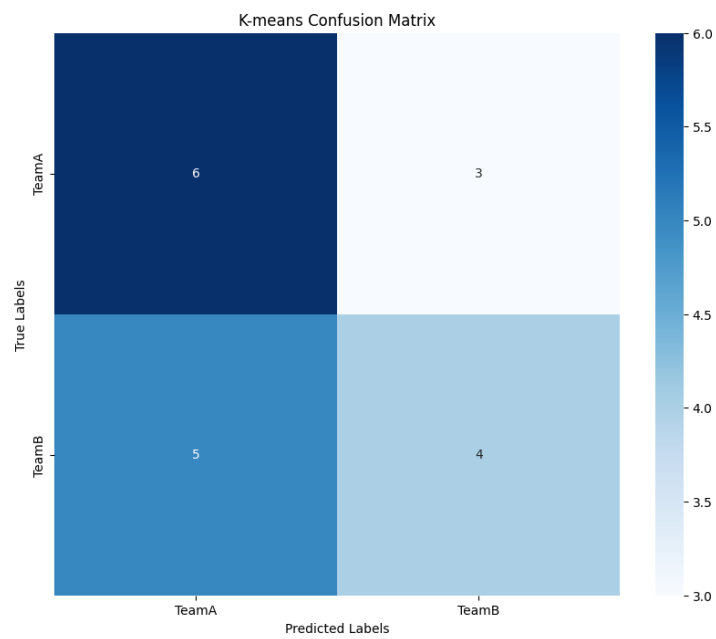


Figure A.8: K-means clustering on initial frame

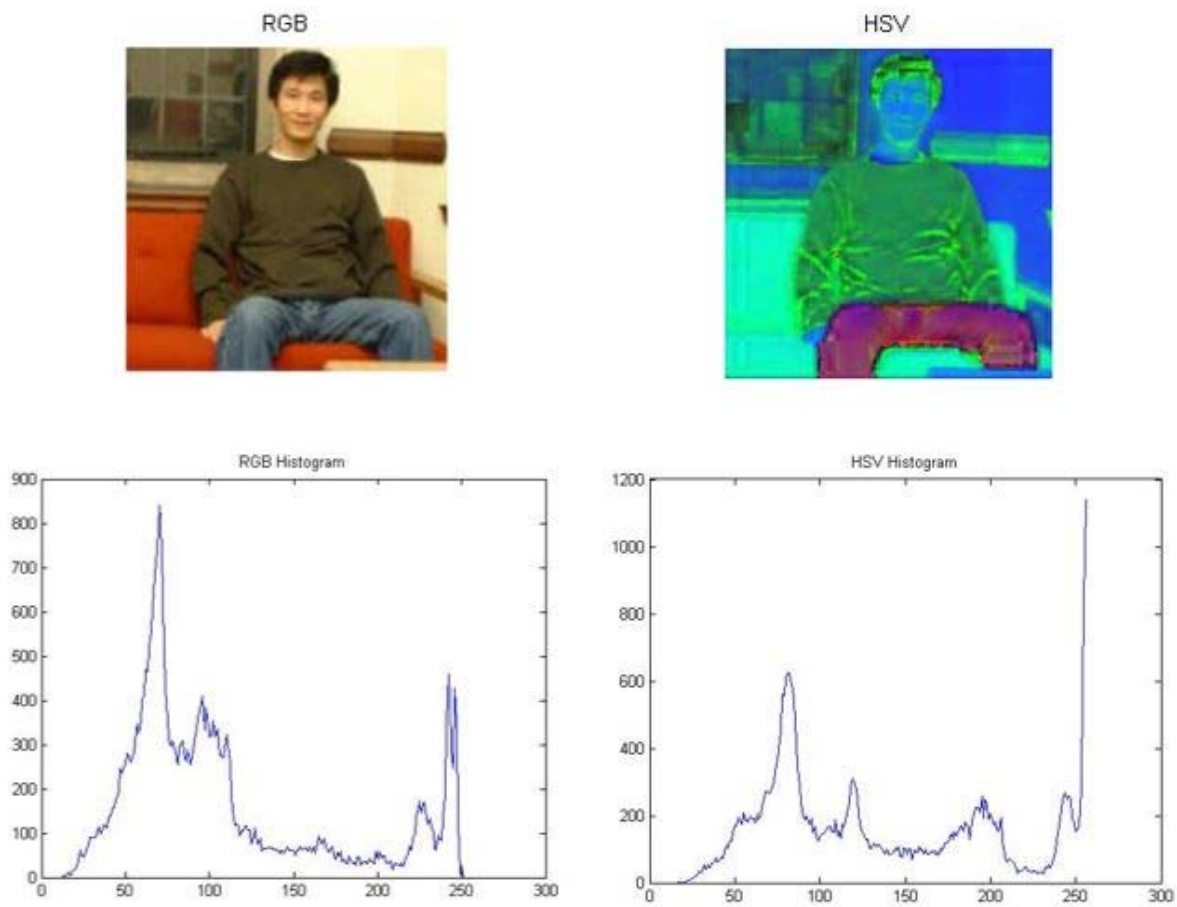


Figure A.9: RGB (left) and HSV (right) histograms example

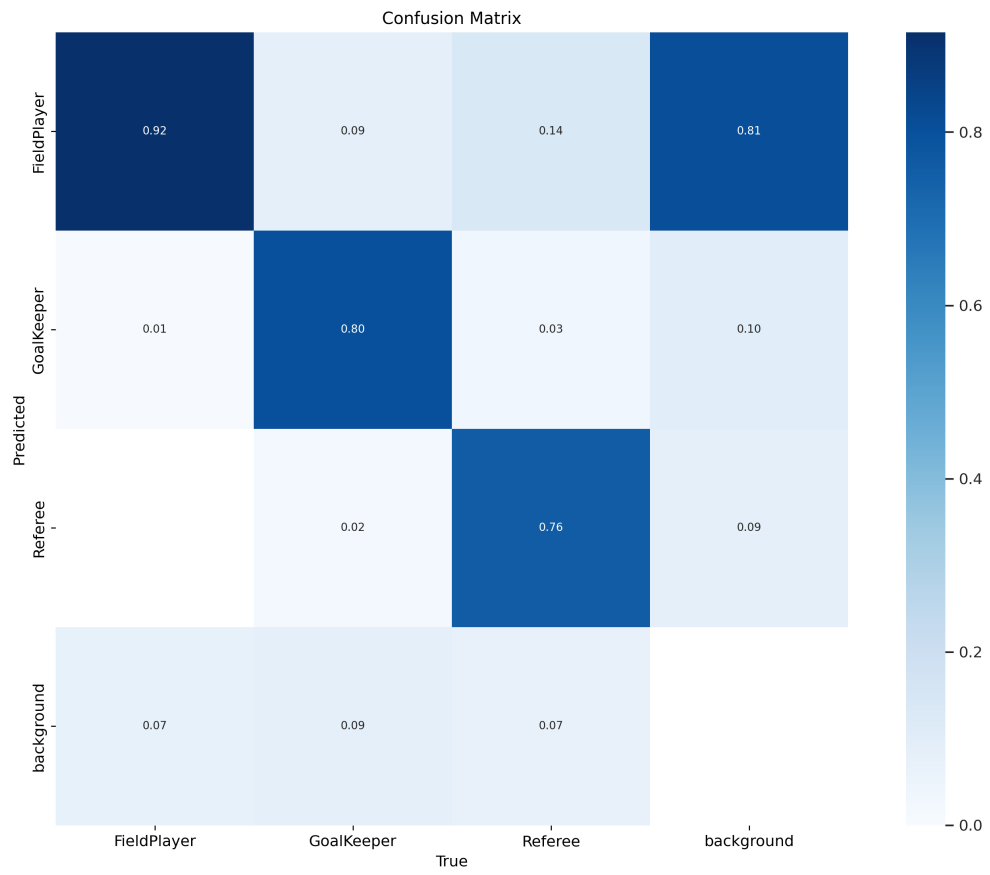
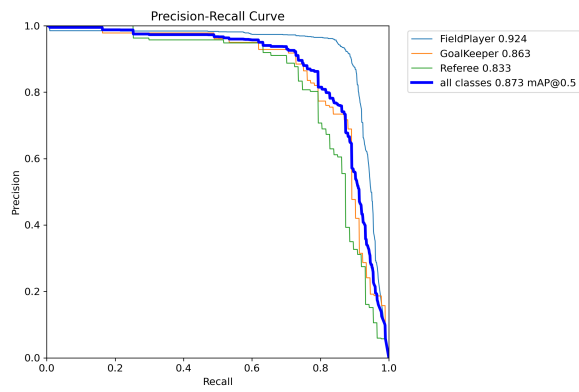


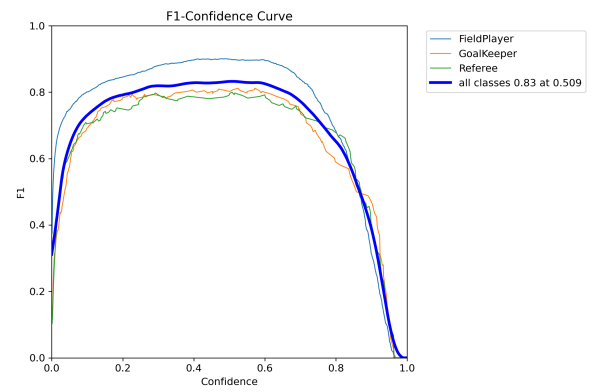
Figure A.10: Confusion matrix yoloV5 training



Figure A.11: Confusion matrix yoloV8 training



(a) Precision-Recall with best mAP@0.5 curve



(b) F1-Confidence curve

Figure A.12: YoloV8

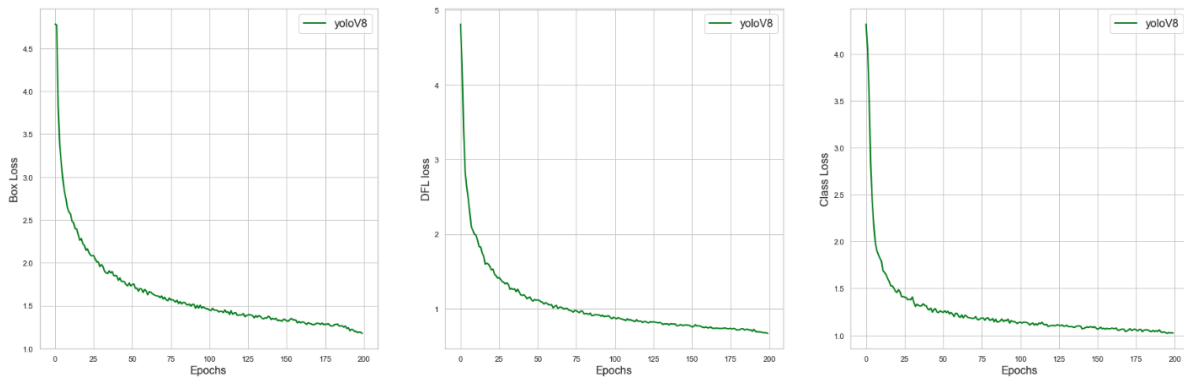


Figure A.13: YOLOv8 training losses

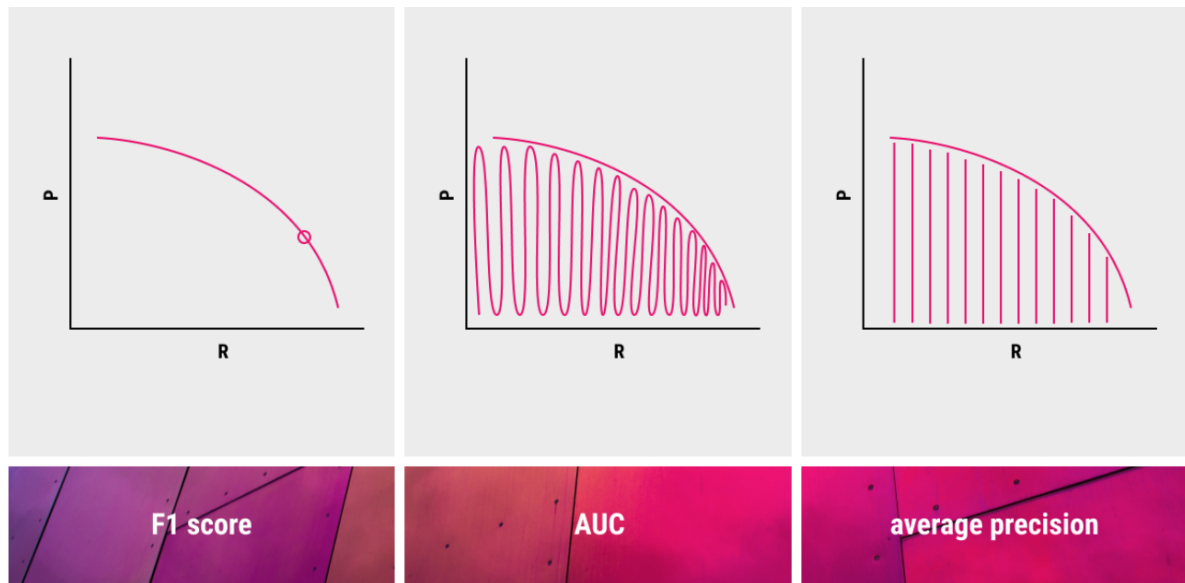
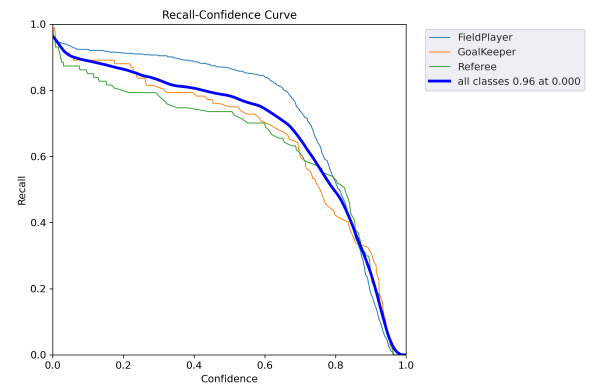
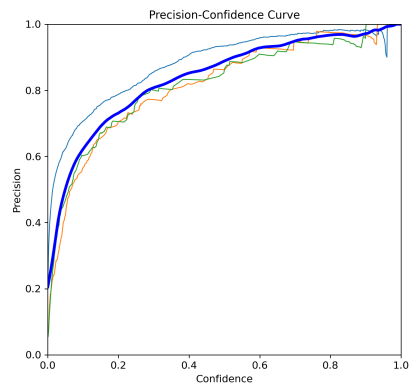


Figure A.14: Illustration and comparison of F1 AUC and AP measures computations

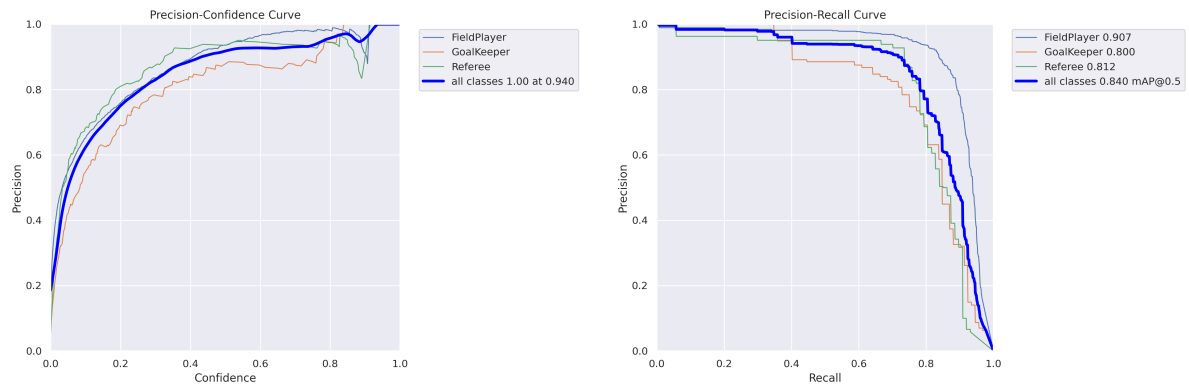


Figure A.15: YOLOv7 detections on initial frame



(a) Precision-Confidence curve customYoloV5 for field hockey (b) Recall-Confidence curve customYoloV5 for field hockey

Figure A.16



(a) Precision-Confidence curve customYoloV8 for field hockey (b) Recall-Confidence curve customYoloV8 for field hockey

Figure A.17

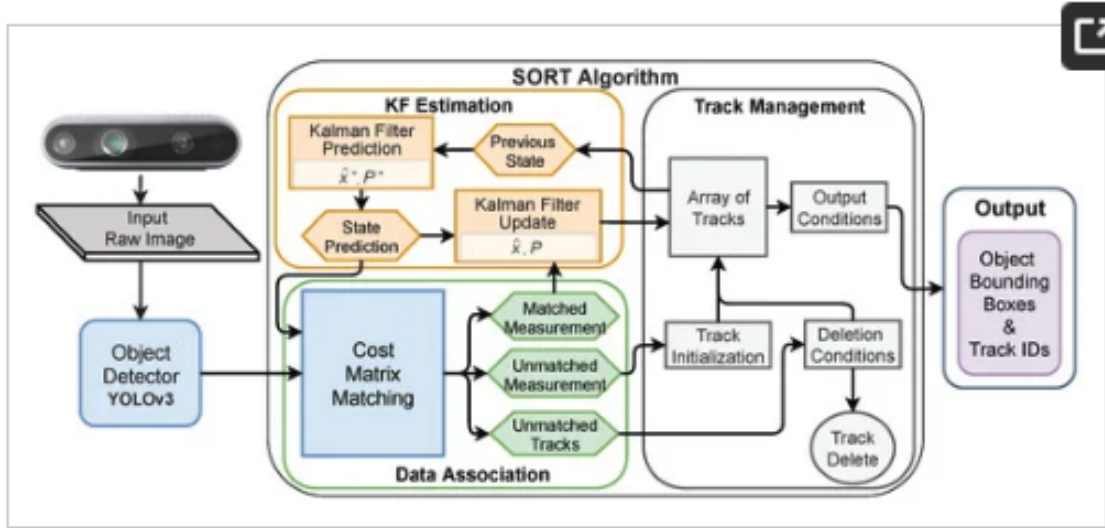


Figure 2. Overview of the object tracking SORT algorithm.

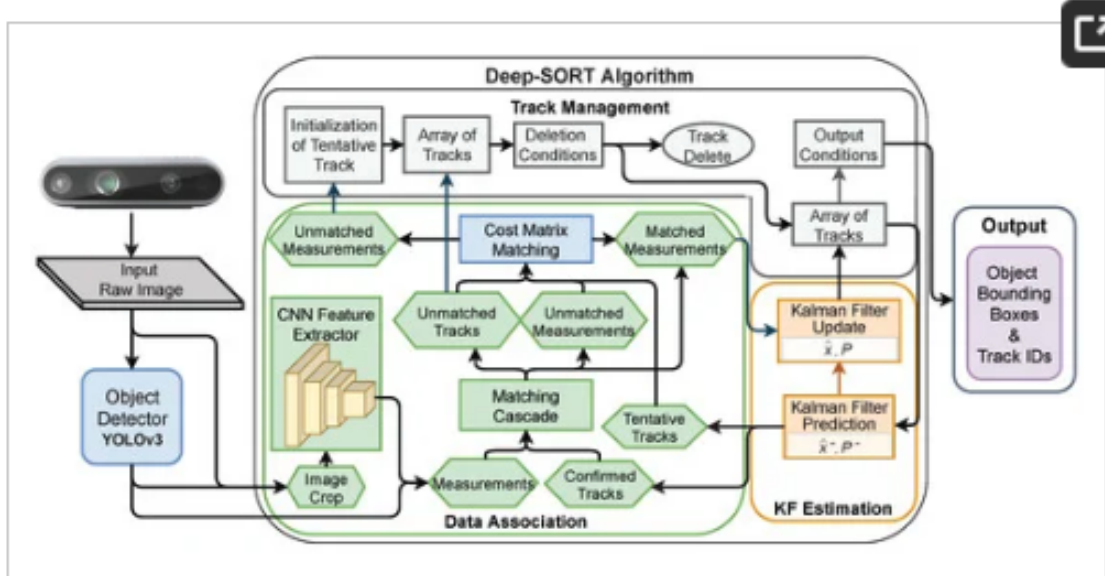


Figure 3. Overview of the object tracking Deep-SORT algorithm.

Figure A.18: Sort and DeepSort algorithm architecture



Figure A.19: Custom YOLOv7 results on initial frame



Figure A.20: Classification example of the most common viewpoints that exist in field hockey

A.3 Supplementary notes

A.3.1 Annotation platform - Amazon Mechanical Turk

Amazon Mechanical Turk platform¹ allows creating tasks consisting in labelling many datasets. The task is posted on the platform and then *workers* can do the necessary tasks against some wage. The idea is very simple but effectively creating the tasks is not suited for the common people, but rather for people with IT and computer science skills. The methodology is as follow:

- Images dataset : it is first necessary to create an excel file that provides all the images links. This means that it is necessary to upload all images into a publicly accessible bucket of images. To do so, Amazon S3 was used². All the images are publicly available.
- Task creation: it is necessary to specify the task, here draw bounding boxes for images classification, and the instructions (in HTML format). See Figure A.21.
- Task publication : since it is open to anyone on the platform, it was necessary to give as much relevant details as possible. Then, propose a price per labeled frame, publish it and wait.

¹<https://www.mturk.com/>

²<https://aws.amazon.com/fr/s3/>

APPENDIX A. APPENDIX A

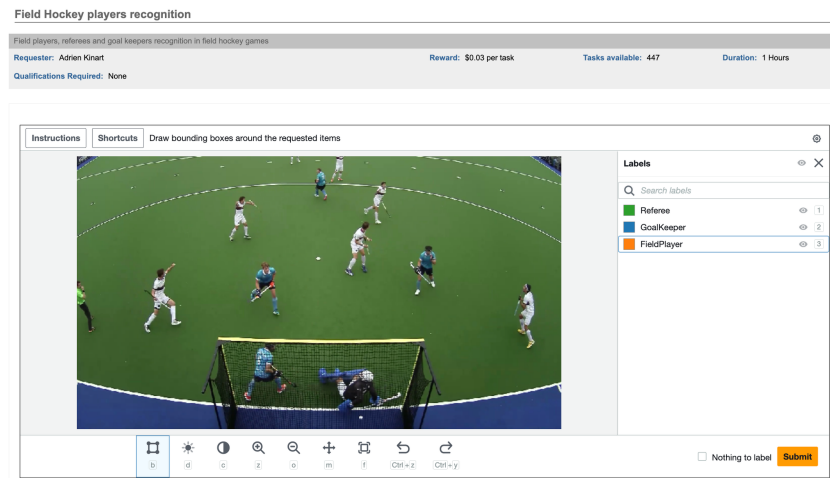


Figure A.21: A task of bounding box doing for objects detection

A.3.2 YOLO format structure

< object class > < x-center > < y-center > < width > < height >

- **<object class>** is an integer representing the class of the object. The class index should start from 0 and increase by 1 for each unique class in the dataset.
- **<x-center>** and **<y-center>** are the coordinates of the center of the bounding box, normalized by the width and height of the image, respectively. The values should be in the range of [0, 1].
- **<width>** and **<height>** are the width and height of the bounding box, normalized by the width and height of the image, respectively. The values should be in the range of [0, 1].

BIBLIOGRAPHY

- [Akenhead and Nassis, 2016] Akenhead, R. and Nassis, G. P. (2016).
Training load and player monitoring in high-level football: Current practice and perceptions.
International Journal of Sports Physiology and Performance, 11(5):587–593.
- [Ali et al., 2012] Ali, M. M. N., Abdullah-Al-Wadud, M., and Lee, S.-L. (2012).
An efficient algorithm for detection of soccer ball and players.
- [Beal et al., 2019] Beal, R., Norman, T. J., and Ramchurn, S. D. (2019).
Artificial intelligence for team sports: a survey.
The Knowledge Engineering Review, 34.
- [Beal et al., 2020] Beal, R., Norman, T. J., and Ramchurn, S. D. (2020).
Optimising daily fantasy sports teams with artificial intelligence.
International Journal of Computer Science in Sport, 19(2):21–35.
- [Bewley et al., 2016] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016).
Simple online and realtime tracking.
In *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE.
- [Bialkowski et al., 2013] Bialkowski, A., Lucey, P., Carr, P., Denman, S., Matthews, I., and Sridharan, S. (2013).
Recognising team activities from noisy data.
In *2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE.
- [Chen and Little, 2018] Chen, J. and Little, J. J. (2018).
Sports camera calibration via synthetic data.
- [Chen et al., 2018] Chen, L., Ai, H., Zhuang, Z., and Shang, C. (2018).
Real-time multiple people tracking with deeply learned candidate selection and person re-identification.
In *2018 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE.
- [Cheng, 1995] Cheng, Y. (1995).
Mean shift, mode seeking, and clustering.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(8):790–799.
- [Csanalosi et al., 2020] Csanalosi, G., Dobreff, G., Pašić, A., Molnar, M., and Toka, L. (2020).
Low-cost optical tracking of soccer players.

- [Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005).
Histograms of oriented gradients for human detection.
In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. IEEE.
- [Direkoglu et al., 2017] Direkoglu, C., Sah, M., and O'Connor, N. E. (2017).
Player detection in field sports.
Machine Vision and Applications, 29(2):187–206.
- [Duarte Moura et al., 2022] Duarte Moura, H., Kholkin, L., Van Damme, L., Mets, K., Leysen, C., De Schepper, T., Hellinckx, P., and Latré, S. (2022).
Low cost player tracking in field hockey.
In Brefeld, U., Davis, J., Van Haaren, J., and Zimmermann, A., editors, *Machine Learning and Data Mining for Sports Analytics*, pages 103–115, Cham. Springer International Publishing.
- [Eetvelde et al., 2021] Eetvelde, H. V., Mendonça, L. D., Ley, C., Seil, R., and Tischer, T. (2021).
Machine learning methods in sport injury prediction and prevention: a systematic review.
Journal of Experimental Orthopaedics, 8(1).
- [Eshel and Moses, 2008] Eshel, R. and Moses, Y. (2008).
Homography based multiple camera detection and tracking of people in a dense crowd.
In *2008 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE.
- [Feichtenhofer et al., 2018] Feichtenhofer, C., Pinz, A., and Zisserman, A. (2018).
Detect to track and track to detect.
- [Fernandes and Oliveira, 2008] Fernandes, L. A. and Oliveira, M. M. (2008).
Real-time line detection through an improved hough transform voting scheme.
Pattern Recognition, 41(1):299–314.
- [Foina et al., 2010] Foina, A. G., Badia, R. M., El-Deeb, A., and Ramirez-Fernandez, F. J. (2010).
Player tracker - a tool to analyze sport players using RFID.
In *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. IEEE.
- [Han et al., 2022] Han, J., Kang, K., and Kim, J. (2022).
Improvement in error recognition of real-time football images by an object-augmented AI model for similar objects.
Electronics, 11(23):3876.
- [Hoernig et al., 2015] Hoernig, M., Herrmann, M., and Radig, B. (2015).
Real-time segmentation methods for monocular soccer videos.
Pattern Recognition and Image Analysis, 25(2):327–337.
- [Horvat and Job, 2020] Horvat, T. and Job, J. (2020).

- The use of machine learning in sport outcome prediction: A review.
WIREs Data Mining and Knowledge Discovery, 10(5).
- [Istasse et al., 2019] Istasse, M., Moreau, J., and De Vleeschouwer, C. (2019).
Associative embedding for team discrimination.
In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2477–2486.
- [Jocher, 2020] Jocher, G. (2020).
YOLOv5 by Ultralytics.
- [Jocher et al., 2023] Jocher, G., Chaurasia, A., and Qiu, J. (2023).
YOLO by Ultralytics.
- [Kalake et al., 2022] Kalake, L., Dong, Y., Wan, W., and Hou, L. (2022).
Enhancing detection quality rate with a combined HOG and CNN for real-time multiple object tracking across non-overlapping multiple cameras.
Sensors, 22(6):2123.
- [Khan et al., 2019] Khan, G., Tariq, Z., and Khan, M. U. G. (2019).
Multi-person tracking based on faster r-CNN and deep appearance features.
In *Visual Object Tracking with Deep Neural Networks*. IntechOpen.
- [Kinart, 2023a] Kinart, A. (2023a).
custom yolov5 detection in field hockey.
<https://youtu.be/-1E0E08rGSc>.
- [Kinart, 2023b] Kinart, A. (2023b).
custom yolov7 detection in field hockey.
https://youtu.be/hP__vbZd-bI.
- [Kinart, 2023c] Kinart, A. (2023c).
custom yolov8 detection in field hockey.
<https://youtu.be/V28dVZyRIEQ>.
- [Kinart, 2023d] Kinart, A. (2023d).
Field hockey dataset for referees, goalkeepers and field players recognition.
kaggle.com/datasets/adrienkinart/FieldHockeyPlayers.
- [Kirillov et al., 2023] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollár, P., and Girshick, R. (2023).
Segment anything.
- [Komorowski et al., 2019] Komorowski, J., Kurzejamski, G., and Sarwas, G. (2019).
DeepBall: Deep neural-network ball detector.

- In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications.
- [Koshkina et al., 2021] Koshkina, M., Pidaparthi, H., and Elder, J. H. (2021).
Contrastive learning for sports video: Unsupervised player classification.
- [Lowe, 1999] Lowe, D. (1999).
Object recognition from local scale-invariant features.
In *Proceedings of the Seventh IEEE International Conference on Computer Vision*. IEEE.
- [Lu et al., 2013] Lu, W.-L., Ting, J.-A., Little, J. J., and Murphy, K. P. (2013).
Learning to track and identify players from broadcast sports videos.
IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(7):1704–1716.
- [Lucey et al., 2021] Lucey, P., Bialkowski, A., Carr, P., Foote, E., and Matthews, I. (2021).
Characterizing multi-agent team behavior from partial team tracings: Evidence from the english premier league.
Proceedings of the AAAI Conference on Artificial Intelligence, 26(1):1387–1393.
- [Manafifard et al., 2017] Manafifard, M., Ebadi, H., and Abrishami Moghaddam, H. (2017).
A survey on player tracking in soccer videos.
Computer Vision and Image Understanding, 159:19–46.
Computer Vision in Sports.
- [Mavrogiannis and Maglogiannis, 2022] Mavrogiannis, P. and Maglogiannis, I. (2022).
Amateur football analytics using computer vision.
Neural Computing and Applications.
- [Mazzeo et al., 2010] Mazzeo, P. L., Spagnolo, P., Leo, M., and D’Orazio, T. (2010).
Football players classification in a multi-camera environment.
volume 6475, pages 143–154.
- [Meinhardt et al., 2022] Meinhardt, T., Kirillov, A., Leal-Taixe, L., and Feichtenhofer, C. (2022).
Trackformer: Multi-object tracking with transformers.
- [Ngo et al., 2010] Ngo, V. A., Yang, W., and Cai, J. (2010).
Accurate playfield detection using area-of-coverage.
In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. IEEE.
- [Okuma et al., 2004] Okuma, K., Taleghani, A., de Freitas, N., Little, J. J., and Lowe, D. G. (2004).
A boosted particle filter: Multitarget detection and tracking.
In *Lecture Notes in Computer Science*, pages 28–39. Springer Berlin Heidelberg.
- [Panwar and Patras, 2022] Panwar, H. and Patras, I. (2022).
Capsule network based contrastive learning of unsupervised visual representations.

- [Pham et al., 2017] Pham, P., Nguyen, D., Do, T., Ngo, T. D., and Le, D.-D. (2017).
Evaluation of deep models for real-time small object detection.
In *Neural Information Processing*, pages 516–526. Springer International Publishing.
- [Pons et al., 2019] Pons, E., García-Calvo, T., Resta, R., Blanco, H., del Campo, R. L., García, J. D., and Pulido, J. J. (2019).
A comparison of a GPS device and a multi-camera video technology during official soccer matches: Agreement between systems.
PLOS ONE, 14(8):e0220729.
- [Redmon et al., 2015] Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2015).
You only look once: Unified, real-time object detection.
CoRR, abs/1506.02640.
- [Redmon and Farhadi, 2016] Redmon, J. and Farhadi, A. (2016).
Yolo9000: Better, faster, stronger.
- [Sander et al., 1998] Sander, J., Ester, M., Kriegel, H.-P., and Xu, X. (1998).
Data Mining and Knowledge Discovery, 2(2):169–194.
- [Seidl et al., 2018] Seidl, T., †, A., †, A., Carr, P., and Lucey, P. (2018).
Bhostgusters: Realtime interactive play sketching with synthesized nba defenses.
- [Sharma et al., 2017] Sharma, R. A., Bhat, B., Gandhi, V., and Jawahar, C. V. (2017).
Automated top view registration of broadcast football videos.
- [Sternig et al., 2011] Sternig, S., Mauthner, T., Irschara, A., Roth, P. M., and Bischof, H. (2011).
Multi-camera multi-object tracking by robust hough-based homography projections.
pages 1689–1696.
- [Sun et al., 2021] Sun, P., Cao, J., Jiang, Y., Zhang, R., Xie, E., Yuan, Z., Wang, C., and Luo, P. (2021).
Transtrack: Multiple object tracking with transformer.
- [Tan et al., 2021] Tan, L., Huangfu, T., Wu, L., and Chen, W. (2021).
Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification.
BMC Medical Informatics and Decision Making, 21(1).
- [Tian et al., 2020] Tian, H., Zheng, Y., and Jin, Z. (2020).
Improved RetinaNet model for the application of small target detection in the aerial images.
IOP Conference Series: Earth and Environmental Science, 585(1):012142.
- [Tsourounis et al., 2022] Tsourounis, D., Kastaniotis, D., Theoharatos, C., Kazantzidis, A., and Economou, G. (2022).
Sift-cnn: When convolutional neural networks meet dense sift descriptors for image and sequence classification.
Journal of Imaging, 8:256.

- [Viola and Jones, 2001] Viola, P. and Jones, M. (2001).
Robust real-time object detection.
volume 57.
- [Wang et al., 2022] Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2022).
Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors.
- [Wang and Li, 2022] Wang, T. and Li, T. (2022).
Deep learning-based football player detection in videos.
Computational Intelligence and Neuroscience, 2022:1–8.
- [Wang et al., 2014] Wang, Y., Zhai, R., and Lu, K. (2014).
Challenge of multi-camera tracking.
- [Wojke et al., 2017] Wojke, N., Bewley, A., and Paulus, D. (2017).
Simple online and realtime tracking with a deep association metric.
- [Xu et al., 2005] Xu, M., Orwell, J., Lowey, L., and Thirde, D. (2005).
Architecture and algorithms for tracking football players with multiple cameras.
IEEE Proceedings - Vision, Image, and Signal Processing, 152(2):232.
- [Zhang et al., 2021] Zhang, Y., Wang, C., Wang, X., Zeng, W., and Liu, W. (2021).
FairMOT: On the fairness of detection and re-identification in multiple object tracking.
International Journal of Computer Vision, 129(11):3069–3087.
- [Zhou et al., 2020] Zhou, X., Koltun, V., and Krähenbühl, P. (2020).
Tracking objects as points.
- [Zhu et al., 2017] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017).
Unpaired image-to-image translation using cycle-consistent adversarial networkss.
In *Computer Vision (ICCV), 2017 IEEE International Conference on*.

