
Event detection in soccer using spatio-temporal data

Jens Christian Christensen Jensen, 20022669

Master's Thesis, Computer Science

November 2015

Advisor: Gerth Stølting Brodal



AARHUS
UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

Acknowledgements

I would especially like to thank my supervisor Professor Gerth Stølting Brodal for making it possible to write the thesis on this subject and for always pointing in the direction of deeper understanding. I would also like to show my appreciation of the talks with Thomas Bull Andersen that helped me enter the domain. I am truly grateful to Ottar Dahle at ZXY for providing data to me in his free time. I would also like to thank Olivier Danvy for his extraordinary teaching and seminar. My former teacher and colleague Peter Hveisel Hansen has been a great inspiration and I would like to express my gratitude for countless interesting discussions and for supporting me in achieving a higher level of education. To Allan Grønlund Jørgensen for interesting talks on machine learning. To my childhood friend for having me stay over when doing exams and courses. To Kristján Rafn Gunnarsson that besides his great hospitality, made doing the master's courses quite an enjoyment.

Last would like to thank my family without whom, I would never had been at this place in my life.

*Jens Christian Christensen Jensen,
Sønderborg, November 27, 2015.*

Acknowledgements

I would especially like to thank my supervisor Professor Gerth Stølting Brodal for making it possible to write the thesis on this subject and for always pointing in the direction of deeper understanding. I would also like to show my appreciation of the talks with Thomas Bull Andersen that helped me enter the domain. I am truly grateful to Ottar Dahle at ZXY for providing data to me in his free time. I would also like to thank Olivier Danvy for his extraordinary teaching and seminar. My former teacher and colleague Peter Hveisel Hansen has been a great inspiration and I would like to express my gratitude for countless interesting discussions and for supporting me in achieving a higher level of education. To Allan Grønlund Jørgensen for interesting talks on machine learning. To my childhood friend for having me stay over when doing exams and courses. To Kristján Rafn Gunnarsson that besides his great hospitality, made doing the master's courses quite an enjoyment.

Last would like to thank my family without whom, I would never had been at this place in my life.

*Jens Christian Christensen Jensen,
Sønderborg, November 27, 2015.*

Acknowledgements

I would especially like to thank my supervisor Professor Gerth Stølting Brodal for making it possible to write the thesis on this subject and for always pointing in the direction of deeper understanding. I would also like to show my appreciation of the talks with Thomas Bull Andersen that helped me enter the domain. I am truly grateful to Ottar Dahle at ZXY for providing data to me in his free time. I would also like to thank Olivier Danvy for his extraordinary teaching and seminar. My former teacher and colleague Peter Hveisel Hansen has been a great inspiration and I would like to express my gratitude for countless interesting discussions and for supporting me in achieving a higher level of education. To Allan Grønlund Jørgensen for interesting talks on machine learning. To my childhood friend for having me stay over when doing exams and courses. To Kristján Rafn Gunnarsson that besides his great hospitality, made doing the master's courses quite an enjoyment.

Last would like to thank my family without whom, I would never had been at this place in my life.

*Jens Christian Christensen Jensen,
Sønderborg, November 27, 2015.*

Contents

Acknowledgments	vii
Acknowledgments	vii
Acknowledgments	vii
1 Introduction	1
1.1 The beautiful game	1
1.2 FC Midtjylland	3
1.3 Spatio-temporal soccer data	3
1.4 (Computer) Science and soccer	4
1.5 Event detection	5
1.6 The model and disposition	6
2 Video-based ball detection and tracking	9
2.1 Camera view classification	11
2.2 Soccer ball detection	12
2.3 Sequential ball detection	18
2.4 Inferring 3D ball position	27
2.5 Overall discussion of reviewed methods	36
2.6 Use of ball size to estimate the distance	37
3 Semantics derived from spatio-temporal soccer data	39
3.1 Semantic features of players	39
3.2 Semantic actions with the ball	42
3.3 A tactical feature	43
4 Machine learning methods for sequential data	45
4.1 Emission probabilities for a hidden Markov model	45
4.2 Recurrent neural networks	46
4.3 Agglomerative hierarchical clustering	48

5	Event detection on player data	51
5.1	Available resources	51
5.2	A visualization and annotation tool	56
5.3	Working with radio-wave based data	56
5.4	Features	58
5.5	Preprocessing	61
5.6	Evaluation	61
5.7	Feature selection	63
5.8	Event segmentation	63
5.9	Putting it all together	64
6	Experiments	65
6.1	Setup	65
6.2	Field estimation	65
6.3	Substitution detection	66
6.4	Discretized	68
6.5	Manual features	68
6.6	Test set evaluation	69
6.7	Recurrent neural networks	71
6.8	Event segmentation	71
7	Discussion	75
8	Conclusion	77
A	Machine learning algorithms	79
A.1	Logistic regression	79
A.2	Random forestA	80
B	Matching graphs	81
B.1	Final result	81

List of Figures

1.1	The soccer field.	2
1.2	Fig. 13 in [27] showing the expected requirements to the type of data to extract.	5
1.3	The model used in the thesis	6
2.1	The model and the parts covered in Chapter 2	11
2.2	Fig. 3 in [87] depicting global view, middle view, close-up and out-of-view.	12
2.3	Fig. 1 in [58] showing different ball appearances.	12
2.4	Leftmost images: Fig. 2 in [26] showing the results of edge detection. Right: Fig. 4 in same publication showing the camera setup, and the lighting challenge.	13
2.5	The hidden Markov model.	18
2.6	Fig. 5 in [47] depicting the drift, diffuse and measure steps of the probabilistic propagation process.	22
2.7	The simple model used in [14].	24
2.8	An illustration of the ball line, between the camera at c and the projected ball on the field at a.	27
2.9	(a) The pinhole camera model. (b) Having the image in front of the camera.	28
2.10	The four classes of transformations presented in [43]. From left to right, isometry, similarity, affine and projective.	30
2.11	Fig. 5(a) and 7 from [17] showing the transformation from image plane to model plane.	32
2.12	From left to right Fig. 2 in [55], Fig. 3 in [59] and Fig. in [93] . .	33
2.13	Fig. 7 in [66] showing particle behavior.	35
3.1	The model and the parts covered in Chapter 3	40
3.2	Fig. 7 from [53]. (a) The motion field for three points and the propagation.(b) Calculation of Ψ using magnitude. (c) Graphical representation of values of Ψ . (d) Two GMMs representing the points of convergence.	40
3.3	Fig. 4 in [19]. Upper row: Mean for each player. Lower row: Result of EM using maximal pairing.	41

3.4	Fig. 4 in [60] showing how sets of passes determines a distribution over which quantized field is passed to. In the end the variance for each field can be calculated.	42
4.1	A merge of Figures 3.3 and 3.4 in [37]. A static view of an RNN on the left and the weight updates through the sequence on the right.	47
4.2	Figures 4.2 and 4.3 in [37]. An LSTM block on the left and a small view of a complete network to the right. The black discs represent component-wise multiplication.	48
5.1	The model and the parts covered in Chapter 5	52
5.2	Match view	52
5.3	Web match summary.	54
5.4	A view of one instance of the annotation tool.	56
5.5	A noisy substitute.	57
5.6	Discretization of the field.	59
5.7	A graph of the output. First half of the EFB match. Square nodes are true events and predicted events ellipses.	62
6.1	EFB second half. Pusic is still on the bench, the signal of Hassan is behind the opponents goal but his state has correctly been labeled as Perm_out, making him part of a substitution.	66
6.2	False negative opponent corners in the test set.	71
6.3	Corner event segmentation	73
B.1	FCK half 1 and 2	82
B.2	VFF half 1 and 2	82
B.3	SE half 1 and 2	83

List of Tables

2.1	Various factors influencing the difficulty of ball detection and tracking	9
2.2	The evaluation data used for ball detection in middle view	15
2.3	Ball detection methods for global or unspecified view.	17
2.4	An overview of the methods used for tracking in the image plane.	23
2.5	An overview of the evaluation used in image plane tracking. . . .	26
2.6	Overview of publications in the 3D domain.	34
5.1	The available resources.	53
5.2	The available resources to work on.	55
6.1	Player substitutions	67
6.2	F_2 -score for learning algorithms on the discretized field.	68
6.3	F_2 -score for learning algorithms on the discretized field.	69
6.4	Backwards elimination for logistic regression on the left and Gaussian on the right	70
6.5	F_2 -score for combinations of input and changing parameters. . .	70
6.6	Final results	71
6.7	F_1 -score for RNNs.	72
6.8	Event segmentation results. Left: the proposed algorithm using agglomerative clustering. Right: Splitting each event into k segments of same size.	72

List of Algorithms

2.1	Hough space calculation for circles.	14
4.1	Agglomerative clustering.	49

Nomenclature

BS	Background subtraction is the method of separating images into background and foreground parts.
CDF	Cumulative distribution function
CHT	Circular Hough transform
codec	Coding/decoding algorithm for compressing/uncompressing streams of data
DoF	Degrees of freedom
ED	Edge detection
EM	Expectation-maximization - an iterative approach to locally optimize a hard problem.
Full HD	Full HD(High Definition), a resolution of 1920×1080
GME	Global motion estimation - estimating the motion between frames typically for encoding video.
GMM	Gaussian mixture model
HMM	Hidden Markov model
LSTM	Long Short-Term Memory neural network
PTZ	Pan, tilt, and zoom. The possible movements of a camera with fixed viewpoint.
RNN	Recurrent neural network
SAP	Search around player - searching around players where tracking of the ball was lost.
Seqs	Sequences
SIR	Sequential importance resampling - resampling at every step which is the standard technique in the first particle filters.
SIS	Sequential importance sampling - a precursor to the particle filter.

Chapter 1

Introduction

If you are first you are first. If
you are second, you are nothing.

Bill Shankly

1.1 The beautiful game

Association Football is the world most watched, most followed and most practiced sport with an estimated number of 250 million players worldwide[1]. It is sometimes also referred to as *the beautiful game* but we will use the more generic term *soccer* here. Many games that involved kicking a ball preceded soccer and some of these are also now official sports sharing the term football in their name, such as American Football and Australian Rules Football. The basic game as it is known today was formalized in the second half of the 19th century. Major soccer events as the *World Cup*, the *Eurocup* and the *UEFA Champions League* draw attention form all over the world. There is also a vast international interest in the national leagues of some of Europes countries such as the Spanish league often referred to as *La Liga* and the English *Premier League*.

The game of soccer is played between two teams of each 11 players on a field normally of size 68×105 meters and approximately depicted in Fig. 1.1 . The objective of the game is to get the ball into the opponents goal, while adhering to and without violating the rules which are formalized as the *Laws of the game* [8] but here referred to as the *rules*. The main rule characterizing the game is that no player can touch the ball with any part of the hands or arms anywhere on the field. The exception is the *goalkeeper* that is allowed to touch the ball with any part of his or her body when the ball is inside the team's *penalty area*. The penalty areas are the two largest rectangles in each half of the field with a size of 18×44 yards. To distinguish the players the goalkeepers wear a shirt of different color than all the other players.

Not allowing use of the hands makes the feet the most important part of the body to pass and dribble the ball around the field with, trying to get it into a

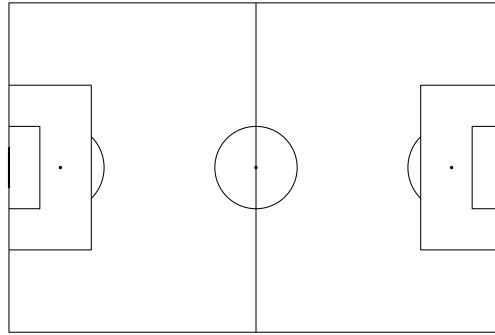


Figure 1.1: The soccer field.

position where it can be shot at the opponents goal, hence the name *football*.

When the ball passes the outer lines of the field known as the touch lines and goal lines a *stoppage* occurs. The side opposite of the one touching the ball last then has to put the ball back into play. On the touch lines it is done by taking a throw-in where the ball is thrown from behind the head with both hands. The feet must touch the ground at release of the ball. If the ball goes into the opponent goal while in play a goal is awarded to the team and the opponent team restarts the game from the center of the field. If the ball went out on a sides own goal line a *goal kick* is taken from the vertical line in the small rectangle inside the penalty area called the *goal area*. If it instead goes out on the other sides half a *corner kick* is taken from the corner of the field where a small arc indicates the legal position for the ball.

There is some allowance for physical contact when fighting for the ball, but when it gets excessive *free kicks* or *penalty kicks* can be awarded to the opposing team by the referee. A penalty kick is given for most infractions committed inside the penalty area which is the larger rectangle close to each goal. A penalty kick is a very good scoring opportunity and the decision to call a foul in the penalty area can therefore be crucial with regards to the result.

Corner kicks, free kicks, and sometimes also throw-ins close to the opponents goal are commonly referred to as *set pieces*.

When infractions are too severe the referee also has the possibility to give out yellow and red cards. A red card results in the player being sent off the field, and the number of players in the team is thereby decreased by one. Two yellow cards to the same player automatically results in a red card. Being one less player on the field is normally a disadvantage and the card-giving decisions by the referee can therefore also in many situations be game changing.

As in many other team sports, *tactics* or *formations* are used to organize the players on the field. There are many variants of these but they tend to organize the players in lines parallel to the goal line. The most well-known tactic is arguably *4-4-2* which denotes that there are 4 defenders, 4 midfielders and 2 attackers. Generally there are typically 3-5 defenders 3-5 midfielders and 1-3 attackers, but the actual placement can vary a lot, which means that there are different subclasses such as *central defender*, *defensive midfielder* etc. Other

tactics are often applied at set pieces such as crowding in front of the nearest goal post on corner kicks.

Given the huge interest in the game, soccer has been a professional sport for many decades. An important concern for most professional clubs is to perform as well in competitions as possible since this often results in increased revenues. At the same time it is necessary to use the normally limited amount of available money, as effectively and efficiently as possible. *Scouting* which is the process of finding new players is very important in both aspects. First of all, having players that fit better in the team allows the team to perform better. Secondly if a player can be bought and then later sold for a much higher amount this has direct impact on the club's economy.

1.2 FC Midtjylland

FC Midtjylland is a danish professional soccer club that was formed on the first of July 1999. The club was formed from two clubs from the 1. division which at the time was the second best danish league. The club won 1. division in the first year, was promoted and has since been playing in the best danish league, Superligaen. 4 times has the club come in at third place, 2 times did it get silver and it won in the season 2014-15[4].

The club was close to bankruptcy in the season 2013-14 but was saved by Matthew Benham, that also came to own it. Matthew Benham owns a company Smartodds that manage data from soccer players around the world. Decisions based on this data led to acquiring the players Awer Mabil form Sydney FC and Daniel Royer from Austria Wien[3] in the summer of 2015.

Scouting is an important but not the only field where the club has a systematic approach. With a specialist coach and dedicated training the club had one of the highest average number of goals scored by set-pieces in all of Europe in the 2014-15 season, coming to score 4 goals from set pieces in a game on 3 occasions.[6].

1.3 Spatio-temporal soccer data

Spatio-temporal soccer data is a set of related positions and time stamps for moving objects in a soccer game. These can be players, the ball, the referee and linesmen. An obvious application of the data in both training and matches is to measure the physical workload of players to measure performance, adjust training and thereby decrease the risk of injuries. Since 2015 the Norwegian company ZXY records spatio-temporal data of the players of FC Midtjylland in all home games using radio-wave technology. The system also records video of the game with a camera array with three cameras, which is mosaiced into a full view of the entire field. The system is described by Pettersen et al. in [71]. Other commercial systems exist of which the most widely known is ProZone[9] which is based on videos placed around the field.

We have received spatio temporal data for the players of 10 home games for the year 2015. Furthermore for some games broadcast video is available of varying quality which we recorded ourselves.

1.4 (Computer) Science and soccer

The scientific fields with relevance to soccer are many and the most prominent are; bio mechanics, sports medicine, physiology, social sciences and humanities[10]. Computer science is a relatively new field to have its methods applied to soccer in various ways and a branch heavily associated to the field is artificial intelligence. Since the way to store the information from a soccer match is usually video, it is natural that computer vision, image analysis, pattern recognition and machine learning are central areas and in relation to these, data mining that gets an increased attention with the availability of spatio-temporal data.

The research in the soccer field starts with Gong et al. [36] in '95 trying to parse soccer programs given different highlights. In '96 Taki [85] analyze the space around players to understand teamwork and Reid and Zisserman [72] analyze a famous old sequence using geometry to see if a goal was scored or not. Seo et al. [80] in '97 try to track both ball and players in video. In '98 Kim et al.[55] assume parabolic trajectories to reason about the position of the ball in 3D in an offline setting, where Reid and North[73] use the shadow of the ball to do it online. Bebie and Berie[16, 17] make 3D reconstruction of a soccer match by manually annotating the position of the ball. Ancona et al.[13] do goal detection using support vector machine classification of ball and no-ball instances. Later Yu et al.[95] observe that the ball only changes direction if it hits the field or a player. In this way, even without knowing the positions on the field of players and ball, it is possible to do some reasoning about the passes being made between players, if trajectories of the ball are given in the image plane.

More research has been performed and regarding ball detection and tracking there are 50+ publications. D'Orazio and Leo[27] in '10 make a review of the video-based research in the field of soccer using three categories to partition the research field. These are: *video summarization*, *augmentation* and *high-level features*. The idea of video summarization is to detect important moments in a match by parsing video to create summaries. Augmentation covers the field of augmenting the viewer experience with e.g. statistics from the match while it is played. The high-level features are considered to be ball- and player detection - and tracking, team statistical analysis and real time event analysis. Even though detection and tracking of ball and players appear in only one category they are also used in the others but with looser requirements to accuracy and processing time. This can be seen from Fig. 1.2

In the ASPOGAMO system[18] presented by Beetz et al. spatio-temporal data is also seen as the fundamental input to a stack of features with each higher feature dependent on the lower and with an increasing semantic level.

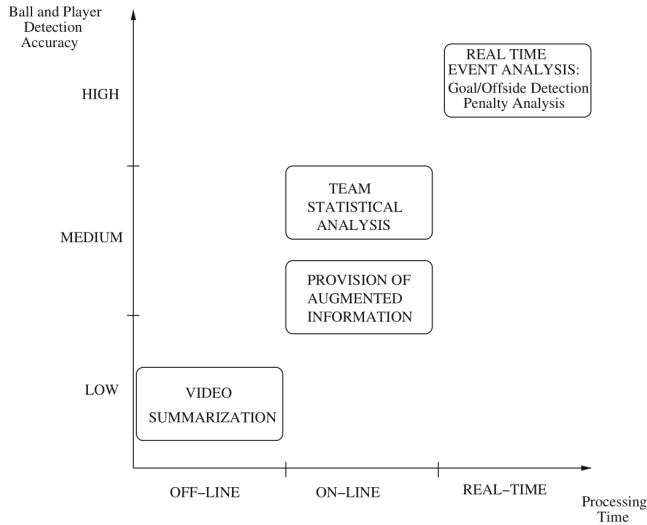


Figure 1.2: Fig. 13 in [27] showing the expected requirements to the type of data to extract.

1.5 Event detection

We will define an *event* to be a stoppage in the game according to the rules thus encompassing e.g. set pieces. Hence passes and dribbles are not classified as events. We will instead denote these as *actions*. We consider an event to have a *start* and an *end* and therefore it also has a *duration*. We will differentiate between an *ideal event* and an *interpreted event*. The start of an ideal event is a point in time where the referee should make a given call according to the rules. The start of an *interpreted event* is defined to be an event that is interpreted by the referee to have happened over a short time interval of e.g. a couple of seconds. Sometimes the start of the event is so obvious that the referee does not need to make a call e.g. when the ball is kicked directly over the touch line without the presence of other players. Other times the referee must make it clear what his/her interpretation of the situation was.

We can also choose to classify events into *ball events* where position of the ball completely determines the start of the event such as corner kicks and goals and *non-ball events* such as substitutions, stoppage due to injury, off-sides etc.

Detecting ideal events is extremely interesting as doing it fully could form the basis of an automated refereeing system. This is however, not an easy task, as especially free kicks and penalty kicks require a substantial model and this is probably only possible using video. The ball events could be easier and we will see literature on using video to do this. A real physical system was applied with success in the 2014 World Cup for goal detection.

The literature is substantial if we instead of helping the referee just want to get interpreted events, i.e. summarize a match, and this is one of the three categories described in [27]. This has mostly been done using video directly and when only considering spatio-temporal data then the position of the ball

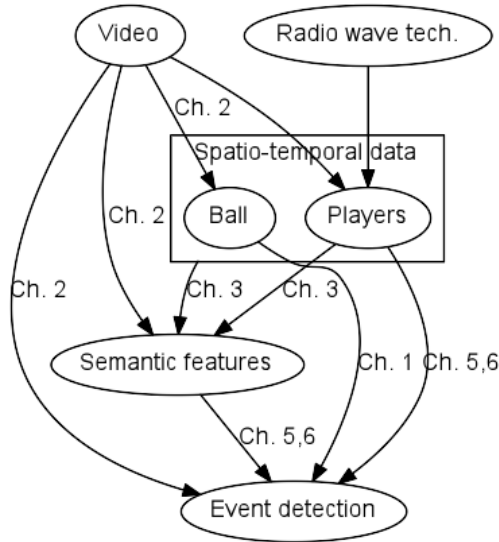


Figure 1.3: The model used in the thesis

is used. Tovinkere and Qian [88] give a long list of actions (which they call events) that they think can be inferred from data such as a player getting up from the ground. The problems as also stated in [27], is that only two events are described in more detail, save and deflection by the goalkeeper, and no indication is given of the matches, events or other relevant information except that it is obtained from a video source. The principal idea is though that if we know the position of the players and the ball then we can infer many interesting things. Gudmundsson and Wolle [38] consider event and action detection based on ProZone player data and manually annotated events. Each event already comes with a time stamp and a location on the field, and the authors reconstruct the ball trajectory based on this data. Levels of events are then created in a bottom-up manner, where on the low-level we have e.g. a “ball-in” event that becomes a “corner kick” event on the next level etc. They note that “yellow card” events are not possible to infer from ball and player data alone. The authors also state that it might be impossible to test the accuracy of the method with the given data.

Event detection using only player data has to our knowledge not yet been done. It is clear though, that there would lie great difficulty in using just player data to infer ideal events, as the players could learn how to behave to trigger the events. Therefore we will have to settle for trying to detect the calls of the referee when using only data from the players.

1.6 The model and disposition

The way that we process the area is depicted in Fig. 1.3. At the top we see the possible raw data that we consider in this work i.e. radio waves and video. Using video to find the ball position and detect events directly is as we have

argued, extremely useful for event detection and in the ideal case could be of help to the referee. Examining this area is the topic of Chapter 2. Video is also the standard approach used by most systems to get player data, but since we have data provided, we will not consider this here directly. It does however appear as part of some of the publications on finding the ball. We do also not elaborate on how to use radio-wave technology to obtain spatio-temporal data, but this is the approach used to deliver the data that we have access to and knowing the source can be important with regards to possible errors.

The other challenge is to do event detection using only player data and we will do this by using machine learning and creating higher-level features. We therefore make a survey in Chapter 3 of how spatio temporal data has been applied in the literature to see what features could be used for event detection. The machine learning methods that we use are not part of the model since other methods could be used. They are also used in relation to some of the other areas so their inclusion would make the Figure more complex and less concise. Chapter 4 describes these methods. The full process from input data to actual event detection on player data is described in Chapter 5. Experiments and discussion thereof based on chapters 4 and 5 are presented in Chapter 6 and we conclude the thesis in Chapter 8.

Chapter 2

Video-based ball detection and tracking

There is only one ball, so you need to have it.

Johan Cruyff

A systematic review of ball detection and tracking publications has been performed. To this end we searched dblp[2] and the first 50 publications at Google Scholar[5] with search terms “soccer ball detect”, “soccer ball track”, and “3D ball soccer”. By reviewing the publications, references were checked to include other relevant publications.

Various factors influence the difficulty of the problem. We have made a coarse overview in Table 2.1 where we on the left side have everything that makes the problem harder and to the right factors that make it easier.

Harder	Factor	Easier
Single	# Frames in sequence	Multiple
3D	Find position in	Frame
Low	Ball information	High
Online	Sequential information	Offline
None	Overlapping views	Multiple
PTZ	Camera dynamics	Fixed

Table 2.1: Various factors influencing the difficulty of ball detection and tracking

A short explanation follows for each of the parameters.

Frames in sequence simply denotes if a method aims to solve the problem of finding the ball in a video sequence or a single image.

Find position in represents if we try to find the coordinates of the ball in the image/sensor plane or in camera or world coordinates. These terms will be elaborated on later.

Ball information is a parameter that covers direct quantitative features such as resolution and size of ball in frame. Ideally it also encompasses the quality of the ball in the frame which is affected by the codec, and the type of camera which is determined by a huge number of parameters where lens and sensor quality are some of the most important. This is too detailed compared to the information that the publications provide. Given a multitude of different resolutions, there will be just two classes for the size of the ball in the frame; high and low ball information which we will explain later.

The *sequential information* factor determines if the proposed algorithm can run in real-time in theory i.e. it is online or if it is inherently offline. Offline meaning that to give an answer at time t it needs to have seen data from time $t + s$ where $s > 0$. The possibility of course exists for a real-time algorithm according to this definition, to not be implementable to cope with the amount of data given, but this is dealt with separately. The distinction is important because we are more interested in the theoretical properties of the approaches than in the speed of a specific hardware platform, as computational resources generally increase over the years.

Overlapping views refer to in the case of more cameras if they are substantially overlapping or simply set up to cover the whole field and thus seldomly having the ball in the view of two cameras at the same time.

Camera dynamics are reduced to *pan*, *tilt* and *zoom*(PTZ) which is based on the following reasoning. First of all none of the publications handle a translating camera i.e. the camera has a fixed viewpoint in the world. Furthermore, we assume that the cameras do not *roll* i.e. while rotation is intuitively understood as looking around horizontally and tilting as nodding ones head up and down, rolling would be the effect of leaning ones head to a side which is not a desirable effect in soccer video footage.

We can divide up the different methodologies hierarchically by the first two rows in Table 2.1:

- Ball detection, that focus on identifying the ball in a single frame.
- Image plane tracking / trajectory generation that use multiple frames but stay in the image plane.
- 3D tracking that focus on getting the ball's position in 3D using multiple frames.

Ball detection is further divided according to the third row, image plane tracking according to the fourth and 3D tracking according to the fourth, fifth and sixth. Let us also note that some dependencies exist between the factors. E.g. finding the ball in a single frame is by definition online, requires no overlapping views, and is largely independent of the camera dynamics.

We can further relate these different approaches to our model in Fig. 2.1.

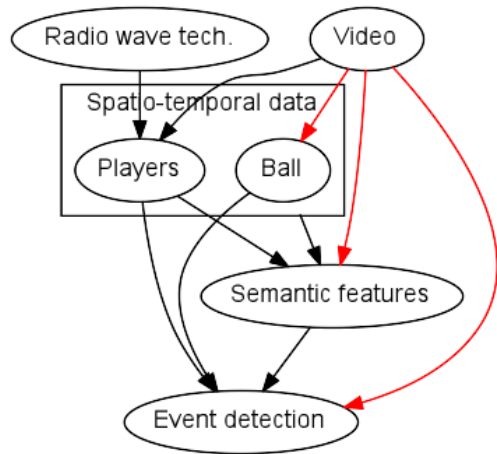


Figure 2.1: The model and the parts covered in Chapter 2

If we can detect the ball in single image we can do goal detection with the right camera setup directly from video, ignoring the absolute position of the ball. Tracking the ball over multiple frames can be used to see if the ball is still for i.e. set piece detection. If it is combined with detection of players, counting passes is also possible. In the model this is depicted as being able to get semantic features directly from video. Finally, the role of 3D tracking is evident since it is exactly spatio-temporal data for the ball.

We will treat these three approaches in the following sections, where we provide tables for summarizing the work. The following legend is used:

P(Publication type):

- C: Conference
- J: Journal
- B: Book chapter

C(Camera setup):

- x F: A number x of fixed cameras
- PTZ: PTZ camera typically from broadcast video

First we will shortly consider the segmentation of broadcast video.

2.1 Camera view classification

In the case that broadcast video is given as input it is necessary to extract the *views* pertaining to the desired cameras. View is also sometimes referred to as *shot* but we choose to use the former term. We will use the naming by Tong

et al.[87] that consider four different classes; *global view*, *middle view*, *close-up view* and *out-of-view* as depicted in Fig. 2.2. In e.g. [70] only three classes are mentioned but for the purpose here it is sufficient to make a distinction between global view and middle view since the ball is rarely present in the others.



Figure 2.2: Fig. 3 in [87] depicting global view, middle view, close-up and out-of-view.

2.2 Soccer ball detection

Common to all of the proposed methods is that the ball has to be detected in as many frames as possible. Sometimes it is not clear if there is a ball or various candidates are present. Often, figures are provided showing some of the ball appearances motivating the work. Fig. 2.3 shows an example.



Figure 2.3: Fig. 1 in [58] showing different ball appearances.

We can choose to see the ball detection as a two-step approach where first some preprocessing is done and afterwards ball candidates are generated. These ball candidates are then treated differently under each specific approach.

An important feature of the input is the resolution of the frames. Combined with the size of the ball in the frame it determines the number of pixels that represent it. A multitude of different resolutions are used in the evaluation part of the publications so we will make a coarse distinction based on a combination of resolution and the previously introduced view types. *High ball-information* is defined as being based on middle view video or global view video in FullHD resolution. *Low ball-information* is thus global view in a lower resolution than FullHD. The reviewed publications considered do not provide different algorithms based on view except Pallavi et al. [70] that consider both global and middle view. In [24] and [64] methods for the middle view are developed but evaluation is also performed on a data set captured by fixed cameras with global view in FullHD.

2.2.1 High ball-information

In the middle view we can see that the ball appears round both from Fig. 2.2 and Fig. 2.4. In the latter we also see the result of applying edge detection to the image. Next we would like to detect the soccer ball which now looks like a circle in the image due to the edge detection. This motivates the coverage of the following theory.



Figure 2.4: Leftmost images: Fig. 2 in [26] showing the results of edge detection. Right: Fig. 4 in same publication showing the camera setup, and the lighting challenge.

2.2.1.1 Background theory - the Hough transform

If a set of points is known to represent a line then least squares fitting can be applied to find the line with the smallest distance to the points. Having multiple lines this can also be done if we know which lines correspond to which points as we can just consider it as the single line problem. However, when an image contains more points that do not belong to any lines or different unknown lines then we need another method. The Hough transform is such a method to find shapes in an image. It is done by converting to parameter space, which sometimes is called the Hough space. Each point will then vote for corresponding parameters in that space and finally local peak values can be looked for, to find the best candidates for the shape. It is apparent that since the parameters generally are real-valued a binnig has to be performed. This means that there is a trade-off between space use and run time of the algorithm on one side and the precision of the found parameters on the other. In Alg. 2.1 we see a naive circular Hough transform. If we let $m = n$ then the time complexity is $O(n^4)$ and the space complexity is $O(n^3)$ and in general if we have p parameters of a shape then it is $O(n^{p+1})$ and $O(n^p)$ respectively. Looping over all the points in the image is necessary in this deterministic algorithm so the hope of doping better in practice is concerned with looping over the shape parameters. First of all, in practice we would not expect to meet the condition in the If-clause very often since most image pixels will not represent edges. Reducing the dimension of H obviously has a direct impact on the time complexity and this could be achieved by lowering the precision and/or we could have narrow restrictions on the radius or the circle center.

Algorithm 2.1 Hough space calculation for circles.

Input: $\text{img}[m,n]$
1: $\text{maxR} = \text{dist}((0,0),(m,n))$
2: $H[m,n,\text{maxR}]$
3: **for** all i **do**
4: **for** all j **do**
5: **if** $\text{img}[i,j] \neq 0$ **then**
6: **for** all k **do**
7: **for** all l **do**
8: $r = \text{dist}((i,j),(k,l))$
9: $r\text{Bin} = \text{int}(r)$
10: $\text{inc}(H[k,l,r\text{Bin}])$
11: **end for**
12: **end for**
13: **end if**
14: **end for**
15: **end for**
16: Return H

2.2.1.2 Literature review

Ancona et al.[13] introduce the problem of detecting the ball with the aim of goal detection using support vector machines. D’Orazio et al.[25] use a similar setup, a single static camera as seen in Fig. 2.4, and the ambition is to run the detection in real-time which they define as faster than 262 frames/second. They also give a definition of when occlusion is happening. If less than 25% of the ball can not be seen then it is visible. Between 25% and 50% is occluded and no-ball cases are beyond 50%. They note that the direction for the gradient is important and not just the magnitude when voting for a circle in the circular Hough transform(CHT). They also improve it to be robust under different lighting conditions. In a later journal publication [26] the method is further improved to a two-step process where the original algorithm is run in the first step. The second step is to process the candidates from step one to extract Haar features and then validate by using a neural network. In [70] the CHT is also used along with Sobel operators, and the Mexican hat filter. Features based on the theory of the scale invariant feature transform(SIFT) are introduced to the domain by Leo et al. [56]. Mazzeo et al.[63] compare different wavelet features for detecting the ball. Another method by Leo et al. [57] based on a 7-step algorithm including the maintenance of a dictionary and classification by naive bayes is compared with the previous work in [56]. De Marco et al. [24] give a randomized approach to circle detection using isophotes curvature analysis as preprocessing. Mazzeo et al. [64] extend this method by extracting features using completed local binary patterns.

2.2.1.3 The ISSIA-CNR Soccer Dataset

In 2009 the D’orazio et al. present a paper on a data set referred to as the ISSIA-CNR Soccer Dataset [28]. The setup consists of six fixed cameras capturing at ~25 frames per second in FullHD resolution for ~120seconds. Calibration information is available for all the cameras and 2D ball and player positions are available for each frame in XML format with all 3026 frames annotated.

2.2.1.4 Evaluation

The evaluation work is substantial but mostly compares to work by the authors themselves or co-authors. This is understandable since none of the other reviewed publications has the specific focus of just detecting the ball. In [57] a comparison is also made with the approach in [70, 77] and the detection accuracy is 87.68% and 82,23% respectively. Pallavi et al.[70] do not provide specific results for detection. The histogram feature in [63] is not far off other more complicated methods. This is only for handling non-occluded balls. For handling occlusion the SIFT based features used by Leo et al. [56] work very well. De Marco et al. [24] only compare the method with the CHT which it beats on the training video with a detection rate of 92% versus 64%, they also provide a comparison with their method to the CHT on [28]. They achieve 86% detection rate versus 79% for the CHT. Mazzeo et al. [64] compare the combination of isophotes curvature analysis and the completed local binary patterns to all the approaches covered previously and find that it is superior for feature extraction with an F_1 -score of around 98%. They report an accuracy of 96.49% on the ISSIA-CNR Soccer Dataset using 7000 frames with a visible ball. The next best achieves 87.68%.

Ref.	P	Evaluation data
[25]	C	512x512, 12 sequences
[26]	J	532x512, 12 sequences
[56]	B	640x480, 200fps, 3560 frames
[63]	J	1280x1024,200fps, 20000frames
[57]	J	1280x1024,200fps, 20000frames
[24][64]	B/J	1024x768,504fps,500

Table 2.2: The evaluation data used for ball detection in middle view

2.2.2 Low ball-information

2.2.2.1 Literature review

The publications dealing with detecting the ball in global view, always do it as a part of tracking the ball either in the image plane or in 3D. The same problems exist as in detection in the middle view and additional problems are also present. With the resolutions used, there are in some cases only a few pixels representing the ball even though it is not occluded. This makes it hard

to detect the ball directly as first noted in [92, 94, 95]. This is also observed in [70] that note the failure of the direct approach using the CHT, given that the radius of the ball is between 1 and 4 pixels.

Since the objective is not the detection in itself, we also include some features here that require the knowledge of multiple frames, but only in an online manner. Examples are velocity and longevity features used by Ren et al. [74][77] or the camera motion by Pallavi et al.[70].

Instead of detecting the ball directly, ball candidates are typically generated by removing everything known not to be the ball. We denote this as the preprocessing step. When using fixed cameras, background subtraction() can be used to leave only the foreground objects. When using a dynamic camera the field is typically found by a range of green colors and then everything outside of it can be discarded. An example of this can be seen in one of the very first publications on the topic by Seo et al.[80]. After the previous operation, connected components labeling(CCL) can be run to enumerate the objects left like the players, the ball, in the case of dynamic cameras the field lines and of course some possible noise.

Different features are then considered to actually find out which of the objects make good ball candidates. Features based on size, shape and color are the most prominent ones as can be seen in Table 2.3.

We have already noted the problem of occlusion. A problem with removing everything not being the ball is that ball pixels will also be removed if the ball is overlapping with, being partially occluded by or even in close proximity of, a player, when assigned to the same object by CCL.

Liang et al.[58] choose to create a mask of white pixels instead of green, however they include a verification based on multiple frames and the Viterbi algorithm, as ball detection. We choose to see this as applying two types of tracking and defer the inclusion of the methodology to the next section.

2.2.2.2 Evaluation

For the global view, the focus is tracking but some provide a separate evaluation of the detection part. Tong et al. [86] compares the run-time of the detection to [25] but do not compare the accuracy. Yu et al. [91] make a comparison citing [26] but do not use that implementation. Instead they feed the ball candidates of their own approach to the CHT ignoring the neural network part and report similar results.

Ref.	Preprocessing	Features
[36]	Gaussian/Lap.	size-/circle threshold
[80]	GM, MF, CCL	TM
[69]	-	Color
[68]	-	Color
[89]	Extract white lines	white regions except players and lines
[92, 94, 95]	CCL	Size, line, color, shape, center color
[75][76]	GM, MCO,BS, CCL[82]	“normalized size, shape and colour”
[86]	-	Shape/color, TM
[22]	BS, CCL	White pixels, TM
[23]	Histogram, CCL	White pixels
[58]	WM, MCO, CCL	Size, MBRratio
[59]	GMM, CCL	[58]
[81][67]	-	Color, size, ratio,separability filter
[91]	[92, 94, 95]	[92, 94, 95]+ penalty mark sieve
[74][77]	BS, CCL[82]	Velocity, longevity, size and color
[48]	-	TM, Players, Image
[87]	GM/Hough	Ecc. Circ.
[66]	-	Centroid, area, orientation
[46]	Histogram-Likelihood ratio,CCL	Ecc, roundness
[90]	Mod.GLA+Sklansky, CCL	Size, FF, Circ.,area ratio
[93]	BS, set of foreground objects	Size, color, shape
[70]	Filtering of non-ball obejcts	Object/camera motion
[14]	BS, labeling	TM
[51]	GRB ² ,ED,CCL	Blob size
[65]	BS as in[54]	Size, Squared error on ellipse
[42]	[65]	[65] + Freeman chain
[24][64]	ED, Isophotes curvature	CLBP[40]

¹GRB refers to the observation that the field color in RGB space exhibits $G > R > B$

BS: Background subtraction

TM: Template matching

Circ.: Circularity

Ecc. Eccentricity

MF: Morphological filtering

GLA: Generalized Lloyd algorithm

GMM: Gaussian mixture model

MCO: Morphological close operation

WM: White mask

GM: Green mask

CM: Camera motion

MBR: Minimum bounding rectangle

Table 2.3: Ball detection methods for global or unspecified view.

2.2.3 Discussion

Ball detection is definitely possible when enough pixels are available. The results in [64] are good but it is necessary to remember that an occlusion is defined as having less than 50% of the ball visible, and all of these are not considered. Since there are 20000 frames in all there are still 13000 frames where the ball either is not present in the view or is having some degree of occlusion higher than 50%. The ISSIA-CNR Soccer Dataset[28], which is publicly available, certainly makes it possible for others to enter the field of research and quickly get an idea of the validity of a new approach.

2.3 Sequential ball detection

The main problem with detecting the ball in a single frame is that it might be partially or fully occluded as dealt with in the previous section. Furthermore it could be far away and in a frame with low resolution very few pixels represent the ball. However, if knowledge from previous and also possibly future frames are available, it should make the problem easier given that the position of the ball is subject to certain restrictions, since it is a physical object. We will first look at some very general and popular theory applied in this area.

2.3.1 Background theory

A graphical model in the probabilistic sense is a graph where nodes represent stochastic variables and edges the conditional dependence between them. The concrete model shown in Fig. 2.5 is a simple such model known as a hidden Markov model(HMM).

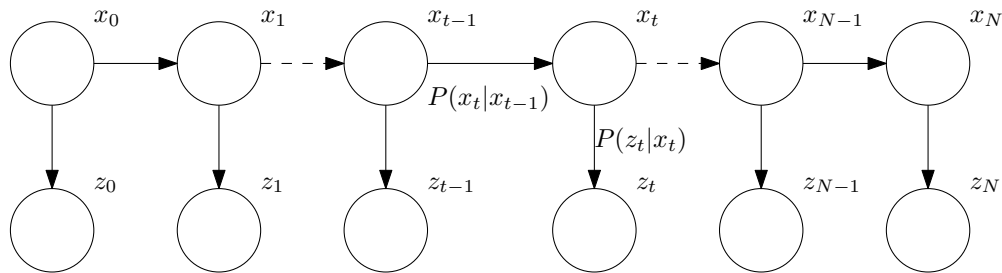


Figure 2.5: The hidden Markov model.

It shows a sequence of hidden variables $X = [x_1, x_2, \dots, x_N]$ that are generally hidden or unknown and observable variables $Z = [z_1, z_2, \dots, z_N]$ that are given. x_{t+1} only depends on x_t making the sequence X a first-order Markov chain. The simplest HMM is a linear model where all variables are discrete. The components of the model are: a $k \times k$ state transition matrix with values for $P(x_t|x_{t-1})$, a $k \times p$ observation matrix representing $P(z_t|x_t)$ and a column vector of length k to give initial state probabilities $P(x_0)$. This model has wide usage in many areas of research, bioinformatics being one of them. For other

applications it can make sense to draw the probabilities for the observables from a probability distribution, typically a Gaussian. Further generalizations exist that we will look at in a moment but let us first consider some problems to solve.

Given such a graphical model different problems arise. One problem is to infer the next hidden state given all previous observations. This is referred to as filtering.

$$P(x_t|z_1, z_2, \dots, z_t) \tag{2.1}$$

It can also be to infer the most probable state given both previous and later observations. This is called smoothing.

$$P(x_t|z_1, z_2, \dots, z_N) \tag{2.2}$$

A third problem is, given a set of observations find the most probable sequence of states to have generated these. This is generally referred to as Viterbi decoding taking its name from the algorithm used to compute it.

$$X^* = \arg \max_X (P(Z, X)) \tag{2.3}$$

The algorithm used to solve the filtering problem when state variables are discrete is known as the forward algorithm and is very similar to the Viterbi algorithm. There is also a backward algorithm which can be used to compute the smoothing problem together with the forward algorithm. When the state variables are not discrete these algorithms do not suffice. The following algorithms solve the filtering problem in more general cases.

2.3.1.1 The Kalman filter

In the case that the observable values are continuous the model in Fig. 2.5 is also called a linear state-space model or a linear dynamical system. The equations corresponding to the model are then:

$$x_t = Ax_{t-1} + w_t \tag{2.4}$$

$$z_t = Hx_t + v_t \tag{2.5}$$

A is the state transition matrix H is the observation matrix and $w_t \sim \mathcal{N}(0, Q)$ and $v_t \sim \mathcal{N}(0, R)$ are the Gaussian noise of state evolution and observation respectively. w_t and v_t are also written w_\bullet and v_\bullet in [79] to denote that the noise is not dependent on t . When tracking objects controlling their own motion a control term Bu_t , where B is the control matrix and u_t is the control input, can also be included in Eq. 2.4, but we do not include it here.

The Kalman filter is of course concerned with estimating the value of x_t which we will denote by \hat{x}_t but the state of the filter also includes a covariance matrix

P_t that tracks the expected error of the filter. In fact, we can treat the state as a Gaussian with mean \hat{x}_t and covariance matrix P_t . This is $P(x_t, z_t)$ i.e. the state a posteriori to the measurement z_t or the posterior. The filter also calculates a priori values or a prior that we denote with \bar{x}_t and \bar{P}_t . Thus we can consider the algorithm as consisting of a prediction step that finds the prior and an update step that uses this to get the posterior.

To find the prior the prediction step uses the following two equations:

$$\bar{x}_t = A\hat{x}_{t-1} \quad (2.6)$$

$$\bar{P}_t = AP_{t-1}A^T + Q \quad (2.7)$$

We see that only variables defined in relation to Eq. 2.4 are used along with the posterior at $t - 1$. The following equations show the update step.

$$K_t = \bar{P}_t H^T (H \bar{P}_t H^T + R)^{-1} \quad (2.8)$$

$$\hat{x}_t = \bar{x}_t + K_t(z_t - H\bar{x}_t) \quad (2.9)$$

$$P_t = (I - K_t H) \bar{P}_t \quad (2.10)$$

The matrix K_t is the Kalman gain matrix. It represents the belief in the model versus the measurement for this specific value of t . It can be defined differently but this has implications for Eq. 2.10 that becomes more complex. Eq. 2.8 is known as the optimal Kalman gain.

The above equations are derived directly by trying to minimize the squared error the filter makes and hence the Kalman filter solves the filtering problem optimally. A derivation of the Kalman filter is quite involved and can be found in [33]. One particular observation in that publication is that the derivation depends directly on the fact that the product of two Gaussians is a Gaussian.

A unifying review of linear Gaussian models including the Kalman filter and the discrete state HMM with Gaussian observations can be found in [79].

2.3.1.2 Particle filter

The main source for this presentation of the particle filter is the popular publication[47] by Isard and Blake that presented it as the CONDENSATION algorithm along with various applications of it in the field of computer vision. The publication is called *CONDENSATION — Conditional Density Propagation for Visual Tracking*, which explains the name of the algorithm. Another name commonly used is *sequential Monte Carlo* but according to Doucet and Johansen[29] this name is better suited for a broader class of algorithms of which the particle filter is just one.

As noted previously the state of the Kalman filter is essentially a Gaussian which is uni-modal, i.e. it only has one peak. Informally we can say that only one value makes sense to pick for the actual position, namely the mean \hat{x}_t . We can therefore not maintain multiple hypotheses of the state. The need to do this is increased with increased non-Gaussianness and non-linearity of the model.

The particle filter solves the filtering problem in the non-linear state-space model with non-Gaussian noise. In that case we have

$$x_t = f(x_{t-1}, w_t) \tag{2.11}$$

$$z_t = g(x_t, v_t) \tag{2.12}$$

where f and g represents the model behavior and w_t and v_t are noise from arbitrary probability distributions.

The fundamental idea of the particle filter is to use a finite set S_t of n samples or particles to approximate the posterior $P(x_t|z_t)$:

$$S_t = \{s_t^{(1)}, s_t^{(2)}, \dots, s_t^{(n)}\}$$

In that way multiple hypotheses can be represented. In many cases there is no way to draw samples directly from the posterior and instead we can use what is known as importance sampling, that samples from a known distribution and then weights the samples to get $P(z|s_t^{(n)})$.

The CONDENSATION algorithm is presented as consisting of three steps; select, predict and measure. If we to begin with ignore the select step, we can consider the predict and measure steps to be equivalent to the steps of the Kalman filter, i.e. evaluating the state and observation respectively, just that the evaluation is performed using importance sampling. We can further divide up the predict step into a deterministic and a stochastic part, called drift and diffuse respectively and consider them steps in a probabilistic propagation process as depicted in Fig. 2.6

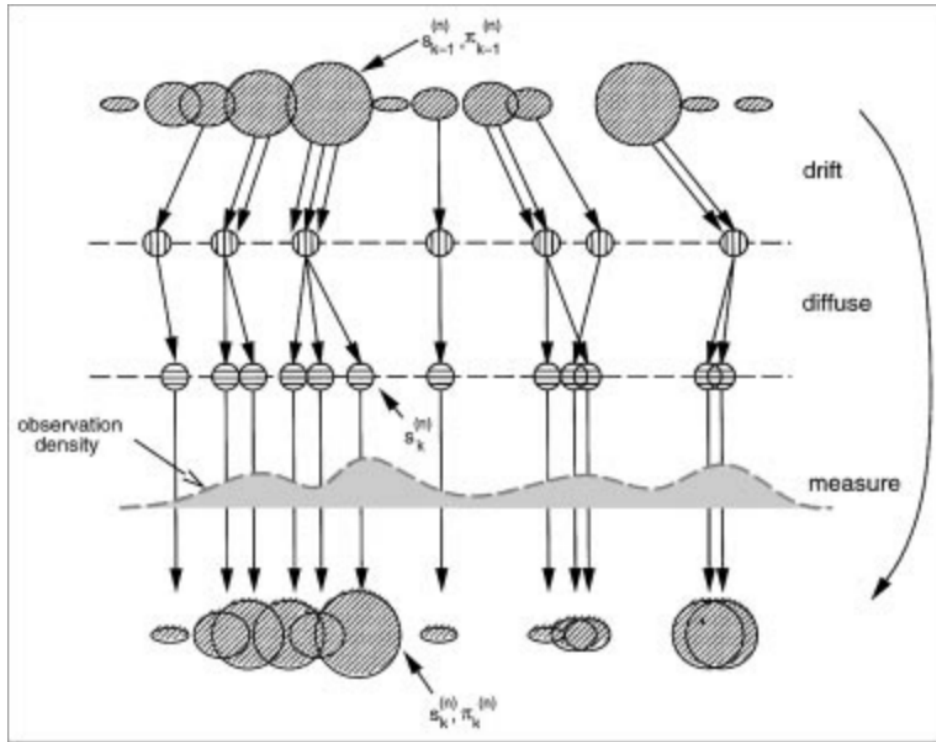


Figure 2.6: Fig. 5 in [47] depicting the drift, diffuse and measure steps of the probabilistic propagation process.

This process then consist of three steps; drift, diffuse and measure that the particles undergo. For the sake of example we divide the function f up into a deterministic part and a noise part equivalent to Eq. 2.6 possibly non-linear and non-Gaussian respectively. Then we view the deterministic part as the drift and the stochastic part as the diffuse. If two particles at time t are sampled from the same particle from time $t - 1$ then they also have equal values after drifting. Application of the noise in the process then makes the particles diffuse. Finally the posterior must be calculated by including the measurement, which is done by calculating and normalizing the weights for the particles based on how well they fit the observation model. This scheme is known as sequential importance sampling(SIS) and it in fact can solve the filtering problem in the non-linear and non-Gaussian case. However, it suffers from a problem called the degeneracy problem which basically means that after few iterations most particles have weight close to zero which makes the computation of the weights pointless[15]. This is the reason for the select step which at each iteration selects particles based on their weights, so as to avoid the computation of irrelevant particles and instead pick more relevant ones using sampling with replacement also known as bootstrapping. The inclusion of this resampling step at each iteration is what characterizes the general particle filter and is also referred to as sequential importance resampling(SIR). It solves the degeneracy problem but introduces the impoverishment problem, which consists in that the particles

end up in very few places because bootstrapping is used. This is especially the case if the state transition has little noise, i.e. the particles do not diffuse much. Many other approaches apart from SIR have therefore been proposed to make better resampling[15].

2.3.2 Literature review

We can consider the problem of detecting the ball either given only past frames or all frames including future ones. In the current context we will discriminate between online ball tracking or offline trajectory generation. We have seen algorithms for the filtering problem and these are also represented in the literature as can be seen in Table 2.4.

Ref.	OL	Tracking	KF	PF	OP
[36]	-	Similarity in neighboring frames			
[80]	-	KF + SAP	X		
[69]	-	FD, SAP, Parabola fit			
[92, 94]	-	Trajectory cand. by KF + selection alg.	X		
[95]	-	[92, 94]+ Gap interpolation	X		
[86]	+	PF		X	
[22]	-	PF+SAP		X	
[23]	-	[22]		X	
[58]	-	KF,DAG+Viterbi	X		X
[81][67]	-	¹ , SAP,transition graph,			X
[91]	-	[95]	X		
[74]	- ²	KF + states + Temporal hysteresis	X		
[87]	-	Graph of trajectories+Dijkstra, CSI	X		X
[90]	-	DAG + 2xMod.Viterbi			X
[70]	-	DAG + DP(Longest path)			X
[14]	+ ³	PF		X	
[51]	+	Dynamic KF + Velocity control	X		

¹ Tracking: $x(t) = x(t-1) + (x(t-1) - x(t-2)) + (x(t-2) - x(t-3))/2$

² A comparison is made. See text.

³ The background is created offline.

OL: Online

KF: Kalman filter

PF: Particle filter

OP: Optimal path

FD: Frame difference

SAP: Search around player

CSI: Cubic spline interpolation

Table 2.4: An overview of the methods used for tracking in the image plane.

There is a column denoting the use of some optimal path graph algorithm. The use of any such algorithm immediately suggests that the offline problem

is considered. First, that is the way the problem is defined probabilistically in 2.3. Secondly, there is no reason to remember past states beyond the very last in the online tracking problem, unless it is desirable that the tracker suddenly “jumps” from one place in the frame to another because a better optimal path was found. None of the methods exhibit this behavior as far as our analysis has gone.

The overall idea in trajectory generation is to remove as much as possible and find good ball candidates[92, 94]. These can then be used to find trajectory candidates and finally find the best trajectory and interpolate gaps.

When attempting to do online tracking the typical approach is to start by detecting the ball in the whole image, and then use a filter to track it using template matching(TM) until it is lost. It is then possible to restart the search on the whole image as depicted in Fig. 2.7. A common idea is to search around the player where it was lost. We call this approach the *search around player* approach(SAP). This obviously incorporates player tracking as well. An extension to this is the approach presented in [81][67] that not only search around players but also the stands, lines and goal posts, and by using a transition graph they can exclude some candidates.



Figure 2.7: The simple model used in [14].

2.3.2.1 Publications using the Kalman filter

In [80] only the use of the filter for player tracking is mentioned explicitly, but it is stated that the same approach is used for ball tracking. Yu et al. [92, 94, 95, 91] apply the filter as a sub procedure of a ball trajectory mining procedure, and also use it for gap interpolation when the ball is occluded. Liang et al. [58] model the position and velocity of the ball in the algorithm giving a 4-dimensional state space. Ren et al. [74] takes modeling even further by using object centroid and velocity and top left and bottom right corners of the bounding box giving an 8-dimensional state space. This allows them to use different ball states based on if player and ball bounding boxes overlap. They also use hysteresis similar to what is used in Canny edge detection. Kim et al. [51] stay in 4 dimensions but use domain specific observations to modify the state of the algorithm indirectly by manipulating the Q and R matrices. They set them to $Q = qI$ and $R = rI$, and depending on the mode which can be measurement, player occlusion or prediction mode, q and r are manipulated to favor the measurement in the first two modes and prediction in the latter. To handle interceptions of the ball by players they directly change the velocity of

the state transition matrix.

2.3.2.2 Publications using particle filters

The main publication on CONDENSATION[47] is cited in [86] and the tracking part is mostly restating the equations of it. Choi and Seo use particle filter in two publications differing mainly in camera setup which is fixed in[22] and dynamic in [23]. They claim to use a novel approach by using accumulated ball trajectories but they do not capture these in the posterior. The model then does not have the Markov property but more importantly they also need an interval of a certain size to batch process the trajectories which makes it an offline method. Ariki et al. [14] model the prior equivalently to [58] and [51] i.e. by position and velocity and Gaussian noise.

2.3.2.3 Applications of ball tracking

Yu et al. [95, 91] use the generated trajectories to count passes. Ariki et al. [14] show in their publication on ball detection that various events can be inferred by simply considering points in time where the ball is completely still for more than 6 seconds.

2.3.3 Evaluation

Of the 19 reviewed publications for online tracking and trajectory generation in the image plane, none of them provide any theoretical proof of their method. 7 do not provide any quantitative evaluation of the method. Of the remaining, all but two, use recorded ground truth of the ball which makes automatic evaluation possible. The authors of [81] and [67] provide manual quantitative evaluation by looking through some sequences and observing if they behave well, and then reporting the found quantities. Two publications make a comparison of the detection rate but only one publication [70] compares the approach quantitatively to another[58] within the same area. In the latter a link to a demo of the system is also given.

Ren et al. [74] compare different sizes of back tracking buffers to see how it affects performance, and it there is substantial improvement.

The modified Kalman filter of Kim et al. [51] at a threshold of 20 pixels gets an accuracy of 78.5%, which is a huge gain over the normal filter that scores 41.5%.

A special note on the use of particle filters is that it is possible to depict the particles in the resulting image, but this is not done by any of the authors.

Ref.	P	C	Evaluation data	QE	GT	S
[36]	C	PTZ	30-min. soccer program	-	-	-
[80]	C	PTZ	< 150 frames	-	-	-
[69]	C	?	320x240,30fps, 300 frames	-	-	-
[92, 94]	C	PTZ	2x1000 frames	X	X	-
[95]	C	PTZ	[95]	X	X	-
[86]	C	?	100 Samples	-	-	RT
[22]	B	2F	1361x480, 800 frames	-	-	-
[23]	C	PTZ	960x540, 600 frames	-	-	-
[58]	B	PTZ	352x288 25fps, 200-1000	X	X	-
[81][67]	C/J	PTZ	480x240, 17279 frames	X	-	-
[91]	J	PTZ	2085+52437 frames	X	X	-
[74]	J	1F	3 seqs, 4800 frames	X	X	-
[87]	B	PTZ	720x576, 25 fps, 1497 frames	X	X	RT
[90]	B	PTZ	2 seqs > 500 frames	-	-	-
[70]	J	PTZ	>1000 frames	X	X	¹
[14]	C	1F	1280 x 720, 800 frames	X	X	²
[51]	C	PTZ	592x320, 1000 frames	X	X	³

¹ 27.2s / frame

² 0.9-3.2s / frame

³ ~170ms/frame

QE: Quantitative evaluation

GT: Ground truth

KF: Kalman filter

PF: Particle filter

S: Speed

RT: Real-time

C: Comparison

Table 2.5: An overview of the evaluation used in image plane tracking.

2.3.4 Discussion and future work

Only Tong et al.[86] and Kim et al. [51] present actual online algorithms. In the first there is no evaluation and in the latter we find that a threshold of 20 pixels is quite large, but we do not know the size of the ball in the experimental data. Tong et al.[87] state that the algorithm runs much faster than real-time but the algorithm in itself uses batches and gap interpolation which obviously makes it an offline algorithm. As is observed in [74] there are large benefits associated with running the algorithm offline, and this seems to be the conclusion, that tracking the ball is really difficult to do in real-time if a high accuracy is wanted. This really comes as no surprise since it seems difficult to reason about where the ball is going next when it is occluded.

As discussed regarding ball detection, the use of a general data set with associated ground truth would make it very fast to get an overview over the effectiveness of the methods presented. Since access to the ISSIA-CNR Soccer

Dataset t[28] is public we would see at as the next step to implement and run methods on this to see what works and what does not. Otherwise, for authors with new ideas in the field, we suppose direct comparison is possible using the data from [58] and [70]. This also applies to online tracking that would be able to test against offline algorithms and pure ball detection as presented previously.

2.4 Inferring 3D ball position

We have seen that we in many cases can find the ball directly and also that we can generate trajectories even if it is difficult to do it online. The 3D tracking problem can be seen as an extension to the image plane problem, where we are asked to provide real world coordinates for the position of the ball.

Assuming that we have detected a ball in a frame and know how to transform this position to world coordinates it will give a line on which the ball will be placed. We will call this line the ball line and a simple model can be seen in Fig. 2.8.

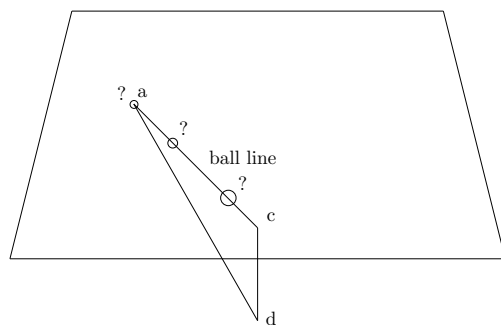


Figure 2.8: An illustration of the ball line, between the camera at c and the projected ball on the field at a .

The 3D tracking problem can thus be seen as solving the following two cases of finding:

1. a transformation between image and model plane
2. the actual position on the ball line

The difficulty of these two challenges are largely related to two of the parameters of Table 2.1 i.e. camera dynamics and overlapping views.

If a camera is fixed the transformation can be calculated once, otherwise it has to be inferred for each frame. Having the transformation not only solves part of the ball problem but is also a fundamental part of tracking players the tecton of which is a priori much simpler.

The other parameter, that we have not considered before in the detection and image plane tracking cases is introduced, which is overlapping views. Overlapping views can be attained by using multiple cameras, and this in theory solves the second problem given the ball is detected in at least two views.

The most difficult case to solve is therefore a setup consisting of a single dynamic camera as the transformation must be obtained for each frame and no other camera is there to help tell where on the ball line the ball is located.

First we will consider some theory related to making transformations between the image plane and the field plane.

2.4.1 Background theory

2.4.1.1 The pinhole camera model

The simplest camera that one can make without the aid of electronics is basically a dark box with a small hole in it. Light will go through the hole and be projected on the opposite surface. If we then put photoreceptive material on this surface we could make a photograph. The drawback of such a camera in practice is that very little light comes through the hole. Modern cameras therefore have one or more lenses that allows a camera to capture more light. This addition however, introduces the need of adjusting the length between the lens(es) and the surface to which is projected, also called the focal length. It also introduces lens distortion turning straight lines into curves depending on the wideness of the angle.

For many purposes of theoretically working with a camera, we can assume that the camera is a point in space, and thereby ignore any parameters associated with more complex models. This model is known as the pinhole camera model.

If working strictly with the pinhole camera model we would have to consider all images in inverted form. It is therefore that we instead of considering the projection through a hole consider the image to be in front of the camera and a line going from the observed point in the world through the given pixel and ending in the camera as depicted in Figure 2.9

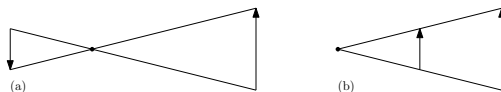


Figure 2.9: (a) The pinhole camera model. (b) Having the image in front of the camera.

2.4.1.2 Homogenous coordinates

Given images in 2D, that come from observations of the real world in 3D, the process of capturing an image is a singular, i.e. irrevertible transformation, a projection. Even though the image would come from an observation of a plane, nothing can be said about the rotation or scale of that plane in the real world without additional information. Some properties are preserved during the transformation. For instance lines are still lines, but parallel lines in the world are not parallel in the image in the general case. In the real plane, parallel lines are degenerate in that they do not intersect. Our intuition might suggest

to think of parallel lines as intersecting at infinity, and it is this idea that the homogenous coordinates allow us to formalize and reason about mathematically. To be able to treat infinity linearly, a third coordinate is needed. We introduce this by looking at the transformations between the two coordinate systems:

$$(x, y) \rightarrow (x, y, 1) \tag{2.13}$$

$$(x, y, z) \rightarrow \left(\frac{x}{z}, \frac{y}{z}\right) \tag{2.14}$$

We can see that if we use 1 for z in Eq.2.14 then it gives the left side of Eq. 2.13. If $z = 0$ then translating to cartesian would give $\left(\frac{x}{0}, \frac{y}{0}\right)$ for any x and y , but in the homogenous world we now have a point at infinity that does depend on x and y . We can also observe that for any $k \neq 0$ $(kx, ky, kz) \rightarrow \left(\frac{x}{z}, \frac{y}{z}\right)$. This means that we can scale a point in homogenous coordinates by whatever we like and it will always correspond to the same cartesian point. If we choose to view the point (x, y, z) as a vector $\mathbf{x} = (x, y, z)^T$ then we can consider \mathbf{x} to be equivalent to any vector $k(x, y, z)^T$ for any $k \neq 0$ and such vectors are then said to be homogenous.

Since points can be scaled we do not need to consider the whole 3D space that an extra coordinate gives. It suffices to consider the two hyperplanes, where z has the values of 1 and 0. We can consider the hyperplane defined by $z = 1$ the real plane. The hyperplane defined by setting $z = 0$ represents points at infinity excluding the origin. **►Must be explained a little better◄**

Remarkable properties hold for typical operations. The intersection point between two lines can be calculated as the cross product of the two line vectors, and similarly the cross product of two points is the line going through them. In fact, any proof on lines can be directly converted to a proof on points and vice versa. This is called the Duality Principle and stated as Result 2.6 in [43]. Here we are mainly using the fact that homogenous coordinates make otherwise nonlinear transformations linear.

2.4.1.3 Homographies and other transformations

A planar projective transformation or homography is a linear transformation on homogenous 3-vectors represented by a non-singular 3×3 matrix:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

, which we could also write $\mathbf{x}' = H\mathbf{x}$, letting \mathbf{x} be $(x \cdot y \cdot z)^T$ and \mathbf{x}' be $(x' \cdot y' \cdot z')^T$.

Since homogenous coordinates are invariant to scaling there are only 8 ratios that determine H and we therefore say that it has 8 degrees of freedom(DoF). Simpler transformations also exist with fewer DoF and we can classify all of them into four groups as depicted in Fig. 2.10.

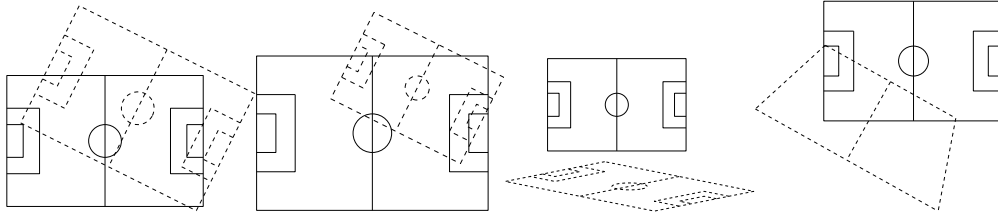


Figure 2.10: The four classes of transformations presented in [43]. From left to right, isometry, similarity, affine and projective.

A Euclidean or isometric transformation uses a 2×2 rotation matrix R with one parameter and translation in 2D by a column vector \mathbf{t} giving 3 DoF.

$$\begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

Length is preserved so we call it an isometry. This is not the case for the similarity transform that uses another parameter s specifying the scale level and thereby has 4 DoF.

$$\begin{bmatrix} sR & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

Here, angles are still preserved. The affine transformation is given by a matrix A instead of sR . A is defined as $A = R(\theta)R(-\phi)DR(\phi)$. and D is a diagonal matrix. The intuition is to rotate by ϕ and then scale in 2D by D , rotate back by $-\phi$, and then rotate by θ . Compared to sR this has 2 more parameters and the affine transformation thus has 6 DoF.

$$\begin{bmatrix} A & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}$$

Still parallel lines are preserved and we can observe that we have not yet actively used the third dimension. However, when the transformation is a projection then we need to handle the intersection of parallel lines. This is done by using the row vector \mathbf{v}^T corresponding to variables h_7 and h_8 in the matrix H . h_9 can now not be enforced to be 1 so the variable v takes its place.

$$\begin{bmatrix} A & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix}$$

The invariant left is the cross ratio of four collinear points.

For more details on the subject e.g. how to decompose the projection matrix and more invariants for each class, we refer the reader to [30] and [43].

2.4.1.4 Estimating a homography

Estimation of a homography can be done by a large number of algorithms. Agarwal et al.[12] make a survey of the theory and provide experiments for 12

different. Here we will look at the most naive which is a *direct linear transformation* (DLT). If we assume that we have a point $p : (x, y)$ in an image and a point $p' : (x', y')$ in the world e.g. on a soccer field we can create vectors representing the homogenous coordinates letting \mathbf{x} be $(x \ y \ 1)^T$ and \mathbf{x}' be $(x' \ y' \ 1)^T$. The homography matrix H relates these as $\mathbf{x}' = H\mathbf{x}$ or

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

To get equations allowing to solve for the parameters we will look at the in-homogenous coordinates corresponding to \mathbf{x}' , i.e. what we get by dividing both sides by the equation of the third coordinate,

$$x' = \frac{x'}{1} = \frac{h_1x + h_2y + h_3z}{h_7x + h_8y + h_9z}$$

$$y' = \frac{y'}{1} = \frac{h_4x + h_5y + h_6z}{h_7x + h_8y + h_9z}$$

that further can be written as

$$-h_1x - h_2y - h_3 + x'(h_7x + h_8y + h_9) = 0$$

$$-h_4x - h_5y - h_6 + y'(h_7x + h_8y + h_9) = 0$$

Based on these equations we can set up a matrix A_i for the i 'th point pair and write the above equations in matrix form as

$$A_i = \begin{pmatrix} -x & -y & 1 & 0 & 0 & 0 & x'x & x'y & x' \\ 0 & 0 & 0 & -x & -y & 1 & y'x & y'y & y'x \end{pmatrix}$$

and thus solve $A_i\mathbf{h} = \mathbf{0}$, where $\mathbf{h} = (h_1 \ h_2 \ h_3 \ h_4 \ h_5 \ h_6 \ h_7 \ h_8 \ h_9)^T$.

If we pick four point pairs we can stack A_1, A_2, A_3 and A_4 into a matrix A and solve for the eight ratios.

Given more than 4 points we have an overdetermined linear system and standard least squares can be used or other suitable error functions. However, in the case of noise it can be preferable to normalize the points to reduce the amplitude of error giving that the impact of the noise depends on the origin and the scale. Last, it is worth noting that the only approach not using points or lines but based on the fast fourier transform, performs better than all others when the error is low in the experiments by Agarwal et al.[12].

We have applied a DLT to a soccer scene and the result can be seen in Fig. 2.11. .

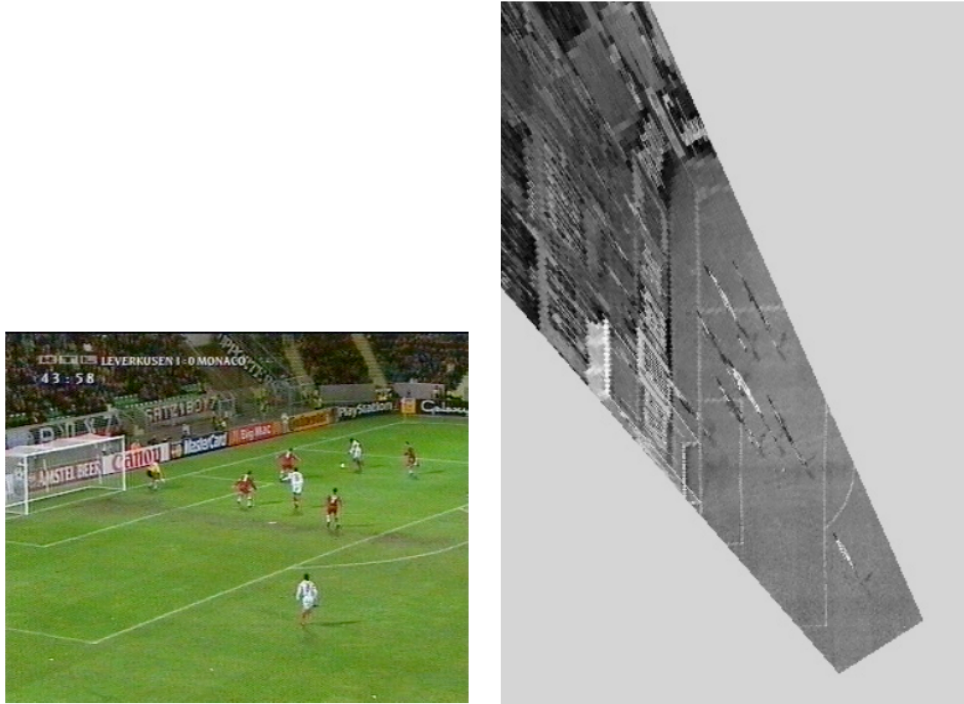


Figure 2.11: Fig. 5(a) and 7 from [17] showing the transformation from image plane to model plane.

2.4.2 Soccer-specific camera calibration

For calibrating the cameras in the soccer domain only two of the research groups considering fixed cameras, state the methods used for calibration. Ren et al. [76, 77, 78] uses an old technique which uses 3D objects. Metzler et al.[65][42] use the method by Zhang[96] that uses a planar surface with a checkerboard pattern and homographies to estimate the parameters.

Given video of a dynamic camera the problem is for each frame to find a homography between the plane and the model, since the parameters pan, tilt and zoom can change frame by frame. Liu et al. [59] note that this can be done by finding a homography matrix H and then correct this along the way based on global motion estimation(GME). This is actually the approach, where it is stated, that is deployed by the authors of all reviewed publications except by Yamada et al. [89]. They match pixels between frame and model by searching “a large parameter space” but it is unclear if this is done automatically.

A homography matrix can, as we have seen, be calculated based on at least four point pairs. In Fig. 2.12 the candidate points used are shown for each publication.

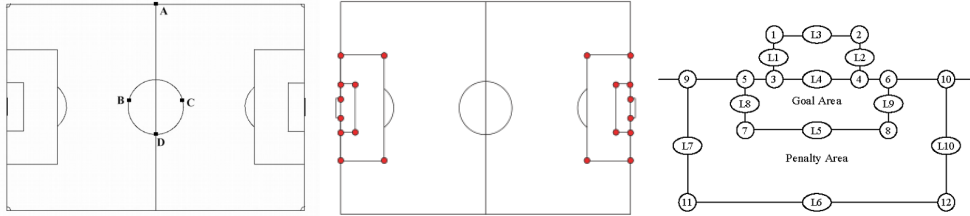


Figure 2.12: From left to right Fig. 2 in [55], Fig. 3 in [59] and Fig. in [93]

In [55] no mention is made of how the width of the field is found. The absence of this absolute knowledge is the motivation for the restricted set of points in [59]. Yu et al. [93] are particularly interested in events close to the goal. No publications claim to find the points in the frames automatically so we assume that it is done manually. However, Yu et al. [93] provide a method “homography tuning” that adjusts the homography when enough points are available. This is based on the Hough transform.

To keep the homography updated or find a new one for each frame Kim et al. [55] use mosaicing. In [59, 73] methods for finding optical flow are used. In [93] the method by Dufuax and Konrad [32] is used.

2.4.3 Overlapping views

The problem of using multiple cameras that overlap to find the 3D position of the ball is geometrically very simple if the ball can be detected in a frame on at least two cameras. If also a perfect camera calibration exists, the 3D position is the intersection of the ball lines and the problem is solved.

Perfect values are not available in practice and in [76] it is shown how to handle the case when the lines do not intersect by simply using least squares on the distances between the ball lines.

Misu et al. [66] present a distributed particle filter that works with any number of cameras. The idea is to have a particle filter receiving particles from all available cameras to infer the ball’s position. If the ball is occluded or a unit is suddenly offline the particles spread out because the uncertainty of the position increases. The state is modeled as position, velocity and acceleration giving 9 dimensions. The state transition is a composite function based on parabolic flight, bounce and decelerating rolling due to friction. The observation model uses centroid, area and orientation as shown in Table 2.3. Ishii et al. [48] specifically choose the Kalman filter over the particle filter due to the computational complexity of the latter in 3D.

Ohno et al. [68] use cameras with overlapping views but it is unclear how they take advantage of this.

2.4.4 Single camera

Inference of the spatial position of the ball captured by a single camera is the most difficult case to handle, and can be the case even in a multiple camera

system if the cameras do not overlap or if the ball is only visible in one view.

2.4.4.1 Offline trajectory generation

The idea used for reasoning about the trajectories in 3D is that the ball follows a parabola thus ignoring air friction. Kim et al. [55] are the first to propose the idea. The height of the ball is estimated via the height of players but the authors mention the preference for using a goal post if it is available. Both start and end of the trajectory have to be annotated manually. Liu et al. [59] note that just fitting a parabola is not robust and they introduce an extra term that depends on the variance of the distances between each pair of subsequent points. This favors trajectories where the ball moves with constant velocity in the xy -plane. Ren et al. [76] also consider the single camera and observe that the ball only changes direction abruptly when it hits the ground, a goal post, a cross bar or a player. If only the ground is nearby it can therefore be inferred that the ball hit the ground and its absolute position is then known at that point in time. Metzler and Pagel [65] introduce a tolerance function and a tolerance factor to allow deviations from a true parabola. They claim to have a real-time algorithm that only has slight delay due to the use of motion history to generate the trajectories, but here we classify it as offline

2.4.4.2 Online tracking

Reid and North[73] use the shadow of the ball to provide an intersecting line for the ball line. This approach requires the presence of shadows created by light sources such as direct sunlight or projectors which are obviously not always there. Furthermore the shadows must be detectable automatically. The authors do the ball and shadow detection manually.

Ohno et al. [68] and Yamada et al. [89] model the 3D position by including gravity and air friction but do not solve for the initial velocity. They claim to solve the problem for fixed and broadcast cameras respectively, but in a later publication by one of the co-authors [67] tracking in 3D is seen as future work.

	P	C	OL		GT
[55]	C	PTZ	-	150 frames[80]	-
[73]	C	PTZ	-	-	-
[68]	C	8F	-	-	-
[89]	C	PTZ	-	-	-
[75] [76, 77, 78]	C/J/J/J	8F	X		X
[59]	J	PTZ	-	720x572	-*
[48]	C	2F	-	640x480, 30 fps, 1297 frames	X
[66]	C	2F	X	1920x540, 14.985 Hz ~5s	X
[93]	C	PTZ	-	5 seqs 172 frames	-*
[65][42]	C	1F	-	Full HD, 140000 frames	-

*Ground truth provided but not for 3D positions

Table 2.6: Overview of publications in the 3D domain.

2.4.5 Evaluation

Yu et al. [93] essentially improve their previous image plane algorithm that suffered from broken trajectories due to camera motion by taking this into account, but they do not generate 3D data. They do not give any evaluation of how well they achieve the correct camera parameters when not observing the goalmouth. Ren et al. [76] present quantitative evaluation based on ground truth. Ground truth is obtained manually at each 25th frame and linearly interpolated and the error is calculated in the xy -plane. They estimate the calibration error to be 2.5m and they report 77% of the positions to be within a margin of 3m. In [77] a correct result lies within 5m, and in [78] it is also 3m. As in the paper on tracking they find that allowing a delay of 2-3 seconds to backtrack the trajectory drastically improve the results as without it the detection rate is around 33%.

Misu et al. [66] show much better results on 4 points spread over 5 seconds but do not state neither the resolution, number of frames nor how the ground truth was obtained. Fig. 2.13 shows how the particles behave after a disconnect and again after reconnecting.

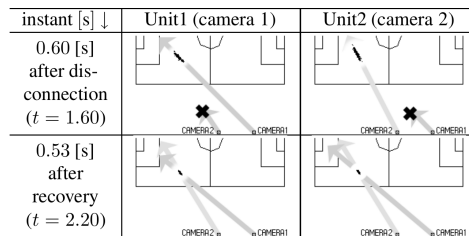


Figure 2.13: Fig. 7 in [66] showing particle behavior.

2.4.6 Discussion

We note that the reviewed publications put little emphasis on making camera calibration automatic. This is especially important for the publications using broadcast video since shifting between different camera views then requires a new manual calibration. Even with only contiguous frames from the center camera, the motion estimation is likely to introduce errors in the homography and we would assume that continuous recalibration based on points on the field would make the algorithm more robust. The publications can therefore in general be considered to provide ideas more than full approaches. An exception to this are the publications by Ren et al. but they suffer from relatively bad results. If the system were to be used for detecting ball out of play, the errors are not anywhere near satisfactory. However, some applications might well do with the accuracy provided.

2.4.7 Future work

When considering the two main parameters, fixation of the cameras and overlapping views, there is one combination that has not been considered, which is overlapping dynamic cameras. This is equivalent to the raw footage taken from a match and even though such data is not obtainable per se, getting such data could be very interesting since most cameras have the ball in their view. This would give almost the same number of overlapping views as there are cameras, something that is only possible in the fixed setups if the cameras are very far away which automatically lowers the ball information parameter.

Ground truth is generally missing from the 3D field. The biggest hurdle is that humans are not able to annotate games directly with sufficient precision. One way to get it would be to capture one game with sufficient cameras, fixed or dynamic, to get the ground truth in 3D. Then later one could consider a subset of these cameras for the explicit research regarding fewer overlapping views.

An application that we have not encountered in the articles considered is the augmentation of the broadcast with the projection of the ball on the ground. When watching a soccer match it can sometimes be difficult for humans to infer the 3D trajectory of the ball. We hypothesize that this would enhance the pleasure of watching games for some. Some experimentation would have to be performed to find out how much added value it would give and how that value is affected by the inevitable error made by such a system.

For using a single view of the ball the possibility of using the ball size to find out where the ball is could be interesting. Given a perfect homography this depends almost entirely on the resolution and very little on the lens distortion since the ball is rarely close to the edges of a frame. We hypothesize that dealing with the problem theoretically would give bounds on the error of the 3D position and thereby also an error for projecting on the field. This would be as a function of the camera setup, the resolution and possibly other input parameters such as errors in the calculation of the projection.

2.5 Overall discussion of reviewed methods

We have seen the problem of finding the ball divided up into detecting it, tracking it and reasoning about its location in 3D. Good recent methods exist to detect the ball when ball information is high. When ball information is low it is very difficult to detect and track in an online manner but generating trajectories is possible although there is a substantial lack of comparisons between the methods. Given that both computational power and bandwidth increase and camera hardware get better dealing with low ball information as defined here can be considered mostly historical. Finding the ball in 3D is certainly possible but the current research is far from giving results that are usable for helping the referee in any way. Considering just detecting interpreted events the idea of Ariki et al. of looking for longer durations of when the ball is completely still, seems like an idea to work on although we know that there are many practical situations where six seconds is too much.

2.6 Use of ball size to estimate the distance

We will try to reason about the 3D position of the ball given a perfect homography and the ball's position in the image. We also assume that we can find the camera's position in the world. Given these assumptions we can consider the problem being that of finding the distance d from the camera to the ball. A ball has a standard size so this in theory makes us able to calculate d within some error margin. The distance to the ball is $d = f \frac{\varnothing}{\varnothing_I}$ where f is the focal length and \varnothing and \varnothing_I is the real diameter of the ball and the diameter of the ball in the image respectively. Let us first assume that the value \varnothing is known exactly. Due to the discrete nature of a digital image we can not expect to measure the exact diameter \varnothing_I , so we will instead consider observing the diameter with some error ϵ so the observed diameter is $\varnothing_I \pm \epsilon$. This gives:

$$f \frac{\varnothing}{\varnothing_I + \epsilon} < d < f \frac{\varnothing}{\varnothing_I - \epsilon} \quad (2.15)$$

For a given image f is fixed and we see that d is proportional to $\varnothing_I \pm \epsilon$ which is important for the theoretical consideration. If we assume \varnothing_I and ϵ to be independent, then we see that we should try to have \varnothing_I as large as possible, to minimize the error on d . This comes as no surprise. However, we do not provide \varnothing_I directly so we want it to depend on something that we can reason about. Let therefore l_h be an imaginary horizontal line segment through the ball center in the observed image and I_h the horizontal resolution of the image I . Then by triangle similarity we have that $\varnothing_I = \frac{\varnothing I_h}{|l_h|}$.

We now give some examples of the percentual error in the distance. We set $\epsilon = 2pixels$ i.e. an error of one pixel at each boundary. Adidas balls used in professional matches are manufactured to within a 1% error, and we assume that other brands deliver the same guarantee. This error will have to be added to the final error.

Example 1 (Main camera - changing resolution): We set $|l_h| = 50m$, $I_h = 720pixels$ and $\varnothing = 0.22m$ and have $\varnothing_I = \frac{0.22m \cdot 720pixels}{50m} \approx 3pixels$, and get a percentual error of $\frac{\epsilon}{\varnothing_I} = 67\%$, which is horrible. However, if we instead grab with an 8K camera we have $|I_h| = 7680pixels$ and $\varnothing_I = \frac{0.22m \cdot 7680pixels}{50m} \approx 34pixels$ and therefore an error in the image of $\frac{\epsilon}{\varnothing_I} = 6\%$, and a total error of 7%. If we assume the ball is at the one of the most distant corner flags and the camera is 5 meters away from the touch line and elevated 5 meters on a standard sized field then $d = \sqrt{(68 + 5)^2 + (105/2)^2 + 5^2} = 90$, the error is then $90m \cdot 0.07 = 6.4m$, which is still a huge.

Example 2 (Ball tracking camera): It would also be possible to zoom closer to the ball if the only objective is to measure the distance. Let us assume that the camera automatically tracks the ball and its own parameters. If V_{hmax} is the maximal horizontal velocity of the ball and R is the frame rate of the camera then the ball maximally travels $F_{max} = \frac{V_{hmax}}{R}$ meters between two

consecutive frames and this gives a lower bound for $|l_h| > 2F_{max}$. If $V_{hmax} = 40\frac{m}{s}$ and $R = 30\frac{frames}{s}$ then $|l_h| > 2.66m$ assuming that the camera maintains the ball at the center of the frame, and the reaction time is negligible. We could therefore go from viewing 50 meters around the ball to 5 and this would allow for a factor 10 decrease in the error, and we would have the ball located within a meter and a half on the ball line in the worst case.

Example 3 (Youth game): In danish U9 and U10 the game is played on a field of 20mx40m with a ball of size 4 and we consider to grab such a game with contemporary smartphone with FullHD resolution from the touch line. $I_h = 1920pixels$ and we set $|l_h| = 40m$. The size of the ball is $\varnothing_I = \frac{0.207m \cdot 1920pixels}{40m} \approx 10pixels$ so $\frac{\epsilon}{\varnothing_I} = 20\%$. When at the corner flag the distance is $d = \sqrt{20^2 + 21^2 + 2^2} = 29$ and the error is thus $29m \cdot 0.2 = 5.8m$. This is without considering the error in ball size that we experimentally have found to be less than 0.0075. The ball could also be measured before the game. Like before we could consider a higher-resolution zooming camera to increase I_h and lower $|l_h|$ but if the camera is hand-held it becomes harder to infer the homography when less of the field is visible.

These three short examples have shown how we can reason theoretically about finding the position in 3D when the ball is not on the ground using only a single camera. We fixed the error to 2 pixels and this could be too little according to [] that find it to be up to . It could also be too much if a good way of measuring is found. We would assume that measuring the diameter could be done robustly through measuring the area of ball. This is a problem however, if the ball travels at high speed and appears as the Minkowski sum of a circle and a line. In this case the diameter would have to be measured directly but in turn there is an increased number of measurements to perform proportional to the length of the line. One approach could be to use regression based on machine learning techniques.

We have only considered a single frame, but if we would consider the ball diameter to be distributed according to a Gaussian pdf, we could do the expectation over several frames averaging out the uncertainty. We could also consider using a Kalman filter to infer the position.

Chapter 3

Semantics derived from spatio-temporal soccer data

The result can sometimes lie;
playing well does not guarantee
success, but it nearly always does.

Xavi Hernandez

We can see in the model in Fig. 3.1 that we are going to cover two applications of spatio-temporal data in this chapter. We will first see research on we can get semantic features from only player data. Since we have covered a large amount of work on how to get spatio-temporal data from the ball we will also consider the features that can be derived when data for both players and ball is available. Finally we provide a feature based solely on player data building on the theory in this chapter.

3.1 Semantic features of players

In this section we look at actions and semantic features such as if a team is in possession or the roles of the players in the team.

Taki et al. [85, 83, 84] introduce the concept of dominant regions which are closely related to Voronoi diagrams. However, in the dominant region the distance measure between a player and a point is the time it takes for the player to reach that point given the players current movement vector and ability with respect to acceleration and velocity.

Kihwan Kim et al. [53] try to predict an approximate location for the point of play evolution using the convergence of a global motion field. They do this by first finding the ground plane motion of all visible players using fixed cameras. Then they interpolate the resulting sparse motion field in to a dense motion field by using a radial function. Finally they find the points of convergence of the motion field in two steps. In step one a confidence measure is calculated based on the magnitude at each point, which is then propagated over the field.

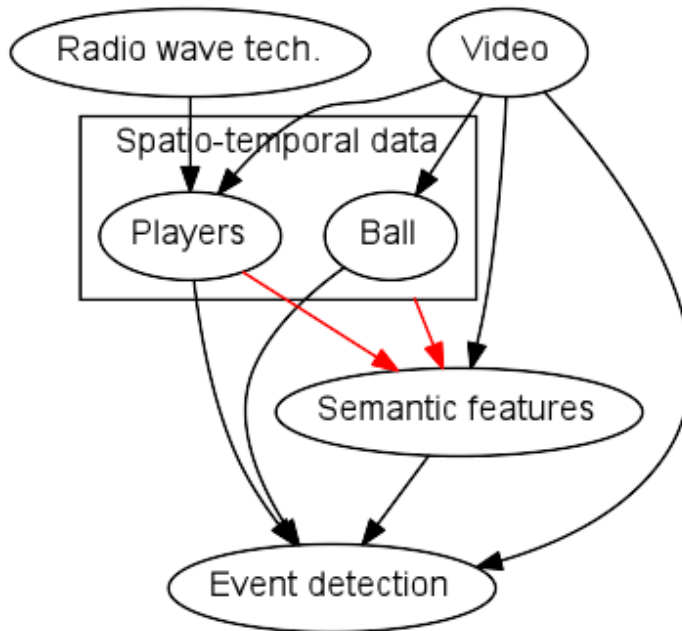


Figure 3.1: The model and the parts covered in Chapter 3

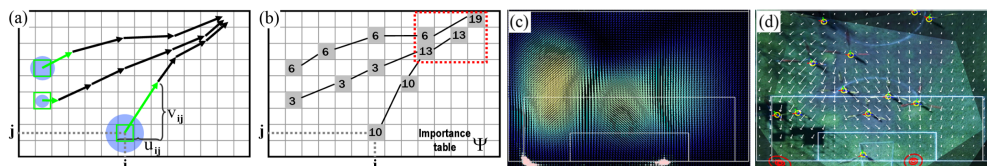


Figure 3.2: Fig. 7 from [53]. (a) The motion field for three points and the propagation.(b) Calculation of Ψ using magnitude. (c) Graphical representation of values of Ψ . (d) Two GMMs representing the points of convergence.

The result is some peaks on the field indicating interesting points. The final points of convergence are found in step 2 of the algorithm. First mean-shift clustering is used and then based on the number of clusters a Gaussian mixture model() is found using expectation-maximization(EM). We provide a modified figure from the publication in Fig. 3.2. Apart from using the information provided for analysis an application mentioned is automatic camera control. A video describing the results is available online[52].

Ho-Chul Kim et al. [50] assume a 4-4-2 tactic and look at the line the 4 defenders form. They define a morphology for the line and four measures; width, depth, angle and distance vector. The example given is measuring the performance of a striker by his influence on the defence line in terms of changing its morphology. The final of the World Cup 2006 is used as the case and they find a most important striker on each team.

Frecken et al. [35] use data provided by Amisco - now owned by the company that owns Prozone - from a UEFA Champions League quarter-final. They test the hypothesis that variability in inter-team centroid distance on both axes is

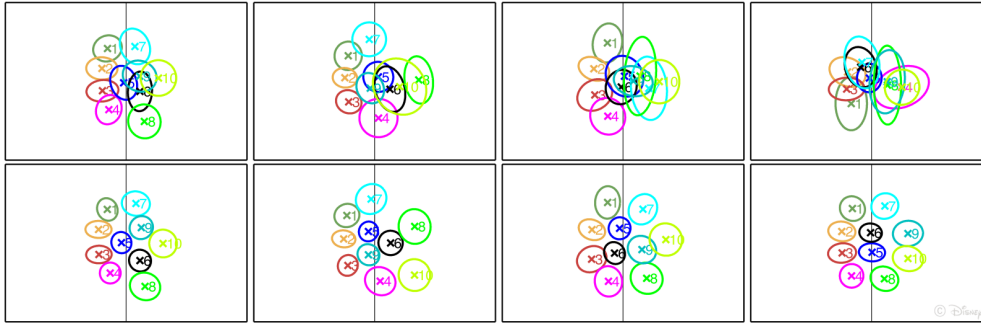


Figure 3.3: Fig. 4 in [19]. Upper row: Mean for each player. Lower row: Result of EM using maximal pairing.

associated with goals or goal-attempts. However, they find that it is more related to passing and reorganization in relation to set-pieces.

Folgado et al. [34] consider the *lwratio* as the ratio between the length and width of a team. In small sided games they note that the ratio is higher for younger teams suggesting that they play more individually and hence it can be seen as a measure of collaborative play. They also consider the centroid distance but the results vary with the number of players (3 or 4) on each team.

Bialkowski et al. [19] observe that simply taking the mean of each players position does not result in a clear overview of the tactic deployed (Fig. 3.3 upper row). They assume that different players take on different roles during a game, and use an approach to iteratively assign players and update roles. They use the Hungarian algorithm which does a min-cost maximal pairing on the distance between role centroid and player. The result is depicted in the lower row of Fig. 3.3. Using this method they find that the team formations do not vary between home and away games, but the centroid or mean does i.e. away teams spend more time in their own half than they do at home.

Bialkowski et al. [21] expand on the approach by giving a more rigorous mathematical analysis based on minimum entropy data partitioning. They also cluster all the tactical formations based on the earth-mover distance. The result of this clustering is that 71% of all formations are found in two clusters. The formation in both cases consists of 4 defenders and 4 midfielders. The difference is in the attack where the teams in one cluster (~50%) play traditionally with strikers side by side and in the other (~21%) with one attacker slightly behind the other. The authors also briefly consider ball events to show that the role-based clustering handles wingers changing from left to right and vice versa.

Bialkowski et al. [20] develop a formation descriptor based on the ideas from the previous two works to predict team identity. By this they predict 67% of all halves correctly. To go the other way they also try to predict the formation given the two teams.

Gudmondsson and Wolle [39] provide 4 different tools for working with spatio-temporal soccer data. The first 2 are considering passing and is therefore not handled here. The third deals with the computation of trajectory clustering and

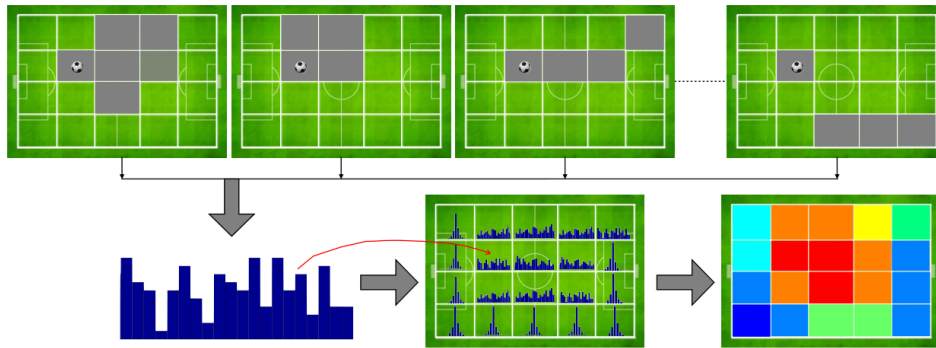


Figure 3.4: Fig. 4 in [60] showing how sets of passes determines a distribution over which quantized field is passed to. In the end the variance for each field can be calculated.

is also considered in [38]. They discuss the use of the discrete Fréchet distance for clustering trajectories of players and ball with respect to time complexity. Specifically they note that clustering trajectories is NP-hard in general and even when an <2 -approximation of the distance between trajectories is allowed. They provide visuals that show clusters obtained by running the chosen algorithm and also running time dependent on the distance parameter and parameter denoting the number of vertices in a trajectory. The last part is called “Correlating clusters” and is concerned with correlating different trajectory clusters in time. The example given is 3 defenders moving in similar trajectories at the same time. Time is the important parameter here and given a way to cluster subtrajectories we can view such a cluster as a set of intervals. They define local correlation to be between subtrajectories sharing at least one point in time.

3.2 Semantic actions with the ball

Hirano and Tsumuto [45] process manual data from the FIFA 2002 World Cup to cluster all pass sequences preceding the 168 goals scored. The methods used are multiscale matching and rough clustering. They end up with 12 clusters where the total number of sequences in the 4 largest clusters is 144. One cluster contains 87 extremely short sequences.

Lucey et al. [60] also use manually annotated data, in this case from the English Premier League. The ambition is to analyze the data in the multi agent plan recognition (MAPR) framework. A team here is a set of agents with a shared plan and a shared mental state. Team behaviours are short coordinated movements or actions. A plan is an ordered set of team behaviours deployed to achieve a goal. They divide up team possessions into segments and quantize the field and look at the entropy for each quantized field with respect to where the ball is played next, depicted in Figure 3.4. They do a specific team analysis down to pinpointing the entropy based on specific players. They also try to predict the home team for each game, which they do successfully in 30% of

the cases beating 19% based on regular match statistics. Combining both yield 47% correct.

Kang et al. [49] introduce several ideas to reason about trajectories of players and ball regarding passing options. They provide several strict definitions of different areas on the field. E.g. safe area and competing area denote the area at time t that a player can receive a pass from player k , safely and in competition with opponents respectively. They propose to make averages of these values, performance measures for players. An experiment with simulated spatio-temporal data is also performed.

Duch et al. [31] create a directed graph based on the pass and shot information provided by UEFA for the 2008 Euro Cup. They measure players by participation in pass sequences that lead to shots on target. Defenders start trajectoreis based on their ball recoveries but it is not clear exactly how this is done. The results of the algorithm fit well with the results of the tournament but it is unclear how much the data has guided the development of the algorithm. However, it would be possible to run the algortihm on later tournaments in follow-up studies. .

In [62] Lucey et al. use the same data and method as in [60] to explore the home advantage. They find that the difference in the occupancy maps as they now describe them, shows that teams in general play more offensively at home than away.

The most recent study by Lucey et al. [61] process 9732 sequences preceding shots to find factors that describe the probability of scoring. In the introduction they claim to use conditional random fields for which they cite a publication, but in the actual text they write that they use logistic regression. One of the findings is that there is a probability of 70.59% chance of scoring if a winger comes down the left passing the ball to a striker in front of goal. This is of course only including the cases were a shot is performed which explains the high percentage, and could be used to indicate a weak point of the analysis.

3.3 A tactical feature

With inspiration in the papers in the two previous sections by Bialkowski, Lucey et al. we propose the following simple tactical feature. Let a set $P = P^{(1)}, \dots, P^{(k)}$ be given of k sets of player positions. A set $P_i = \{p_1^{(i)}, \dots, p_n^{(i)}\}$ of n positions for player i , where $p_t^{(i)} = (x_t^{(i)}, y_t^{(i)})$. We will think of these positions as a time series but note that hey do not have to be. We define the mean over all players for some $t \in [1, n]$ as $\mu_t = \frac{1}{k} \sum_{i=1}^k p_t^{(i)}$. We refer to this as the *player mean*. We also have the mean over all positions or the *match mean* for each player to be $\mu^{(i)} = \frac{1}{n} \sum_{t=1}^n p_t^{(i)}$. The mean over all positions and players is: $\mu = \frac{1}{k} \frac{1}{n} \sum_{i=1}^k \sum_{t=1}^n p_t^{(i)}$.

Let us now define the relative position of a player i at index t to the mean to be $r_t^{(i)} = \mu^{(i)} - p_t^{(i)}$ i.e. the position relative to the match mean. We also define the distance $d_t^{(i)} = \|r_t^{(i)}\|_2$

The absolute error over these distances is now the value of what we call the *Tactical Formation* score or TF for short.

$$TF(t) = \sum_{i=1}^k d_t^{(i)}$$

This value indicates how close the team is from playing the standard formation. When the players are spread out or very close together the score goes up even though the formation is kept. To solve this problem we choose to normalize the score based on the sum of the distances to the player mean i.e.

$$AE(t) = \sum_{i=1}^k \|\mu_t - p_t^{(i)}\|_2$$

and we then get the *Normalized Tactical Formation* (NTF) score

$$NFT(t) = \frac{TF(t)}{AE(t)} = \frac{\sum_{i=1}^k \|\mu^{(i)} - p_t^{(i)}\|_2}{\sum_{i=1}^k \|\mu_t - p_t^{(i)}\|_2}$$

Chapter 4

Machine learning methods for sequential data

Machine learning is typically divided into three fields, supervised learning, unsupervised learning and reinforcement learning. Reinforcement learning is about iteratively making an agent learn to act in an environment and could be used to make a robot learn to play soccer. We will not use reinforcement learning here. Supervised and unsupervised learning both have as input a data set. In supervised learning the data set comes with labels and the task is to learn a model given the data and the labels, that can classify future instances. In unsupervised learning the task is to classify the instances without any labels. We will look at both of these here with the special focus on sequential data.

Machine learning models can be further divided into discriminative and generative models. From a probabilistic point of view, we model the joint probability of a data point x and a label y i.e. $P(y, x)$ with a generative model, where a discriminative model models the conditional probability $P(y | x)$.

4.1 Emission probabilities for a hidden Markov model

The HMM is often used in bioinformatics to predict e.g. gene structures or protein membranes and it typically uses a discrete set of emission probabilities corresponding to e.g. amino acids. The model then uses a $K \times L$ matrix B that denotes the probability $P(x_n | z_n)$, where K is the number of hidden states and L is the number of observable states.

One way to handle continuous values of the emission is to assume that for each state a probability distribution, e.g. a Gaussian is responsible for generating the emissions. We can define such emission distribution ψ_k , $k = 1, \dots, K$:

$$(x_n | z_n = k) \sim \psi_k$$

Another way to provide the emission probabilities is by using a discriminative classifier. We will call such a model a *hybrid HMM-dc* where *dc* can stand for any discriminative classifier. When using e.g. an artificial neural network(ANN)

the model is known as a hybrid HMM-ANN. This kind of model has been used with success in speech recognition. The main discriminative classifier that we use is regularized logistic regression based on the well-known practice to start in any learning problem using a linear classifier [11]. The derivation of the model can be found in Appendix A.

There is one theoretical note that is important to make, however. An emission probability is defined to be $P(x_n | z)$ i.e. the probability of seeing the observation x_n given the state z_n . A discriminative classifier like logistic regression or an ANN can be considered to model $P(z_n | x_n)$ i.e. given the observation x_n what is the probability of being in state z_n . Using Bayes' theorem we know that these should be proportional:

$$P(x_n | z_n) = \frac{P(z_n | x_n)P(x_n)}{P(z_n)} \quad (4.1)$$

For the sake of running the Viterbi algorithm to get the most likely states we can ignore $P(x_n)$ as it does not depend on the state. However, Eq. 4.1 would suggest that we should divide by $P(z_n)$. We instead choose to ignore this factor also leading to modelling $P(x_n | z_n) = P(z_n | x_n)$. This is now an altogether different model, but if we look at what actually happens in the Viterbi algorithm we can understand why this makes sense. The Viterbi algorithm calculates the Viterbi table ω using dynamic programming and in particular the n'th entry is:

$$\omega_n = P(x_n | z_n) \max P(z_n | z_{n-1})\omega_{n-1}$$

If we by ω' denote the modified Viterbi table from using $P(z_n | x_n)$ then it is instead:

$$\omega'_n = P(z_n | x_n) \max P(z_n | z_{n-1})\omega_{n-1}$$

This effectively changes what we compute with the Viterbi algorithm but we are only interested in finding the best sequence of labels using a discriminative classifier which this approach allows.

4.2 Recurrent neural networks

The following is a very short overview of neural networks and their application to making predictions on sequences based on the book by Alex Graves et al. [37].

Artificial neural networks take their inspiration in the structure of the brain using a weight matrix and activation functions to model neurons and the synapses between them. We hereafter assume that all neural networks treated here are artificial and just call them neural networks. A *feed forward neural network* is what is often meant when just referring to the usage of neural networks. The “feed forward” term refers to the fact that there are no cycles between the cells or neurons. Logistic regression can actually be seen as the simplest of such networks but is generally not considered as a neural network since it does not

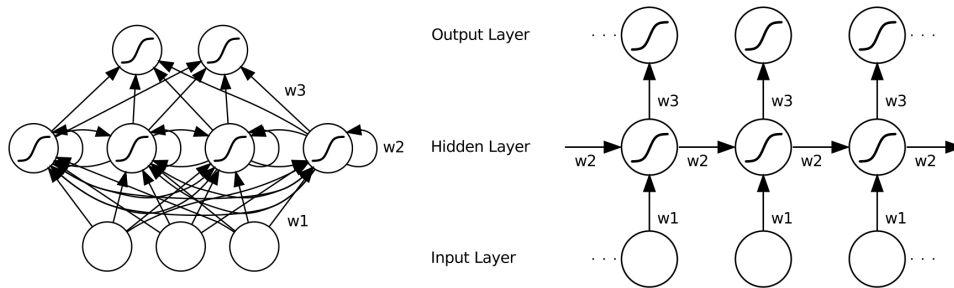


Figure 4.1: A merge of Figures 3.3 and 3.4 in [37]. A static view of an RNN on the left and the weight updates through the sequence on the right.

model non-linearities. Neural networks are sometimes also called *multi layer perceptrons*, which suggest the similarity. Adding a single layer known as a hidden layer is however enough to introduce this feature. Such a network could be used to classify images of ball candidates when detecting a soccer ball or for numerous other tasks. As we have just argued we can also use them for providing the emission probabilities of an HMM. A *recurrent neural network* () on the other hand, uses cycles to encode information over time and can therefore be trained to predict the labels of or classify sequences on its own. An RNN is depicted in Figure 4.1 where we see both the connections in the static case and the updates of the weights w_1, w_2 , and w_3 over time for three cells, one in each layer.

RNNs are prone to a the *problem of the vanishing gradient* which basically means that over time the gradient that was calculated in the output layer early in the sequence will have little impact later. The most successful method to accommodate the problem has been the *Long Short-Term Memory* () neural networks. In an LSTM layer a cell is substituted with a block containing three gates as depicted in Figure 4.2.

The idea is that the input gate determines how much of the input to send to the cell and similarly for the output gate that determines how much of the output signal to pass on. The forget gate and the peep-holes represented by the dashed lines were later additions to the model. The forget gate allows memory to be reset and the peep-holes allow the gates to observe the state of the cell without being dependent on the output which could be close to zero due to the output gate not firing. If the input gate is not firing it effectively prevents the cell state from being overridden, thus maintaining its state for as long a period as necessary.

►put this in own section on NN◄ Not surprisingly, the weight matrix that results from introducing all the dependencies can get quite large, and training big networks can therefore take a long time even using large amounts of contemporary hardware. This is not the only problem as many weights mean many degrees of freedom and this in turn makes the model much more prone to overfitting. A well-known way to prevent overfitting for neural networks has been to stop training early. Other ways can be by introducing artificial noise

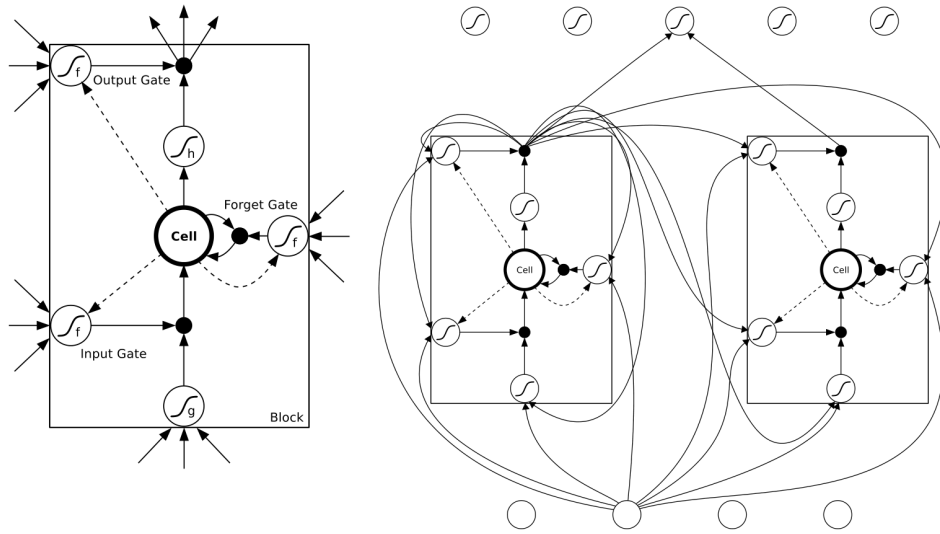


Figure 4.2: Figures 4.2 and 4.3 in [37]. An LSTM block on the left and a small view of a complete network to the right. The black discs represent component-wise multiplication.

somewhere in the training process. A recent method by Hinton et al. [44] is the use of *dropout* which consists in providing a single value that determines the probability with which the output of a cell is ignored. The authors note that dropout can be used at both hidden and input layers. For the hidden layer they consistently use a dropout value of 0.5, while it should be lower when used on the input. They also show how very different features are extracted when using dropout on a standard data set. In the case of an LSTM layer in the keras library [7] used in this work, the dropout is applied to the output.

4.3 Agglomerative hierarchical clustering

Clustering is one of the main branches in unsupervised learning. The input is an unlabeled data set and the problem is to find a set of clusters and assign each data point to one. Many of the most popular algorithms such as *k*-means or GMM clustering takes a number *k* as parameter and finds labels for all data points. The problem that these two methods solve is NP-hard and random initialization of clusters is therefore typically used, which makes the algorithms non-deterministic. Hierarchical clustering differs both in that it does not need an explicit value for *k* and in that it is deterministic. Instead of creating one set of clusters it builds a hierarchy of clusters where at the top there is only one cluster with all the data points and on the bottom each data point has its own cluster. The hierarchy can be constructed either bottom-up or top-down which are referred to as agglomerative and divisive hierarchical clustering respectively. Here we will only use agglomerative clustering and pseudo code for the algorithm can be seen in Alg. 4.1

Algorithm 4.1 Agglomerative clustering.

- 1: Assign each x_i to a cluster $C_i \in C$
 - 2: **while** $|C| > 1$ **do**
 - 3: $k, l = \arg \min(\text{link}(C_i, C_j))$ for all $i < j$
 - 4: merge C_k and C_l into $C_{k,l}$
 - 5: remove C_k, C_l from C , add $C_{k,l}$
 - 6: **end while**
-

The running time of the algorithm is cubic, since the while loop gives a factor $O(n)$ and line 3 gives a factor $O(n^2)$.

The output can be visualized by a dendrogram that depicts which clusters are merged and the distance used to merge the clusters. This means that there are two measures we can use to get the output of the algorithm. By starting from the top we can go down until we have a desired number k of clusters. We can also define a distance d and go down until d is smaller than the distance used to merge the clusters.

While the distance between two data points of spatial data from e.g. soccer players obviously could be picked as the Euclidean distance, there are more ways of generalizing this to merge clusters with more than one element. This is provided as the distance function link that is used in line 3 also known as the linkage criteria. Various linkage criteria exist and they all need a distance function d to compare two individual data points. One of the most widely used ways to link the clusters is the Ward criteria that finds the intracluster variance of cluster $C_i \cup C_j$:

$$\text{Ward}(C_i, C_j) = \frac{1}{|C_i| + |C_j|} \sum_{x \in C_i \cup C_j} d^2(x, \mu_{C_i \cup C_j})$$

What is really interesting is the ease with which the algorithm can be used on sequential data. Instead of considering the upper triangle of the distance matrix we just consider the superdiagonal i.e. line 3 becomes $k, l = \arg \min(d(C_i, C_{i+1}))$ for all i . This also immediately cuts a factor $O(n)$ off of the time complexity.

Chapter 5

Event detection on player data

In a football match, everything is complicated by the presence of the other team.

Jean-Paul Sartre

Here we will describe how to use machine learning to do event detection using semantic features of player data and player data alone. This is the final part of the model as depicted in Figure 5.1.

In Section 5.1 we describe the resources that are available and how to use these. In 5.2 we present a helpful tool to solve many of the tasks. In 5.3 we look at how to process the data to get the relevant values. How to generate features is handled in 5.4. Preprocessing the data before using the learning algorithms is in 5.5. In 5.6 we show how to evaluate the predictions, and in 5.7 we present the approach to feature selection. A method to increase the number of classes is in 5.8. In 5.9 we put the things together to lead into the next chapter on the experiments.

5.1 Available resources

5.1.1 Data and video

We summarize the available resources in Table 5.1. We have data for ten matches and with around 12-13 players on average, 50 minutes per game and a frequency of 20hz there are around 10^7 data points. This can seem as many and it certainly is in terms of running e.g. an $O(n^3)$ algorithm on them. Since it is sequential data, the data points are highly dependent so we choose to use a frequency of 2hz, as a trade off between running time and loss of data.

The origin of the coordinate system is placed at the lower left corner flag considering the video in Figure 5.2. The x -axis coincides with the touch line and the y -axis with the left goal line. In the following

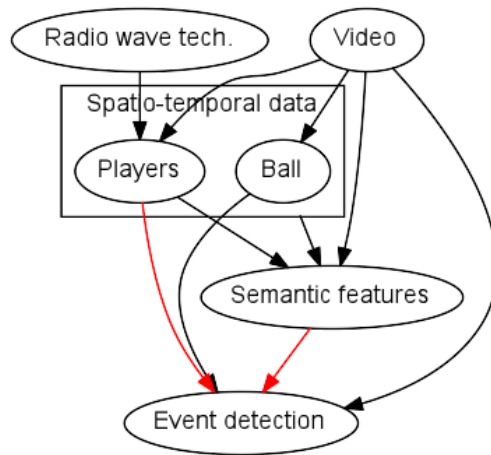


Figure 5.1: The model and the parts covered in Chapter 5



Figure 5.2: Match view

Opponent	Date	Half	Video	Field p.	Total p.
OB	23-2	1	-	10	13
		2	-	10-9	13
Esbjerg fB	8-3	1	-	9	12
		2	-	9-8	13
Hobro IK	21-3	1	BC	10	13
		2	BC	10-9	13
Silkeborg IF	17-4	1	BC ¹	8	12
		2	BC	8	13
Brøndby IF	25-5	1	BC	9	14 ²
		2	BC	9-7	14 ²
SønderjyskE	7-6	1	-	8	12
		2	-	10-9	11
FCK	14-8	1	-	10	11
		2	-	10-8	11
Randers FC	16-10	1	-	9	12
		2	-	9-6	12
Viborg ff	31-10	1	-	10	13
		2	-	10	12
Esbjerg fB	8-11	1	-	10	13
		2	-	10-9	13

¹ Only last 15 minutes.

² 4 opponent players

p.: Players

BC: Broadcast video

Field players $a-b$, a :maximum players, b :minimum players

Table 5.1: The available resources.

KAMPREFERAT		
FC MIDTJYLLAND		BRØNDBY IF
0'	Spilstart	
9'	Mål (0 - 1)	Pukki ⚽
11'	Mål (1 - 1)	Pusic ⚽
12'	Hjørnespark	Brøndby IF
16'	Skud på mål	Larsson
18'	Advarsel	Agger 🟡
20'	FC Midtjylland	Hjørnespark
22'	Poulsen 🟡	Advarsel
27'	Igboun > Lauridsen ⚽	Mål (2 - 1)
29'	Mål (2 - 2)	Hjulsager > Szymanowski ⚽
32'	Andersson	Skud på mål
34'	Skud på mål	Pukki
44'	Skud på mål	Nørgaard
45'+1'	Hjørnespark	Brøndby IF
45'+1'	Mål (2 - 3)	Larsson > Agger ⚽
45'+2'	Stut på halvleg	

Figure 5.3: Web match summary.

5.1.2 Matches and events for training, validation and testing

When we have video for a match we can annotate it ourselves, but since we do not have video for all matches we instead use annotations posted on the web. An example can be seen in Figure 5.3.

Since these are annotated with minute precision and also sometimes are off by a minute or two they cannot be used for annotating the training sets. We will instead use these for validation and testing. The events that are annotated also determine which events it makes sense to try to predict. We want events to be well-defined which invalidates free kicks since these in most matches are annotated only when they are close to the goal, which is too imprecise a notion. Other events like throw-ins and goal-kicks are not annotated at all. We are therefore left with corners and goals. Table 5.2 shows how we divide up the matches. It clearly does not make sense to try to predict opponent goals, and for the other events we can consider the data set quite small.

Ideally we would have a labeled data set that we could split into training and test set, do cross validation on the training set and finally apply the model to the test set. This is not possible under our circumstances, specifically if we would do cross validation it would be on only a few events. We also find that given so few matches, setting aside four matches for a single run of the model is a waste. Also, even if the model would do very well we could not be sure if we are just very lucky given the uncertainty about the annotations. We therefore use the test set as a second validation set, not to optimize a single model but to compare all the models. Hence we will use it multiple times in the end of the process but not to make any direct decisions such as parameter selection on this. This will allow us also to test the whole approach set forth here, and will be quite interesting since we consider the case to be non-trivial.

Opponent	Corners	Opp. corners	Goals	Opp. goals
Training set				
Hobro IK	1	4	3	0
Silkeborg IF	7 ¹	4	1	0
Brøndby IF	4	4	2	3
Total	12	12	6	3
Validation set				
OB	6	3	3	0
Esbjerg fB(1)	0	9	3	0
Esbjerg fB(2)	8	4	5	1
Total	14	16	11	0
Test set				
SønderjyskE	3	2	2	0
FCK	3	10	0	0
Randers FC	2	8	2	1
Viborg FF	5	7	2	4
Total	13	27	6	5

¹ 9 corners in the match but only 7 annotated.

Opp.:Opponent

Table 5.2: The available resources to work on.

5.1.3 Annotating events

To predict events we take advantage of the many data points available even though the number of events is small. We annotate the data at the start and end of an event, i.e. when the ball goes out of and when it is put back into play, following the model put forth in the introduction. We define a *corner out event* as the moment the ball goes out for a corner and a *corner kick event* when the ball is moved again from its position close to the corner flag. A *goal event* is when the ball crosses the goal line and a *kick-off event* is when the ball is put back into play by the opposing team from the center line. The corners can be annotated well, since it is both an interesting situation when the ball goes out, because it is close to a goal, and it is interesting when the corner kick is taken for the same reason. The same applies to the *goal event*. The *kick-off event* however is often not part of the production as repetitions of the goal are broadcasted instead. This leaves for a manual interpretation based on the visualized data.

To label the data we consider the data between two corresponding events to represent a state e.g. a corner state. These can automatically be generated from the start and end events described above. We will use state and event interchangeably when the context is clear.

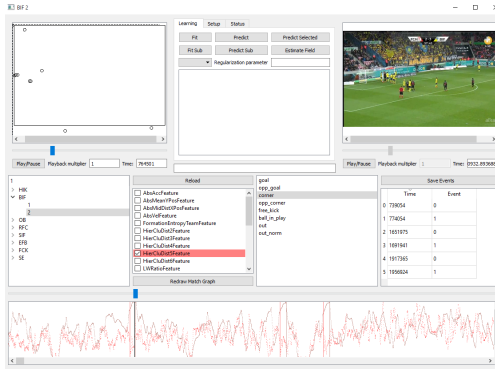


Figure 5.4: A view of one instance of the annotation tool.

5.2 A visualization and annotation tool

Since we work with positions of players on a soccer field, it is valuable to be able to visualize the data. To this end we construct a small application with a graphical user interface using Python, Qt and the OpenCV framework which can be seen in Figure 5.4.

This allows for visualization at different speeds, synchronization and annotation of data and video. It also supports fitting machine learning models to the data and visualizing predicted probabilities and classes and annotated events. Furthermore semantic features can be visualized and new features can be implemented and reloaded while the program is running to work on already loaded data.

5.3 Working with radio-wave based data

There are several issues that we identify that make the data hard to deal with.

1. Some players are not there. E.g. the goal keeper is not present in any of the matches. As stated in it could be that a player forgets to put on the belt.
2. The team of the players is not annotated and in the received data a few players from other teams are included.
3. The data can have large amounts of noise in it. A unit could be malfunctioning giving spurious values or the signal could be lost.
4. The teams switch halves at half-time so all values have to adjusted accordingly.
5. Data is given for both players on the field and off the field so it is necessary to distinguish between these.

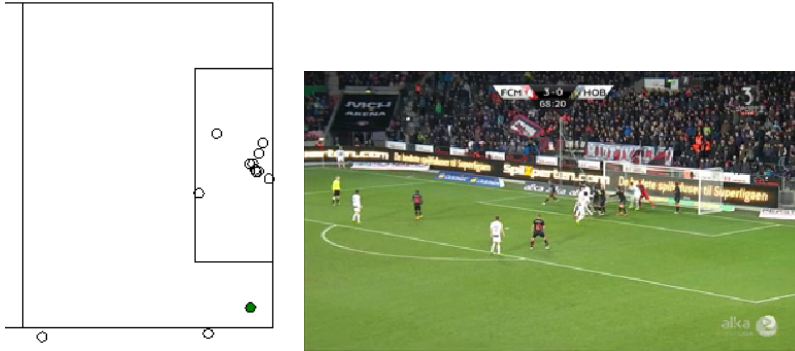


Figure 5.5: A noisy substitute.

1. Missing players As stated in [71] the only way to deal with missing players, when not considering video, is to make algorithms robust enough.

2. Other players Regarding players from other teams a list is kept of the ids of players in FCM. 3. Finding the position of the field lines makes it possible to invert the field. 4. Having the field makes it easy to see if the player position is inside or not but not if the player is playing.

3. Handling noise We will classify the signal according to three classes; standard, lost, and noisy. Standard signal is a signal that has noise similar to the specifications provided by ZXY[71]. A lost signal can be considered to be a player missing over some time but still appearing in the data. We simply load the data into an array according to the start of the first data point and load the positions in to the relevant positions. This means that all values including the timestamp are set to zero, and are in a sense marked this way. A noisy signal can be observed by comparing the trajectories of the players in data and on video or by directly observing unnatural movements in the data. It is therefore not easily detectable but an example is shown in Figure 5.5 where the substitute is probably not entering the field during normal play. The noisy signal is really difficult and we currently have no way of handling this. An idea is to find a measure that captures a players deviation from his or her regular relative position on the field.

4. Field size/location detection To detect the field location we annotate the data each time a player is on a goal or touch line. We then use four univariate gaussians, two using the x -coordinate and two using the y -coordinate representing each line, and fit these to the points. The means are then the expected position of each line and we even have a measure for the certainty of its location. We will perform an experiment to infer the size of the field this way, although we will not use the measurements given that ZXY claims that the field is well calibrated. A re-calibration has also been performed since the first games so it would not make sense to use those values based on few data points.

5. Players that are actually playing It seems easy to check if a player is playing or not. Substitutes are not annotated but this would be quick to do manually. This however gives a problem if players accidentally switched belts or if a player is out with a head injury for 5 minutes it would definitely not make sense to include this player. A red card would also have to be annotated and we are not really looking for manual labor. Instead we could compare with the field location found previously. This would solve it for players being substituted, out for injury treatment or red cards. However, a player normally leaves the field to take a corner kick or throw-in, or as normal part of play close to the goal and touch lines. Here the player would seem essential to infer this kind of event. It could also be the case that a substitute has a noisy signal and therefore appears as being on the field, and could seriously bias the data for the playing players as can be seen in 5.5.

The task is thus to create a model that predicts if we should include the player as part of the current data or not. Based on the previous analysis we consider a SUBSTITUTE state. Any player at any time that has started the game as a substitute and not yet entered the field is in this state. We also have a general FIELD state that is given to players on the field. We also consider PLAYING_OUT, TEMP_OUT and PERM_OUT states, that all have in common that they require the player to previously to have been in the FIELD state. They denote the following three situations: PLAYING_OUT: a player whose position is outside the field for as part of the play, and that does not require permission from the referee to reenter. TEMP_OUT: a player is in this state if the player needs permission to reenter. PERM_OUT, a player in this state has left the field and does not enter it any more in the game.

We use a the HIK match to annotate for each player their states and use these for counting the initial probabilities and transition matrix in an HMM. The emission probabilities are calculated using the Gaussians from 4. We set the cumulative distribution function() for each such that the center of the field is very close to 0 and being outside the field has probability close to 1 for at least one of them. Given a position (x, y) we take the maximum over all the CDFs and use this as the probability of being out of the field. We then run the Viterbi algorithm for each players and this will give us which state they are in. Finally create a mask based on the states that denote the player is playing, which are the FIELD and PLAYING_OUT states.

In an experiment we will see how well we handle substitutions i.e. the transition from the SUBSTITUTE state and the transition to the PERM_OUT state.

5.4 Features

Given data of players and a mask to filter the players that are playing, the next question to answer is: what vector should we use as input to an algorithm? At each time step we are given the coordinates of some of the players. Using these directly is absolutely non-trivial, since no ordering is given per se. We choose

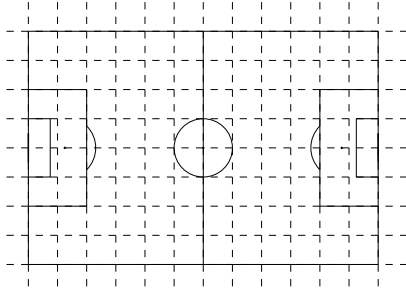


Figure 5.6: Discretization of the field.

to use two different approaches. The first is to discretize the field into a grid and then use each grid cell as a parameter. The second is to manually extract features using some of the ideas in Chapter 3. Discretization is automatic but blows up the dimension considerably. Manual features have the advantage that they are few, but since they are manual they have to be constructed.

5.4.1 Discretization

We discretize the field based on a single parameter hor_cells that determines the number of horizontal cells that the field is divided into. ver_cells is the number of cells vertically and is two thirds of hor_cells . Additionally there are cells all around the field. An example is illustrated in Figure 5.6 with $hor_cells = 12$ giving an input vector of dimension $(12 + 2) \cdot ((\frac{2}{3}12 + 2) = 140$.

For each time step we scan through the players and increment the cell corresponding to the position.

5.4.2 Manual features

We want to ignore from which side of the field a corner is taken. The first step to achieve this is to define the line $l_H : b$ as the unique line passing through the two penalty spots where b is the only parameter of the line i.e. the value of the intersection with the y -axis.

Let the positions of k players be given by, $P = [(x_1, y_1), (x_n, y_n)]$. We denote with $\mu^{(P)} = (\mu_x^{(P)}, \mu_y^{(P)})$ the arithmetic mean of P , divided in to its 2D coordinates. We leave out the superscript notation when it is clear which set we are dealing with. We also use X and Y to denote the the sets of x_i 's and y_i 's respectively.

To create features with symmetric y axis we define the line $l_H = b$ as the unique line defined by the two penalty spots, and consider the distance to this line as the feature. This explains how the mean is treated.

$$\text{MeanX} \quad \mu_x^{(P)}$$

$$\text{AbsDMeanY} \quad |\mu_y^{(P)} - b|$$

Similarly instead of considering maximum and minimum of two values y_1, y_2 we define $Dmax(y_1, y_2) = \max(|y_1 - b|, |y_2 - b|)$ and $Dmin(y_1, y_2) = Dmax(y_1, y_2) - |y_1 - y_2|$ which in the way they are defined have the property that: $Dmax(y_1, y_2) - Dmin(y_1, y_2) = |y_1 - y_2|$. These definitions generalize trivially to more parameters. We define some features of max and min values:

$$\text{MaxX} \quad \max(X)$$

$$\text{MinX} \quad \min(X)$$

$$\text{MaxD} \quad Dmax(Y)$$

$$\text{MinD} \quad Dmin(Y)$$

We also define features that give the other coordinate for the player having an extreme value. These are assumed to be important for corner kicks, as just having a player close to the oppoent goal line or outside the touch line, could happen in other situations, such as goal attempts or throw-ins respectively.

$$\text{MaxXD} \quad |y_j - b| \text{ where } j = \arg \max_i(X)$$

$$\text{MinXD} \quad |y_j - b| \text{ where } j = \arg \min_i(X)$$

$$\text{MaxDX} \quad x_j \text{ where } j = \arg \max_i(X)$$

$$\text{MinDX} \quad x_j \text{ where } j = \arg \min_i(X)$$

Based on the previous features it is also easy to define the width and length and the ratio used by Folgado et al. [34] presented in Chapter 3.

$$\text{Width} \quad Dmax(y_1, y_2) - Dmin(y_1, y_2)$$

$$\text{Length} \quad \max(X) - \min(X)$$

$$\text{lwratio} \quad \frac{\max(X) - \min(X)}{Dmax(y_1, y_2) - Dmin(y_1, y_2)}$$

We also consider the velocity P', X', Y' and acceleration P'', X'', Y'' of the players defined similar to P, X, Y

$$\text{Mean[Vel|Acc]X} \quad \mu_x^{(P')}, \mu_x^{(P'')}$$

$$\text{Abs[Vel|Acc]} \quad \sum_{i=1}^n \sqrt{x_i'^2 + y_i'^2}, \sum_{i=1}^n \sqrt{x_i''^2 + y_i''^2}$$

To model how close the k nearest players stand we use the height of the dendrogram using hierarchical clustering.

HierClukdist The distance d used for the $k + 1$ th cluster when doing agglomerative hierarchical clustering with *ward* linkage and euclidean distance between two clusters C_1 and C_2 of players.

We also model the distance to the midfield line that we define similarly to l_H as l_V , given by the value c that represents the x -value for the midfield line. We expect this to be useful for detecting kick-offs.

AbsMidDistX $| \max(X) - c |$

Finally we also include the features that we developed Section 3 representing the tactical formation, the sum of distances, and the former normalized by the latter.

5.5 Preprocessing

Before we put the data into the learning algorithms we process it in two ways. The input vectors are always scaled so that each feature has zero mean and unit variance. This is important in many cases i.g. when using a penalization term since otherwise small valued features will be penalized much more than large valued because they need much higher weights to have an impact on the classifier.

Since the amount of data points for each class are extremely uneven, some learning algorithms will take a very long time to adjust to the classes with few labels. We therefore implement a sampling scheme to upsample the classes with low frequency and/or downsample the classes with high frequency. Most learning algorithms also allow for adjusting the weight of the classes and this is generally better than upsampling since it does not affect running time. Our framework allows for both options to be turned on and off.

5.6 Evaluation

The algorithms are trained by minimizing cost functions and after predicting on new data the output is the most likely classes for a given input. Due to the annotation of validation and test data we do not have any duration of the events that we try to predict. In the end we can therefore not test how well we actually cover the state, but only if we more or less predict an event at the time of the annotation. The way that we evaluate is by creating a bipartite graph where the nodes are the true and predicted events respectively. If a predicted and true event are of the same type and overlap they are connected by an edge. Since the predicted events are with precision in seconds we could have a problem if the second counter is close to 0. We therefore add *slack* in the following way; if the seconds counter is below 30 we also include the previous minute. Likewise if above 30 seconds. An edge is then added if two events overlap. Finally we perform a maximal matching to get the correctly predicted events. An example of the results can be seen in Figure 5.7. Since it would be too tedious to examine all these graphs we also create summaries for each event of the true positives, false, positives, false negatives, precision recall. For each run of any algorithm the data is saved to log files for later retrieval.

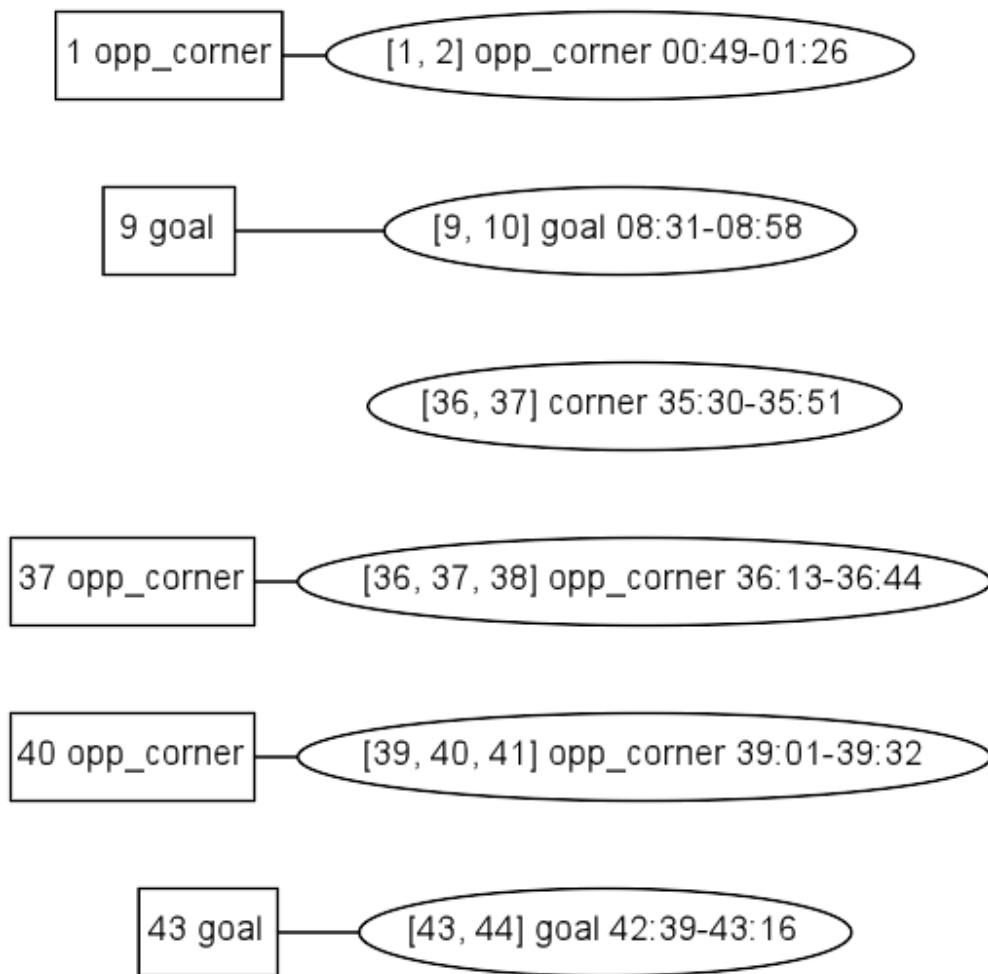


Figure 5.7: A graph of the output. First half of the EFB match. Square nodes are true events and predicted events ellipses.

5.7 Feature selection

Feature selection is the process of finding the best features in our case to classify the data. First of all it is interesting to see which features are the most important seen from a semantic point of view. Secondly it can be important to reduce running time of the algorithms. By increasing the bias of the learner it can also help to make a better classifier if it successfully eliminates noisy variables. According to Guyon and Elisseeff [41] greedy feature selection in particular exhibit the last two properties, fast running time and resilient to overfitting. Two types of methods exist; we can choose to start from no features and add one at a time or we can start from all the features and take the least important away. These approaches are referred to as *forward selection* and *backwards elimination* respectively. We use backwards elimination as it guarantees progress, which is not the case for forward selection when one parameter can not classify anything on its own.

5.8 Event segmentation

One problem that can be expected given that the annotation are just based on when the ball crosses a line and is put into play again, is that there is a high intra-event variance. By just assigning one class to the whole event we force the classifier to consider all the data. An idea to try out is to segment the event into different phases, to make the classifier more specific, and put more constraints on the classes. The question is, if this in any way can improve any relevant measure, and if it can on which criteria should we choose the number of clusters and which features should be used as input.

The problem is more formally that we are given a set S of m sequences $S = \{X_1, X_2, \dots, X_m\}$ of varying length i.e. $X_i = (\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \dots, \mathbf{x}_{n_i}^{(i)})$ for $\mathbf{x}_j^{(i)} \in \mathbb{R}^d$ for $1 \leq i \leq m, 1 \leq j \leq n_i$. If we for each $1 \leq i \leq m$ are given indexes $a_1^{(i)}, a_2^{(i)}, \dots, a_{k+1}^{(i)}$ then we define a *phase* $p_l = \{x_j^{(i)} | 1 \leq i \leq m, a_l^{(i)} \leq j < a_{l+1}^{(i)}\}$. The final measure is the sum of the variances of the phases i.e. the following:

$$J(k) = \sum_{l=1}^k Var(p_l)$$

We now have a cost function that we would like to minimize. The first question is, given a value for k how to find meaningful values for $a_1^{(i)}, a_2^{(i)}, \dots, a_{k+1}^{(i)}$ for each X_i . We do this by hierarchical clustering with the Ward linkage criteria. Since many features are present we choose to pick a single feature for which we try to find the smallest value. We can then iterate over all the features to find the feature and k with the lowest value of the cost function J .

$$Best_k = \arg \min_k J(k)$$

5.9 Putting it all together

For both the discretized and manual set of inputs we use an HMM and try to find the best way to provide emission probabilities using Gaussians, logistic regression, neural network and random forest. A short explanation exists for each one in Appendix A **►write on neural network◄**. For the Gaussian and logistic regression-based models we use feature selection to find the best features based on the validation set. We then try to combine the models and adjust their parameters to find the model that we believe is best before running the algorithm on the test set. We evaluate all the algorithms on the test set to also evaluate our approach. We also run both an RNN and an LSTM on the data and compare the results with the previous. Furthermore we test the event segmentation against just splitting the event uniformly.

As evaluation we use the F_2 score to put more emphasis on recall than precision. To make it very concrete we can imagine the system used by humans to detect events of interest having accompanying video and here it will either be very easy to see that a point in time does not correspond to a particular event, or the event will be so similar as to also be of interest.

Chapter 6

Experiments

Quality without results is
pointless. Results without quality
is boring.

Johan Cruyff

6.1 Setup

For event detection we train on the annotated data from the matches in the training set: HIK, SIF and BIF. We predict on the validation set. We run the experiments on an i7-4790K CPU with a Geforce 970GTX GPU and 8GB RAM.

For all algorithms we use class weights and scaling to 0 mean and unit variance. For each learning algorithm we use the following default settings unless stated otherwise with the input being vectors of dimension n :

Logistic regression: $C = 1$,

Gaussian mixture model, $k = 1$, i.e. a single Gaussian.

Neural network: dropout $p = 0.5$ in hidden layers, a single hidden layer with size n/p , sigmoid activation functions, softmax in output. The optimizer is SGD with keras default values.

Random forest: sklearn default values, number of trees is 10 and the number of features is \sqrt{n}

6.2 Field estimation

Fitting a Gaussian to each set of out events on the Hobro IK match, we estimate the placement of the field lines in the coordinate system defined by the data.

Near touch line $\mu = 0.55$ $\sigma = 0.82$

Far touch line $\mu = 68.71$ $\sigma = 0.57$

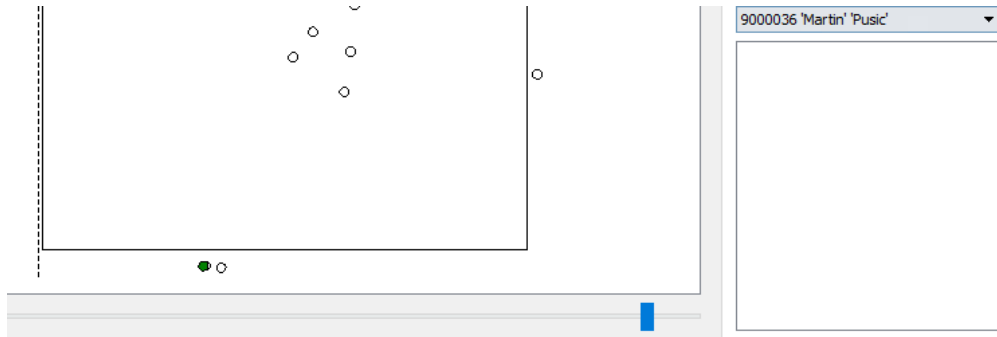


Figure 6.1: EFB second half. Pusic is still on the bench, the signal of Hassan is behind the opponents goal but his state has correctly been labeled as Perm_out, making him part of a substitution.

Left goal line: $\mu = 0.85$ $\sigma = 0.88$

Right goal line: $\mu = 103.82$ $\sigma = 0.92$

Most fields in the professional soccer world today have the standard recommended size by FIFA, 105×68 . Our results suggest that the home field of FCM is actually 2 meters shorter. There are of course both noise in the data, uncertainty in the annotation and the possibility of error in the calibration.

6.3 Substitution detection

Given the field location obtained in Experiment 6.2 we predict which classes the players belong to when they are out of the field and based on this we can predict the substitutions. The results can be seen in Table 6.1

We can also observe that the algorithm basically works. We correctly identify players leaving and entering the field. As for the various artifacts there are explanations for all of them.

A substitution is completely missed in the SE match because it was annotated to have happened in the 46th minute when it was actually made at half-time. This could be solved by checking which players actually are on the field between the two halves.

There is only a single false positive which is in the OB match in the minute that the second half ends. The reason is that Igboun is outside the field when the data that we have stops.

There are many false negatives. In the EFB match, data for Pusic's belt is actually there but it was worn by another player that day as can be seen in Figure 6.1. The data for Rasmussen in SIF and Pusic in RFC are there but the signal is completely lost. In the RFC game alle players lose the signal once in a while which explains why it takes longer to detect that Royer and Mabil have left the field. The rest of the false negatives are due to missing data for the players.

We can conclude from this experiment that we made a robust model that works.

<p>EFB 2</p> <p>True events</p> <p>58 Hassan \leftrightarrow Larsen</p> <p>67 Rasmussen \leftrightarrow Pusic</p> <p>83 Poulsen \leftrightarrow Urena</p> <p>Predicted events</p> <p>58 Hassan \leftrightarrow Larsen</p> <p>67 Rasmussen \leftarrow</p> <p>83 Poulsen \leftrightarrow Urena</p>	<p>BIF 2</p> <p>True events</p> <p>57 Dickoh \leftrightarrow Sisto</p> <p>72 Hassan \leftrightarrow Urena</p> <p>83 Banggard \leftrightarrow Bach Bak</p> <p>Predicted events</p> <p>57 Dickoh \leftarrow</p> <p>72 Hassan \leftrightarrow Urena</p> <p>83 Banggaard \leftarrow</p>
<p>OB 2</p> <p>True events</p> <p>62 Uzochukwu \leftrightarrow Hassan</p> <p>71 Pusic \leftrightarrow Rasmussen</p> <p>83 Bach Bak \leftrightarrow Rømer</p> <p>Predicted events</p> <p>62 Uzochukwu \leftrightarrow Hassan</p> <p>71 Pusic \leftrightarrow Rasmussen</p> <p>83 Bach Bak \leftarrow</p> <p>94 Igboun \leftarrow</p>	<p>SE 2</p> <p>True events</p> <p>46 Bach Bak \leftrightarrow Rømer</p> <p>71 Sisto \leftrightarrow Rasmussen</p> <p>78 Olsson \leftrightarrow Urena</p> <p>Predicted events</p> <p>71 Sisto \leftarrow</p> <p>78 Olsson \leftrightarrow Urena</p>
<p>HIK 2</p> <p>True events</p> <p>71 Pusic \leftrightarrow Urena</p> <p>78 Andersson \leftrightarrow Duelund</p> <p>87 Sparv \leftrightarrow Uzochukwu</p> <p>Predicted events</p> <p>72 Pusic \leftrightarrow Urena</p> <p>79 Andersson \leftarrow</p> <p>87 Sparv \leftrightarrow Uzochukwu</p>	<p>FCK 2</p> <p>True events</p> <p>61 Olsson \leftrightarrow Andersson</p> <p>61 Pusic \leftrightarrow Rasmussen</p> <p>79 Sisto \leftrightarrow Onuachu</p> <p>Predicted events</p> <p>61 Olsson \leftrightarrow Andersson</p> <p>61 Pusic \leftarrow</p> <p>78 Sisto \leftarrow</p>
<p>SIF 2</p> <p>True events</p> <p>63 Duelund \leftrightarrow Andersson</p> <p>68 Sparv \leftrightarrow Rasmussen</p> <p>Predicted events</p> <p>63 Andersson \rightarrow</p> <p>68 Sparv \leftarrow</p>	<p>RFC 2</p> <p>True events</p> <p>69 Duelund \leftrightarrow Pusic</p> <p>82 Royer \leftrightarrow Mabil</p> <p>94 Sisto \leftrightarrow Gemmer</p> <p>Predicted events</p> <p>69 Duelund \leftarrow</p> <p>85 Royer \leftarrow</p> <p>97 Sisto \leftarrow</p>

Table 6.1: Player substitutions

	Gaussian	LR	NN	RF
5	0.10	0.52	0.57	0.53
10	0.12	0.51	0.55	0.54
15	0.1	0.46	0.53	0.51
20	0.08	0.46	0.60	0.54
25	0.08	0.49	0.52	0.51
30	0.08	0.42	0.56	0.47
35	0.08	0.44	0.62	0.50
40	0.08	0.40	0.59	0.50
45	0.10	0.41	0.63	0.45
50			0.61	0.40

Table 6.2: F_2 -score for learning algorithms on the discretized field.

No players leaving the field for a short time are taking to be substituted and no noisy substitutes suddenly enter the field to leave it again. This achievement is solely due to the HMM, as the Gaussians only can tell what the probability of being on the field is, not if the player is actually playing or is a substitute.

6.4 Discretized

We test for the value of *hor_cells* starting at 5 and going to 50 in steps of 5 for all four learning algorithms. For the random forest we take the mean over 3 runs of the algorithm. The neural network has the samples shuffled, a batch size of 32 and is trained for 30 epochs.

The Gaussian model definitely performs very poorly suggesting that a single Gaussian is a poor way to represent a state using this input. We also see that the winner is the hybrid HMM-ANN and it can be considered surprising that it keeps performing well even with a high number of parameters as a value of *hor_cells* of 50 gives 1820 input parameters and a hidden layer of double the size. We attribute this to the Viterbi algorithm. To get an idea of the impact of dropout we also ran the experiment without dropout. The result was 0.59 so to get a better idea of the impact one would have to take the mean over many runs.

6.5 Manual features

We will now run the algorithms with the input vectors being the manual features that we extract. First we use all of them, then we extract features for the two simplest models, the logistic regression and the Gaussian HMM and finally we combine the models with the discretized data from before.

	Precision	Recall	F_2
Gaussian	0.26	0.88	0.57
LR	0.66	0.85	0.81
NN	0.45	0.83	0.71
RF	0.87	0.65	0.69

Table 6.3: F_2 -score for learning algorithms on the discretized field.

6.5.1 Using all features

We use all the manual features. For the non-deterministic algorithms, the neural network and the random forest we take the best results over several runs.

In Table 6.3 we see how the picture changes. The Gaussians still give the worst result, but the model has a the highest recall of all the methods. Interestingly logistic regression not only performs best, but at the same time we get a deterministic result. Since we use the F_2 -score the neural network performs better than the random forest.

6.5.2 Feature selection

We run backwards elimination on the deterministic models. Logistic regression gets to 0.85 and just including the features MeanX and HierCluDist3 gives a result of 0.75, suggesting that these features are very good at describing the data.

6.5.3 Combining with the discretized

We would expect to get better results by combining the two inputs, both the discretized and manual. Tweaking the parameters of the algorithms is also done. In Table 6.5 we see the results.

The random forest consistently lies very close to 100% precision when using the discretized features as well. If this were the goal we would be very happy. We do see that the feature selection has given us a classifier with a very nice recall, finding 19 out of 20 events in the matches. Since we only have one validation set that we optimize the parameters against it would be naive to think that we will get the same performance on the test set. In fact, running the full algorithm on the training sets gives only a score of 0.70 even though the algorithm is trained on this data. We have the same problem with the other algorithms if we pick the best with respect to the validation set. However problematic it might be we nonetheless choose to continue with the chosen model.

6.6 Test set evaluation

We run the logistic regression model with $C=10$ and with the features that give the best performance on the validation set. We see the complete results in Table 6.6.

Logistic regression		Gaussian	
0.806	AbsMidDistXPosFeature	0.627	LengthFeature
0.814	HierCluDist5Feature	0.647	HierCluDist4Feature
0.833	StdDevFeature	0.668	WidthFeature
0.833	WidthFeature	0.675	AbsMidDistXPosFeature
0.833	MinDPosXFeature	0.673	MinXPosYFeature
0.833	LengthFeature	0.675	MeanXVelFeature
0.841	MinXPosXFeature	0.681	StdDevFeature
0.845	HierCluDist2Feature	0.688	AbsAccFeature
0.849	MinXPosYFeature	<-	0.704 MinDPosYFeature
0.849	MinDPosYFeature	<-	0.720 TeamFormationFeature
0.849	MeanXAccFeature	<-	0.742 HierCluDist3Feature
0.841	MaxXPosXFeature	0.759	HierCluDist6Feature
0.829	LWRatioFeature	0.739	MeanXPosFeature
0.814	NormFormTeamFeature	0.762	HierCluDist5Feature
0.803	HierCluDist4Feature	0.742	LWRatioFeature
0.814	HierCluDist6Feature	0.743	MaxXPosXFeature
0.833	MaxDPosXFeature	0.749	MinDPosXFeature
0.826	AbsVelFeature	0.732	AbsVelFeature
0.811	AbsMeanYPosFeature	0.726	MeanXAccFeature
0.783	TeamFormationFeature	0.693	MaxDPosXFeature
0.781	MaxXPosYFeature	0.708	HierCluDist2Feature
0.723	MaxDPosYFeature	0.672	MaxXPosYFeature
0.735	MeanXVelFeature	0.651	AbsMeanYPosFeature
0.746	AbsAccFeature	0.588	MinXPosXFeature
0.524	HierCluDist3Feature	0.560	MaxDPosYFeature
0.000	MeanXPosFeature	0.000	NormFormTeamFeature

Table 6.4: Backwards elimination for logistic regression on the left and Gaussian on the right

Alg.	Parameters	Precision	Recall	F_2
LR	C=10, manual + feature selection	0.64	0.95	0.87
LR	C=10, hor_cells=5, manual + feature selection	-	-	0.79
NN	batch size=10, hor_cells=20 + manual	0.51	0.80	0.72
RF	hor_cells=10 + manual	1.00	0.61	0.66
RF	manual	0.85	0.71	0.73
RF	complete trees “bagging”	0.62	0.76	0.72
RF	complete trees “bagging”, hor_cells=10	0.69	0.76	0.74

Table 6.5: F_2 -score for combinations of input and changing parameters.

All	T Pos	T Neg	Total	Precision	Goal	T Pos	T Neg	Total	Precision
P Pos	25	25	50	0.5	P Pos	1	3	4	0.25
P Neg	9				P Neg	3			
Total	34			F_2	Total	4			
Recall	0.74			0.67	Recall	0.25			

Corner	T Pos	T Neg	Total	Precision	Opp_corner	T Pos	T Neg	Total	Precision
P Pos	8	9	17	0.47	P Pos	16	13	29	0.55
P Neg	3				P Neg	3			
Total	11				Total	19			
Recall	0.73				Recall	0.84			

Table 6.6: Final results

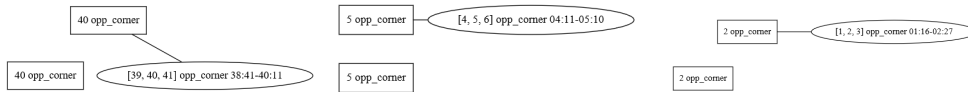


Figure 6.2: False negative opponent corners in the test set.

We see that we have higher recall than precision which makes sense since we extracted the features with that purpose. The goal detection is really poor but does not have a great impact on the final result given that there so few instances. Since we can not inspect video to see what actually happened we can take a look at the produced matchings in Figure 6.2. It shows both why a pairing is necessary in the evaluation and a fundamental problem, namely that of detecting corners be it opponent or not, that are very close to each other.

6.7 Recurrent neural networks

We experiment with a an RNN with two hidden layers. The input layer and hidden layers are either standard or using LSTM block. We run with a batch size of one which means a half at a time, and for 200 epochs. The activation of the layers is tanh and we apply dropout only at the hidden layers.

We use all manual features. We evaluate on the training set and the matches EFB and OB.

We see from Table 6.7 that the simple RNN loses to the LSTM in score while it is considerably faster given that it is a much simpler model. Using also discretized data gives many more hidden units. We discretize using $hor_cells = 30$ and a dropout of 0.5. With this the LSTM achieves 0.56 with only 30 epochs.

6.8 Event segmentation

We run the presented algorithm for event segmentation for each event on all the training matches to find the best feature and an optimal k . We also run

dropout	Simple	LSTM
None	0.24	0.54
0.1	0.21	0.58
0.2	0.27	0.49
0.3	0.25	0.39
0.4	0.31	0.49
0.5	0.27	0.45
Runtime / exp	~55m	~2h

Table 6.7: F₁-score for RNNs.

Event	feature	k	$J(k)$	Event	feature	k	$J(k)$
corner	MaxXD	2	0.33	corner	MaxXD	2	0.76
opp_corner	NTF	2	0.91	opp_corner	-	2	-
goal	MaxDX	2	0.41	goal	AbsMidDistX	2	0.63
	MeanX	3	0.29		MaxX	3	0.65
	AbsMidDistX	4	0.27		AbsMidDistX	4	0.67
	AbsMidDistX	5	0.53		MeanX	5	0.82

Table 6.8: Event segmentation results. Left: the proposed algorithm using agglomerative clustering. Right: Splitting each event into k segments of same size.

the algorithm just splitting each into equal sized segments. The results can be seen in Table 6.8 left and right respectively. We can see that there is a reason to use agglomerative clustering for splitting each event, as the values of $J(k)$ are consistently lower meaning that there is lower intra-phase variance. As for the features we are a little surprised to see XArgMaxD which we recall is the x -value of the player with maximum d -value, but the others make sense in that they capture that the players are close to the opponents goal when scoring and then move back for the kick.off.

We can see that apparently the best feature for splitting a corner event is the d -value of the player with highest x -value. We can see a typical example in Figure 6.3 where often the player closest to the opponents goal in the x -direction will try to also get close in the y -direction giving a lower value for d than the corner kick taker marked in green that will eventually be the player with highest x -value and thus giving a very high value for d . This is only so because the corner kick takers of FCM are selected to take corners that curve towards the goal and thereby be in a position of highest x -value. We can therefore not necessarily expect this to hold for other teams.

For the opponent corners we see the normalized team formation feature. This suggests that the players are defending in formation, which is then broken when they go into position of defending the corner kick.

Unfortunately we have not been able to make it help in the classification task.

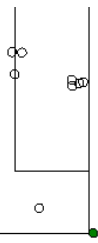
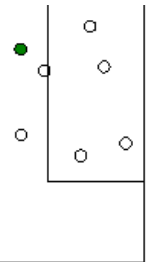


Figure 6.3: Corner event segmentation

Chapter 7

Discussion

We find the approach to event detection using spatio-temporal data quite interesting. We can see that the problem is definitely not a trivial one, neither in the setup nor in finding the correct algorithms. The discretization of the field can be considered a rather naive approach although with reasonable results. The hand-crafted features with logistic regression were selected here as the optimal algorithm but as convolutional neural networks show state-of-the-art results on many problems it would also make sense to attack the discretized field with them to get an automatic extraction of features instead of creating them by hand.

There are a multitude of other things to try out to improve the results as well. It would be very interesting to see a more context-aware neural network in the HMM. This could be a neural network using a window to get context on both sides or an RNN. This could possibly help in one of the very difficult problems that consists, namely separating corners that come in succession. The problem is difficult because of the very little time i.e. a couple of seconds that can be between two corners. This means that slight errors in annotation and synchronization can disturb a high percentage of the labels. There are also very few labels with very low weight given that they are annotated by the non-event label. Also, using the kind of evaluation that we do and feature selection can promote a learner that is very bad at finding the whole duration of an event but can separate some corners. This is undesirable behaviour and disturbs the whole process of creating a good model. One approach to solving the problem - although an unfruitful one - was to segment the events in to phases. Another approach would be to label consecutive corners differently or the labels between them. Further it would be interesting to make a postprocessing e.g using an RNN to classify each candidate event sequence. It could then be classified into single corner, double corner, corner with substitution etc.

We could also try combining some of the best manual features although it can be difficult to know exactly which to use, based on feature detection given that it is greedy and not an exhaustive search. A genetic algorithm could be interesting here.

We must of course remember the setup and what can actually be expected from

the problem. To really attack the problem well, we believe that the data should be annotated using video to get the exact start and end of each event. This would make the validation much more purposeful and avoid that a corner of one second triggers a true positive if it is within two minutes of the true event. It would also allow for annotating the matches completely with other events and thereby get more classes.

Chapter 8

Conclusion

For the purpose of detecting the ball in 3D to help the referee make decisions we have made an extensive review of the literature. At the moment this is generally not possible, due to too much noise, but detection of the ball in video is certainly possible when the data contains enough information. We have also presented work on how to reason about the distance of the ball to a single camera and shown that it depends heavily on the resolution used. For detecting events based on player data we have made an overview of some of the theory that already uses spatio-temporal data, and used some of these features. For the actual detection we have provided a number of approaches resulting in a final model consisting of an HMM using emission probabilities from a logistic regression classifier. We have had to settle for only 3 partly annotated matches but still find around 6 out of 10 corners, and goal events in unseen data with a precision rate of 50%. A direct problem to solve is successive events of the same type. We proposed a method to help with this problem but it showed to be of no value in practice. In the presence of more annotated data it is our firm belief that much higher scores can be achieved.

Appendix A

Machine learning algorithms

A.1 Logistic regression

Logistic regression is a linear classifier that uses a soft threshold as output. As is the case also in linear regression and the perceptron we want to estimate a weight vector \mathbf{w} and a bias term b in what we call the signal:

$$\mathbf{w}^T \mathbf{x} + b$$

The perceptron uses the sign function to bring the values of the signal in $-1, 1$. Linear regression uses the signal directly. The problem with the perceptron is that it gives a binary output and it is therefore difficult to optimize. Linear regression makes no classification as the signal is used directly. In logistic regression the sigmoid function

$$\theta(s) = \frac{e^s}{1 + e^s}$$

is used, which gives continuous values in the interval $]0, 1[$. These values can then be interpreted as probabilities of belonging to one class or the other. A nice property of the sigmoid function is the following:

$$\theta(-s) = \frac{e^{-s}}{1 + e^{-s}} = \frac{e^s e^{-s}}{e^s 1 + e^s e^{-s}} = \frac{1}{e^s + 1} \quad (\text{A.1})$$

$$\theta(-s) = \frac{1}{e^s + 1} = \frac{1 + e^s - e^s}{e^s + 1} = \frac{1 + e^s}{e^s + 1} - \frac{e^s}{e^s + 1} = 1 - \theta(s) \quad (\text{A.2})$$

In a noise-free setting we would have a target function f that given values \mathbf{x} produce labels y in one of two classes corresponding to $y = 1$ and $y = -1$. We want our classifier h to model $P(y|\mathbf{x})$ i.e. the probability of $y = 1$ which we can write as:

$$P(y|\mathbf{x}) = \begin{cases} h(\mathbf{x}) & y = 1 \\ 1 - h(\mathbf{x}) & y = -1 \end{cases}$$

Now let us consider $h(\mathbf{x}) = \theta(\mathbf{w}^T \mathbf{x})$, where we for ease of notation define $x_0 = 1$ and $w_0 = b$. According to Eq. A.1 and A.2 we have that

$$P(y|\mathbf{x}) = \begin{cases} \theta(\mathbf{w}^T \mathbf{x}) & y = 1 \\ 1 - \theta(\mathbf{w}^T \mathbf{x}) & y = -1 \end{cases} = \theta(y\mathbf{w}^T \mathbf{x}) \quad (\text{A.3})$$

$$P(y|\mathbf{x}) = \theta(y\mathbf{w}^T \mathbf{x}) = \frac{1}{e^{-y\mathbf{w}^T \mathbf{x}} + 1} \quad (\text{A.4})$$

Given N independent data points $(x_1, y_1), (x_2, y_2) \dots (x_N, y_N)$ the maximum likelihood is defined as $\prod_{n=1}^N \theta(y\mathbf{w}^T x_n)$, but instead of maximizing this quantity it is more convenient to minimize the log likelihood given by:

$$\begin{aligned} -\log \prod_{n=1}^N \theta(y\mathbf{w}_n^T x_n) &= \sum_{n=1}^N -\log \theta(y_n \mathbf{w}^T x_n) \\ &= \sum_{n=1}^N \log \frac{1}{\theta(y_n \mathbf{w}^T x_n)} \\ &= \sum_{n=1}^N \log (e^{-y_n \mathbf{w}^T \mathbf{x}_n} + 1) \end{aligned}$$

Where we get the last line using Eq. A.4. We can define this measure to be the error on the training set and by finding the gradient we can use gradient descent to approximate the optimal solution.

An important note to make is that even though the classifier is linear there is no restriction on how we manipulate a vector \mathbf{x} before it is put into the algorithm. We can thus input powers and fractions between values and in that way make it classify data sets that are not linear per se. Doing this systematically however, quickly blows up the dimensionality.

A.2 Random forestA

A *random forest* is a method for discriminative classification based on the use of multiple decision trees. Given that it uses multiple classifiers it is known as an *ensemble* method. A decision tree has the property that it can fit anything, if allowed arbitrary depth. In machine learning terms we say that it has 0 bias. The problem is that it is deemed to overfit the data i.e. have a high variance. *Bagging* is an acronym for *bootstrap aggregation* and also uses multiple classifiers. It uses bootstrapping on the training data for each classifier and then aggregates them such that an instance is classified based on a voting scheme over all the trees. A random forest uses the scheme from bagging but also picks a random subset of the features that it considers for each tree.

When using a random forest in practice there are - as in most more complicated algorithms - many parameters that can be tweaked affecting the final output. We have to pick the number of trees in the forest and the number of features to use for each tree. To these comes the option to truncate the trees themselves, limiting the number of nodes or the depths of the trees.

Appendix B

Matching graphs

B.1 Final result



Figure B.1: FCK half 1 and 2

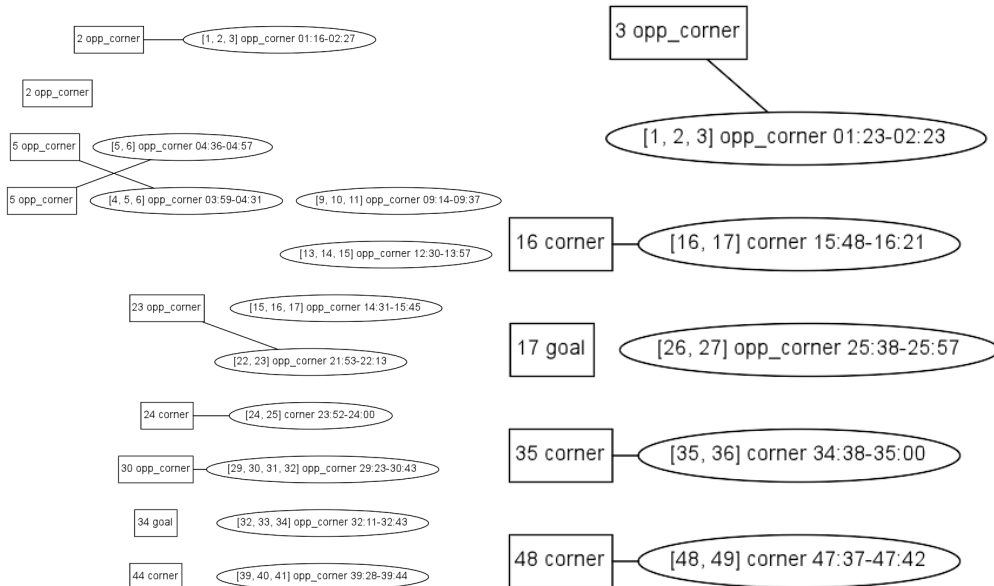


Figure B.2: VFF half 1 and 2

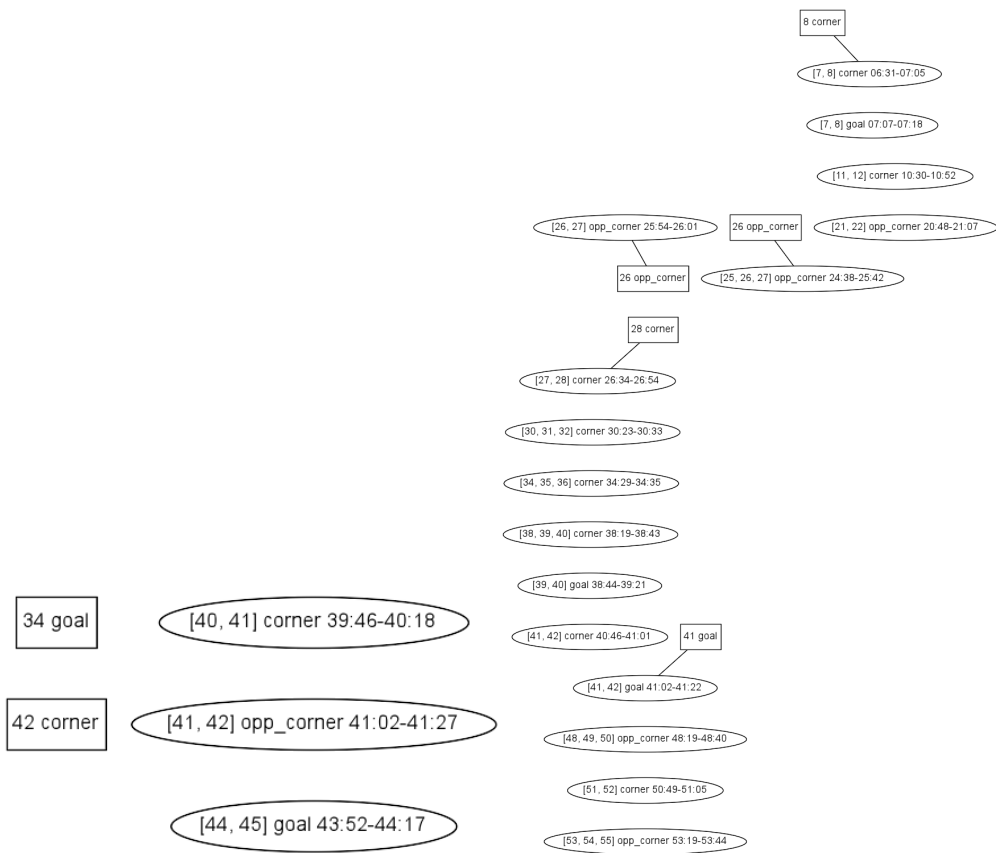


Figure B.3: SE half 1 and 2

Bibliography

- [1] Association football. "https://en.wikipedia.org/wiki/Association_football", 2015.
- [2] dblp computer science library. "<http://dblp.uni-trier.de>", 2015.
- [3] Efter et Års hype: Nu finder fcm spillere med 'big data'. "http://ekstrabladet.dk/sport/fodbold/dansk_fodbold/superligaen/fc_midtjylland/efter-et-aars-hype-nu-finder-fcm-spillere-med-big-data/5657344", 2015.
- [4] Fc midtjylland. "https://da.wikipedia.org/wiki/FC_Midtjylland", 2015.
- [5] Google scholar. "<https://scholar.google.com>", 2015.
- [6] How data, not humans, run this danish football club. "<https://decorrespondent.nl/2607/How-data-not-humans-run-this-Danish-football-club/230219386155-d2948861>", 2015.
- [7] Keras. "<https://keras.io>", 2015.
- [8] Laws of the game. "http://www.fifa.com/mm/Document/FootballDevelopment/Refereeing/02/36/01/11/LawsofthegamewebEN_Neutral.pdf", 2015.
- [9] Prozone. "<http://prozonesports.stats.com/subsector/football/>", 2015.
- [10] Scientific program of 8th world congress on science and football. "<http://wcsf2015.ku.dk/scientificprogram/>", 2015.
- [11] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*. AMLBook, 2012.
- [12] Anubhav Agarwal, CV Jawahar, and PJ Narayanan. A survey of planar homography estimation techniques. *Centre for Visual Information Technology, Tech. Rep. IIT/TR/2005/12*, 2005.

- [13] N Ancona, G Cicirelli, A Branca, and A Distante. Goal detection in football by using support vector machines for classification. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, volume 1, pages 611–616. IEEE, 2001.
- [14] Yasuo Ariki, Tetsuya Takiguchi, and Kazuki Yano. Digital camera work for soccer video production with event recognition and accurate ball tracking by switching search method. In *Multimedia and Expo, 2008 IEEE International Conference on*, pages 889–892. IEEE, 2008.
- [15] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188, 2002.
- [16] Thomas Bebie and Hanspeter Bieri. Soccerman-reconstructing soccer games from video sequences. In *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on*, volume 1, pages 898–902. IEEE, 1998.
- [17] Thomas Bebie and Hanspeter Bieri. A video-based 3d-reconstruction of soccer games. In *Computer Graphics Forum*, volume 19, pages 391–400. Wiley Online Library, 2000.
- [18] Michael Beetz, Nicolai von Hoyningen-Huene, Bernhard Kirchlechner, Suat Gedikli, Francisco Siles, Murat Durus, and Martin Lames. Aspogamo: Automated sports game analysis models. *International Journal of Computer Science in Sport*, 8(1):1–21, 2009.
- [19] Alina Bialkowski, Patrick Lucey, Peter Carr, Yisong Yue, and Iain Matthews. "win at home and draw away": automatic formation analysis highlighting the differences in home and away team behaviors. *Proceedings of MIT Sloan Sports Analytics*, 2014.
- [20] Alina Bialkowski, Patrick Lucey, Peter Carr, Yisong Yue, Sridha Sridharan, and Iain Matthews. Identifying team style in soccer using formations learned from spatiotemporal tracking data. In *Data Mining Workshop (ICDMW), 2014 IEEE International Conference on*, pages 9–14. IEEE, 2014.
- [21] Alina Bialkowski, Patrick Lucey, Peter Carr, Yisong Yue, Sridha Sridharan, and Iain Matthews. Large-scale analysis of soccer matches using spatiotemporal tracking data. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 725–730. IEEE, 2014.
- [22] Kyuhyoung Choi and Yongdeuk Seo. Probabilistic tracking of the soccer ball. In *Statistical Methods in Video Processing*, pages 50–60. Springer, 2004.
- [23] Kyuhyoung Choi and Yongduek Seo. Tracking soccer ball in tv broadcast video. In *Image Analysis and Processing-ICIAP 2005*, pages 661–668. Springer, 2005.

- [24] Tommaso De Marco, Marco Leo, and Cosimo Distanto. Soccer ball detection with isophotes curvature analysis. In *Image Analysis and Processing—ICIAP 2013*, pages 793–802. Springer, 2013.
- [25] Tiziana D’Orazio, Nicola Ancona, Grazia Cicirelli, and Massimiliano Nitti. A ball detection algorithm for real soccer image sequences. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 1, pages 210–213. IEEE, 2002.
- [26] Tiziana D’Orazio, Cataldo Guaragnella, Marco Leo, and Arcangelo Distanto. A new algorithm for ball recognition using circle hough transform and neural classifier. *Pattern recognition*, 37(3):393–408, 2004.
- [27] Tiziana D’Orazio and Marco Leo. A review of vision-based systems for soccer video analysis. *Pattern recognition*, 43(8):2911–2926, 2010.
- [28] Tiziana D’Orazio, Marco Leo, Nicola Mosca, Paolo Spagnolo, and Pier Luigi Mazzeo. A semi-automatic system for ground truth generation of soccer video sequences. In *Advanced Video and Signal Based Surveillance, 2009. AVSS’09. Sixth IEEE International Conference on*, pages 559–564. IEEE, 2009.
- [29] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009.
- [30] Elan Dubrofsky. Homography estimation. Master’s thesis, UNIVERSITY OF BRITISH COLUMBIA (Vancouver, 2009).
- [31] Jordi Duch, Joshua S Waitzman, and Luís A Nunes Amaral. Quantifying the performance of individual players in a team activity. *PloS one*, 5(6):e10937, 2010.
- [32] Frederic Dufaux and Janusz Konrad. Efficient, robust, and fast global motion estimation for video coding. *Image Processing, IEEE Transactions on*, 9(3):497–501, 2000.
- [33] Ramsey Faragher et al. Understanding the basis of the kalman filter via a simple and intuitive derivation. *IEEE Signal processing magazine*, 29(5):128–132, 2012.
- [34] Hugo Folgado, Koen APM Lemmink, Wouter Frencken, and Jaime Sampaio. Length, width and centroid distance as measures of teams tactical performance in youth football. *European Journal of Sport Science*, 14(sup1):S487–S492, 2014.
- [35] Wouter Frencken, Harjo de Poel, Chris Visscher, and Koen Lemmink. Variability of inter-team distances associated with match events in elite-standard soccer. *Journal of sports sciences*, 30(12):1207–1213, 2012.

- [36] Yihong Gong, Lim Teck Sin, Chua Hock Chuan, Hongjiang Zhang, and Masao Sakauchi. Automatic parsing of tv soccer programs. In *Multimedia Computing and Systems, 1995., Proceedings of the International Conference on*, pages 167–174. IEEE, 1995.
- [37] Alex Graves et al. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer, 2012.
- [38] Joachim Gudmundsson and Thomas Wolle. Towards automated football analysis: Algorithms and data structures. In *Proc. 10th Australasian Conf. on Mathematics and Computers in Sport*. Citeseer, 2010.
- [39] Joachim Gudmundsson and Thomas Wolle. Football analysis using spatio-temporal tools. *Computers, Environment and Urban Systems*, 47:16–27, 2014.
- [40] Zhenhua Guo, Lei Zhang, and David Zhang. A completed modeling of local binary pattern operator for texture classification. *Image Processing, IEEE Transactions on*, 19(6):1657–1663, 2010.
- [41] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, 3:1157–1182, 2003.
- [42] Josef Halbinger and Juergen Metzler. Video-based soccer ball detection in difficult situations. In *Sports Science Research and Technology Support*, pages 17–24. Springer, 2013.
- [43] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- [44] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
- [45] Shoji Hirano and Shusaku Tsumoto. A clustering method for spatio-temporal data and its application to soccer game records. In *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 612–621. Springer, 2005.
- [46] Yu Huang, Joan Llach, and Sitaram Bhagavathy. Players and ball detection in soccer videos based on color segmentation and shape analysis. In *Multimedia Content Analysis and Mining*, pages 416–425. Springer, 2007.
- [47] Michael Isard and Andrew Blake. Condensation-conditional density propagation for visual tracking. *International journal of computer vision*, 29(1):5–28, 1998.
- [48] Norihiro Ishii, Itaru Kitahara, Yoshinari Kameda, and Yuichi Ohta. 3d tracking of a soccer ball using two synchronized cameras. In *Advances in Multimedia Information Processing-PCM 2007*, pages 196–205. Springer, 2007.

- [49] Chan-Hyun Kang, Jung-Rae Hwang, and Ki-Joune Li. Trajectory analysis for soccer players. In *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on*, pages 377–381. IEEE, 2006.
- [50] Ho-Chul Kim, Oje Kwon, and Ki-Joune Li. Spatial and spatiotemporal analysis of soccer. In *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 385–388. ACM, 2011.
- [51] Jong-Yun Kim and Tae-Yong Kim. Soccer ball tracking using dynamic kalman filter with velocity control. In *Computer Graphics, Imaging and Visualization, 2009. CGIV'09. Sixth International Conference on*, pages 367–374. IEEE, 2009.
- [52] Kihwan Kim. Motion field to predict play evolution in dynamic sports scenes. <https://www.youtube.com/watch?v=jrksnCR1S0s>, 2010.
- [53] Kihwan Kim, Matthias Grundmann, Ariel Shamir, Iain Matthews, Jessica Hodgins, and Irfan Essa. Motion fields to predict play evolution in dynamic sport scenes. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 840–847. IEEE, 2010.
- [54] Kyungnam Kim, Thanarat H Chalidabhongse, David Harwood, and Larry Davis. Real-time foreground–background segmentation using codebook model. *Real-time imaging*, 11(3):172–185, 2005.
- [55] Taeone Kim, Yongduek Seo, and Ki-Sang Hong. Physics-based 3d position analysis of a soccer ball from monocular image sequences. In *Computer Vision, 1998. Sixth International Conference on*, pages 721–726. IEEE, 1998.
- [56] Marco Leo, Tiziana D’Orazio, Paolo Spagnolo, Pier Luigi Mazzeo, and Arcangelo Distanto. Sift based ball recognition in soccer images. In *Image and Signal Processing*, pages 263–272. Springer, 2008.
- [57] Marco Leo, Pier Luigi Mazzeo, Massimiliano Nitti, and Paolo Spagnolo. Accurate ball detection in soccer images using probabilistic analysis of salient regions. *Machine vision and applications*, 24(8):1561–1574, 2013.
- [58] Dawei Liang, Yang Liu, Qingming Huang, and Wen Gao. A scheme for ball detection and tracking in broadcast soccer video. In *Advances in Multimedia Information Processing-PCM 2005*, pages 864–875. Springer, 2005.
- [59] Yang Liu, Dawei Liang, Qingming Huang, and Wen Gao. Extracting 3d information from broadcast soccer video. *Image and Vision Computing*, 24(10):1146–1162, 2006.
- [60] Patrick Lucey, Alina Bialkowski, Peter Carr, Eric Foote, and Iain Matthews. Characterizing multi-agent team behavior from partial team tracings: Evidence from the english premier league. In *AAAI*. Citeseer, 2012.

- [61] Patrick Lucey, Alina Bialkowski, Mathew Monfort, Peter Carr, and Iain Matthews. "quality vs quantity": Improved shot prediction in soccer using strategic features from spatiotemporal data. MIT Sloan Sports Analytics Conference, 2014.
- [62] Patrick Lucey, Dean Oliver, Peter Carr, Joe Roth, and Iain Matthews. Assessing team strategy using spatiotemporal data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1366–1374. ACM, 2013.
- [63] Pier Luigi Mazzeo, Marco Leo, Paolo Spagnolo, and Massimiliano Nitti. Soccer ball detection by comparing different feature extraction methodologies. *Advances in Artificial Intelligence*, 2012:6, 2012.
- [64] PL Mazzeo, P Spagnolo, M Leo, T De Marco, and C Distanti. Ball detection in soccer images using isophote’s curvature and discriminative features. *Pattern Analysis and Applications*, pages 1–10, 2015.
- [65] JÄErgen Metzler and Frank Pagel. 3d trajectory reconstruction of the soccer ball for single static camera systems. In *MVA’13*, pages 121–124, 2013.
- [66] Teruhisa Misu, Atsushi Matsui, Masahide Naemura, Masahiro Fujii, and Naomi Yagi. Distributed particle filtering for multiocular soccer-ball tracking. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 3, pages III–937. IEEE, 2007.
- [67] Jun Miura, Takumi Shimawaki, Takuro Sakiyama, and Yoshiaki Shirai. Ball route estimation under heavy occlusion in broadcast soccer video. *Computer Vision and Image Understanding*, 113(5):653–662, 2009.
- [68] Yoshinori Ohno, Jun Miura, and Yoshiaki Shirai. Tracking players and estimation of the 3d position of a ball in soccer games. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 1, pages 145–148. IEEE, 2000.
- [69] Yoshinori Ohno, J Miurs, and Yoshiaki Shirai. Tracking players and a ball in soccer games. In *Multisensor Fusion and Integration for Intelligent Systems, 1999. MFI’99. Proceedings. 1999 IEEE/SICE/RSJ International Conference on*, pages 147–152. IEEE, 1999.
- [70] V Pallavi, Jayanta Mukherjee, Arun K Majumdar, and Shamik Sural. Ball detection from broadcast soccer videos using static and dynamic features. *Journal of Visual Communication and Image Representation*, 19(7):426–436, 2008.
- [71] Svein Arne Pettersen, Dag Johansen, Håvard Johansen, Vegard Berg-Johansen, Vamsidhar Reddy Gaddam, Asgeir Mortensen, Ragnar Langseth, Carsten Griwodz, Håkon Kvale Stensland, and Pål Halvorsen. Soccer video and player position dataset. In *Proceedings of the 5th ACM Multimedia Systems Conference*, pages 18–23. ACM, 2014.

- [72] Ian Reid and Andrew Zisserman. Goal-directed video metrology. In *Computer Vision-ECCV'96*, pages 647–658. Springer, 1996.
- [73] Ian D Reid and A North. 3d trajectories from a single viewpoint using shadows. In *BMVC*, volume 50, pages 51–52. Citeseer, 1998.
- [74] Jinchang Ren, James Orwell, and G Jones. Generating ball trajectory in soccer video sequences. 2006.
- [75] Jinchang Ren, James Orwell, Graeme Jones, Ming Xu, et al. A general framework for 3d soccer ball estimation and tracking. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 3, pages 1935–1938. IEEE, 2004.
- [76] Jinchang Ren, James Orwell, Graeme Jones, Ming Xu, et al. Real-time modeling of 3-d soccer ball trajectories from multiple fixed cameras. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(3):350–362, 2008.
- [77] Jinchang Ren, James Orwell, Graeme A Jones, and Ming Xu. Tracking the soccer ball using multiple fixed cameras. *Computer Vision and Image Understanding*, 113(5):633–642, 2009.
- [78] Jinchang Ren, Ming Xu, James Orwell, and Graeme A Jones. Multi-camera video surveillance for real-time analysis and reconstruction of soccer games. *Machine Vision and Applications*, 21(6):855–863, 2010.
- [79] Sam Roweis and Zoubin Ghahramani. A unifying review of linear gaussian models. *Neural computation*, 11(2):305–345, 1999.
- [80] Yongduek Seo, Sunghoon Choi, Hyunwoo Kim, and Ki-Sang Hong. Where are the ball and players? soccer game analysis with color-based tracking and image mosaick. In *Image Analysis and Processing*, pages 196–203. Springer, 1997.
- [81] Takumi Shimawaki, Jun Miura, Takuro Sakiyama, and Yoshiaki Shirai. Ball route estimation in broadcast soccer video. In *Proc. ECCV-2006 Workshop on Computer Vision Based Analysis in Sport Environments*, pages 26–37. Citeseer, 2006.
- [82] Chris Stauffer and W Eric L Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2. IEEE, 1999.
- [83] Tsuyoshi Taki and Jun-ichi Hasegawa. Dominant region: a basic feature for group motion analysis and its application to teamwork evaluation in soccer games. In *Electronic Imaging'99*, pages 48–57. International Society for Optics and Photonics, 1998.
- [84] Tsuyoshi Taki and Jun-ichi Hasegawa. Visualization of dominant region in team games and its application to teamwork analysis. In *Computer Graphics International, 2000. Proceedings*, pages 227–235. IEEE, 2000.

- [85] Tsuyoshi Taki, Jun-ichi Hasegawa, and Teruo Fukumura. Development of motion analysis system for quantitative evaluation of teamwork in soccer games. In *Image Processing, 1996. Proceedings., International Conference on*, volume 3, pages 815–818. IEEE, 1996.
- [86] Xiao-Feng Tong, Han-Qing Lu, and Qing-Shan Liu. An effective and fast soccer ball detection and tracking method. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 4, pages 795–798. IEEE, 2004.
- [87] Xiaofeng Tong, Tao Wang, Wenlong Li, Yimin Zhang, Bo Yang, Fei Wang, Lifeng Sun, and Shiqiang Yang. A three-level scheme for real-time ball tracking. In *Multimedia Content Analysis and Mining*, pages 161–171. Springer, 2007.
- [88] Vasanth Tovinkere and Richard J Qian. Detecting semantic events in soccer games: Towards a complete solution. In *null*, page 212. IEEE, 2001.
- [89] Akihito Yamada, Yoshiaki Shirai, and Jun Miura. Tracking players and a ball in video image sequence and estimating camera parameters for 3d interpretation of soccer games. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 1, pages 303–306. IEEE, 2002.
- [90] Junqing Yu, Yang Tang, Zhifang Wang, and Lejiang Shi. Playfield and ball detection in soccer video. In *Advances in Visual Computing*, pages 387–396. Springer, 2007.
- [91] Xinguo Yu, Hon Wai Leong, Changsheng Xu, and Qi Tian. Trajectory-based ball detection and tracking in broadcast soccer video. *Multimedia, IEEE Transactions on*, 8(6):1164–1178, 2006.
- [92] Xinguo Yu, Qi Tian, and Kong Wah Wan. A novel ball detection framework for real soccer video. In *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 2, pages II–265. IEEE, 2003.
- [93] Xinguo Yu, Xiaoying Tu, and Ee Luang Ang. Trajectory-based ball detection and tracking in broadcast soccer video with the aid of camera motion recovery. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 1543–1546. IEEE, 2007.
- [94] Xinguo Yu, Changshen Xu, Qi Tian, and Hon Wui Leong. A ball tracking framework for broadcast soccer video. In *Multimedia and Expo, 2003. ICME'03. Proceedings. 2003 International Conference on*, volume 2, pages II–273. IEEE, 2003.
- [95] Xinguo Yu, Changsheng Xu, Hon Wai Leong, Qi Tian, Qing Tang, and Kong Wah Wan. Trajectory-based ball detection and tracking with applications to semantic analysis of broadcast soccer video. In *ACM*, pages 11–20, 2003.

- [96] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.