University of London
Imperial College of Science, Technology and Medicine
Department of Computing

Computer Vision and Machine Learning for In-Play Tennis Analysis: Framework, Algorithms and Implementation

Silvia Vinyes Mora

supervised by Prof. William Knottenbelt

Submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of Imperial College London and
the Diploma of Imperial College London.
October, 2017.

Abstract

Statistical analysis has become an essential part of professional sports to the point that every major professional team employs expert analysts. With the widespread availability of high definition streaming data, increased processing power and algorithmic advances, statistical models are undergoing an evolution from static coarse-grained pre-play models based on simple statistical data to finer-grained dynamic in-play models that exploit spatio-temporal data and event streams.

Our primary hypothesis is that Computer Vision (CV) and Machine Learning (ML) research has opened up the opportunity to develop automated systems to collect the fine-grained data needed to support these more sophisticated models. We have chosen tennis as our focus on account of it being a highly structured sport which has enthusiastically embraced technology, but our research is also applicable to other sports.

In this work, we propose a novel framework for the application of CV and ML techniques to the in-play spatio-temporal analysis of tennis using commodity hardware with the ultimate objective of obtaining insights related to players' performance or prediction of future events. Our framework consists of three-layers: Vision, Classification and Modelling, each of which features various algorithmic innovations.

For the Vision Layer we propose a multi-camera system able to detect the player and ball positions in real-time, at over 60 fps. Their 3D inferred location has an error lower than 10 cm in 80% of the frames. The Classification Layer uses data from the previous layer to obtain high-level information. Our contributions within this layer are two-fold: (1) insightful visualizations of Vision Layer data and automatic extraction of high-level statistics (2) the first deep Neural Network for fine-grained action recognition in tennis which yields highly competitive results: 88.16% one versus all accuracy for classifying backhands, forehands and serves and 47.22% in classifying 12 finer-grained actions. The Modelling Layer incorporates the data obtained for the previous two layers to gain insights about the game. For this layer, we survey and critically examine different approaches of using tennis spatio-temporal data for prediction and analysis.

Our work opens the door to a deeper understanding of tennis and other sports leading to new approaches to coaching, better analysis of rivals for developing strategies and in-play match analysis to enhance spectators' experience, amongst many other applications.

Acknowledgements

I would like to express gratitude to:

- My supervisor, Professor William Knottenbelt, for his enthusiastic endorsement and guidance throughout the PhD. Will has been an excellent supervisor offering full support in my personal and professional development. Will is very inspirational and passionate about research, spreading these around him. I want to thank him for his invaluable trust and support throughout the PhD.
- The tennis coaches Matt Willcocks, Richard Hawkes, Darren Emery and Oren Holtzman for sharing their insights from years of coaching experience in tennis.
- My family and friends, whose love and support has been the force in overcoming the difficult times of my PhD and the best possible company to celebrating my successes.
- My parents Jordi and Lola for giving me all their best for my success since I was born. They have been an extraordinary source of love, motivation and support throughout my PhD. I could not thank them enough for this. Their interest in knowledge, perseverance and dedication has been the best example for me. Jordi, offering fascinating discussions constantly and giving me feedback about my thesis. Lola, for her trust and being an example of courage.
- My sister Marina for being my ally in sharing PhD experiences, discussing scientific matters, and offering her trust and support all along the PhD.
- My closest friends and fellow researchers Carla Campos, Silvia Gimeno, Victor Estella, Danielle Belgrave, Claudia Schultz, Cristian Florindo and Jean Kossaifi, for their friendship. Thank you for making my days more fun, having fascinating discussions and your unconditional support.
- All my friends who have constantly been there to provide their love and support, especially Marta Caballero, Aisling McCabe, Gergana Ivanova and Sara Pagani.
- The Computing MSc students who have worked in projects under my supervision. It has been a very enriching and rewarding experience.

- Jordi Constans and Ramon Roman for their useful feedback on this thesis.
- Dr Amani El-Kholy, Mrs Ann Halford, Ms Teresa Ng and Mr Hassan Patel for their constant support.
- The Engineering and Physical Sciences Research Council (EPSRC), for funding the Doctoral Training Award (DTA) that covered the tuition fees of this research.
- To my examiners Dr Gordon Hunter and Professor Yi-Ke Guo for their insightful discussion and helpful feedback.

Dedication

To my grand-father Pep Viñas, for being a source of inspiration.

And to my family, Jordi, Lola and Marina for their unconditional love and support.

‘Numbers have an important story to tell. They rely on you to give them a voice.’

Stephen Few

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"One absolutely cannot tell, by watching, the difference between a .300 hitter and a .275 hitter. The difference is one hit every two weeks."

Michael Lewis

"Moneyball" [2]

1.1 Motivation

Sports analysis has always been of great interest, and for many years coaches have been trying to design the best training programs and to implement the best strategies to win. For a long time, the standard approach was to watch previous matches and analyse them using domain knowledge. It is easy to realise that such 'historical data' is a small sample since the coach has to physically watch each match and is also subject to bias.

In fact, "Moneyball", a book by Michael Lewis published in 2004 [2], represented a breakthrough in showing the power of statistical data in sports. The book relates the real story of how the Oakland Athletics baseball team, which was at a clear disadvantage regarding budget, used statistical data to find players of underestimated value in the market, and thus built a team competitive with major teams. The term "Moneyball" has even become part of the standard vocabulary to describe this statistically-driven approach to make technical decisions in baseball.

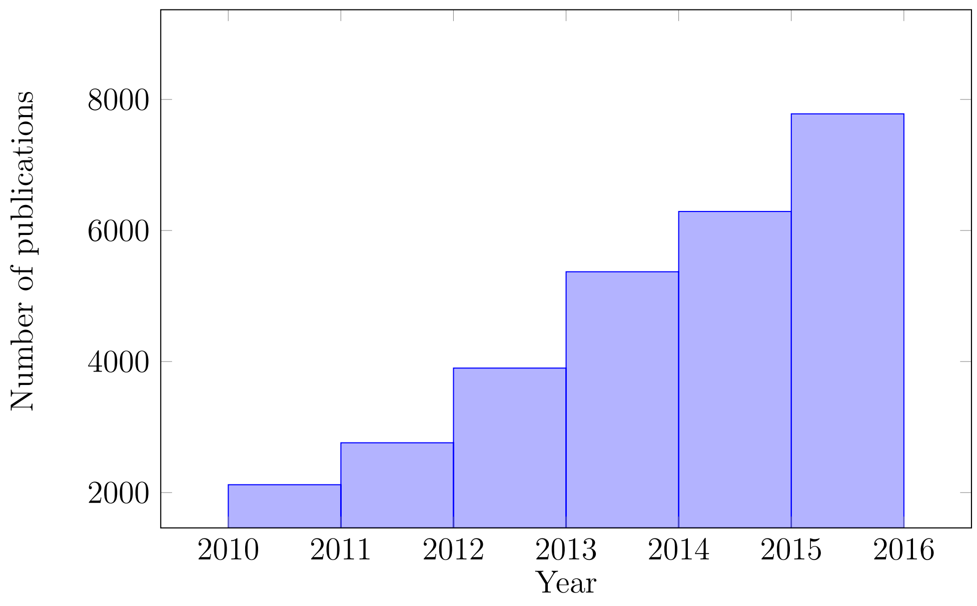


Figure 1.1: 'Sports Analytics' publications by year of publication, according to Google Scholar.

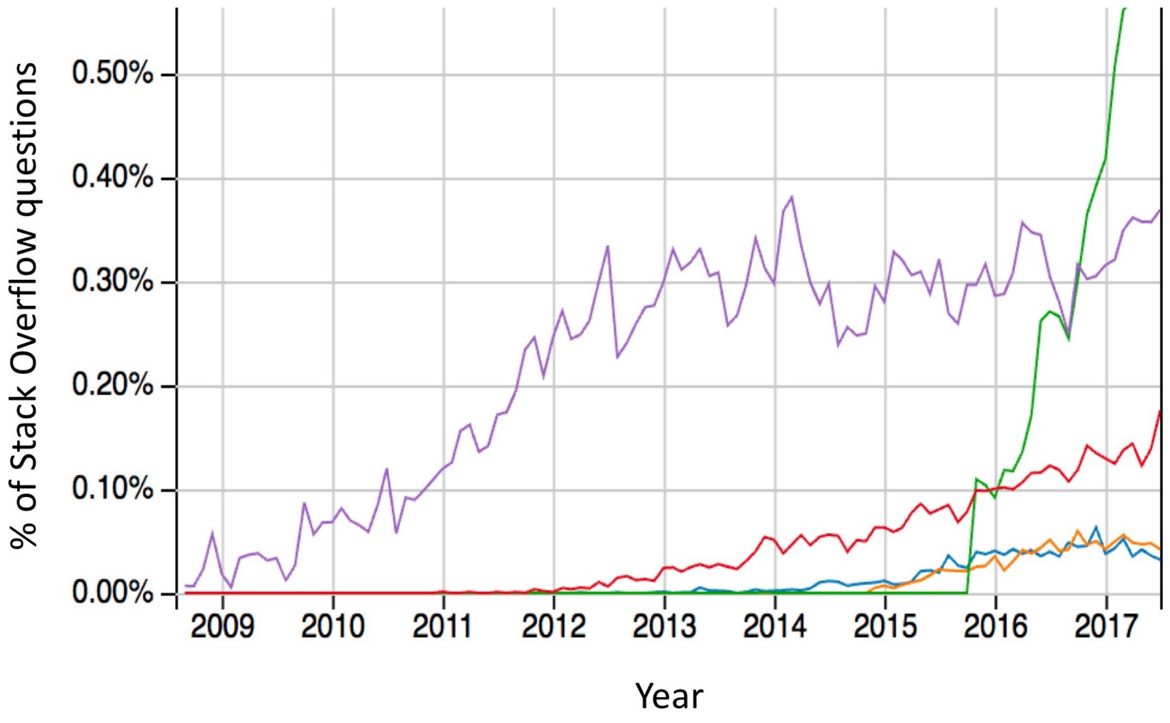
At that time, the data that was available to professional sports analysts was restricted to manually annotated events such as the batting average (in baseball), number of first serves in (in tennis), number of matches won by a particular team or goals scored by a given player (in soccer). Therefore, most of the modelling and statistical techniques employed to analyse and predict sports were built for the analysis of coarse-grained manually annotated data. It is undeniable that such analysis has led to successful game analysis, as in the example of "Moneyball". Nonetheless, it has a number of disadvantages:

- The level of precision that can be possibly obtained is limited. For instance, it is impossible to detect the exact ball trajectory by simply watching a match. This leads to coarse-grained data.
- Data collection can be subjective and inaccurate. The bias in the annotator and his/her mental state can interfere with objectivity in the data. For instance, assessing the speed of the ball (as fast or slow) can be subjective to prior knowledge on the player and annotators can also miss shots or overlook important events.
- Collecting such data is tedious and requires hiring experts. There is a restricted number of people with the necessary skills and willing to do the job. This leads to a corpus of data which is sparse and limited.

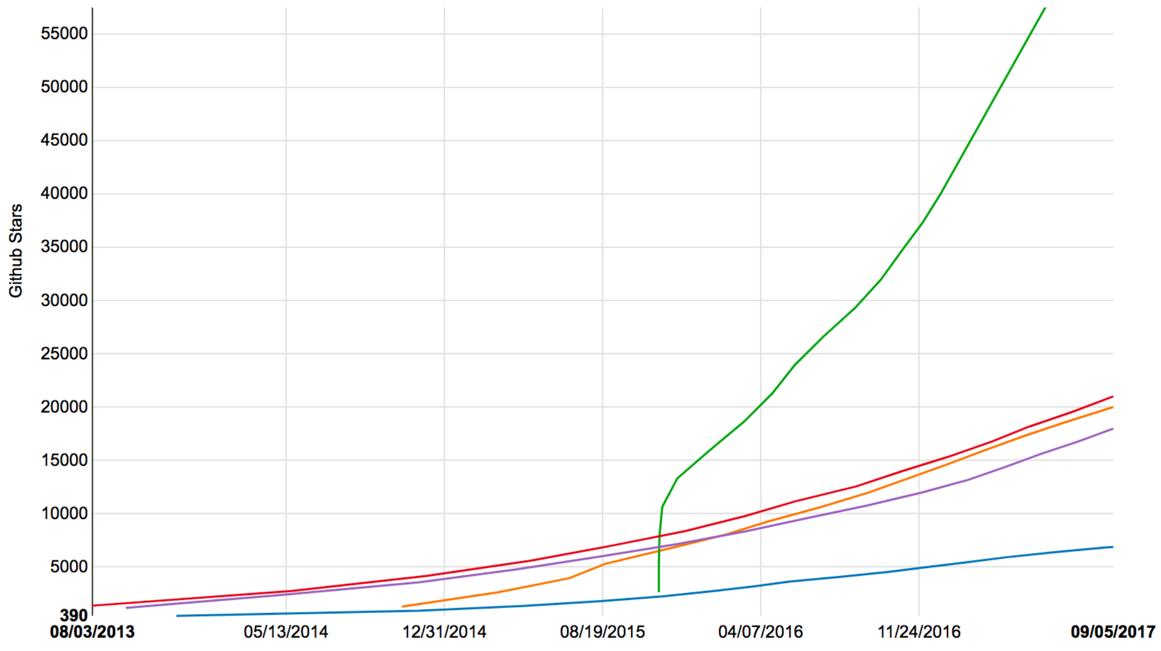
Since then, the interest in sports statistics has only grown, and this is reflected by the number of publications on the subject, shown in Figure 1.1. The rise in sports analysis can be traced back to a number of factors. Its growth is a result of the convergence of a number of technological trends:

- The proliferation of the Internet in the late 1990s started the data revolution that we are still experiencing today. It meant that statistical information about players and matches in a significant number of sports were publicly available and with it came the possibility of obtaining more substantial corpora of data per player and per team. The rise of online streaming has also enabled real-time analysis and the replay of a larger amount of previous matches leading to a better analysis of the opponents.
- The sheer increase in the processing capabilities of CPUs has been dramatic: while 1 GFLOP/s cost US$30 000 in 1997, today the figure is less than US$0.10[1]. This is before taking into account the emergence of general-purpose GPUs with thousands of processor cores capable of yielding upwards of 6 000 GFLOPs with a power consumption lower than 200W. This has enabled increases both in volumes of data that can be handled and the sophistication of the algorithms that can be applied.
- Regarding the algorithms themselves, the number of open source projects has experienced an exponential growth. Companies like Microsoft, Google and Facebook are now major contributors to open source, with more than 10 000 employees contributing to open source projects in GitHub (one of the major software development platforms). GitHub had 5.8M active users and 91.4M active open source repositories in the past year and contributions have increased more than 7 times in the last 3 years[2]. This has been especially notable in the areas of Computer Vision and Machine Learning. Figure 1.2 shows the trends of the interest in major Computer Vision and Machine Learning open source libraries (tensorflow, OpenCV, scikit-learn, theano and caffe) according to GitHub and Stack Overflow (largest online community of developers) metrics.

[1]Source: `https://en.wikipedia.org/wiki/FLOPS`
[2]Source: `https://octoverse.github.com/`



(a) Evolution of StackOverflow Searches[3].



(b) Evolution of GitHub Stars (attributed by other users to projects they like)[4].

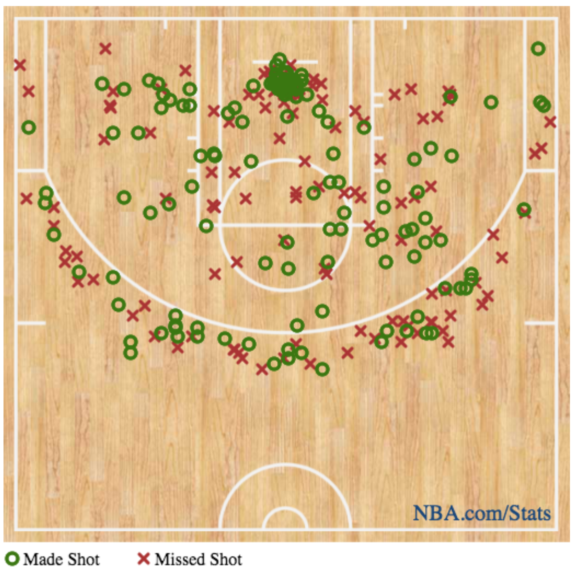
Figure 1.2: Evolution of StackOverflow searches and GitHub stars for popular Computer Vision and Machine Learning open source libraries: ● Tensorflow, ● Theano, ● scikit-learn,● Caffe, ● OpenCV.

[3]Graph rendered using `http://www.timqian.com/star-history/` on the 6th of September 2017.

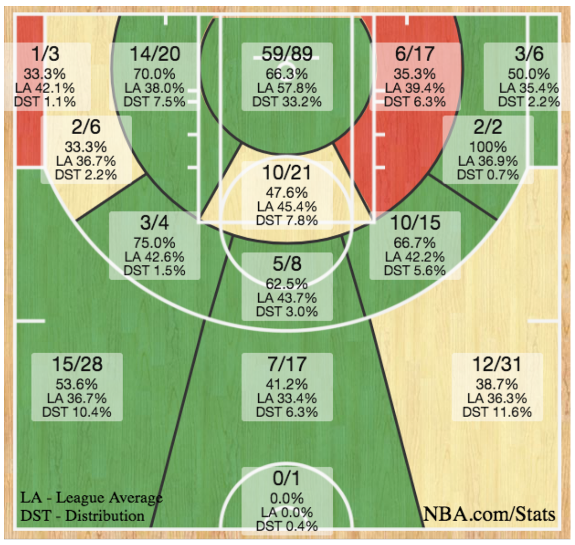
[4]Graph rendered using `https://insights.stackoverflow.com/trends` on the 6th of September 2017.

Advances in sports analytics have made it an indispensable component of elite sports coaching and sports broadcasting. As Leigh Steinberg puts it in Forbes: "Today, every major professional sports team either has an analytics department or an analytics expert on staff" [3]. It is also common for sports commentators to use historical information in their analysis to keep the spectators interested. Also, it has represented new business opportunities in many directions, for example providing consultancy services for the sports industry (e.g. ATASS, STATS) or offering insights to traders in sports betting (e.g. Stratagem, SmartOdds).

Partnerships between sports clubs or sports competitions and statistical experts as well as the integration of technology are revolutionising many sports. For example, in US Basketball, the NBA has a partnership with STATS and SportsVU to expand statistical data and provide advanced metrics to teams and fans. They provide data including speed and distance covered by a player, rebounding opportunities, defensive impact or field goal attempts. Figure 1.3 shows visualisations of the field goals attempted by Kevin Durant on the 2016-17 Playoffs Finals. One of the biggest impacts of analysis in basketball has been the shift in reducing the number of mid-range shot (two points) attempts in favour of three-point shots. This trend can be observed in Figure 1.4 and has received the name of "MoreyBall", as a combination of "Moneyball" and Daryl Morey, analyst and Houston Rockets general manager.



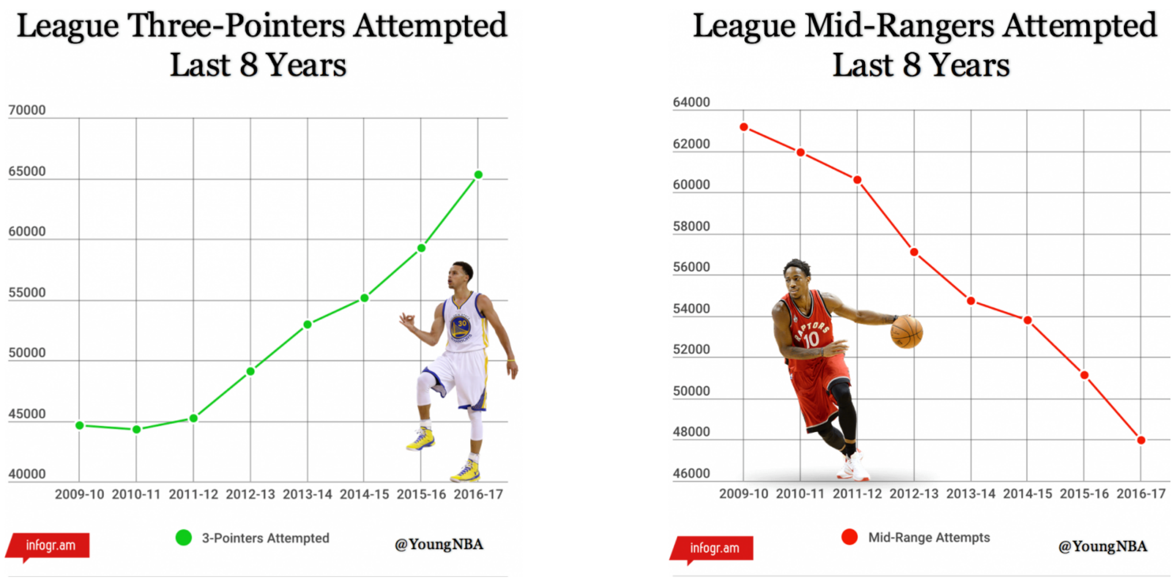
(a) FGA Missed and Made.



(b) FGA by Shot Zone.

Figure 1.3: Field goals attempted by Kevin Durant on the 2016-17 Playoffs final[5].

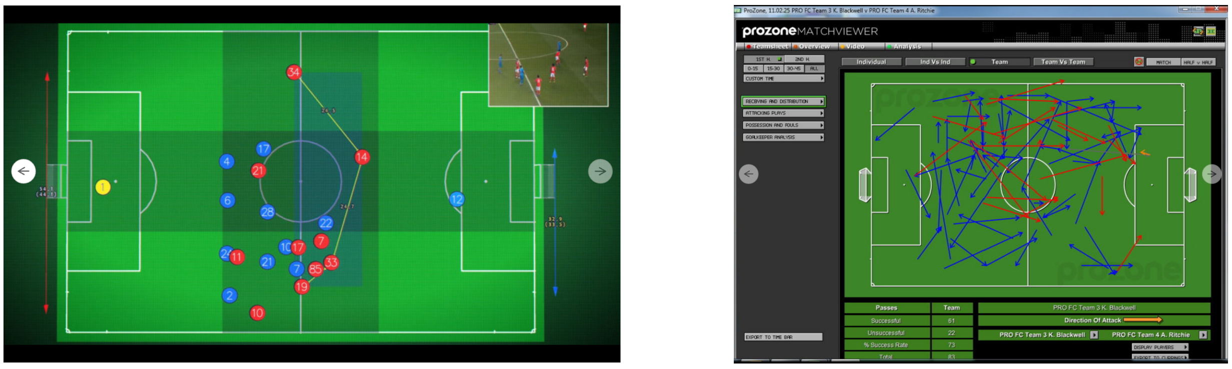
[5]Source: `http://stats.nba.com/`



(a) Number of 3-point Attempts. (b) Number of Mid-Range Attempts.

Figure 1.4: Evolution of shot attempts in the NBA league from 2009 to 2017[6].

Another interesting example is the S. L. Benfica soccer club. They use over ten sensors per player to gather data like player location (see Figure 1.5) or heart rate. They also have teams of data scientists working with Microsoft Azure Cloud Platform to analyse the extracted data and get insights into recovery times or stress levels of individual players to help in deciding who should play in the next match or obtain personalised training programs to optimise performance and reduce injuries. One of the greatest benefits from their analytic approach to coaching has come from the financial side, and over the last six years player transfers have resulted in income in excess of £270 million.



(a) Player Locations. (b) Player Trajectories.

Figure 1.5: Benfica S. L. player tracking technology from Prozone[7].

[6]Source: `https://bballbreakdown.com/2016/12/16/the-nba-3-point-revolution/`
[7]Source: `https://arstechnica.co.uk/science/2017/05/football-data-tech-best-players-in-the-world/`

These are only two examples, but technology and statistical analysis are becoming increasingly integrated into coaching, prediction, and broadcast for many other sports such as cricket (Figure 1.6) or golf (Figure 1.7).

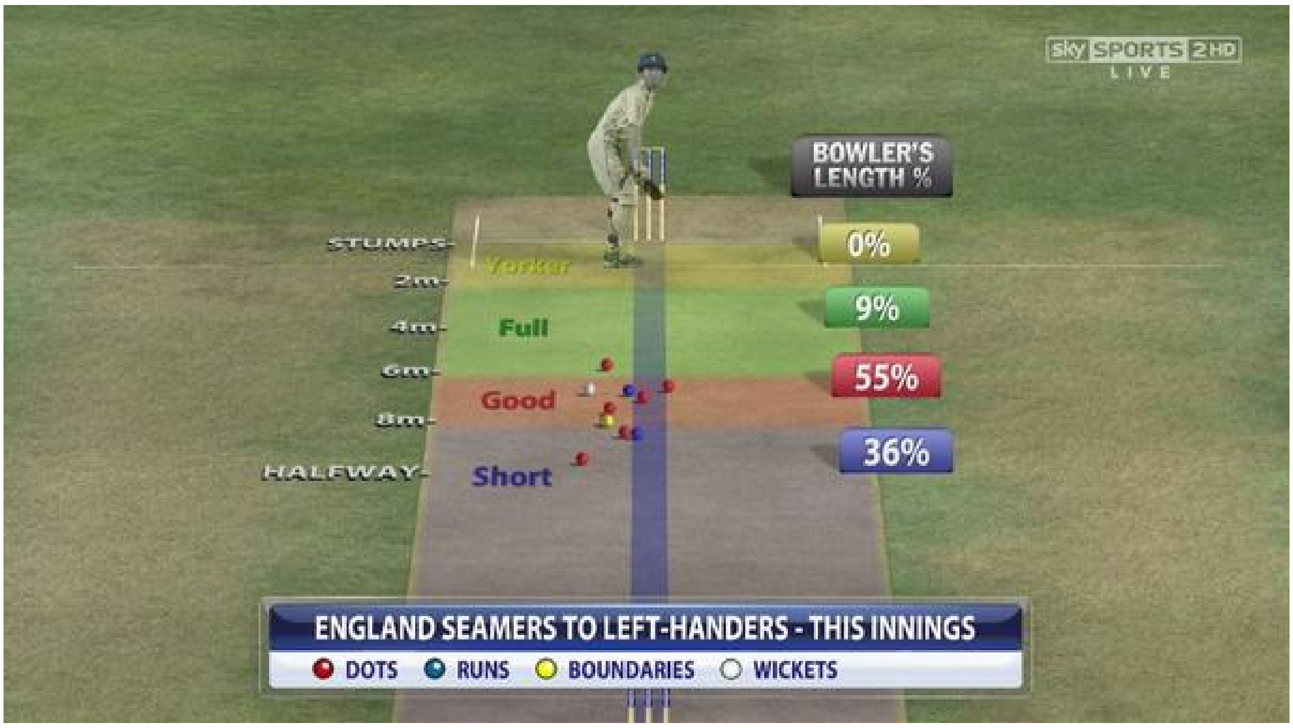
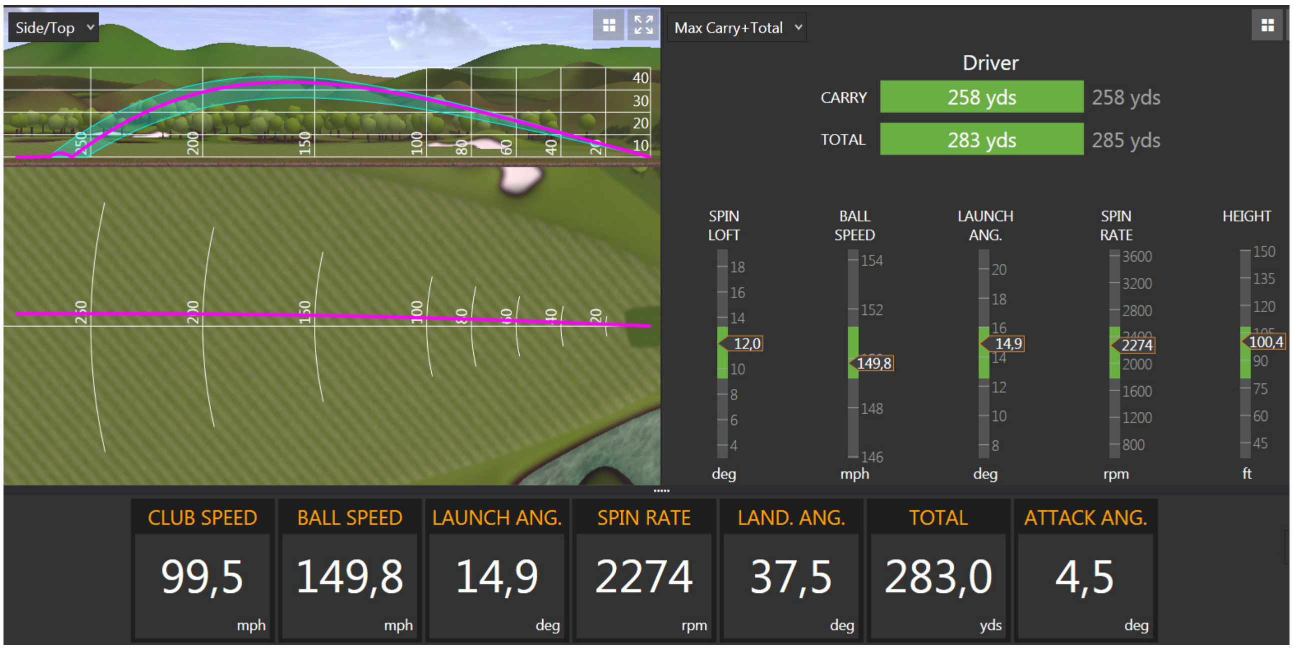


Figure 1.6: Cricket pitchmap by Hawk-Eye[8].



CARRY	258 yds	258 yds
TOTAL	283 yds	285 yds

CLUB SPEED	BALL SPEED	LAUNCH ANG.	SPIN RATE	LAND. ANG.	TOTAL	ATTACK ANG.
99,5 mph	149,8 mph	14,9 deg	2274 rpm	37,5 deg	283,0 yds	4,5 deg

Figure 1.7: Golf trackman technology for analysing shot trajectory[9].

[8]Source: Sky sports Twitter account.
[9]Source: `http://www.leechapmangolf.com/en/technology/trackman/`

In this work, we decided to focus on the sport of tennis although our research is also applicable to other sports. Tennis is a popular global sport – the Association of Tennis Professionals (ATP) estimates that last year 4.3 million fans attended ATP events and the TV audience reached more than 880 million spectators across the season. It is a highly exploited sport also from the analysis point of view in that a large number of statistical models have been built for the prediction of match outcomes. This has been possible due to the facility of access to a large corpus of historical data such as the percentage of first or second serves in, the percentage of points won in the first or se'cond serve and break points converted. Tennis is also particularly suited to build mathematical models since the hierarchical scoring system restricts the possible outcomes to a defined space (whereas the number of possibles scores in a football match can significantly vary), as we will see in Section 2.1. Also, in "singles" tennis, there are only two players on court who move in two distinct and defined spaces, reducing complexity when compared to team sports, in which one has to distinguish players from one another and model their interactions. Finally, tennis is a sport that has enthusiastically embraced technology, as can be seen in the following timeline:

- **1974** – First electronic line judge invented by Geoffrey Grant (tennis player) and Robert Nicks (electronics engineer). It was a system that processed data from pressure sensors positioned under the court carpet and was able to discriminate between the pressure exerted by the player or ball and determine its position, shown in Figure 1.8. It was used in the Men's World Championship in Dallas and the Ladies Virginia Slims in Los Angeles.

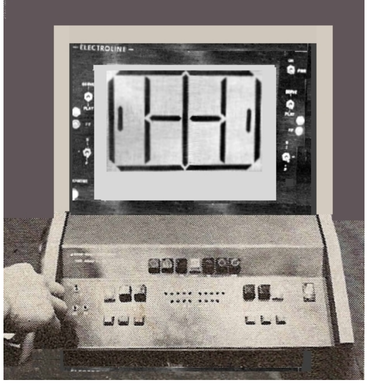


Figure 1.8: First electronic line judge[10].

[10]Source: `https://en.wikipedia.org/wiki/Electronic_line_judge`

- **1980** – “Cyclops” system introduced in Wimbledon. It was an electronic line judge system used for serves only, based on the use of infra-red beams that produced an audio signal if the beam signal was cut by the ball. The system was later introduced in the US Open and the Australian Open.

- **1989** – Measurement of the serve speed using a radar gun. This system was originally invented by Jon L. Barker Sr. and Ben Midlock for the military in World War II. The radar gun sends a number of radar pulses and is also able to receive them when reflected by the ball (or any object). By knowing the frequency of the sent and received pulses the system can determine speed based on the Doppler effect.

- **2002** – Hawk-Eye introduced in Wimbledon. Hawk-Eye is a complex system comprising up to ten high-speed cameras, able to track the ball with high accuracy, obtain its real-world position and display a reconstruction of any bounce, see Figure 1.9

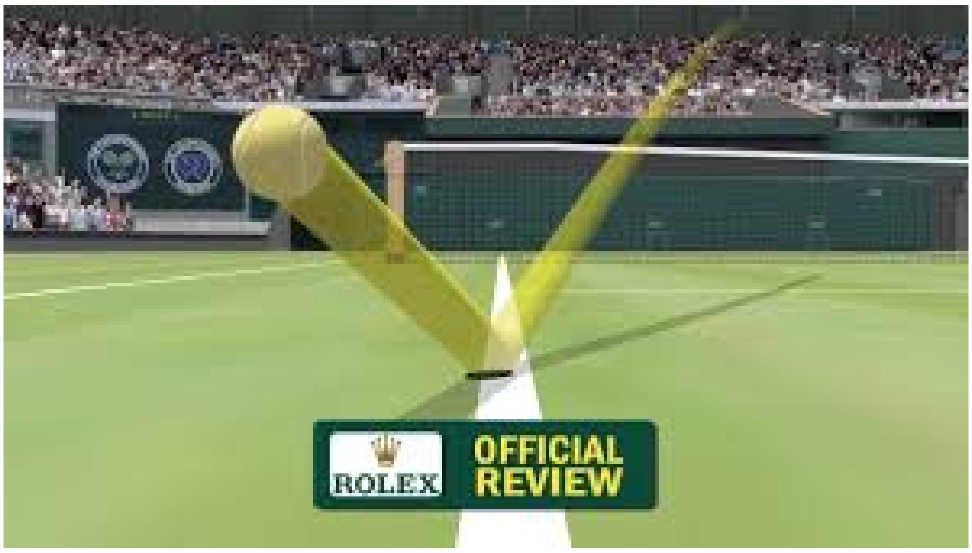
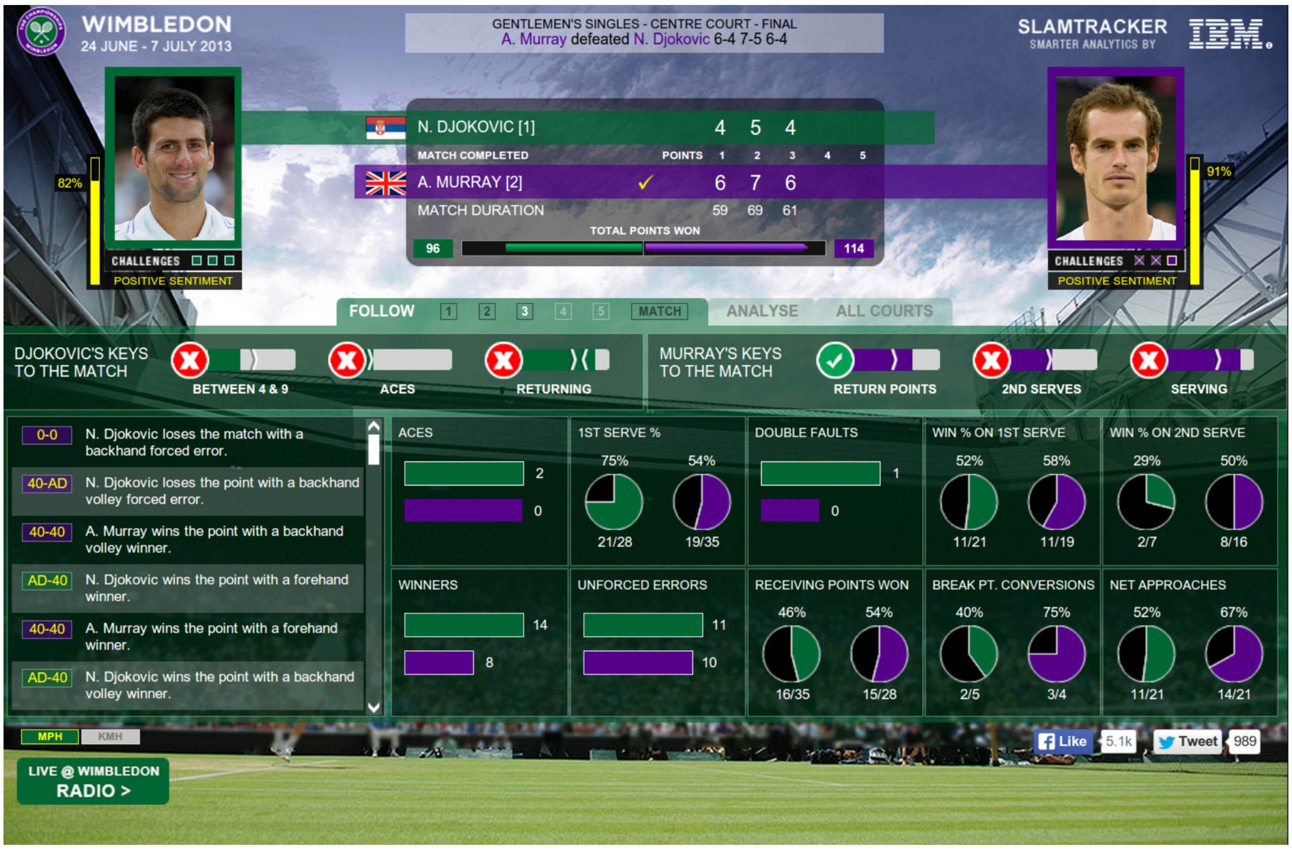


Figure 1.9: 3D reconstruction of tennis shot by Hawk-Eye[11].

- **2006** – Challenges introduced in Wimbledon. Players can request a Hawk-Eye-mediated line call if they do not agree with the decision of the umpire, with a maximum of 3 unsuccessful challenges allowed per set. In a tie-break the challenge counter is reset and players are allowed a maximum of 3 unsuccessful challenges again, which is reset every

[11]Source:`http://trolltennis.com/2016/11/19/saturday-school-10-technological-headways-of-tennis/`



GENTLEMEN'S SINGLES - CENTRE COURT - FINAL
A. Murray defeated N. Djokovic 6-4 7-5 6-4

	1	2	3	4	5
N. DJOKOVIC [1]	4	5	4		
A. MURRAY [2] ✓	6	7	6		
MATCH DURATION	59	69	61		

MATCH COMPLETED
TOTAL POINTS WON: 96 – 114

DJOKOVIC'S KEYS TO THE MATCH: BETWEEN 4 & 9 ✗, ACES ✗, RETURNING ✗
MURRAY'S KEYS TO THE MATCH: RETURN POINTS ✓, 2ND SERVES ✗, SERVING ✗

0-0	N. Djokovic loses the match with a backhand forced error.
40-AD	N. Djokovic loses the point with a backhand volley forced error.
40-40	A. Murray wins the point with a backhand volley winner.
AD-40	N. Djokovic wins the point with a forehand winner.
40-40	A. Murray wins the point with a forehand winner.
AD-40	N. Djokovic wins the point with a backhand volley winner.

	Djokovic	Murray
ACES	2	0
1ST SERVE %	75% 21/28	54% 19/35
DOUBLE FAULTS	1	0
WIN % ON 1ST SERVE	52% 11/21	58% 11/19
WIN % ON 2ND SERVE	29% 2/7	50% 8/16
WINNERS	14	8
UNFORCED ERRORS	11	10
RECEIVING POINTS WON	46% 16/35	54% 15/28
BREAK PT. CONVERSIONS	40% 2/5	75% 3/4
NET APPROACHES	52% 11/21	67% 14/21

Figure 1.10: Slamtracker analysis for the Wimbledon 2013 final[12].

12 games. Challenges were subsequently adopted by 80 other ATP/WTF tournaments up to date.

- **2012** – Introduction of IBM Slamtracker, an application presenting real-time scores and statistics (between 15 and 25 parameters for each point) to augment the experience of fans, as shown in Figure 1.10.

- **2013** – Hawk-Eye starts tracking the players.

Unfortunately, not all of the collected data is publicly available, especially spatio-temporal data. The state-of-the-art technology involved in officiating and collecting data in sports is the Hawk-Eye system [4], mentioned above. This system is extremely accurate but also highly sophisticated, comprising between 8 to 10 high-speed cameras (up to 1 000 fps) and an extremely powerful computer. The fact that this technology is equipment-intensive, costly and requires

[12]Source: `http://www.wimbledon.com/`

expertise to install it on a court restricts its availability to the high profile venues of major tournaments. In addition to this, the spatio-temporal data is of highly restricted access and Hawk-Eye does not collect high-level data (e.g. type of shot played).

Another notable system for spatio-temporal data collection in tennis is the Playsight Smart-Court [5], released around 2013. Similar to Hawk-Eye, it provides tracking of the players and ball but also includes some additional analysis of the game. It is a less sophisticated version of Hawk-Eye regarding equipment and, for instance, only uses five cameras. Different from Hawk-Eye, it is fully automated and, once installed in a court, it does not require an expert to be present. However, the system is not mobile and only accessible to players with access to courts in which a Smart-Court system has been installed, making it an elitist system only available to a small proportion of tennis players. This is illustrated by the fact that currently, there are only 7 tennis Smart-Courts installed in the UK, 6 of which are in the Greater London area.

Even though the availability of tennis spatio-temporal data is limited and there is a lack of historical data, the potential for using spatio-temporal data to build intra-point tennis models has been recently exposed [6]. The authors obtained promising results, but their rally models do not incorporate the serve, lack historical reach because they have limited access to Hawk-Eye mediated data and do not use high-level information like 'type of shot', since this is not currently extracted by Hawk-Eye. In 2009, G. Hunter and K. Zienowicz [7], succesfully developed a Markov model to simulate tennis rallies based on the type of stroke. The data used in their work were manually annotated types of stroke from videos, limiting the amount of data that could be used in their models. Another contribution has come from D. Spanias [8] who presented low-level intra-point Markov chain models of rallies and serves based on spatio-temporal information; however, the authors lacked sufficiently detailed data to apply this approach.

1.2 Objectives

To further develop fine-grained models in tennis, limitations in the availability of spatio-temporal tennis data have to be overcome, and a fully automated and accessible system for data collection is required. We believe that advances in Computer Vision and Machine Learning have made it possible to obtain accurate spatio-temporal tennis data from commodity hardware in an unobtrusive and affordable manner in real-time. However, for such a system to be useful in the analysis and understanding of intra-point dynamics in tennis, it must be built within a framework that connects raw visual data to tennis analysis. Such a framework can bring new insights related to players' strategies, their strengths, and weaknesses or the prediction of future events, amongst other opportunities.

In this thesis, we investigate how this goal can be achieved within the context of tennis, which entails the achievement of the following specific objectives:

- Design a framework for the application of Computer Vision and Machine Learning to in-play tennis analysis using commodity hardware.
- Build a Vision system able to find the player and ball position in a tennis match or training session with the constraints of being accurate, easy to install, unobtrusive, real-time and affordable in terms of financial cost.
- Analyse spatio-temporal data to automatically obtain high-level statistics and offer insightful visualisations.
- Classify fine-grained tennis actions from videos.
- Place the achieved work within the context of existing research in the area of spatio-temporal data collection in tennis. This requires overcoming the challenge of comparing different approaches to visual data collection in sports as no benchmarks exist for most sports research.

1.3 Contributions

The main contribution of our work is the design and implementation of the framework shown in Figure 1.11 and the algorithmic innovation developed within this context. The framework was designed for the application of Computer Vision and Machine Learning techniques to tennis with the objective of enabling the fine-grained analysis of tennis, the prediction of future events and the revealing of insights from an affordable, real-time (over 60 fps) and unobtrusive system. In this thesis, we investigate how these goals can be achieved within the context of tennis, but our research is also applicable to other sports. Our approach presents a three-layered framework with a Vision Layer, a Classification Layer, and a Modelling Layer. They compose a hierarchical structure in which each layer uses the information from the previous ones to reach a higher level of abstraction. The basis of our model is the Vision Layer which retrieves the relevant spatio-temporal information from tennis video feeds, mainly the ball trajectory and player position. The output of the Vision Layer is fed to the Classification layer to obtain high-level information including the number of shots in a rally, movement patterns and shot classification (backhand return, smash, net approach etc.) Finally, the Modelling Layer incorporates the data obtained in the previous two layers into a tennis model to understand players' patterns and strategies and predict future events. In this thesis, the Modelling Layer is presented as a review of work done by other researchers and without analysing our own data. With this chapter we wanted to show the potential of our framework and the data collected in it. We provide a critical analysis of different state-of-the art techniques, and show how our framework addresses some of its limitations.

Our objective is to show how such a framework can lead to the same quality of analysis, or even higher level than that provided by highly sophisticated systems such as Hawk-Eye, but using commodity hardware. For this, we also present algorithmic innovation within the context of the individual layers by taking advantage of the advances in Computer Vision and Machine Learning techniques.

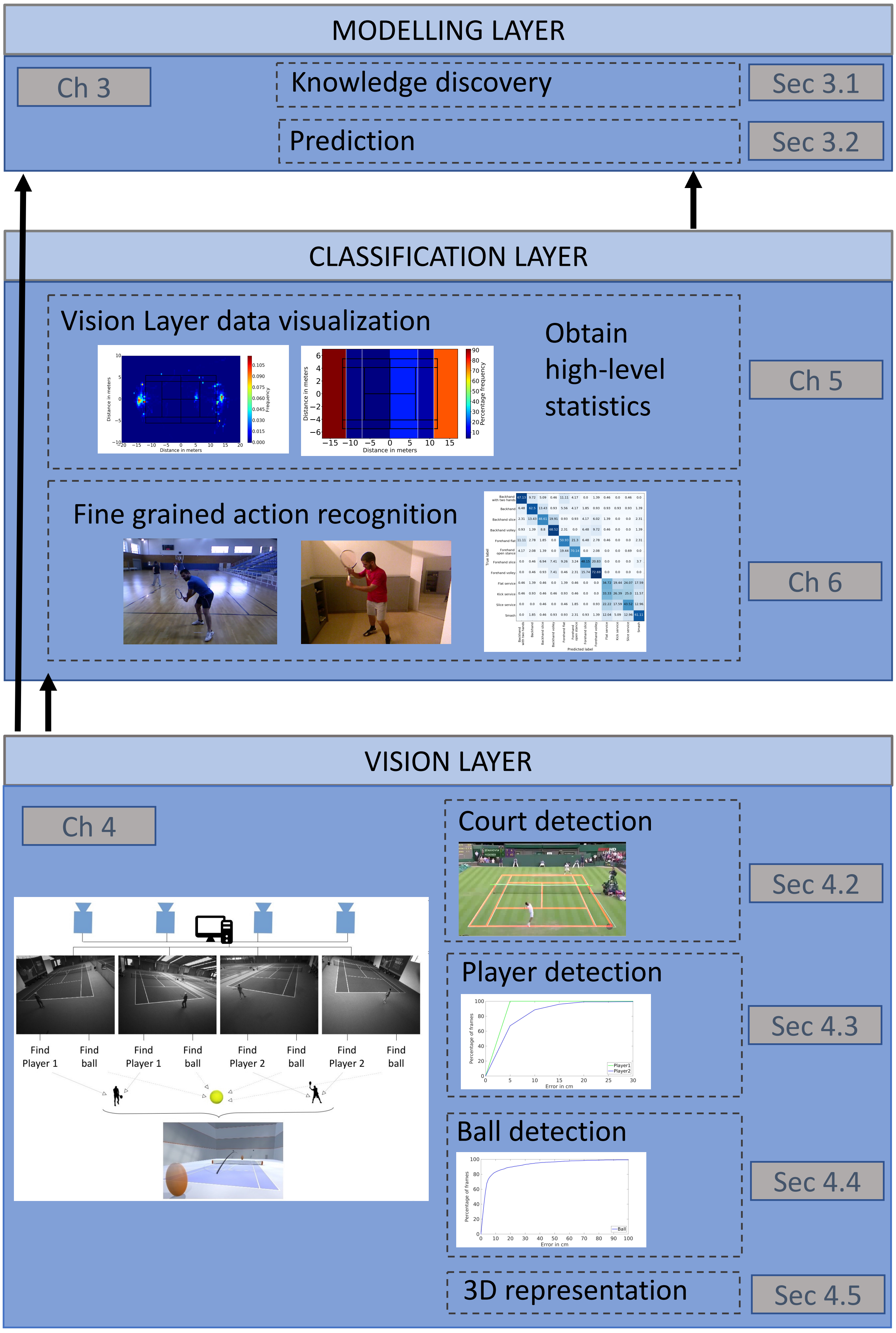


Figure 1.11: Framework overview.

1.3.1 Vision Layer

We present a low-cost spatio-temporal data collection system for tennis using four cameras and commodity hardware. It has the following capabilities:

- Court detection: We develop an algorithm for the automatic detection of the court lines. The algorithm has three main steps: select court pixels using domain knowledge, fit lines to court pixels with the Hough Transform algorithm and find correspondences between these and specific court lines (e.g. the baseline). This last step is done by evaluating the court fitting for different possible configurations and selecting the best one; we also compute a level of confidence for each court detection.

- Player tracking: We perform player detection in two steps: (1) reconstruction of a background image by selecting the pixels that are most stable across a number of frames and (2) selection of the largest object (i.e. the player) of moving pixels within an automatically determined region of interest. The player position is represented in two ways: a bounding box enclosing the whole player and the centre of mass of the area within the contour of the player. In all the tested frames, the predicted centre of mass was inside of the ground truth bounding box of the player and its distance from the centre of the ground truth bounding box never exceeded 40 pixels (representing an error of 20% of the bounding box in 80% of the frames).

- Ball tracking: For this, we select moving foreground pixels by frame differencing and cluster them into ball candidates. Then the best candidate is selected using a random forest approach based on shape (defined in terms of Hu moments [9]) and ball trajectory. The ball position was detected with an error lower than 2 pixels for more than 75% of the frames, and never exceeded 6 pixels; this was a very good result considering that the ball has a size of up to just 10 pixels (depending on camera proximity).

- Infer the 3D player and ball coordinates relative to the court: Knowing the court lines it is possible to estimate the position of each of the four cameras and obtain the 3D coordinates of the objects of interest through triangulation. Both the ball and player predicted locations had an error lower than 10cm for 80% of the frames. We consider these results to be of good quality, especially considering the size of the player and ball, i.e. approximately 50cm $\times$ 170cm $\times$ 20cm and 7cm of diameter respectively.

- Visualization: We replicate the results in a 3D virtual environment using Unity[13] for visualisation purposes.

1.3.2 Classification Layer

The work relating to this layer illustrates the kinds of higher-level analysis that can be enabled by the data emerging from the Vision Layer. This includes the extraction and visualisation of high-level tactical information as well as a component for fine-grained action recognition in tennis.

High-Level Tactical Information

From the data extracted in the Vision Layer, we can obtain high-level information such as the timing of the strokes of each of the players, number of shots in a rally, maximum ball speed or area covered by each player. We can also show visualisations that provide insights by displaying frequency maps of player movement, ball trajectory, and ball landing position.

Action Recognition

We develop and train a novel deep neural network for domain-specific action recognition in tennis. We performed our research on a challenging and publicly-available dataset containing RGB low-definition videos of fine-grained tennis actions, called THETIS [10]. The actions are:

- backhand (two handed)
- backhand
- backhand (slice)
- backhand (volley)

[13]`https://unity3d.com/`

- forehand (flat)
- forehand (open stance)
- forehand (slice)
- forehand (volley)
- service (flat)
- service (kick)
- service (slice)
- smash

This is the first application of neural networks to fine-grained tennis action recognition, and one of the few applications to fine-grained actions in general. Most of the available research in the field of sports action recognition is performed on datasets that are not accessible to the public, and with this work we also aim at encouraging other researchers to use publicly available datasets so their work can be compared.

We obtain very promising results of 47.22% accuracy of one class against all others in classifying the previously listed 12 fine-grained actions and 88.16% for classifying backhands, forehands and serves. We also find some interesting results when applying the system to amateur, compared with professional, players and show that our neural network can also be used in applications beyond tennis.

1.3.3 Modelling Layer

To address the Modelling Layer, we survey different methods that incorporate visual information for the analysis of tennis matches. We did not find a published account of the different approaches that use spatio-temporal data to analyse tennis and considered that such a piece of work was essential not only to bring together previous research but also to provide a sense of the current developments in the field.

1.4 Declaration of Originality and Publications

I declare that I am the sole author of the work written in this thesis. Results from collaborations are clearly indicated and ideas and results from other researchers are fully cited.

The following peer-reviewed publications emerged as part of the work in this PhD:

- S. Vinyes Mora, G. Barnett, C. da Costa-Luis, J. Garcia Maegli, M. Ingram, J. Neale, and W. J. Knottenbelt, “A Low-Cost Spatiotemporal Data Collection System for Tennis,” in *Proceedings of the 5th Mathematics in Sport Conference*, (Loughborough, UK), June 2015.
  (Chapter 4)

- S. Vinyes Mora and W. J. Knottenbelt, “Spatio-Temporal Analysis of Tennis Matches,” in *Proceedings of the 3rd ACM KKD Workshop on Large Scale Sports Analytics*, (San Francisco, California, USA), August 2016.
  (Chapter 5)

- S. Vinyes Mora and W. J. Knottenbelt, “Deep Learning for Domain-Specific Action Recognition in Tennis,” in *Proceedings of the 3rd IEEE International CVPR Workshop on Computer Vision in Sports*, (Honolulu, Hawaii), July 2017.
  (Chapter 6)

1.5 Thesis Outline

This thesis is structured in the chronological order of our work rather than the order of layers in Figure 1.11 as we believe it provides a better understanding of our research:

Chapter 2 describes the sport of tennis, providing information about its rules and scoring system and reviews the evolution of modelling of the sport. Then an overview of the fields of Computer Vision and Machine Learning is provided in conjunction with a description of the most relevant techniques in each area.

Chapter 3 surveys the research that has been done in the analysis and prediction of tennis using spatio-temporal data. Advantages and current limitations are reviewed to provide not only the first survey in this area of research but also to motivate the work presented in the rest of this thesis.

Chapter 4 presents our novel system for the real-time extraction of spatio-temporal data in tennis using unobtrusive, portable and affordable technology. The proposed system detects the 2D location of the tennis ball and players in videos fed by 4 cameras. Then, their 3D location is obtained via triangulation. This chapter also presents a quantitative evaluation of the detection accuracy, which is not common in the area of object detection in sports.

Chapter 5 analyses the data extracted from the Vision Layer system, presented in the previous chapter, to automatically obtain higher level tactical information and show visualisations of the data.

Chapter 6 presents our application of deep learning to action recognition in tennis. We present a robust algorithm able to classify fine-grained tennis actions directly from RGB videos and with the ability to generalise to other domains.

Chapter 7 concludes the thesis by reviewing the most relevant points presented and discusses limitations and future work.

Chapter 2

Preparation for the Big Game: Background

This chapter provides a brief explanation of the game of tennis and reviews the evolution of techniques for its analysis, from classical to contemporary approaches. Theoretical background in the fields of Computer Vision and Machine Learning is provided, in conjunction with selected algorithms in these areas that are pertinent to our work.

2.1 The Tennis Game Explained

Tennis is a racket sport that entails competition between two opponents, who can be individual competitors or teams of two players. It is played on a court of standard dimensions; Figure 2.1 shows the court dimensions and nomenclature associated with it. The detailed rules of tennis are publicly available from the International Tennis Federation (ITF) website[1]. Here we provide an overview of the fundamentals of its scoring system. It has a hierarchical structure in which winning points leads to winning games, winning games leads to winning sets, and finally winning sets leads to winning the match. Playing a point starts with a service and for a service to be valid, the serving player must serve from behind the baseline, and the ball must go over the

[1]Source: `http://www.itftennis.com/officiating/rulebooks/rules-of-tennis.aspx`

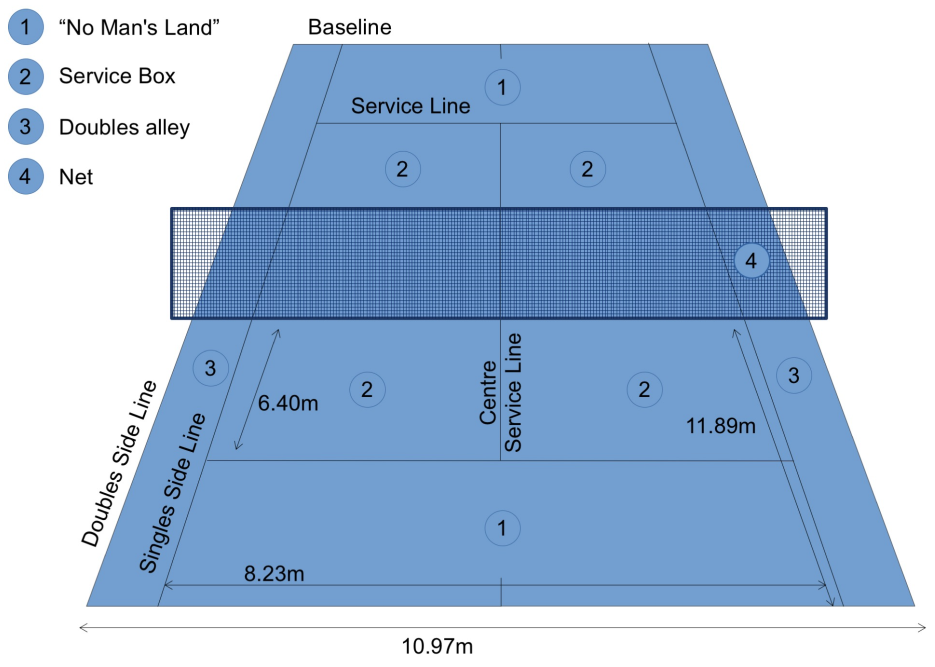


Figure 2.1: Diagram of a tennis court with its associated nomenclature and dimensions.

net cleanly and land within the diagonally-opposite service box. When the ball correctly lands within the diagonally-opposite service box having touched the net, it is called a "Net" or "Let" and the serve is repeated without counting it as a fault. Otherwise, it counts as a fault. If the latter occurs, the player who was serving can play a second serve. If the second serve is also a fault, the opposing player wins the point. If the service is valid, a rally follows in which the players return the ball back and forth. Each player must return the ball inside of the court and before it bounces twice. Failing to do so results in ending the point, which is won by the last player to have returned the ball within the regulations. The player serving changes at the end of each game and players change ends after every odd game (1st, 3rd etc.) The server must stand on the right side for his serve at the beginning of the game and alternate sides for every point. In a standard game, each player earns points as follows: no points – "love", first point – 15, second point – 30, third point – 40, fifth point – "Game". If both players get three points (score of 40 – 40), the score is "Deuce", and new rules apply: the player winning the next point has "Advantage", and if he/she wins another point he/she wins the game. Otherwise the score returns to "Deuce". Therefore to win a game from "Deuce" a player must score two consecutive points. The first player to win at least 6 games by a margin of at least two games with respect to the opponent's score wins a set. If both players reach 6 games, different rules can be used. The two most common are: 1) "Tie-break set", where a tie-break game is played in which points are scored 0, 1, 2 etc. and the first player to win 7 points, with a margin of at least 2 points wins the set, and 2) "Advantage set", where the set continues until a margin of two games is reached. In general, women play matches to best of 3 sets matches at all levels of competition, while men play best of 5 sets in certain major tournaments, and best of 3 sets otherwise.

2.2 Markov Chains

Markov chains, named after the mathematician Andrey Andreyevich Markov, are stochastic processes satisfying the Markov property of "no-memory" in which the evolution of a process is conditional only on the current state at each stage. Changes of state are called transitions, the probabilities associated with the transitions called transition probabilities and a transition matrix is usually used to specify these variables. Figure 2.2 is a diagram representing a Markov chain with three states: s_1, s_2 and s_3, and its corresponding nine transition probabilities: p_{11}, p_{12}, ..., p_{33}, in which p_{xy} is the probability associated with the transition from state x to state y. Table 2.3 presents the transition matrix associated with this Markov chain. The transition probabilities follow the property $\sum_j p_{ij} = 1$ for all states i, since for each state i, the probabilities of moving to state j (where j could equal i) must sum to 1.

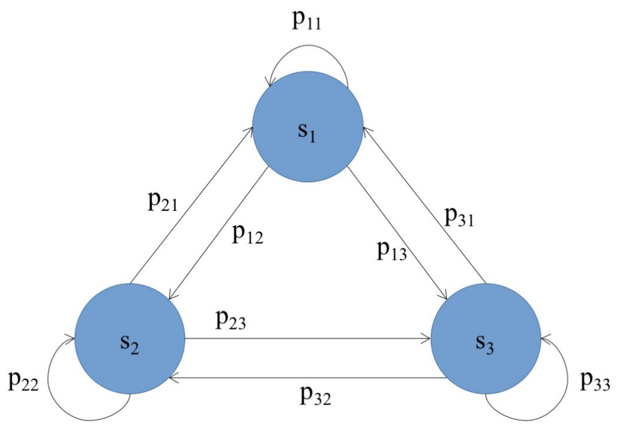


Figure 2.2: Markov chain example.

		To state		
		s_1	s_2	s_3
From state	s_1	p_{11}	p_{12}	p_{13}
	s_2	p_{21}	p_{22}	p_{23}
	s_3	p_{31}	p_{32}	p_{33}

Figure 2.3: Transition matrix.

2.3 Classical Tennis Models

Most tennis prediction models belong to one of these classes: regression models, paired comparison, and point-based models. An overview and in-depth comparison between them has recently been published by Kovalchik [14]. The first two directly predict the winner of a match from statistical data, while the latter calculates the probability of winning a match from the probability of winning a point on serve, better reflecting the dynamics of a tennis game.

In 1960, John G. Kemeny and James Laurie Snell presented one of the earliest applications of Markov Chains to model tennis using the fixed probability of winning a point as its building block [15]. This model represents the basis for most tennis point-based models. Using the Barnett and Clarke formulation [16], the Markov Chain states are the game scores and the probability $P(a,b)$ of player A winning the game when the score is a,b[2] is calculated based on the probability p of player A winning a point:

$$P(a,b) = pP(a+1,b) + (1-p)P(a,b+1)^3 \tag{2.1}$$

Similarly, the probability of winning a set is calculated from the probability of winning a game and the probability of winning a match from the probability of winning a set. Details of the calculations can be found in [17].

Another commonly used formulation to predict the probability of winning a tennis match from the probability of winning a point is that of O'Malley [18]:

$$M_3(p,q) = S(p,q)^2[1 + 2(1 - S(q,p))] \tag{2.2}$$

$$M_5(p,q) = S(p,q)^3[1 + 3(1 - S(p,q)) + 6(1 - S(p,q))^2] \tag{2.3}$$

[2]Using scoring 0, 1, 2, etc. rather than 0, 15, 30 etc.

[3]This is the original formula in the paper , but Dr Gordon Hunter notes the formula should be P(a, b) = pP(a - 1, b) + (1- p)P(a, b - 1)

$M_3(p,q)$ and $M_5(p,q)$ are the conditional probabilities of winning a best-of-three-sets and best-of-five-sets matches respectively, based on the probability of winning a point on serve p and on return q. $S(p,q)$ is the probability of winning a tiebreaker set as a function of p and q, which is calculated from the probability of winning a game and tie-break. Readers interested in the detailed formulation are directed to [18].

All of the above models assume that tennis points are independent and identically distributed. In 2001, Klaasen and Magnus made a significant contribution to the field by showing that the assumption, even though not strictly true, is reasonable for forecasting applications [19].

The application of point-based models described above to forecasting requires knowing the probability of winning a point. The estimation of this value is not straightforward and requires taking into account the ability of both players in the match. For this reason, over the years, effort has been put into refining the calculation of the probability of winning a point:

- In 2005, Barnett and Clarke estimated the probability of winning a point on serve and return by taking into account the abilities of both players in the match [20]. Their model also considers the effects of court surface by adapting to specific tournaments. They estimate the percentage f_{ij} of points won by player i when serving against player j as follows:

  $$f_{ij} = f_t + (f_i - f_{av}) - (g_j - g_{av}) \tag{2.4}$$

  Here f_t represents the average fraction of points won on a serve in the current tournament. The second term $(f_i - f_{av})$ reflects the ability of player i serving by calculating how much the fraction f_i of points won on serve by player i exceeds the average fraction f_{av} of points won on serve across professional players. The third term $(g_j - g_{av})$ represents the ability of player j returning as the excess of the fraction g_j of points won on return by player j, compared to the average fraction g_{av} of points won on return across professional players. Similarly, the combined fraction g_{ji} of points won on return by player j when player i is serving is estimated as:

  $$g_{ji} = g_t + (g_j - g_{av}) - (f_i - f_{av}) \tag{2.5}$$

Assuming that the second serve is successful, the fraction of points won on serve and return are estimated as follows:

$$f_i = a_i b_i + (1 - a_i)c_i \tag{2.6}$$

$$g_i = a_{av} d_i + (1 - a_{av})e_i \tag{2.7}$$

where a_i is the probability of player i getting the first serve in, a_{av} is the average probability of successful first serves across professional players, b_i is the proportion of points won from successful first serves by player i, c_i is the proportion of points won from successful second serves by player i, d_i is the proportion of points won by player i when receiving a first serve and e_i is the proportion of second serve points won by player i when receiving.

This new definition of the probability of winning a tennis point is more accurate than previous formulations, since it takes into account the skill of both players in the match. However, the main drawback is that better players play amongst each other more often than with weaker players and vice-versa. Therefore, in Equations (2.4) and (2.5) player j might be stronger than player i, but f_j might be lower than f_i because player j generally faces stronger opponents.

- In 2012, Knottenbelt, Spanias and Madurska refined further the formulation of the probability of winning a point and addressed the previously mentioned drawback [21]. They introduced the concept of common opponents, who are players that have faced both players of the current match on a previous occasion. In the common opponent analysis model, the data selected for predicting a match between two players A and B is of the matches in which they played common opponents, since players A and B may have only faced each other rarely, or not at all. The metric Δ_i^{AB} is used to represent the advantage of player A over player B with respect to a common opponent C_i:

$$\Delta_i^{AB} = (spw(A, C_i) - (1 - rpw(A, C_i))) - (spw(B, C_i) - (1 - rpw(B, C_i))) \tag{2.8}$$

where $spw(a, b)$ and $rpw(a, b)$ are the proportions of service and return points won by a against b respectively. Players A and B are the players of the match to be predicted, and C_i is their i^{th} common opponent.

This value can be used to influence the probability of player A and of player B winning a game, set and match when they play each other in point-based models. For example, applying it to O'Malley's formula results in the following probability $Pr(A\ beats\ B\ via\ C_i)$ of player A winning a best-of-three-sets match against player B, when using the data involving the common opponent i:

$$Pr(A\ beats\ B\ via\ C_i) \approx \frac{M_3(0.6 + \triangle_i^{AB}, (1 - 0.6)) + M_3(0.6, (1 - (0.6 - \triangle_i^{AB})))}{2} \tag{2.9}$$

Equation (2.9) is calculated as the average of two match probabilities following O'Malley's formulation in Equation (2.2). The value 0.6 is chosen to represent the average probability of a professional player winning a point on serve against another professional player. As shown by O'Malley [18], the exact value of this parameter has negligible effect on the match winning probability, since the primary determiner of the match winning probability is the difference in serve winning probabilities.

In the first match probability, the probability of player A winning a point on serve is the average probability of winning a point on serve plus $\triangle_i^{AB}$, which is the advantage of player A over player B with respect to the common opponent i. In the second match probability, the probability of player A winning a point on return is 1 minus the probability of B winning a point on serve. This last term corresponds to the average probability of winning a point in serve minus $\triangle_i^{AB}$, which is the advantage of player A over player B with respect to the common opponent i. Then, this value can be averaged over N common opponents to find the probability of A winning a match against B as follows:

$$P_{avg}^{AB} = \frac{\sum_{i=1}^{N} Pr(A\ beats\ B\ via\ C_i)}{N} \tag{2.10}$$

This model has been shown to achieve an average match win prediction accuracy of 63.3%[4] over the prediction of matches in the following tournaments: ATP 2014 – 64.9%, WTA 2014 – 61.2%, ATP 2015 – 66.2% and WTA 2015 – 61.0%. However, a major disadvantage of this approach is that it uses a limited amount of the available statistical data, since it ignores matches with players outside of the set of common opponents.

- In 2013, Goldsack and Knottenbelt developed the impact model to address the limitations of the common opponent model [22]. Their approach for reflecting players' skill is by looking at their impact on the key dimensions of their opponents' performance when facing them in a match. This metric is calculated for impact on service and return separately. The impact Δp_s^j of player j serving is:

$$\Delta p_s^j = \frac{1}{K_j} \Sigma_{k=1}^{K_j} (p_s^{ik} - p_s^i) \tag{2.11}$$

where $k = 1, ..., K_j$ are the matches played by j, p_s^{ik} is the probability of winning a point on serve by player i in match k and their average performance is p^i.

Historical data can be combined into the player's impact to estimate the proportion of service points won by player A against player B:

$$spw(A, B) = spw_A + \Delta p_s^B \tag{2.12}$$

where Δp_s^b is the serve impact of player b. This can also be done for returns points won (rpw). Finally, match prediction is achieved using the O'Malley formula from Equation (2.2) and the estimated values for $spw(a, b)$ and $rpw(a, b)$, the proportion of service and returning points won by a against b respectively. This model outperformed the common-opponent model, obtaining an average match win prediction accuracy of 66.0% over the prediction of matches in the following tournaments: ATP 2014 – 67.5%, WTA 2014 – 65.3%, ATP 2015 – 68.3% and WTA 2015 – 62.9%.

[4]Although [14] mentions that in 2014 ATP, "higher ranked players won 68% of matches".

As we have seen above, for models that reflect the dynamics within a tennis game, the standard is to use Markov Chains based on the probability of winning a point and, for more than 50 years, research has focused on improving accuracy in predicting the probability of winning a point. As discussed next, this trend has been broken in recent years.

2.4 Evolution of Tennis Models

In recent years, the field of tennis modelling has been evolving towards more sophisticated and finer-grained models, interested in understanding what happens at the point level. This type of model is also particularly suitable for in-play analysis, with the ability of updating winning probabilities as the match progresses. In this direction, Spanias presented an extension of classical tennis models with a Markov Model for a tennis service and rally using spatio-temporal information [8]. Figure 2.4 shows a diagram for modelling the service using location information ('Positioning choice') and higher-level tactical information ('Serve speed' and 'Type of service'). Similarly, the rally model in Figure 2.5 also uses spatio-temporal information to determine who will win the rally, and therefore who will win the point. Unfortunately, the authors were not able to test their approach because the data necessary for these models is lacking or of highly restricted access. It is interesting to note that the data used in Figure 2.5 could also be obtained from acoustic data, as has been shown in [23]. However, in this thesis we focus on spatio-temporal information obtained from visual data.

Researchers from the Queensland University of Technology, Disney Research Pittsburgh and the Australian Institute of Sport have had access to a selected amount of Hawk-Eye data and have shown how it can be used to develop finer-grained models [6; 24–26]. Their research has provided insightful analysis of a tennis match and will be reviewed in more detail in the next chapter. However, their work is still limited by availability of data, and two of the main drawbacks in their work comes from lack of historical data and high-level data such as the type of shot executed.

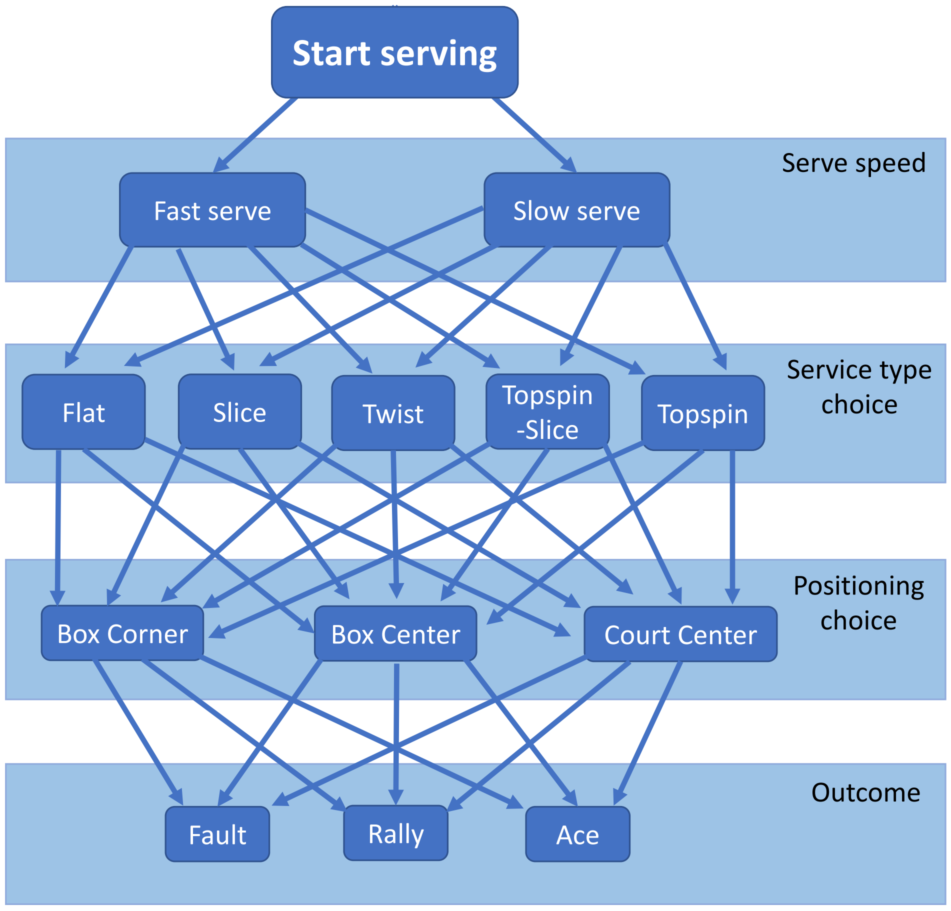


Figure 2.4: Markov chain for a serve, based on [8].

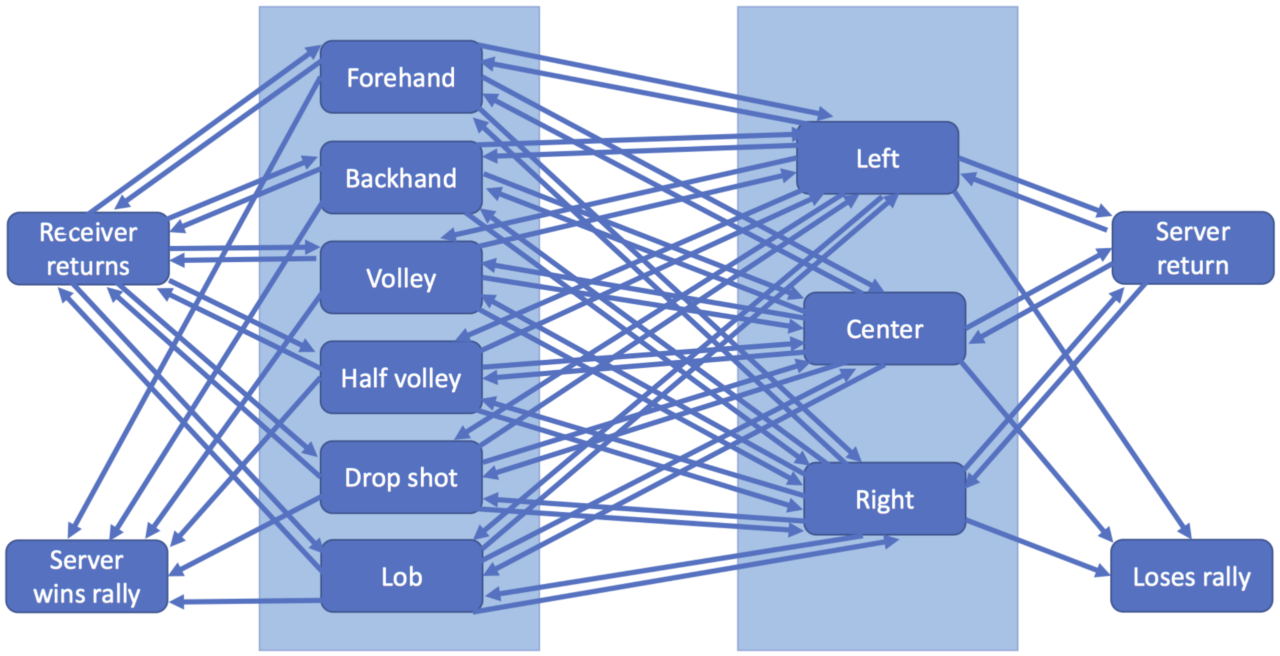


Figure 2.5: Markov chain of a rally, based on [8].

2.5 Computer Vision

Computer Vision as a field was born from the effort of making machines able to understand visual information such as images or videos. For humans, processing such information seems straightforward and feels effortless [27]. For example, it is easy for us to detect how many people are in a photograph and recognise who they are, provided that we know them, understand 3D shapes and textures from objects around us so that we can interact with them or identify objects such as the table or chairs even when illumination changes.

Given this, one might wonder why it is such a difficult task for computers. First of all, even if vision feels effortless to us, it is a very complex task for our nervous system [28; 29]. Research in Neuroscience to understand the visual system has been of prime importance for many years and we still do not entirely understand its functioning. In fact, Professor Mriganka Sur of MIT's Department of Brain and Cognitive Sciences said that "half of the human brain is devoted directly or indirectly to vision"[5]. It is hard to estimate the exact percentage of the brain dedicated to vision but is it widely accepted that the proportion is very high[6]. In addition to this, being able to process visual information also requires knowledge about the world. For instance, humans have the knowledge of what components constitute faces, animals or objects.

Hubel and Wiesel received a Nobel Prize for their research in the visual processing of the brain. In 1962 they performed one of the experiments that have most influenced the fields of Neuroscience and Computer Vision [30]. The study involved measuring the activity of neurons from cats that were looking to a projection of images of light patterns in front of them. From this, Hubel and Wiesel discovered that some neurons, which they named 'simple cells', were responding only to edges with a precise orientation and in exact locations while others, that they called 'complex cells', were activated by edges in any position and to specific motion patterns. This work was the inspiration for SIFT feature extraction, a well-known feature extraction algorithm in Computer Vision [31]. Figure 2.6 is a diagram showing the specific response of a 'simple cell' to the orientation of the stimulus.

[5]Source: `http://news.mit.edu/1996/visualprocessing`

[6]`http://channel.nationalgeographic.com/brain-games/articles/brain-games-watch-this-perception-facts`

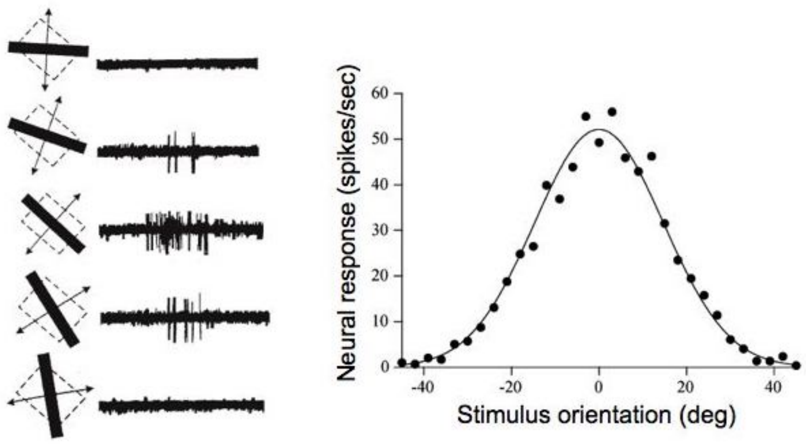


Figure 2.6: Tuning curve for neuron in Visual Cortex[7].

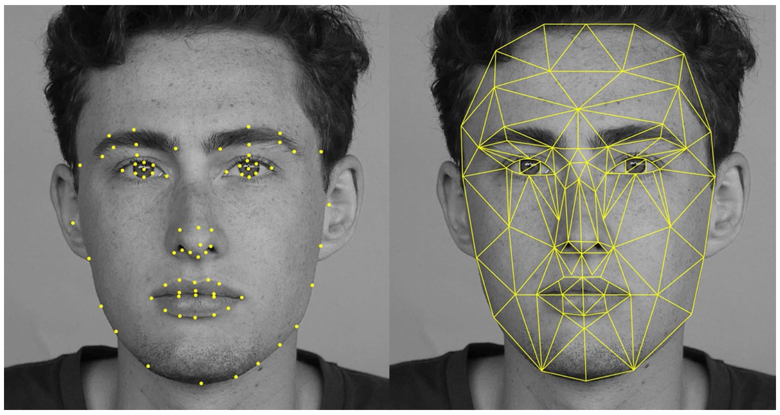


Figure 2.7: Face keypoints and mesh for facial recognition applications[8].

Computer Vision has been evolving in a great number of directions such as object recognition, face recognition (Figure 2.7), pose estimation (Figure 2.8) or image captioning (Figure 2.9). As a result of the advances in the field, we can see many applications of computer vision in our everyday lives. Some examples of these are:

- Postal Service: Postal automation involves automatically reading addresses, most of which are handwritten and therefore require Computer Vision techniques able to process this data.

- ePassport gates: Many airports have electronic gates able to automatically read the relevant information from biometric passports, scan the photograph and recognize whether the person standing in front of the machine is the same as in the photograph.

[7]Source: `http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/V1/lgn-V1.html`
[8]Source: `https://petapixel.com/2016/06/30/snapchats-powerful-facial-recognition-technology-works/`

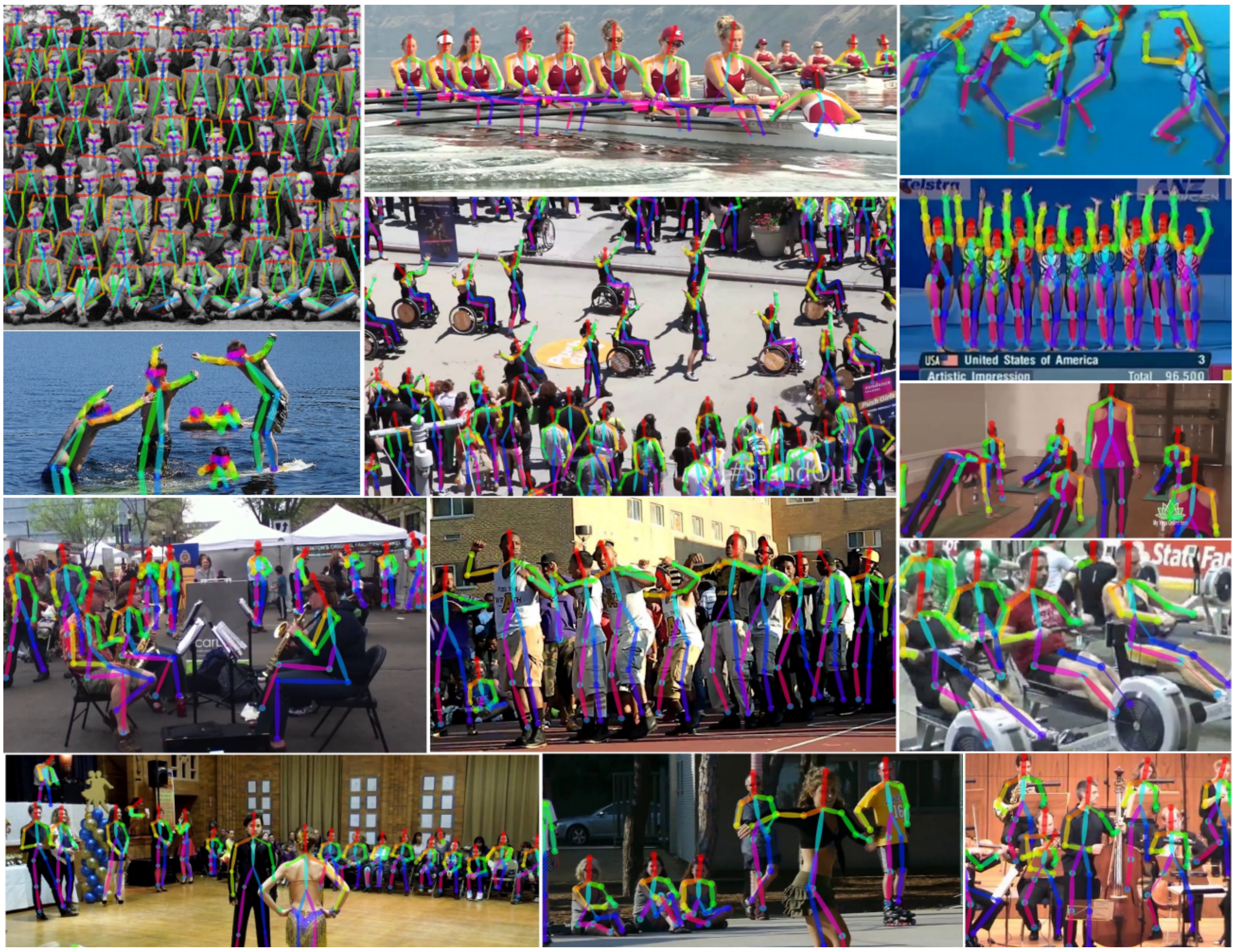


Figure 2.8: Pose estimation, from [32].

- Autonomous cars: self-driving cars have already been commercialized (e.g. by Tesla) and they require an understanding of their surroundings including road signs, moving objects and people around them, which is achieved using advanced Computer Vision techniques.

- Augmented reality: in the context of vision, this technology integrates computer-generated images with real-world ones. This requires recognising objects in the environment so that the computer-generated images can be superimposed in a coherent way. It has been integrated into games (e.g. Pokemon Go), messaging applications (e.g. Snapchat) and has potential in many other areas such as retail.

For a more thorough account of the field of Computer Vision, interested readers are invited to refer to [34].

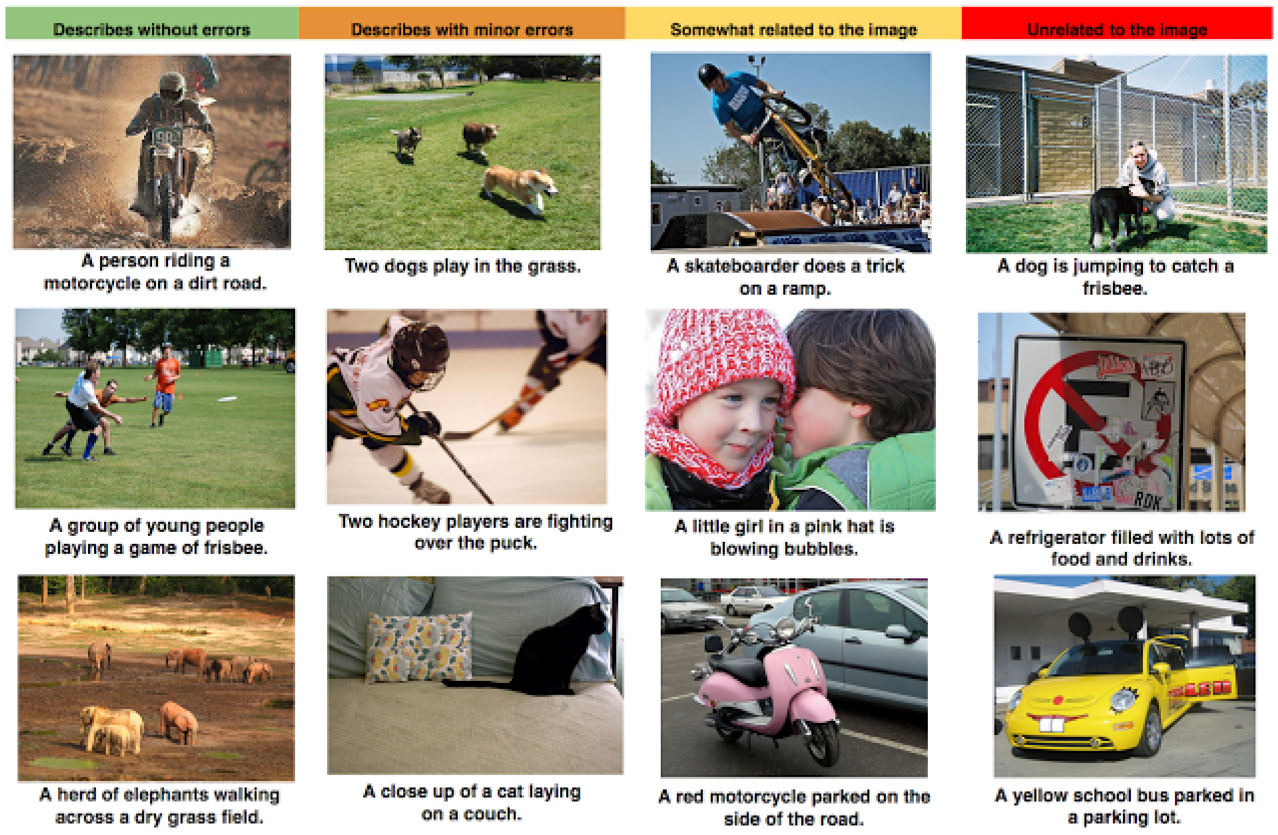

Describes without errors
Describes with minor errors
Somewhat related to the image
Unrelated to the image
A person riding a motorcycle on a dirt road.
Two dogs play in the grass.
A skateboarder does a trick on a ramp.
A dog is jumping to catch a frisbee.
A group of young people playing a game of frisbee.
Two hockey players are fighting over the puck.
A little girl in a pink hat is blowing bubbles.
A refrigerator filled with lots of food and drinks.
A herd of elephants walking across a dry grass field.
A close up of a cat laying on a couch.
A red motorcycle parked on the side of the road.
A yellow school bus parked in a parking lot.

Figure 2.9: Image captioning, from [33].

2.5.1 Optical Flow

Optical flow is used to represent the apparent motion of an image. James J. Gibson, an American psychologist pioneer in visual perception, was working in preparing training films for pilots in World War II when he developed his optical flow theory [35]. He proposed that when moving directly towards a target, that target seems to be motionless and the surroundings seem to go in the opposite direction with nearer objects moving faster, as shown in Figure 2.10. In Computer Vision, optical flow represents the apparent movement of objects between consecutive frames and works under the assumption that the object itself does not change colour over the frames and illumination remains uniform. Based on this assumption, if the pixel with coordinates (x, y) at time t moves by a distance dx and dy in the x and y directions respectively during the time interval dt:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{2.13}$$

where $I(x, y, t)$ represents the intensity of the pixel with coordinates (x, y) at time t. From this, the following optical flow equation can be obtained by Taylor series approximation:

$$\frac{\delta I}{\delta x}u + \frac{\delta I}{\delta y}v + \frac{\delta I}{\delta t} = 0 \tag{2.14}$$

where u and v are the optical flow components in the x and y direction and $\frac{\delta I}{\delta x}$, $\frac{\delta I}{\delta y}$ and $\frac{\delta I}{\delta t}$ are the derivatives of the image with respect to x, y and t. The optical flow vector is $[u, v]$ and different techniques exist to calculate it from the above equations; in our work, we used the Lucas–Kanade Method [36].

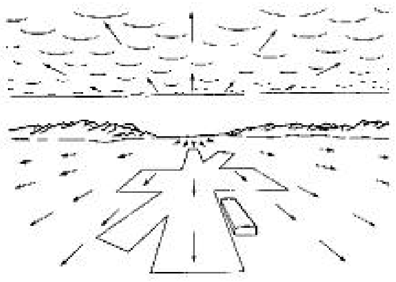


Figure 2.10: The idea of optical flow (shown by the arrows) was hypothesised by Gibson[9].

2.5.2 Feature Extraction

For many computer vision applications such as image classification or object detection, instead of processing the raw image as a matrix of colour values, a feature representation is used because it is supposed to meaningfully represent the image in a lower dimensional space. The choice of features is crucial to the success of the classifier (or any other algorithm) that will be used to complete the computer vision task (e.g. image classification, object detection, etc.) A large range of feature extraction algorithms exist: SIFT [31], Harris [37], HOG [38], etc. Here we will describe HOG and SIFT since they are standard feature representations used in action recognition. [39] is a survey of action recognition, revising the most popular feature extraction techniques for this application.

[9]`http://www.users.totalise.co.uk/~kbroom/Lectures/gibson.htm`

Histogram of Oriented Gradients (HOG)

HOG was initially designed for pedestrian detection [38], and it consists of four main steps:

1. Divide the image into patches of 8×8 pixels[10].

2. Calculate the gradients for each pixel: In image processing, the gradient ∇I of an image I measures its change in intensity (colour value), as illustrated in Figure 2.11. As a more formal definition, it corresponds to the partial derivative of the image value I with respect to the x and y directions:
$$\nabla I = \left[\frac{\delta I}{\delta x}, \frac{\delta I}{\delta y}\right] \tag{2.15}$$
A commonly used technique to perform such operation in images is to apply the $\left[-1, 0, 1\right]$ filter kernel to the image in the x and y directions separately.

3. Calculate a 9-bin[11] histogram per patch: The application of the previous step results in a gradient vector (with magnitude and direction) for each pixel in the 8×8 patch. A 9-bin histogram is created to represent the collection vectors of all pixels in the 8×8 patch. Each bin in the histogram corresponds to an angle range, representing the direction of the vector and the height of each bar is calculated by adding up the magnitude of vectors with the corresponding angle, as shown in Figure 2.12.

4. The resulting feature vector is a normalisation and concatenation of the histograms for every patch.

Scale Invariant Feature Transform (SIFT)

SIFT is a feature extraction algorithm that detects local keypoints and descriptors invariant to scaling and rotation, published by David Lowe in 1999 [31]. Therefore, SIFT features can be used to represent a given object and recognise it in images where the angle of view or scale is different.

[10] 8×8 are the dimensions originally used in [38], although other sizes can be used
[11] 9 bins performed best for the human detection experiments of [38]
[12] Source: `https://en.wikipedia.org/wiki/Image_gradient`

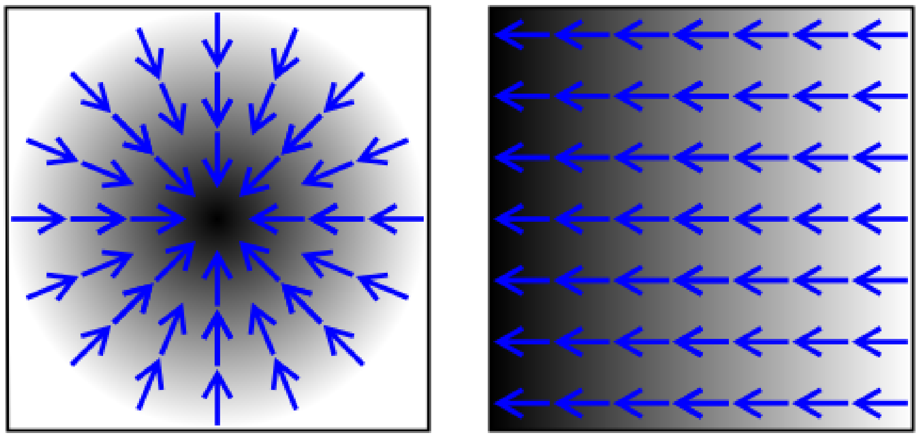


Figure 2.11: Image gradients vectors represented by blue arrows[12].

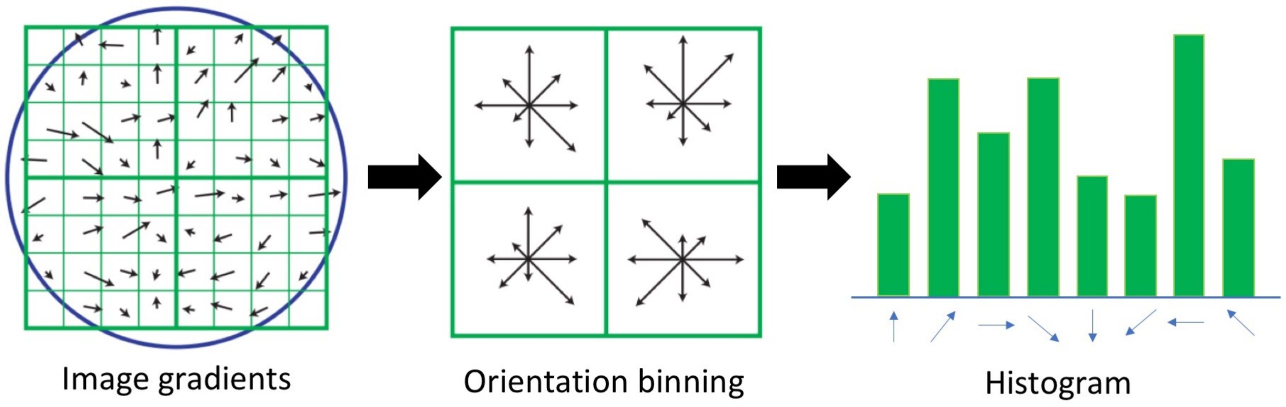


Figure 2.12: Diagram of the steps to construct a histogram of gradients, adapted from [40].

The first step in the SIFT algorithm is finding key points in the image. One of the most common approaches to find key points is the application of the Differences of Gaussians (DoG) function to the image at various scales. The first step in this process requires the application of a Gaussian kernel to the image. A 1-dimensional Gaussian distribution with mean μ and standard deviation σ^2 can be formalized as Equation (2.16) and as Equation (2.17) for D-dimensions, with $\boldsymbol{\mu}$ the D-dimensional mean vector and $\boldsymbol{\Sigma}$ the $D \times D$ covariance matrix [41].

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2} \tag{2.16}$$

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} \tag{2.17}$$

Applying a Gaussian kernel to an image results in blurring it and the value of σ determines the level of blurring. Taking the difference between images with different blurs is like performing a band-pass filter operation, selecting only a range of spatial frequencies. This is performed at various scales of the image. This process is shown in Figure 2.13 and has the name of Gaussian pyramid. Salient points are detected by selecting pixels which represent local extrema in the space and scale dimension. Selected points are refined by discarding points belonging to an edge or with little contrast. Each key point is assigned a dominant orientation based on gradients, as described above and for each key, a descriptor is generated as a histogram of gradients for the key point and its neighbouring pixels. For more details about this process, the reader is directed to [31]. The main difference between SIFT and HOG is that the first is a local descriptor and the latter provides features for the entire image. SIFT is used to recognise objects by matching key points with corresponding characteristics, as illustrated in Figure 2.14.

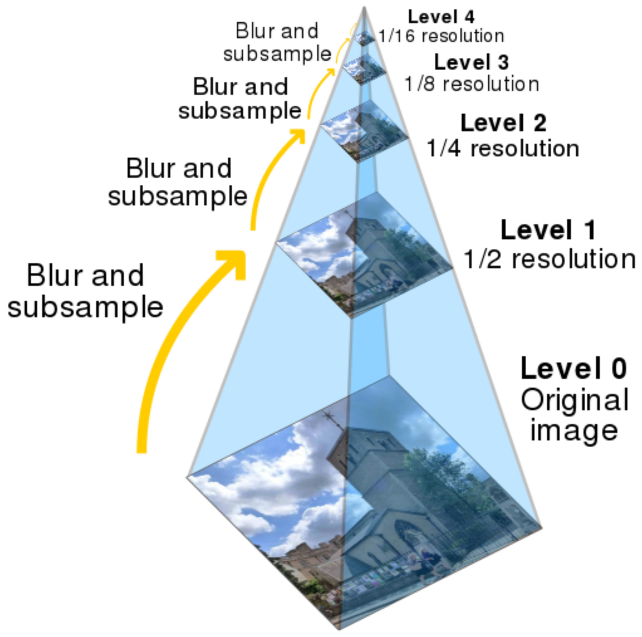


Figure 2.13: 5-level image pyramid diagram[13].

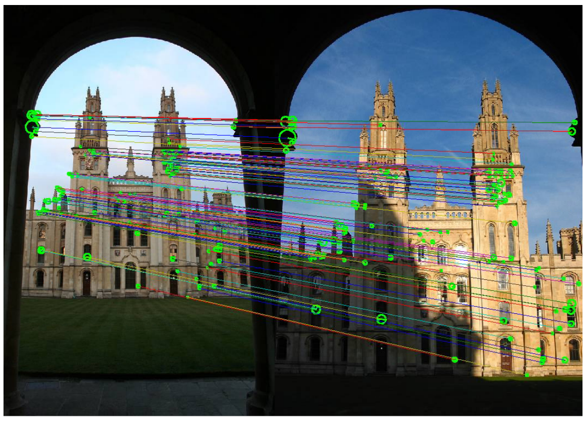


Figure 2.14: SIFT feature matching[14].

[7]Source: https://en.wikipedia.org/wiki/Pyramid_(image_processing)

[8]Source: http://www.robots.ox.ac.uk/~vgg/share/SearchPractical2012.html

2.5.3 Hough Line Transform

Cartesian vs Polar Coordinates

The Cartesian coordinate system specifies each point in the plane by a pair of coordinates x_0 and y_0 along the x and y orthogonal axes, as shown in Figure 2.15. The polar coordinate system is also two-dimensional, determined by a point (the origin) and a polar axis. These are shown as the point O and x axis in Figure 2.15, respectively. A point a in the polar coordinate system is defined by the distance ρ from the origin to the point and the angle θ between the polar axis and the line passing through O and a. In this section, polar coordinates are expressed as $(\theta, \rho)_P$ and Cartesian coordinates as $(x, y)_C$.

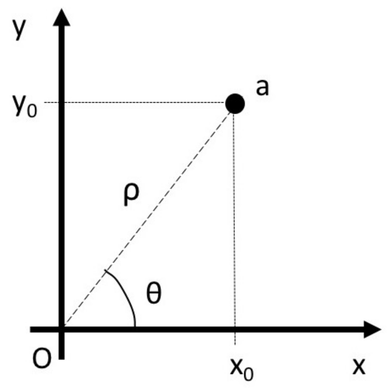


Figure 2.15: Point a in a polar and Cartesian coordinate system.

Line Definition

In a Cartesian coordinate system, a line can be defined by the $y = ax + b$, equation where a and b are the slope and the y-intercept, respectively. However, vertical lines cannot be represented in this form because they do not intersect with the y axis and this is why, here, we describe lines in the polar coordinate system. In the latter system, a line can be defined by the following equation:

$$\rho = x \cos\theta + y \sin\theta \tag{2.18}$$

where θ and ρ are the polar coordinates (defined above) of the point of intersection between the line and its normal through the origin. The set of all lines passing through a given point with coordinates $(x, y)_C$ are all pairs of ρ and θ values satisfying Equation (2.18).

Hough Space

Lines can be represented in the Hough space by its two parameters θ and ρ. Therefore a line can be mapped to the θ and ρ point in the Hough space. The mapping of a line to a point in the Hough space is shown in Figure 2.16.

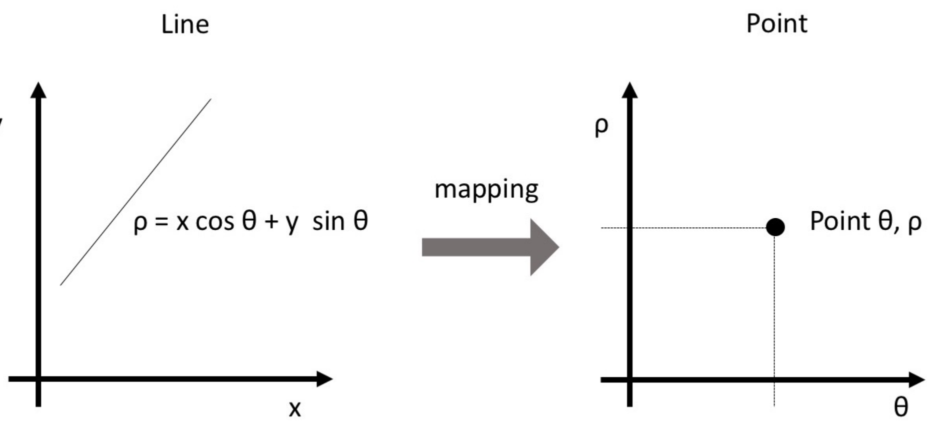


Figure 2.16: Mapping of a line to a point in the Hough space, adapted from [42].

Hough Transform for Line Detection

We have just seen that a line can be mapped to a point in the Hough space. Mapping all possible lines through a given point to the Hough space results in a sinusoidal curve, as shown in Figure 2.17. Given two distinct points, the mapping of all possible lines through each of them will results in two distinct sinusoidal curves, shown in Figure 2.18. Their intersection is a point in the Hough space, which represents a line (cf. Figure 2.16) passing through both points.

Hough Transform Algorithm

The Hough Transform algorithm can detect lines even in the presence of noise. It has to be performed on binary images that contain the edges of the source image only. The algorithm

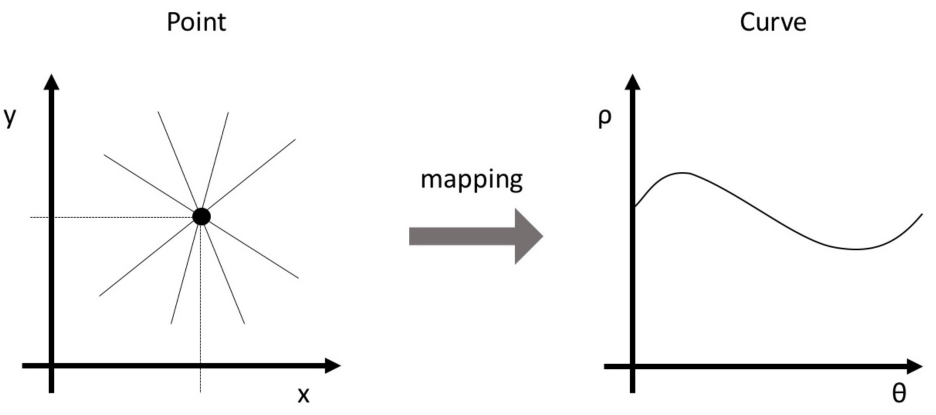


Figure 2.17: Mapping of all line through a point to the Hough space.

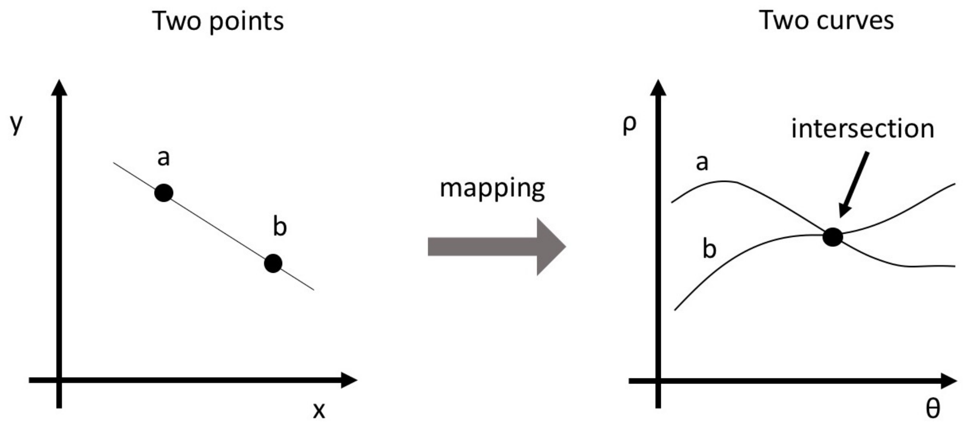


Figure 2.18: Mapping of all lines through point a and all lines through point b to the Hough space. The mapping back of their intersection defines the line crossing points a and b.

works by finding all lines that can cross each of the edge points; then the possible lines that are common to a higher number of points are selected.

As an example, Figure 2.19a presents a source image as a 100×100 matrix containing three edge points $a = (0,0)_C$, $b = (100,0)_C$ and $c = (0,100)_C$. For each of these points, we ought to find all possible lines that cross it, which can be achieved by solving Equation (2.18). Plotting all pairs of θ and ρ values that satisfy the previous equation for a given point $(x,y)_C$ in a 2D Cartesian coordinate system with θ and ρ in the x and y axes respectively results in a sinusoid. Plotting the solutions to Equation (2.18) for each point a, b and c of our example results in the blue, red and green curves of Figure 2.19b[15].

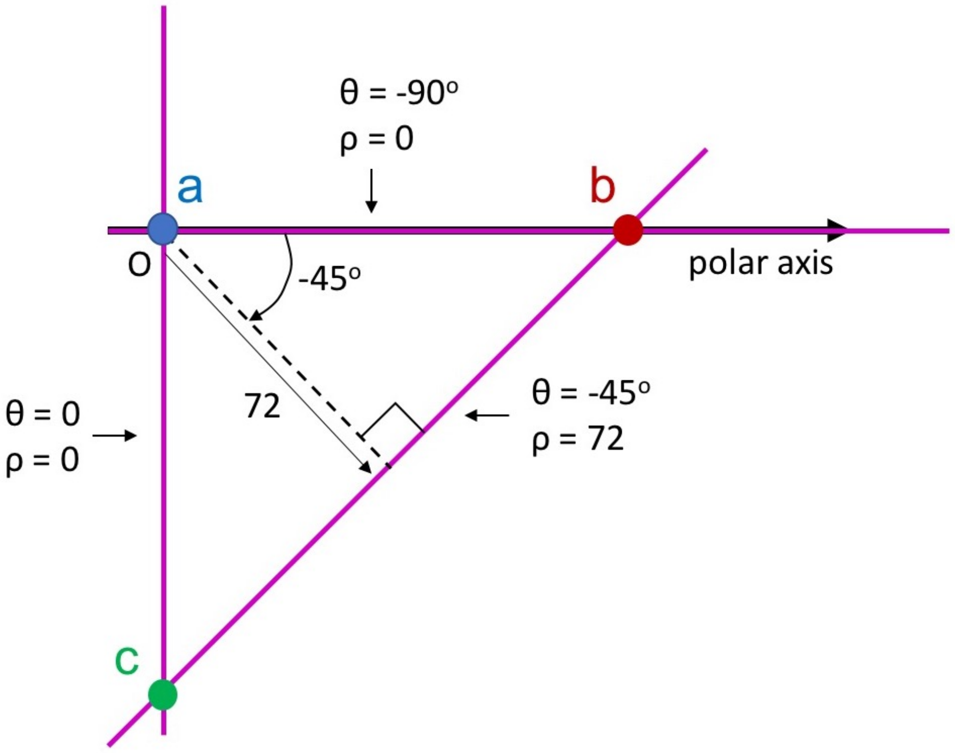
[15] ρ_{min} and ρ_{max} are determined by calculating the length of the diagonal of the image

Based on the above, an accumulator array with a discrete set of bins representing pairs of (θ,ρ= values is calculated, as shown in Figure 2.19b. Each edge point is said to "vote" for the bins of θ and ρ pairs defining lines that cross it, according to Equation (2.18). For example the bin corresponding to the intersection of the green and red curves in Figure 2.19b are the θ and ρ values defining a line (according to Equation (2.18)) that passes through the points b and c, and thus has 2 votes.

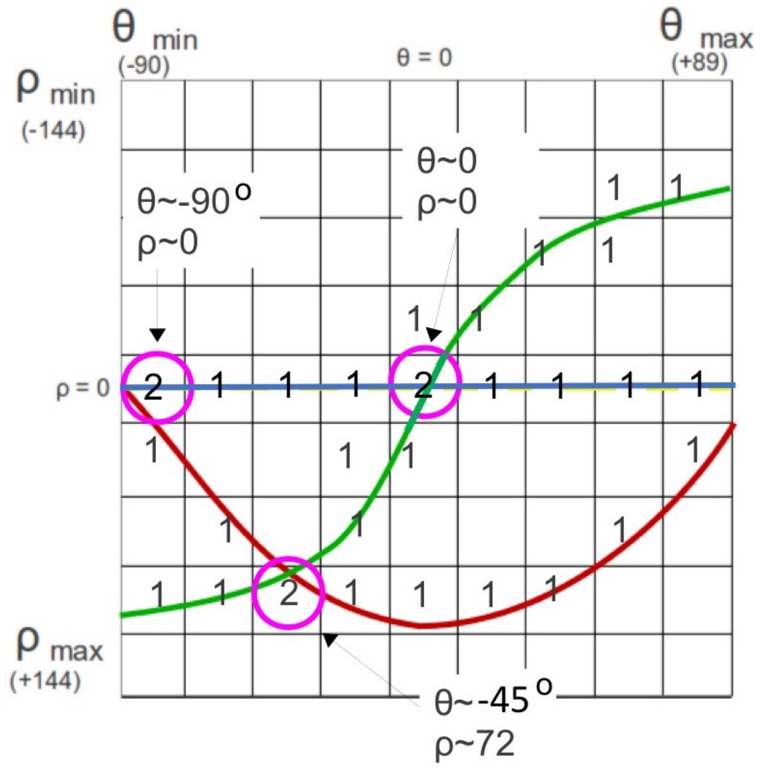
In our example, Figure 2.19b has three bins with two votes each, shown by the circles, at the bins with values $(0, 0)_P$, $(-90°, 0)_P$ and $(-45°, 72)_P$. Therefore the lines defined by these values according to Equation (2.18) are said to have 2 votes each. Since these are the most voted lines, they are selected as the lines detected in our example of Figure 2.19a and are plotted in magenta in Figure 2.19a. Indeed, each of these three lines passes through two points in Figure 2.19a (a & b, a & c and b & c). Our example consists of three non-aligned points. Thus the maximum number of points that a line can cross is two. Accordingly, there are no bins with more than two votes in Figure 2.19b. However, using this algorithm with a higher number of points (e.g. the edges of an image) results in an accumulator array with a higher number of votes per bin than in our example. In such case, a threshold on the minimum number of votes required to select a line of interest must be determined.

Probabilistic Hough Transform

In this project, we have used the probabilistic Hough transform algorithm that is implemented in OpenCV [43] that is based on [44]. The aim of probabilistic Hough transform is to reduce the number of edge points that are evaluated but still maintain accuracy, improving the time complexity over its non-probabilistic version. In this particular version of the probabilistic Hough transform, only a random subset of points is considered for line detection. Accordingly, the threshold on the minimum number of votes required to select a line must be adjusted to the reduced number of points used in line detection.



(a) 2D image with three points a, b and c and the lines that connect them (magenta).



(b) Hough transform accumulator array of (a).

Figure 2.19: 2D matrix containing three points and the accumulator array used to find lines.

2.6 Machine Learning

In 1959, Arthur Samuel first introduced the term “Machine Learning” as the “field of study that gives computers the ability to learn without being explicitly programmed”[16]. He is the author of what is thought to be the first program with the capacity to learn: a checkers program [45]. In 1997, Mitchell proposed the following definition: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E” [46]. Amongst the many possibilities, tasks T can refer to classification or regression, E to the training dataset, which contains examples, and P can refer to the accuracy or error rate of the model [47].

The field of Machine Learning is vast, and its applications encompass a wide variety of problems such as classification, regression, clustering or dimensionality reduction. However, Machine Learning algorithms can be divided into three broad categories according to the type of data they learn from [48]:

- Supervised learning: This class of Machine Learning algorithms learn from data labelled with a class or value, and the objective is to find patterns in the data in order to predict these labels for unseen data. An example of this can be handwritten digit recognition. In this case, the Machine Learning algorithm receives as input many examples of digits written by different people and each example is labelled with the correct digit. The objective is to associate one of the ten possible digits to new unseen examples by finding patterns in the previous data.
- Unsupervised learning: The algorithm is input data without specific labels associated with it. The objective is to find patterns and structures to, for example, cluster data into meaningful groups. For example, customers can be grouped according to their behaviour to make recommendations.

[16]Originates from “Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort” in [45]

- Reinforcement learning [49]: In this type of learning an agent takes actions that modify the state of its environment and leads to some rewards. The objective of the agent is to select the actions that will maximise its reward. This type of learning is very suitable for automated playing of Atari games, in which the reward can be the score [50].

One of the biggest challenges in machine learning is overfitting, which occurs when the model fits the training data too closely and loses the capacity of generalisation. This occurs when a model is too complex for the amount of data used in training. The model is said to be memorising the data. It learns noise and details irrelevant to the task and fails to generalise to new data [48]. Many machine learning algorithms consist of three main steps: learning from the data, tuning the parameters and testing. To avoid overfitting, it is usual to divide the data into three groups: training, validation and test, corresponding to the previously mentioned steps.

Machine learning techniques are used in a wide range of applications and many multimedia applications also involve Machine Learning:

- Speech recognition: this corresponds to translating spoken language into text. Machine Learning algorithms are trained with audio clips, labelled with the corresponding written text and can be fine-tuned using voices from particular users. In 2016, researchers at Microsoft presented a Neural Network for speech recognition that achieved results on par with human performance [51]. Applications of speech recognition are part of our everyday lives with Siri[17], Google Home [52] and Amazon Alexa[18].
- Recommender systems: such algorithms have the objective of finding patterns in what people are interested in (products purchased, movies, music or restaurants) and recommend new products [53; 54]. One approach to recommender systems for online sales is to group people by their purchase behaviour and recommend products bought by individuals in the same group.
- Computational Creativity: this term has been defined as "The study and support, through computational means and methods, of behaviour exhibited by natural and artificial sys-

[17]`https://machinelearning.apple.com/2017/10/01/hey-siri.html`
[18]`https://developer.amazon.com/docs/alexa-voice-service/api-overview.html`

tems, which would be deemed creative if exhibited by humans." by Wiggins [55]. In recent years, deep neural networks have been able to generate novel images [56] or transfer style [57], as shown in Figure 2.20. The fact that the algorithm picks up style but keeps the coherent structures in the images (the same objects appear in the modified image) is remarkable.

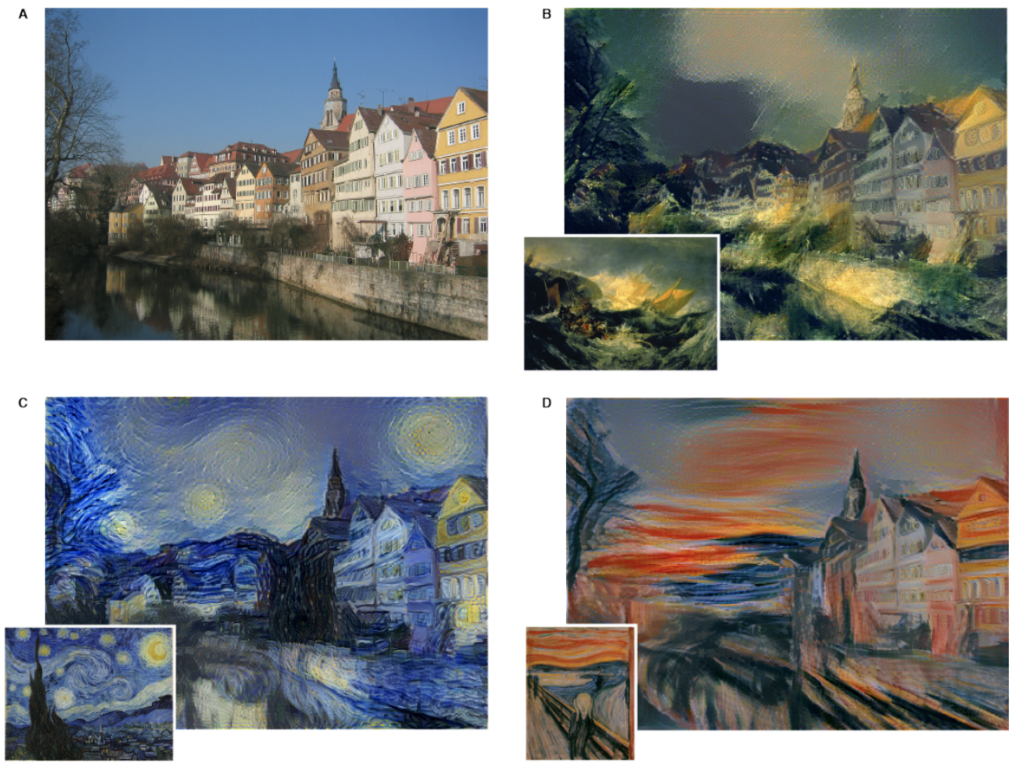


Figure 2.20: Style transfer to image A, from [57].

A more detailed account of Machine Learning foundations can be found in [41].

2.6.1 Artificial Neural Networks

Artificial Neural Networks are computer systems inspired by the functioning of the nervous system [58; 59]. The biological brain has neurons that are interconnected and in a highly structured way in order to process information [60]. Neural networks also have processing units called 'neurons' which are based on models of their biological counterparts. The simplest neural network, presented in 1958 by the American psychologist Frank Rosenblatt, consists of a single

[18]Source: `https://deepdreamgenerator.com/`

layer and is called perceptron [61]. Figure 2.21 is a diagram of a perceptron with only one neuron, two inputs x_1 and x_2, which correspond to the features of instance x, and output y. This simple model is a linear binary classifier in which the output is calculated as:

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i + b > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

with n the number of inputs to the neuron (2 in our example), w_i the weight for input x_i and b is the bias, this value is a constant used to shift the decision boundary to 0.

Training consists in using labelled examples to find the optimal weight values for the correct prediction of labels in unseen examples. The first step in the training process is the random initialization of weights. Then they are updated using labelled examples through the iteration of the two steps below applied to the complete set of examples available for training:

1. Calculate the output $y_k(t)$ from current weights $\mathbf{w}(t)$ for a given example k, using Equation (2.19). This is the predicted label for k at time t.

2. Update the weights as follows:

$$w_i(t+1) = w_i(t) + (j_k - y_k(t))x_{k,i} \tag{2.20}$$

   with $w_i(t+1)$ the updated i^{th} weight at time $t+1$, j_k the correct label of example k, $y_k(t)$ its predicted label based on the weights $\mathbf{w}(t)$ at time t and $x_{k,i}$ the i^{th} feature (input) of the k^{th} example.

As such, neural networks learn by example, adjusting their weights to optimize their performance to a specific task. The end of the learning process can be determined in a number of ways [47]:

- Convergence: all samples are classified correctly.

- Number of iterations: a pre-determined number of iterations is reached.

- Inadequate progress: weight change is below a chosen threshold, and the network is assumed to not be‘learning’ any more.

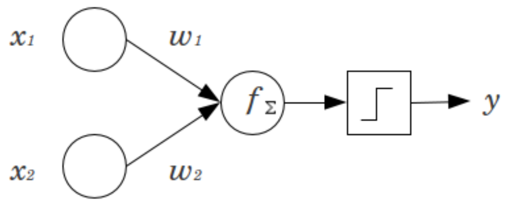


Figure 2.21: Diagram of a perceptron[19].

Based on the simple model presented above, more complex neural networks can be built, such as the one shown in Figure 2.22 with 2 hidden layers, and some neural networks have even more than 40 layers of neurons, hence the name of “deep” neural networks [47]. Each layer takes as input the output of the previous layer and the last one outputs the result. Each layer of a deep neural network has the same principles as the perceptron: they also have an activation function to calculate the output, given the inputs and their weights (cf. Equation (2.19)) and a weight update function (cf. Equation (2.20)). In theory, neural networks with more layers can model data with more complex structures [47]. The formulation of their activation function and weight update is described below.

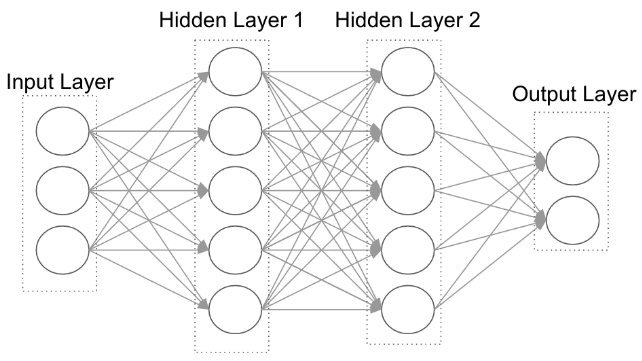


Figure 2.22: Neural network with 2 hidden layers[20].

Activation Functions: They are used to calculate the output of a neural network layer given its inputs and weights, as Equation (2.19) does for the perceptron. The most common activation functions, some of which shown in Figure 2.23, are [47]:

[19]Source: `https://glowingpython.blogspot.co.uk/2011/10/perceptron.html`
[20]Source: `http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/`

- Linear: this activation is proportional to the input and therefore is not suitable to model data with complex (non-linear) patterns. Another drawback of this is that values can blow-up in large networks.

- Sigmoid: it is non-linear, differentiable and within the [0,1] range. y values tend to concentrate at the extremes 0 and 1, which makes it suitable for classification tasks. However, this also leads to the problem of "vanishing gradients" as extreme x values cause only very small changes in y and learning may stall.

- Tanh (hyperbolic tangent): it is similar to the sigmoid activation but within the range [-1,1] and with steeper gradients.

- ReLU (Rectified Linear Unit): it is also non-linear but within the $[0, \infty)$ range. For all negative values, neurons have a 0 activation which can cause "dying" neurons, as neurons can have a 0 gradient and stop responding to the input. This is not a problem because the activations are more sparse, but the network also has the possibility to stop responding if too many neurons die.

Currently, the default recommendation is to use ReLU activation [47; 62].

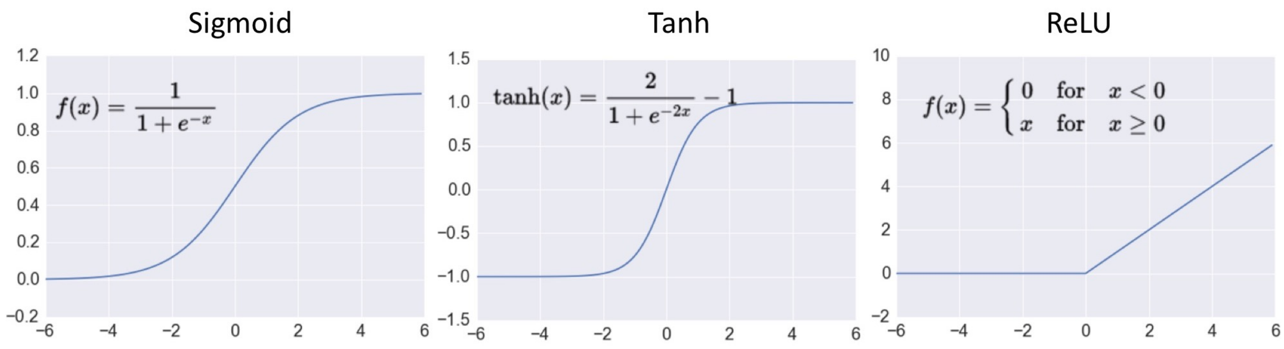


Figure 2.23: Different activation functions for neural networks[21].

Weight updates: In Equation (2.20) we showed how weights are updated based on the predicted and correct labels. For deep neural networks, a different method must be used to take into account the contribution of each neuron to the output and adjust each of their weights

[21]Source: `http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/`

accordingly. This calculation is called backpropagation and requires the definition of a loss function to quantify the prediction error. For example, the Euclidean distance between predicted outputs and data labels can be used. A common technique for backpropagation is gradient descent. It consists in moving towards local minima in the error function by calculating the gradient of the loss function with respect to the parameters of the network (e.g. weights). Figure 2.24 is a plot of a sample loss function with respect to parameters θ_0 and θ_1 and by following the gradient descent steps in black, parameters θ_0 and θ_1 yielding local minima can be found. It is important to be aware of the possibility of getting trapped in sub-optimal local minima.

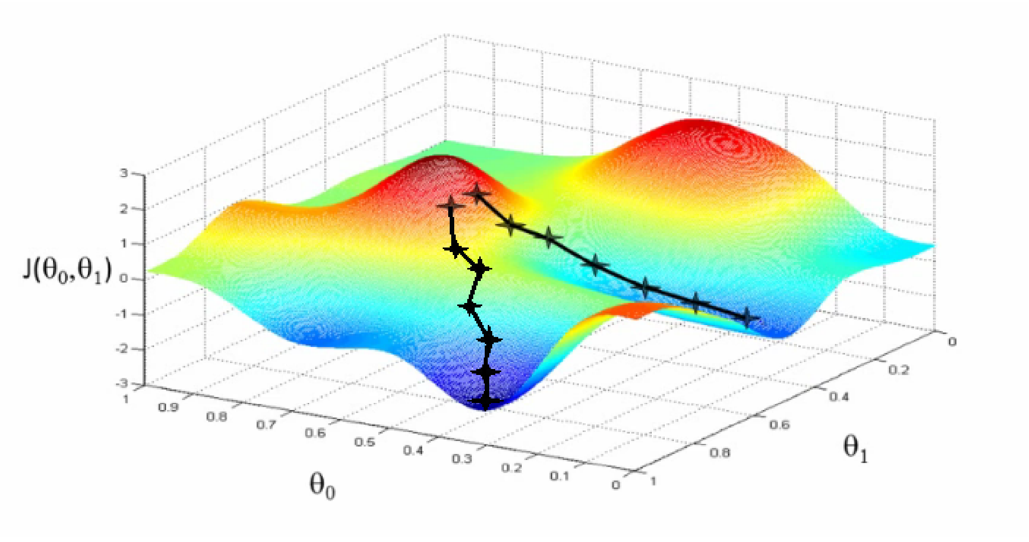


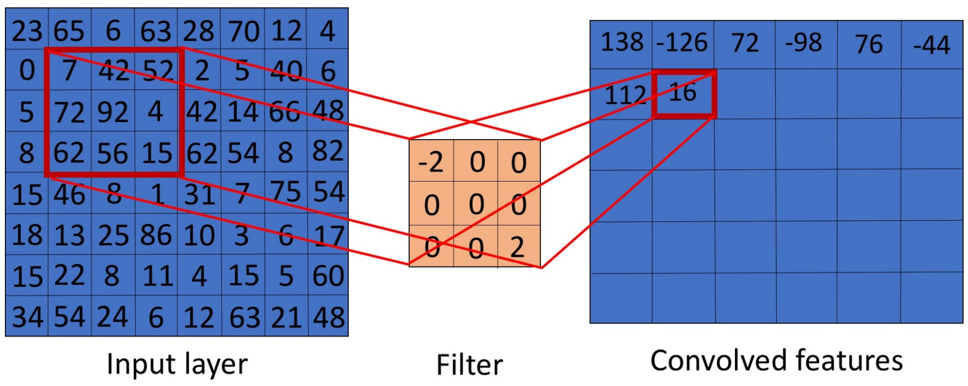
Figure 2.24: Loss function J with respect to parameters θ_0 and θ_1 and gradient descent steps shown in black[22].

Deep neural networks have been demonstrated to be incredibly powerful in the context of computer vision in tasks such as face recognition, object detection and object recognition [63]. Convolutional neural networks are a type of deep neural network that have consistently won object recognition competitions since 2012 [64; 65]. Convolutional neural networks use a combination of the following types of layers [47]:

- Fully connected layer: every neuron of layer $(l-1)$ has a connection to every single neuron in layer l.
- Convolutional layer: a filter is applied to patches of neurons from the previous layer in a sliding manner, by calculating their dot product, until the previous layer has been

[22]Source: `https://www.analyticsvidhya.com/blog/2017/03/`

processed entirely. For example, if layer $l-1$ is the input layer consisting of a 96×96 image and we decide the filter to be of size 8×8, each neuron in layer l will have 64 neurons as an input (8×8) and the resulting convolved layer will have 96×96 neurons (if padding is used, for example adding zeros around the border). This type of layer is suitable for images by finding local features while keeping spatial relationships. This is illustrated in Figure 2.25, with an 8×8 input, a 3×3 filter and no padding.



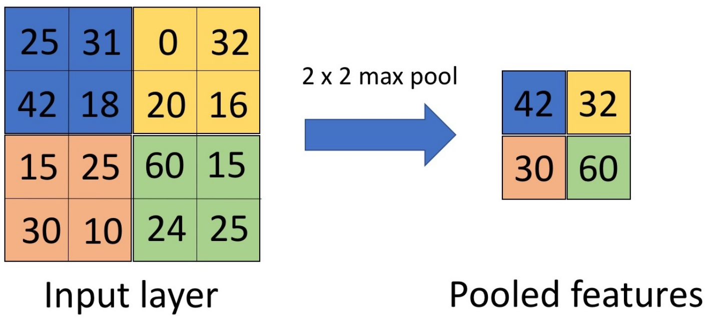
23	65	6	63	28	70	12	4
0	7	42	52	2	5	40	6
5	72	92	4	42	14	66	48
8	62	56	15	62	54	8	82
15	46	8	1	31	7	75	54
18	13	25	86	10	3	6	17
15	22	8	11	4	15	5	60
34	54	24	6	12	63	21	48

-2	0	0
0	0	0
0	0	2

138	-126	72	-98	76	-44
112	16				

Figure 2.25: Convolutional layer using a 3×3 filter. In this case, since no padding was used, the convolved layer is 6×6.

- Pooling layer: this layer is generally used after a convolutional layer to reduce the dimensionaly of the features. For each $m \times n$ region of the image, the max or mean feature value from the convolutional layer is calculated (called max-pooling and mean-pooling layer respectively) resulting in a more compact and generalizable feature representation of the image, as shown in Figure 2.26. This is supposed to lead to better image classification as features are less location dependent and are more robust against small translations [63].



25	31	0	32
42	18	20	16
15	25	60	15
30	10	24	25

42	32
30	60

Figure 2.26: Max-pooling layer example.

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) [66–68] are a type of neural network with the ability to learn time dependencies, making them very suitable to process sequences. At time t, the output h_t of the RNN is calculated by taking as inputs its previous output h_{t-1} and the current element of the sequence x_t as follows:

$$h_t = f(W_{xh} \times x_t + W_{hh} \times h_{t-1} + b) \tag{2.21}$$

where W_{xh} and W_{hh} are weight matrices, b the bias and f is the output activation function [69].

LSTMs

RNNs are suitable for learning time dependencies but, when applied to long sequences, the gradient of the loss function (cf. Figure 2.24) is likely to vanish [70]. This is because gradients at each time-step are calculated using the chain rule for backpropagation and therefore the first layers of the recurrent network (earlier time steps) can end up with a very small gradient. In 1997, a type of RNN called LSTM was introduced to overcome this issue [71]. LSTMs are particularly suited to learn long-term dependencies in sequences, such as in video classification or speech processing. They are composed of memory cells which contain a memory state c that is updated with the new inputs, but controlled by gates determining which information to keep and what to forget.

The cell state c_t is updated by "forgetting" some information through the multiplication of the previous cell state c_{t-1} and the forget gate f_t and by adding new information, controlled by the input gate i_t. Finally, the output gate o_t, controls the output of the cell h_t. The LSTM implementation that we used is based on [69; 72] (see Figure 2.27) and the calculations of the activations are as follows:

Input gate

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{2.22}$$

Forget gate

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{2.23}$$

Memory cell state

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{2.24}$$

Output gate

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{2.25}$$

Hidden state

$$h_t = o_t \tanh(c_t) \tag{2.26}$$

2.6.2 Random Forest Classifier

A random forest is an ensemble learning method, introduced by Breiman and Cutler that can be used for regression and classification [73; 74]. It builds a number of decision trees, each of which outputs a class number and the mode of these classifications is taken as the final result.

Decision Trees are a learning method that can be used for regression and classification [75] (called regression and classification trees respectively). In a simple classification tree, one of the features of the input data is handled at each node, and the data space split accordingly. For instance, let us define a dataset with N examples that we want to classify into two classes A and B and each example is characterised by three features $[x_1, x_2, x_3]$. Figure 2.28 shows how data is divided according to each feature relative to a particular threshold for each of them. At the end are the leaf nodes containing the class name.

In our work, we used random forests for classification, based on the work of Breiman in 1999 [73; 74]. Here, each tree is trained with a sub-sample of all the data and sub-samples are not mutually exclusive. At each level, a random subset of features is used to split the data space (instead of just one feature as in Figure 2.28) and the tree is grown to a maximum size without pruning. This leads to trees with a large set of terminal nodes, which are likely to overfit the data [76]. However, even though individual trees might be overfitting, their combination in a random forest is much less prone to this effect, as shown by Breiman in [73].

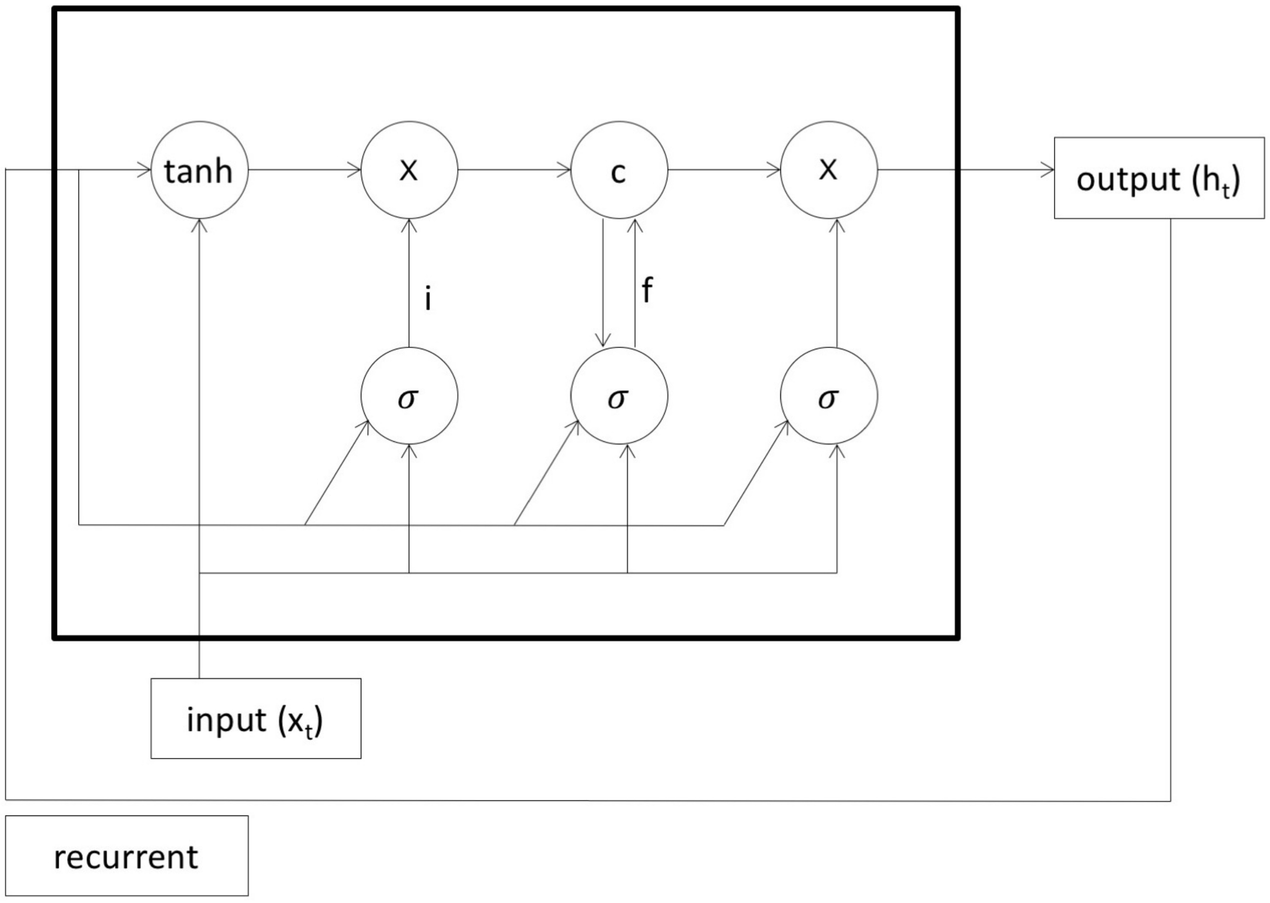


Figure 2.27: LSTM cell architecture. The memory of the LSTM cell is stored in c. Through time and as new inputs are fed in, the memory state is updated controlled by the forget and input gates. Both have as input: (1) the previous memory state c_{t-1}, (2) the previous output h_{t-1} and (3) the current input x_t; their activation is calculated as described in Equations (2.22) and (2.23). The cell state is modified by multiplying the old memory c_{t-1} by the forget gate, which will determine how much of the old memory to keep. Then, new memories (input activation) are added, controlled by the input gate. The output, which is then fed back to the network, is calculated as in Equation (2.25). σ represents the sigmoid function, Diagram inspired by [69].

In the random forest implementation that we have used (from [43]), each tree is trained with the same parameters and using $\sqrt{n}$ features (with n the total number of potential features) at each node and 2/3 of the examples in the dataset, as recommended in [77]. Random forests do not need cross-validation, and the error is calculated by making predictions for each example using only the trees that were trained without using that particular example; these are called "out of bag" (oob) examples [73]. The error is calculated as the ratio of oob misclassified examples over N observations.

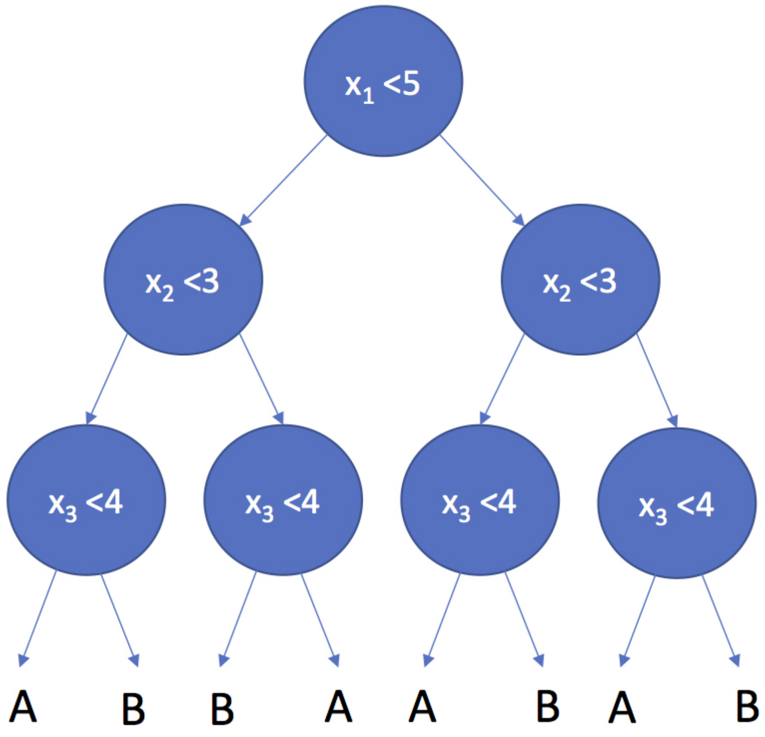


Figure 2.28: Example of classification tree.

2.6.3 K-means Clustering

The k-means algorithm is a non-probabilistic clustering technique introduced in 1982 by S. Lloyd [78]. It consists in partitioning N observations into K clusters with centres $\boldsymbol{\mu}_k$ (mean of the cluster k observations). These are randomly initialized and determined by the algorithm. If we define our observations as D-dimensional in a Euclidean space, an observation $\mathbf{x}_n$ is a D-dimensional vector of the features of example n. According to the k-means algorithm, $\mathbf{x}_n$ will belong the cluster j if its Euclidean distance[23] to $\boldsymbol{\mu}_j$ is the smallest amongst all K clusters. The matrix $\mathbf{r}$ is a two dimensional bit array used to indicate to which cluster an example belongs to, if x_n belongs to cluster j, r_{nj} will be 1 and r_{nk} for any other cluster k will be 0. The aim is to choose cluster centres that minimise the distance between intra-cluster observations and maximise inter-cluster distances, which using the notation in [41] can be formalised as minimising J with respect to r_{nk} and $\boldsymbol{\mu}_k$:

$$J(\mathbf{r}, \mu_1, \dots, \mu_K) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||\mathbf{x}_n - \boldsymbol{\mu}_k||^2 \tag{2.27}$$

[23]Other distance metrics can also be used.

To find the best $\mathbf{r}$ and $\boldsymbol{\mu}_k$ values, iterations of a two-step process are usually used:

- Expectation: $\boldsymbol{\mu}_k$ is fixed (random initialization) and J minimized with respect to r_{nk}. We assign each sample to a cluster.
- Maximization: minimize J with respect to $\boldsymbol{\mu}_k$ with r_{nk} fixed. We update the centres of the clusters.

and this is called the Expectation Maximisation (EM) algorithm [79]. The algorithm can give different results for different initializations.

2.6.4 Support Vector Machine (SVM)

An SVM is a machine learning model most commonly used in supervised learning for classification [80]. Given a set of labelled observations $(\mathbf{x}_1, y_1)$, $(\mathbf{x}_2, y_2)$, , $(\mathbf{x}_t, y_t)$ with $y_i = +/-1$ (belonging to two classes), the SVM will find the hyperplane best-separating the examples from the two classes, ensuring that the distance from the hyperplane to the closest observation from each class is maximised, as illustrated in Figure 2.29. The hyperplane is defined as:

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{2.28}$$

where $\mathbf{w}$ and b are the parameters of the hyperplane. The positive support vector is the data point, denoted $\mathbf{x}_+$, that lies closest the hyperplane defined by:

$$\mathbf{w}^T\mathbf{x}_+ + b = 1 \tag{2.29}$$

Equivalently, the negative support vector is denoted $\mathbf{x}_-$ and defined by:

$$\mathbf{w}^T\mathbf{x}_- + b = -1 \tag{2.30}$$

From this, the distance between the positive support vector to the hyperplane of Equation (2.28) is:

$$|\mathbf{w}\mathbf{x} + b|/||\mathbf{w}|| = \frac{1}{||\mathbf{w}||} \tag{2.31}$$

and similarly for the negative support vector, resulting in a margin (see Figure 2.29) of $\frac{2}{||\mathbf{w}||}$. Maximising this margin can be achieved by solving the following minimization problem:

$$\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} ||\mathbf{w}||_2 \tag{2.32}$$

subject to

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1 \text{ for } i = 1...N \text{ and } y_i = -1, 1 \tag{2.33}$$

with y_i the class. This is a quadratic convex optimization problem, and therefore it has a unique minimum.

If the data is not linearly separable, the “kernel trick” can be used [81]. This consists in mapping the data points to another space where the data is linearly separable by applying a transformation function (e.g. polynomial, radial basis, sigmoid etc.) It is called a trick because instead of applying the transformation function to the data and optimizing Equation (2.32), kernels can be directly used in Equation (2.32).

In cases of noisy data, a soft margin can be used, which allows misclassifications by adding a penalty proportional to the distance between the outlier and the correct side of the hyperplane [82; 83].

Applications to regression problems [80; 84] or multi-class classification [85; 86] have been developed, but these will not be discussed here as they are not used in this thesis.

2.6.5 Bayesian/Belief Networks (BN)

Bayesian Networks, also called belief networks, are probabilistic graphical models that use a directed acyclic graph with edges representing probabilistic relationships between random variables, called nodes. In such networks, each node has a conditional probability table determined by its parents in which a node x with parent a has a probability distribution defined as follows:

$$p(x) = p(x \mid a)p(a) \tag{2.34}$$

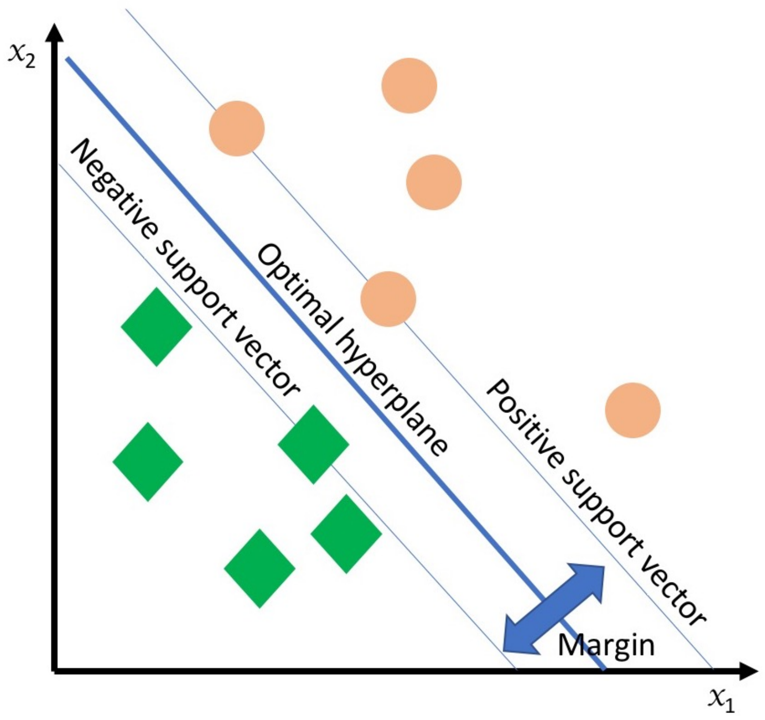


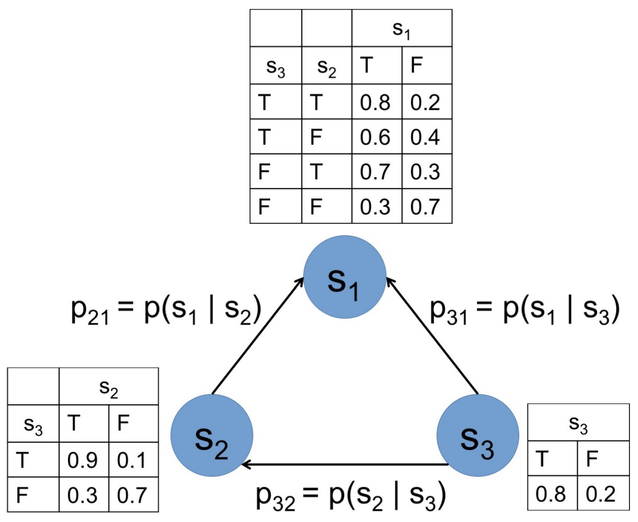
Figure 2.29: Diagram of a Support Vector Machine.

Figure 2.30 represents a simple BN with three random variables (s_1, s_2, s_3) and the probabilities associated with it such that $p_{ij} = p(s_j|s_i)$ [41]. In the figure, s_3 influences s_1 and s_2, and s_2 influences s_1, with the corresponding conditional probability tables reflecting these dependencies. Figure 2.31 is a diagram representing the same network but with an additional node s_4. The only difference with respect to the previous BN is that s_1 is also influenced by s_4. In this diagram s_4 and s_3 are conditionally independent since there are no arcs connecting them (s_4 and s_2 are also independent); this means that the probabilities of s_3 and s_2 are not affected by s_4. Equations (2.35) and (2.36) relate to Figure 2.30 and Equations (2.37) and (2.38) to Figure 2.31.

Dynamic Bayesian Network

BNs can be also used to model sequences [87], this type of BN is called Dynamic BN (DBN) [41]. In DBNs the probability of a node at time t might be dependent on its probability at $t-1$, as shown in Figure 2.32. Therefore:

$$p(x_t) = p(x_t \mid a, x_{t-1})p(a, x_{t-1}) \tag{2.39}$$



		s_1	
s_3	s_2	T	F
T	T	0.8	0.2
T	F	0.6	0.4
F	T	0.7	0.3
F	F	0.3	0.7

	s_2	
s_3	T	F
T	0.9	0.1
F	0.3	0.7

s_3	
T	F
0.8	0.2

Figure 2.30: Simple Bayesian network with conditional probability tables.

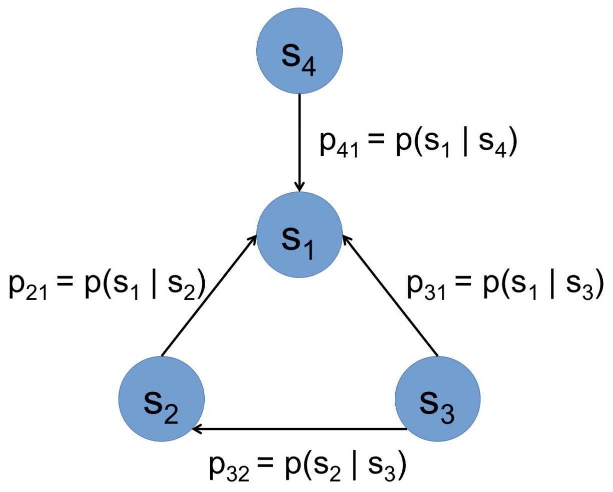


Figure 2.31: Bayesian network with independent variables.

$$p(s_1) = p(s_1 \mid s_2, s_3)p(s_2, s_3) \quad (2.35)$$

$$p(s_2) = p(s_2 \mid s_3)p(s_3) \quad (2.36)$$

$$p(s_1) = p(s_1 \mid s_2, s_3, s_4)p(s_2, s_3, s_4) \quad (2.37)$$

$$p(s_2) = p(s_2 \mid s_3)p(s_3) \quad (2.38)$$

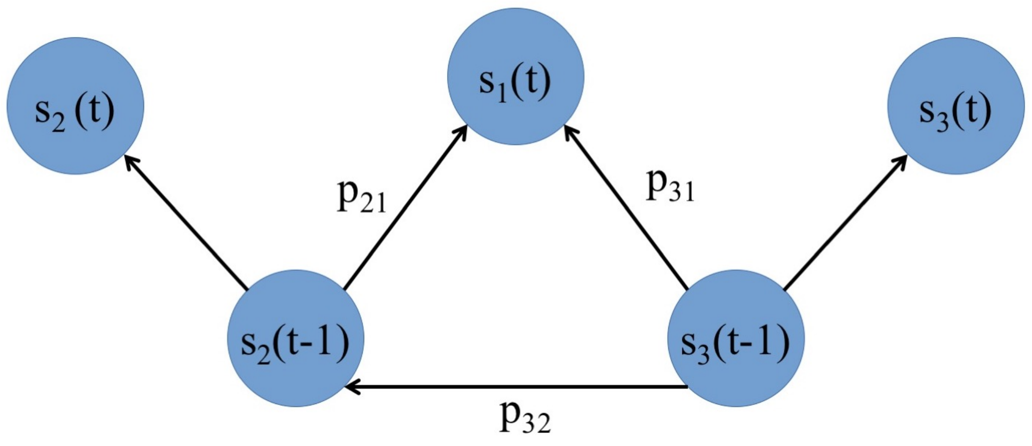


Figure 2.32: Dynamic Bayesian Network.

DBNs can be used to represent different models [88] such as Kalman Filters [89] and Hidden Markov models [90].

2.6.6 Gaussian Mixture Model

A Gaussian mixture model (GMM) assumes that data points are generated from a linear superposition of Gaussian distributions [91], as illustrated in Figure 2.33. Using the formulation of [41], a Gaussian distribution for a D dimensional vector $\mathbf{x}$ is specified as:

$$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \tag{2.40}$$

where $\boldsymbol{\mu}$ is the D-dimensional mean vector, $\boldsymbol{\Sigma}$ is a $D \times D$ covariance matrix, and $|\boldsymbol{\Sigma}|$ is the determinant of $\boldsymbol{\Sigma}$. From this, the GMM is given as:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.41}$$

with K Gaussian distributions $\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, each of which is a component of the GMM with its individual mean $\boldsymbol{\mu}_k$ and covariance $\boldsymbol{\Sigma}_k$. Each π_k takes a real value and is the mixing coefficient, representing the contribution of component k to the GMM.

In the work that will be described later, GMM will be used to model the probability distribution of ball impact locations in the continuous space of the tennis court.

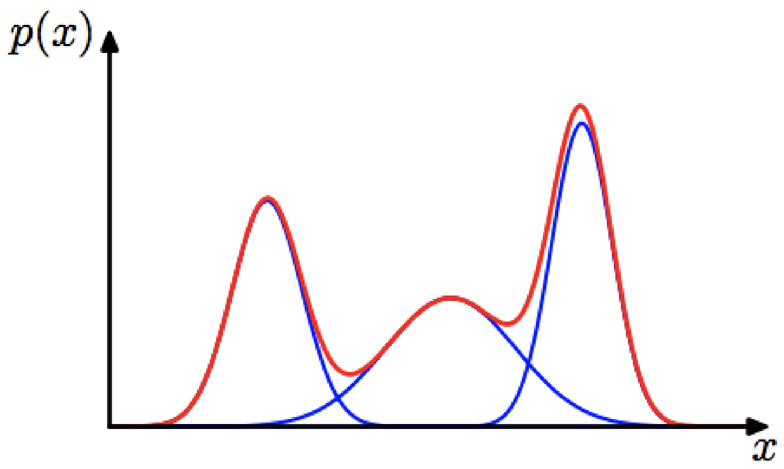


Figure 2.33: One dimensional Gaussian mixture distribution (red) with three components (blue), from [41].

Chapter 3

The Arrival of a New Player: Visual Data in Tennis Analysis

In the previous chapter, we have reviewed classical approaches for modelling tennis (Section 2.3) and we have also seen how interest in using spatio-temporal information for this task is rising. This chapter focuses on critically reviewing current research in this direction, and our contribution is two-fold. First, we address the lack of comprehensive surveys that include different approaches to incorporating spatio-temporal data in the analysis of tennis. Second, we discuss the limitations of present work to motivate the need for a framework like the one proposed in this thesis. This chapter relates to the Modelling Layer of our framework (cf. Figure 1.11).

We review research in spatio-temporal-based tennis analysis from two different perspectives: knowledge discovery and prediction. By knowledge discovery, we refer to finding patterns and depicting tactics and strategies, while prediction is about estimating future events such as the location or type of a serve or the next shot and even who will win the match. The aim of this chapter is to show the motivation to our the research and applicability of our framework in addressing current limitations of state-of-the-art tennis analysis. This is why this chapter focuses on reviewing methods of current analysis rather than applying the reviewed techniques to our own data.

3.1 Why Use Visually Derived Spatio-Temporal Data?

Professional tennis players are able to quickly assess a situation and make decisions based on their domain knowledge and current environment. In receiving from a serve at 200km/h, the opponent has 500ms [92] to predict the ball trajectory, position himself/herself and execute a stroke. It has been shown that professional tennis players have enhanced anticipation skills in predicting the ball landing location of incoming shots [93] and higher visual perception abilities, with respect to for example speed discrimination [94]. Therefore, to understand and predict tennis players' decisions, it seems natural to analyse the visual cues available to them.

When using coarse-grained event statistics, as shown in Section 2.3, one can predict the outcome of a match by building a model that uses the probability of winning a point as its building block. However, using visually derived spatio-temporal data can enable us to understand what happens at a finer-grained level and develop models for in-play analysis and prediction. From this, new insights into players' strengths, weaknesses and strategies can be obtained. Prediction models can start from predicting the type and location of the next shot and estimate the probability of winning a point, revealing how it evolves as the point progresses. This can help in understanding the contribution of each shot to winning a point. Finally, it can also lead to an in-play prediction of the probability of winning the set, game and finally the match.

3.2 Challenges

Incorporating spatio-temporal data to the modelling of tennis has many advantages but also poses significant challenges:

- Spatio-temporal data such as ball trajectory or player position are continuous in time and space. It is challenging to manipulate and extract salient features from this continuous data.
- The player and ball positions have to be merged with other data such as score or type of shot, and ways in which to combine this multi-modal information have to be investigated.

- The availability of spatio-temporal data in tennis is of more restricted access than that of coarse-grained statistics, which are publicly available on the Internet. When available, the amount of spatio-temporal data per player, surface or tournament is highly limited.

- The accuracy of spatio-temporal data is harder to assess. Hawk-Eye is considered extremely accurate but still has its detractors [95; 96], and it is risky to trust other sources of manually annotated data (due to the reasons exposed in Section 1.1) or from data collection systems that do not provide quantitative evaluations (cf. Chapter 4).

3.3 Knowledge Discovery

One of the primary goals in tennis analysis is finding patterns to reveal strategies and discover the strengths and weaknesses of players. In 1996, the United States Tennis Association published a book listing 58 tennis tactics patterns [97], listed in Appendix A. These were found empirically by professionals in the field. Spatio-temporal data is a great asset to uncover these patterns automatically and even find new ones. Such data is very rich but also hard to handle, due to the factors listed above, and this is where knowledge discovery comes to the fore. Finding ways in which to cluster or represent this data is crucial, and this section outlines the most relevant approaches that have been proposed. An overview of the timeline of advances in tennis knowledge discovery from spatio-temporal data is shown in Figure 3.1, including a reference to the work of Intille and Bobick in American football [98; 99], from which subsequently presented approaches to spatio-temporal tennis data analysis originate.

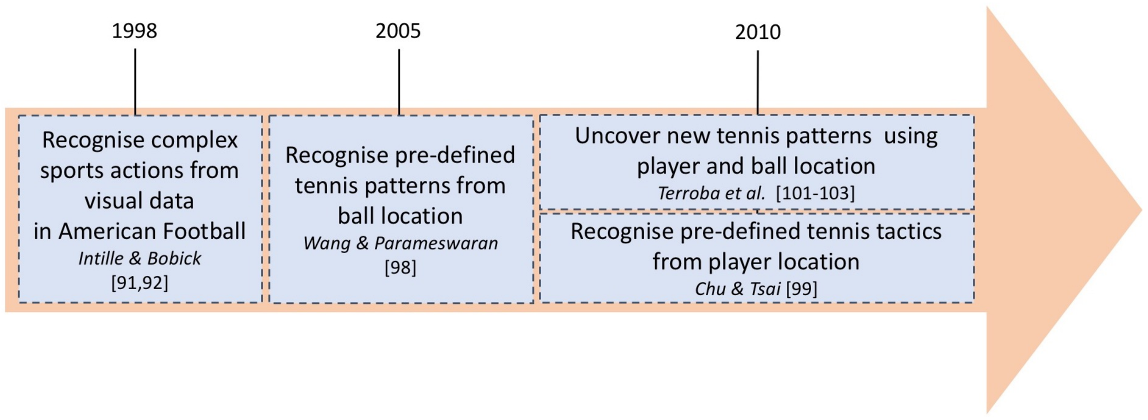

1998
2005
2010
Recognise complex sports actions from visual data in American Football
Intille & Bobick [91,92]
Recognise pre-defined tennis patterns from ball location
Wang & Parameswaran [98]
Uncover new tennis patterns using player and ball location
Terroba et al. [101-103]
Recognise pre-defined tennis tactics from player location
Chu & Tsai [99]

Figure 3.1: Timeline of advances in tennis knowledge discovery from spatio-temporal data.

3.3.1 The Start

One of the earliest work using spatio-temporal data in sports analysis dates back to 1998. At that time, Intille and Bobick [98; 99] presented work to recognize complex multi-agent actions in American Football from visual data. Their work was inspired by advances in object recognition in computer-vision. In 1990, Grimson proposed a model for object recognition consisting of using image features to search a tree, where nodes represented object edges and leaf-nodes represented the objects themselves [100]. A major drawback of this approach is that its complexity makes exhaustive search intractable. To approach this issue, Grimson and Lozano–Perez developed the idea that 'massive low order consistency typically implies correctness' [101] and this is exploited in Intille and Bobick's approach for complex action recognition. They represent each complex multi-agent action (e.g. curl or catch pass[1]) with a temporal graph that is pre-defined using domain knowledge, as shown in Figure 3.2. The main building blocks for this graph are:

- Belief nodes: these are Bayesian Networks, explained in Section 2.6.5, for predicting single-agent goals from visual data, such as a pass thrown from a given player. The visual data used is based on player location and includes velocity, position relative to other players or to meaningful court landmarks. These are denoted by a B in Figure 3.2 and G is the main belief node.
- Evidence nodes: these are directly evaluated from the data and describe temporal, spatial and logical relationships between agents and objects. In Figure 3.2 they are denoted by E.

With this approach, the authors correctly recognise 21 out of 25 plays evaluated from manually annotated visual data of player positions. In contrast to tennis, this work was done in a team sport in which player interactions are more complex, and their formation is a substantial factor into understanding strategies. Nonetheless, as we will see, using a Bayesian Network from visual information to uncover patterns of play can also be successfully applied to tennis.

[1]American Football patterns.

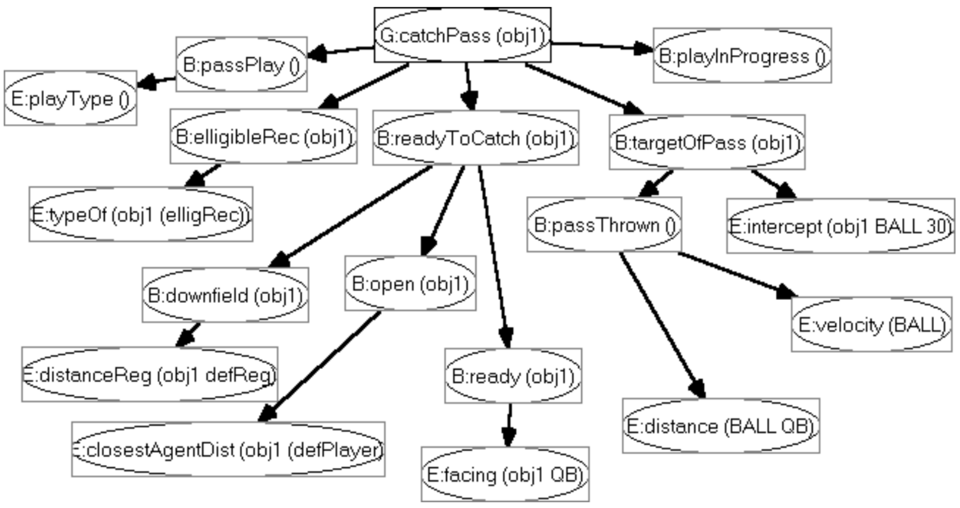


Figure 3.2: Visual Network for catch pass, from [99]. B stands for Belief node, G for the main Belief node (Goal) and E for Evidence nodes.

3.3.2 Finding Patterns in Tennis

Strategy is defined as 'A plan of action designed to achieve a long-term or overall aim' [102]. According to a publication from the International Tennis Federation, strategy in tennis can be defined as 'the overall game plan for a certain match' [103] and 'the practical application of the strategy during the match' is referred as tactics. A good tennis strategy will use a game plan that takes advantage of the player's own strengths and the opponent's weaknesses [104]. Tactics used to achieve a particular strategy are achieved through patterns of play [97]. Therefore, understanding and dissecting tennis tactics corresponds to finding patterns of play and analysing them taking into consideration whether they led to losing or winning a point. These patterns are the repetition of a particular sequence of tennis shots and finding them is what we address in this section.

In 2005, Wang and Parameswaran [105] used ball information alone to classify tennis shots into the 58 official patterns mentioned earlier, and listed in Appendix A. In their approach, two separate Bayesian Networks (BNs) were used to cluster:

- Ball landing positions into 9 areas, as shown in Figure 3.3.
- Trajectories into 10 clusters: 5 for forward plays and 5 for backward plays, as shown in Figure 3.4.

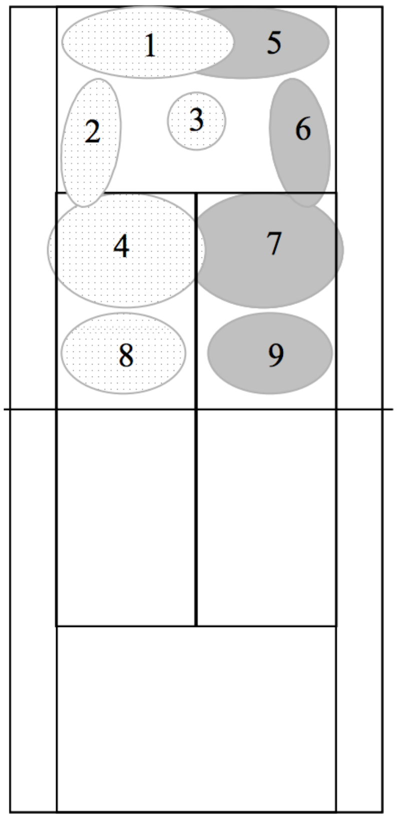


Figure 3.3: Ball landing positions clusters, from [105].

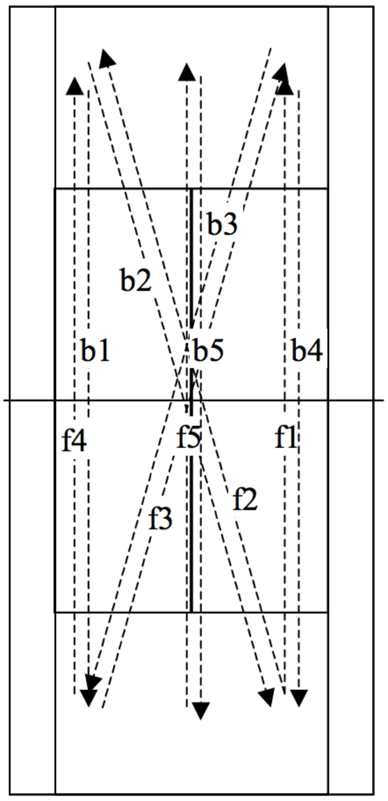


Figure 3.4: Ball trajectory clusters, from [105].

Interestingly, an additional BN is used at a higher-level and incorporates both ball landing position and trajectories, since the two are correlated, and this BN works in tandem with the low-level ones to cluster ball landing position and trajectories. Using this technique, each of the 58 pre-defined tennis rally patterns can be formulated as a sequence of trajectories and ball landing positions, represented by their cluster numbers. The authors provide an empirical evaluation of two patterns tested and classified correctly.

Later on, in 2010, Wei-Ta Chu and Wen-Ho Tsai [106] used player position to find patterns without incorporating the ball position, contrasting with the previous work from [105]. The authors argued that such data is easier to extract from videos and sufficient for the task. Also different from the previously described work that aimed at recognising pre-defined tennis patterns, the authors in this paper [106] are interested in discovering tactics in addition to identifying patterns of play. The tactics targeted in their work are: passing ball, moon ball, drop shot, volley and unforced error.

A two-level framework is developed to accomplish this objective: first, pattern detection and second, tactics analysis. In this framework, a symbolic representation is defined for the last 5-second segment of play, noting player positions every 10 frames. This symbolic representation corresponds to the concatenation of the following attributes, for each player and each consecutive pair of frames:

- Location at frames $t-1$ and t.
- The direction of movement of each player, with the discrete values 'left', 'right' or 'still' for horizontal movement and 'up', 'down' or 'still' for vertical movement (towards or away from the net).
- The speed of movement of each player in the horizontal and vertical directions split into three categories: 'fast', 'medium' or 'still'.

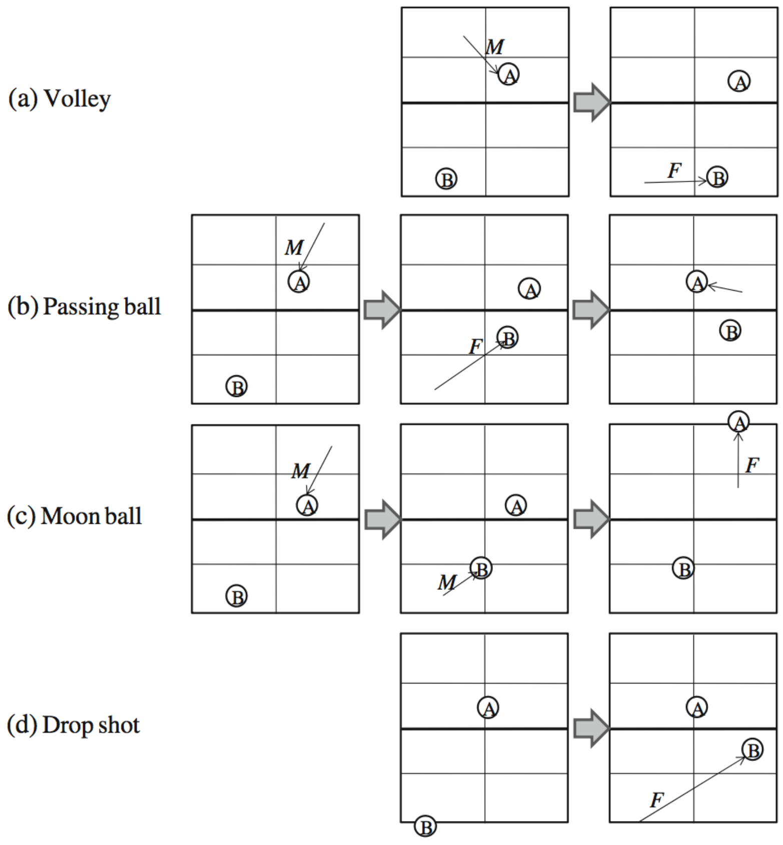


Figure 3.5: Examples of moving patterns for different tactics, from [106]. F: fast, M: medium, A: player A and B: player B.

At the second-level representation, each pair of possible combinations of attributes in the symbolic representation is assigned a meta-symbol, to represent the attributes of both players involved.

Once this framework is set, the next step consists in learning the association between sequences of meta-symbols and pre-determined tactics: passing ball, moon ball, drop shot, volley and unforced error; Figure 3.5 illustrates examples of moving patterns for these techniques. With that objective in mind, 30 example plays are manually retrieved for each tactic and transformed into meta-symbol sequences; these are used to represent a tree for each tactic. This representation allows selecting which meta-symbols occur frequently enough to be considered characteristic of a given tactic. After this step, each tactic has some 'template' sequences of movement associated with it, and new patterns are recognized using a dynamic programming approach to find the Largest Common Subsequence [107].

The evaluation was performed over 10 tennis matches comprising 7 distinct tactics and an F-measure[2] of 0.74 was obtained. These results are not straightforward to analyse since the authors present a full pipeline from player detection in a video to pattern matching, and errors in player detection may cause errors in tactic analysis. In fact, the authors mention that pattern matching on clay-courts led to poorer results probably because court-line detection is harder on this surface [106]. In general, the framework proposed is interesting and could be highly valuable as it would be possible to analyse which tactics lead to winning a point, game, set or match. Regarding the game analysis produced in this paper, however, the main insight is that 'The player who issued fewer unforced errors won the game'. This is widely accepted as being true for tennis.

Also in 2010, Terroba et al. proposed a framework for pattern discovery in tennis [108–110]. In contrast to [105], the authors try to discover patterns through data mining rather than classifying pre-determined patterns. In their papers a *rally* is formalized as a sequence of *events* (single strokes) defined by 7 attributes: identity of player (1 or 2) hitting the ball, stroke type (10 different stroke types), positions of each player (as a pair of integer coordinates) and

[2]A combined measure of precision and recall; the highest possible score is 1.

ball speed after the stroke (as ‘normal’, ‘slow’ or ‘fast’). A similarity measure between events is introduced to handle such multi-modal data, and the similarity between two events is defined as the sum of the similarity functions applied to each attribute of the event (e.g. the similarity function for the player position between two events is their Euclidean distance). This results in a similarity score that is used to solve the pattern mining problem.

Once this framework is set, the aim of the paper is to identify the most frequent sequences in general and for a particular player. The authors calculate the support for a given template sequence in a match by looking at how many sequences in that match are similar to it. A *minimum support* threshold is established to determine the support required by a template sequence for it to be considered ‘frequent’. By tuning the different thresholds, the user can decide how general the frequent patterns can be. Using this method, the authors analyse 7 hours of video containing around 3 000 events and find interesting common tennis patterns. Figure 3.6 shows service patterns mined from the match between Na Li and Serena Williams at the 2010 Australian Open semifinal [109].

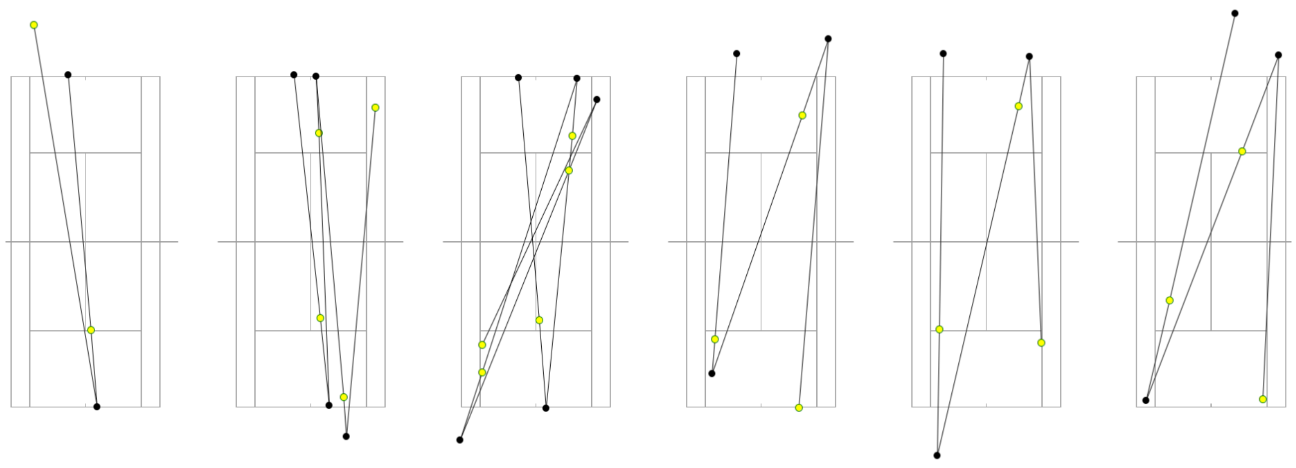


Figure 3.6: Successful service patterns from 2010 Australian Open semifinal, from [109]. Black dots are the player positions when hitting the ball and yellow dots are ball bounce positions.

3.3.3 Knowledge Discovery Until Now

The work that we presented shows that a common approach is to discretize data such as ball speed into ‘slow’, ‘medium’ or ‘fast’, or player position into court areas. By discretizing continuous data, the search space becomes tractable but some information is lost. This opens the question on which data should and which should not be discretized. The experiments discussed in this section are not sufficient to determine this and further research should be completed to answer that question. Also, if discrete information is sufficient to understand the strategies of players, this implies that any spatio-temporal data could contain a small error without interfering with the final classification result in most cases.

Another interesting line of future investigation arises from the knowledge discovery studies discussed here: in order to evaluate a player's strategy and performance, is it better to use pre-defined patterns, find them automatically or investigate how to combine the two? As above, further experiments are required to compare these different strategies.

3.4 Prediction

As mentioned in Section 3.3.1, Initille and Bobick presented in 1998 what is considered by many to be the seminal work in spatio-temporal based sports analysis [98; 99]. However, it was not until 2013 that spatio-temporal based prediction in tennis really took off. That year, the first paper that analysed Tennis Hawk-Eye data appeared, from the collaboration between researchers at the Queensland University of Technology, Disney Research and the Australian Institute of Sport [111]. More publications by the same authors followed in the following years up to this day. They represented a breakthrough in the prediction of tennis events, and while investigating prediction, knowledge discovery also emerged. This series of papers will be reviewed in this section. They all use data collected by Hawk-Eye from the 2012 Australian Open Men's draw (2012 – 2014 for [24]) and their main achievements, shown in the timeline of Figure 3.7, are:

- Predict the type of the next shot – Section 3.4.1
- Predict the location of the next shot – Section 3.4.2
- Predict the style and point of impact of serves – Section 3.4.3
- Predict the outcome of a shot – Section 3.4.4

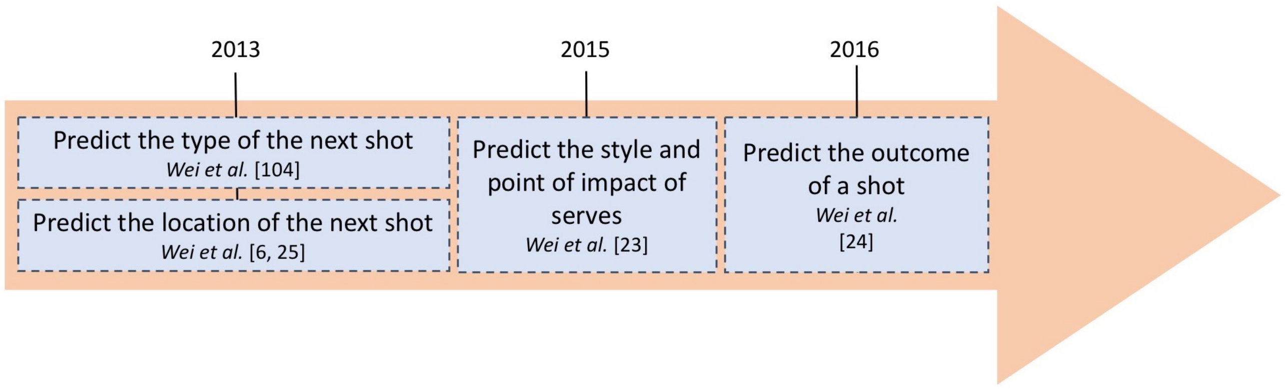

2013
2015
2016
Predict the type of the next shot
Wei et al. [104]
Predict the location of the next shot
Wei et al. [6, 25]
Predict the style and point of impact of serves
Wei et al. [23]
Predict the outcome of a shot
Wei et al. [24]

Figure 3.7: Timeline of advances in tennis prediction using spatio-temporal data.

3.4.1 Predict the Type of the Next Shot

In 2013, a Dynamic Bayesian Network (DBN, explained in Section 2.6.5) was used to model the sequence of events in a tennis point[3] and predict the type of the next shot (returning shot, winner or error) [111]. In such a model the probability of the next shot $\mathbf{z}^t$ at time t of being of a certain type given the type and Hawk-Eye obtained characteristics of the previous shot ($\mathbf{z}^{t-1}$ and $\mathbf{x}^{t-1}$ respectively) can be obtained through Bayes' rule for a sequence of states:

$$P(\mathbf{z}^t \mid \mathbf{z}^{t-1}, \mathbf{x}^t) = \frac{P(\mathbf{x}^t \mid \mathbf{z}^t)P(\mathbf{z}^t \mid \mathbf{z}^{t-1})}{P(\mathbf{x}^t \mid \mathbf{z}^{t-1})} \tag{3.1}$$

However, we know that the previous state $\mathbf{z}^{t-1}$ can only be a returning shot. Otherwise, the point is over and $\mathbf{z}^t$ would not even exist. Since the previous state is always the same (a returning shot), the next state is not conditioned on it. Therefore, the DBN can be simplified into a Bayesian Network (BN), and the previous formulation becomes:

$$P(\mathbf{z}^t \mid \mathbf{x}^t) = \frac{P(\mathbf{x}^t \mid \mathbf{z}^t)P(\mathbf{z}^t)}{P(\mathbf{x}^t)} \tag{3.2}$$

This is calculated for each possible state $\mathbf{z}^t$ ($\mathbf{z}^t_{return}$, $\mathbf{z}^t_{winner}$ and $\mathbf{z}^t_{error}$), and the one with highest probability is selected as the predicted state.

[3]Excluding the serve.

The authors experimented with different subsets of Hawk-Eye features to find out which characteristics are best predictors of the next shot. According to their results, the combination of speed, impact location of the shot and the location of the players' feet leads to the best prediction: 68.52% AUC[4] for winners and 76.09% for errors. Including the number of shots in a rally worsens performance and authors believe that this is due to the high variance of this parameter. Another possible explanation is that for longer rallies, the outcome of each shot is less predictable as the influence of the serve diminishes, as shown in [7].

To further increase the prediction accuracy, the authors explore adaptive model techniques that take into account the opponent's behaviour. They withhold data from a single given opponent, and the model trained in two phases: first using non-withheld data and then refined with the withheld data. The withheld data does not contain feet locations, so it must be compared to the non-adaptive model that also excludes feet location. When doing that, using speed and impact location, accuracy improves from 63.62% to 77.28% for the prediction of winners and from 61.12% to 71.85% for errors prediction. We can expect that prediction including players' feet location will behave similarly and improve with the adaptive model.

3.4.2 Predict the Location of the Next Shot

After that, the authors predict the next shot location using the type of the previous shot and its speed, location and angle [6; 26]. This is considerably more challenging than predicting the next type of shot because the output space for shot type is small and discrete while locations are continuous.

This is achieved using a local clustering method to select historical shots that were preceded by shots with similar speed, location and angle to the incoming shot and their impact position represented in a 2D GMM[5] as shown in Figure 3.8b. Separately, each half-court is divided into 4 areas (plus an area outside this region), and a Dynamic Bayesian Network (DBN) is used to find the probabilities of the shot falling in each of these discrete areas as shown in Figure 3.8a.

[4]Represented as the area under the curve plotting correct classification rate against false positives rate, with the perfect score being 1.

[5]Gaussian Mixture Model, explained in Section 2.6.6

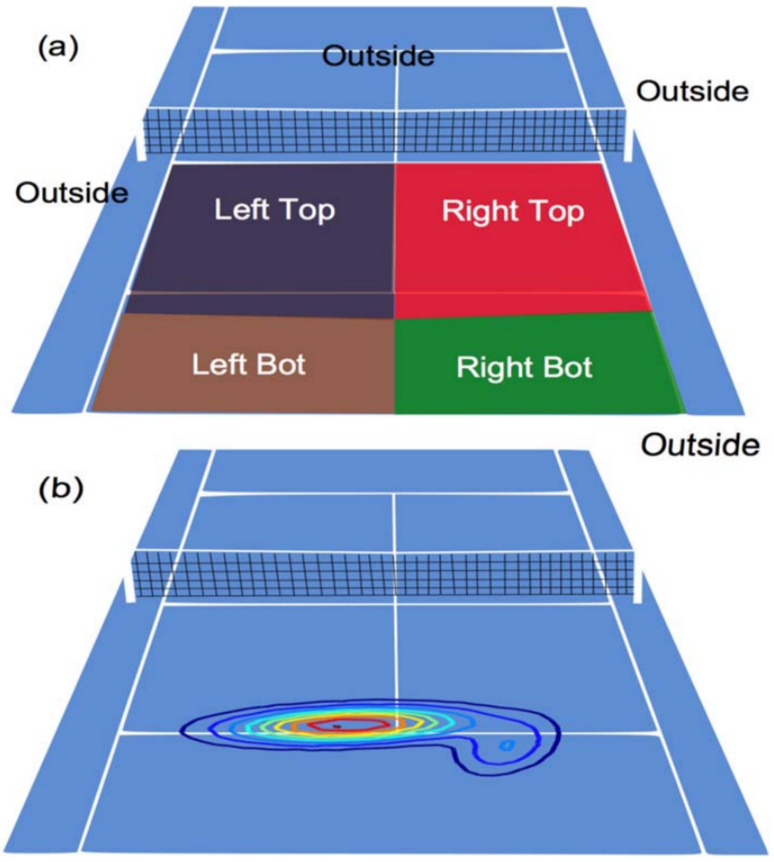


Figure 3.8: Prediction of shot locations based on zones (a) and continuous regions (b), from [6].

The weights of the GMM are adjusted with these calculated probabilities and an example the results is shown in Figure 3.9. Finally, two adaptive model techniques are used to improve accuracy: (1) pre-game adaptation is used in the same manner as described in Section 3.4.1 and (2) on-line adaptation is implemented by including data from shots of the current match.

Regarding results, the model without adaptation performed best when using speed, start location and player position – as for the prediction of shot type – reaching an average 74% AUC over the five zones. Using pre-game adaptation increased the results by 3% – 7.4% (except for one of the zones, for which prediction success stayed the same). As expected, the performance of the on-line adaptation model increased with time, the AUC going from an average of 68.52% in set 1 up to 82% average in set 5. The continuous-output predictions without adaptation had an error of 1.7m for two of the players and 2.3m for the other one on average.

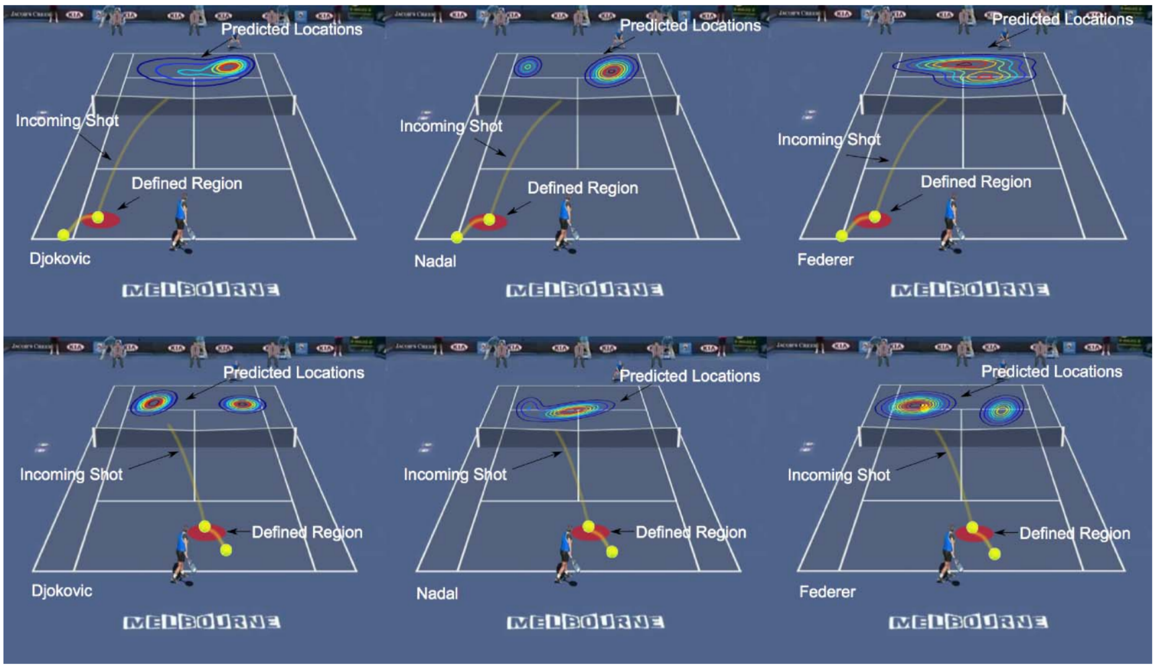


Figure 3.9: Prediction of shot locations, from [6].

3.4.3 Predicting the style and point of impact of serves

The work presented so far is only concerned for predictions in the rally and excluding the serve. However, the importance of the service for winning a tennis match is widely acknowledged [112] and it is the focus of the next piece of work in the series of papers using Hawk-Eye data presented here [24]. The main contribution of that paper is the discovery of players' serve styles and, based on this, the prediction of a service type and location dependent on the current match context.

Serve Styles

First, each service is classified using manually defined classes based on whether the service was valid and the direction of the service, to advantage or deuce court. Then, serves within each category are clustered into 7 groups using the unsupervised k-means clustering technique, described in Section 2.6.3. For this, each service is divided into two trajectories – before and after bouncing – and each trajectory is represented as the ball's locations and dynamics at 100 positions along the trajectory. This is illustrated in Figure 3.10 but showing only 3 points per trajectory instead of 100 for clarity purposes. The distance between two serves is calculated as the mean of the pairwise Euclidean distance amongst the 200 points representing each serve, 100 for the serve trajectory before bouncing and 100 for the trajectory after the bounce. From this, 7 styles of serve were uncovered on each side of the court, 14 in total, Figure 3.11 shows clusters for serve trajectories on one-half of the court. Once clusters of serve trajectories are established, the authors define a player's serving style in terms of a normalised histogram of each of the 14 types of services, as shown in Figure 3.12.

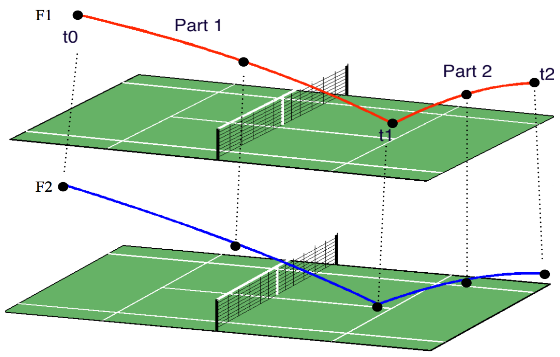


Figure 3.10: Representation of two serve trajectories with 3 points per trajectory, from [24].

Serve Prediction

Prediction is performed as a classification task in which the objective is to predict the cluster to which the next serve will belong, amongst the 7 possibilities for either ad or deuce court. This is performed with a random forest classifier (see Section 2.6.2). The best results are achieved using feature vectors that capture the match score, and the serve styles of the player who is serving and that of his/her opponent. This technique achieves 27.8% accuracy with respect to predicting the cluster to which the next serve will belong.

The authors provide examples of the insights obtained from their work, and this is an excerpt to illustrate it: *'For Nadal, in normal situations his main serve tends to be down the line. In break-point situations, however, almost half the time (48%) he serves wide of the court which is totally different. Djokovic, on the other hand, does the opposite, in normal points he tends to serve wide, but in break-points, he goes down the middle'* [24].

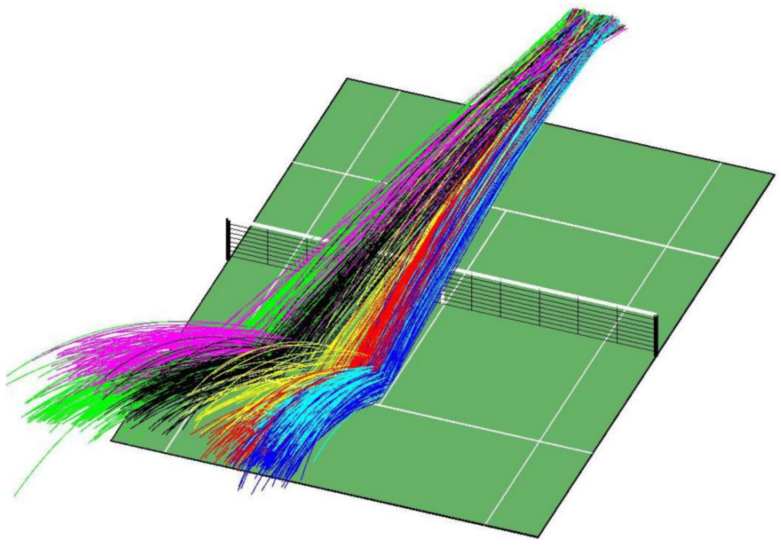


Figure 3.11: Serve clusters for the advantage court, from [24].

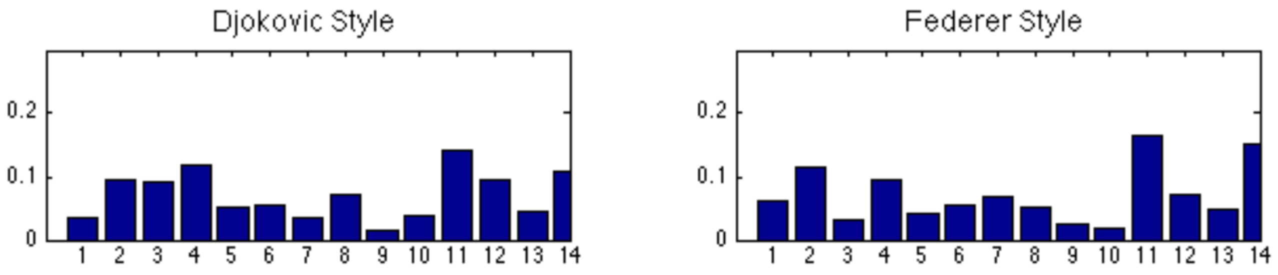


Figure 3.12: Serve style histogram for Djokovic and Federer, from [24].

3.4.4 Predicting the Outcome of a Shot

In 2016 a method to predict the outcome of a point using player's style and match context was presented [25]. Predicting the outcome of a point is achieved by predicting whether the next type of shot will be a winner or an error, which corresponds to a shot type classification task. This is performed using a random decision forest classifier (see Section 2.6.2). The shot features used by the classifier are:

- Angle, maximum height, average speed and instantaneous speed of the shot.
- Location of each player at the start and end of each stroke.
- Ground Stroke Speed Ratio: ratio between the ball speed of successive shots.
- Ground Stroke Weight Ratio: the ratio between the distance of the player making the

current shot to the baseline and that of the opponent on the previous stroke.

- Lateral Player Movement Ratio: ratio of lateral distance covered by a player between successive strokes.

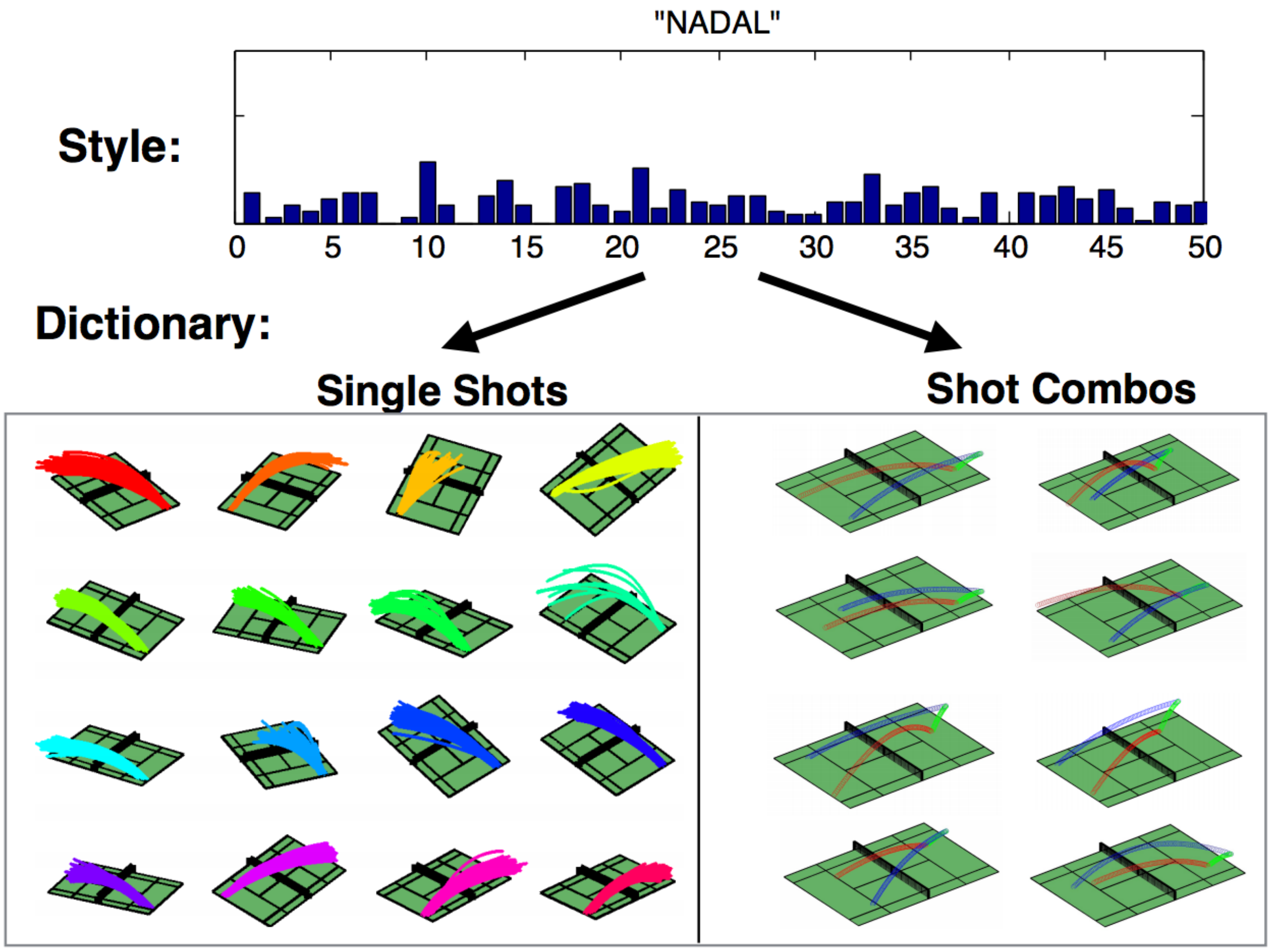


Figure 3.13: Style histogram for Nadal, from [25].

Then, this simple model is enhanced using player style and context. A player's style (*style priors*) is defined using a histogram of single shots and shot combinations, and the context corresponds to specific characteristics of the game such as score, court surface and other external factors such as temperature.

The players' style is defined similarly to the previously described work on serve style [24]. A shot is characterised by sampling a number of points along its trajectory, and a dictionary is learnt by clustering these using a K-means approach. Building the shot dictionary is based on:

- Using assignment of single shots/shot combinations into items of the dictionary and minimising their reconstruction error.

- Minimising classification error when predicting the winner of a point using dictionary words.

As in the previous section [24], once clusters of shots were established, the authors defined a player's style as a normalised histogram for each type of shot and shot combination, as illustrated in Figure 3.13.

The authors evaluate their method in a test set of 9 349 shots and assess the performance of their prediction of who will win a point using only shot features and also using style and/or context. Incorporating style and context features yielded the best results, for which they achieved a 0.38 Root Mean Squared Error[6] when including shot index, set score and point score information. Figure 3.14 illustrates how the point winning probability evolves as the point progresses.

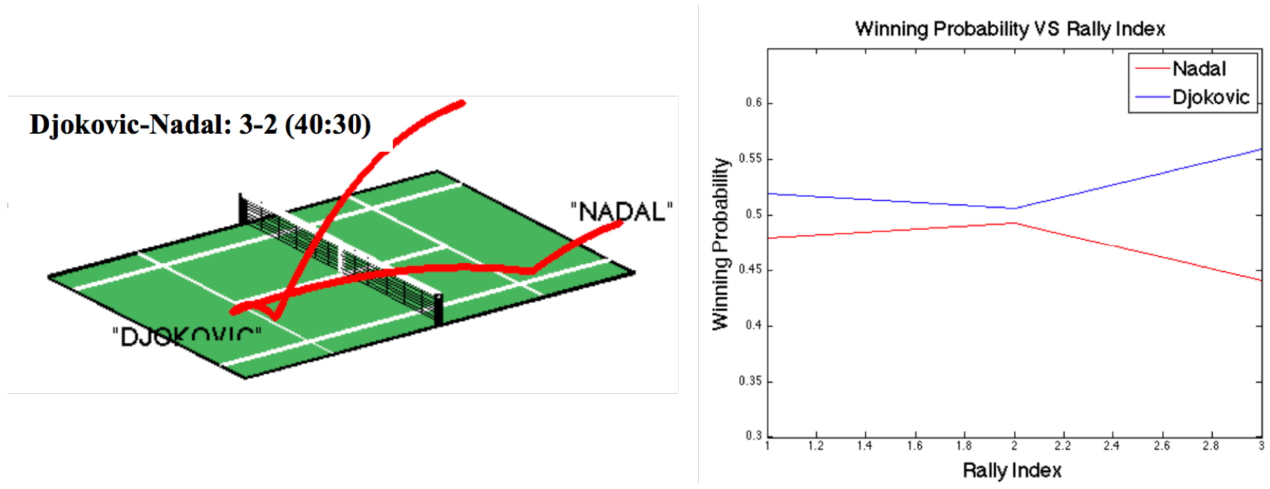


Figure 3.14: Point winning probability with style and context descriptors, from [25]. Rally index is the shot index (first, second etc.) in the rally excluding the serve.

[6]This is calculated as $\sqrt{p(1-p)}$, with p the confidence in the prediction that the player who won the points would do so and p(1-p) the variance for a binomial distribution.

3.5 Current Limitations

The work discussed in this chapter provides ways to find tennis patterns from spatio-temporal data obtained from videos, whether the patterns be new ones or pre-defined. We also have seen how spatio-temporal data can be used to represent a player's style and make predictions at the shot level. Previous work into the analysis of tennis from spatio-temporal data is promising but also has some limitations:

1. Methods using manually annotated data, such as [105], cannot be widely applied. Firstly, as we mentioned in Section 1.1, there is a limitation in the precision of data that can be extracted manually, as the human visual system cannot record the exact bounce location at the mm level. Secondly, the data so obtained can be biased by the annotators' mental state or their previous knowledge of the players. Finally, annotators must have expertise in the field to produce accurate results. The availability of qualified people and the cost associated with it limits the number of matches that can be realistically analysed.

2. Methods that implement automatic data collection evaluate their results in different ways, most of the time in a qualitative manner (explained in Chapter 4). The lack of benchmark datasets and evaluation protocols impedes the comparison between techniques and the assessment of their accuracy.

3. The prediction work that we presented in Section 3.4 is based on Hawk-Eye data. Hawk-Eye is currently only installed for major tournaments, and only researchers at Australian research institutions have access to data collected at the Australian Open. Prediction models adapted to the opponent need data from the two specific players, further reducing the amount of data available. Therefore, the models presented here that adapt to the opponent can only be applied to players that competed in most matches in the tournament (Djokovic, Federer and Nadal) and can only be adapted using data from that tournament, since no other Hawk-Eye data is available to the researchers [6; 24–26; 111]. In addition to this, researchers using Hawk-Eye data have access to the information collected by the system but have no control for obtaining additional data. For example, Hawk-Eye

does not currently perform action recognition, and this is not used in models that use Hawk-Eye data, but researchers have no way to implement action recognition for this data.

4. The lack of a unified framework for knowledge discovery and prediction hampers the comparison of results obtained from different techniques. Because different researchers use different data extraction techniques, different sources of data (different tournaments, for example), different data analysis techniques (shots belonging to clusters, exact impact location of a shot or play patterns defined differently) it becomes challenging to compare performance between different approaches. In our opinion, this reflects how the field is still in an exploratory phase.

5. This chapter has shown promising directions of research for knowledge discovery and prediction in tennis using spatio-temporal data. However, there is still need for research in understanding how to translate these results into the real-world and have an impact on changing play and training strategies for tennis players.

Figure 3.15 shows how the limitations described above fit in the context of spatio-temporal analysis of tennis, incorporating the research described in this chapter. As can be seen, limitations 1, 2, 3 and 4 can be addressed by a unified framework for the automatic extraction and analysis of spatio-temporal tennis data, and this is our focus in the remainder of this thesis.

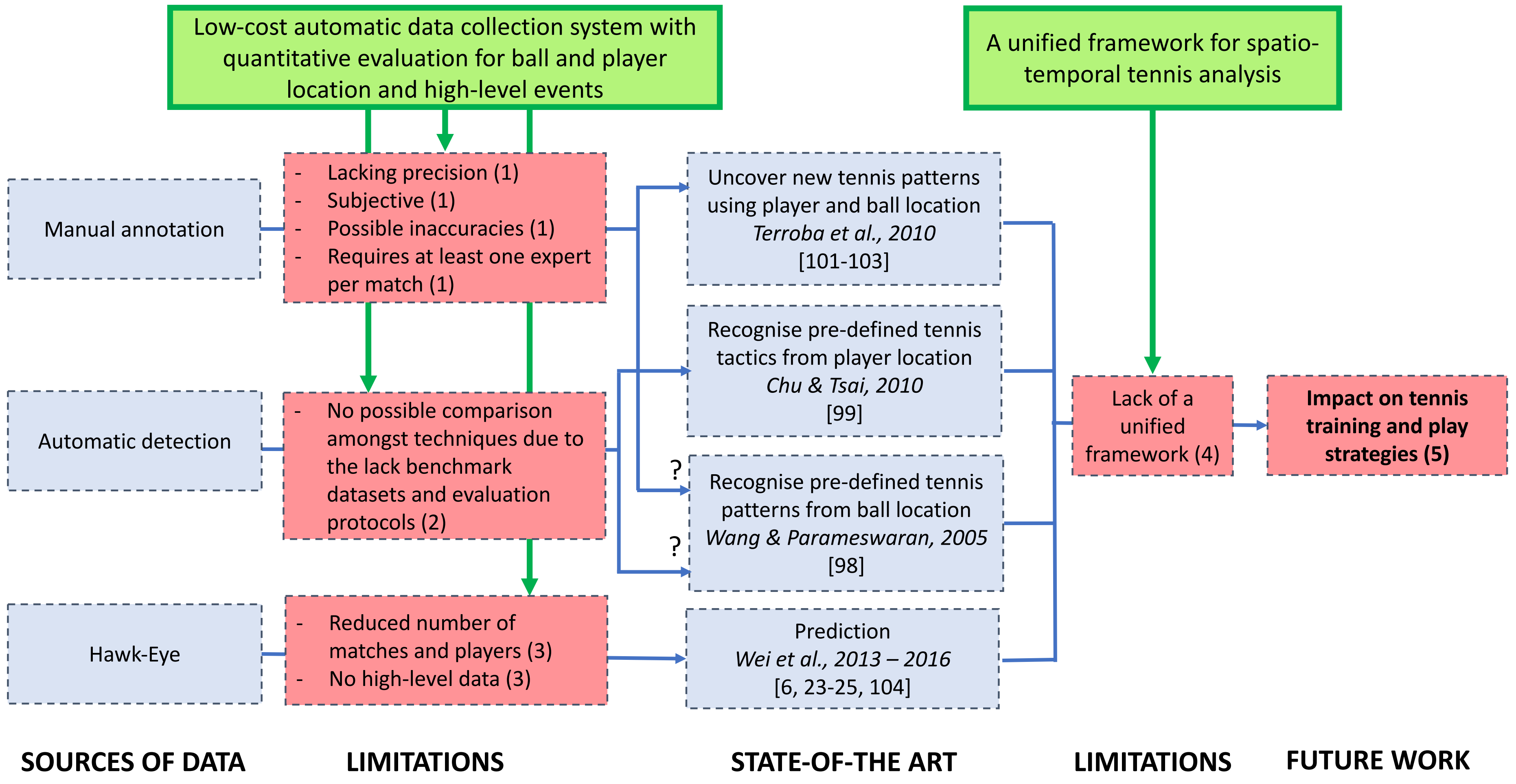

Low-cost automatic data collection system with quantitative evaluation for ball and player location and high-level events
A unified framework for spatio-temporal tennis analysis
Manual annotation
Automatic detection
Hawk-Eye
- Lacking precision (1)
- Subjective (1)
- Possible inaccuracies (1)
- Requires at least one expert per match (1)
- No possible comparison amongst techniques due to the lack benchmark datasets and evaluation protocols (2)
- Reduced number of matches and players (3)
- No high-level data (3)
Uncover new tennis patterns using player and ball location Terroba et al., 2010 [101-103]
Recognise pre-defined tennis tactics from player location Chu & Tsai, 2010 [99]
?
?
Recognise pre-defined tennis patterns from ball location Wang & Parameswaran, 2005 [98]
Prediction Wei et al., 2013 – 2016 [6, 23-25, 104]
Lack of a unified framework (4)
Impact on tennis training and play strategies (5)
SOURCES OF DATA
LIMITATIONS
STATE-OF-THE ART
LIMITATIONS
FUTURE WORK

Figure 3.15: Data sources and limitations for state-of-the-art tennis spatio-temporal analysis. In blue, state-of-the-art research papers and the data sources used in them. In red, current limitations and work that is yet to be developed, with an index relating to the previous limitations list. In green, our contribution and its effect within this context. ‘?’ is used because the work referenced mentions the use of ‘labelled videos’ without specifying how they were labelled.

Chapter 4

Seeing Like a Coach: Real-Time Data Collection

In the previous chapter, we have seen how the analysis of spatio-temporal data can lead to useful in-play prediction at the shot level and bring insights into players' tactics and style. We have also observed that most limitations for research in this area originate from a lack of widely available data. With the objective to overcome such a challenge, we have developed a low-cost system for collecting spatio-temporal tennis data in real-time. This chapter focuses on the hardware and software components of our system, the algorithms developed for our technology and the analysis of their performance. It is based on a paper that was presented at the Mathsports International Conference in Loughborough in 2015 [11]. I supervised the implementation of this work as postgraduate student group project by George Barnett, Casper da Costa-Luis, Jose Garcia Maegli, Martin Ingram and James Neale. It was heavily based on my first year PhD research in monocular tennis video analysis.

The main contribution from this chapter is our low-cost spatio-temporal tennis data collection system, with its novel design and implementation. In addition to this, we also provide an overview and comparison of existing tennis ball and player detection algorithms, which is currently missing in the literature. This chapter addresses the Vision Layer of our framework shown in Figure 1.11.

4.1 System Requirements and Components

In this chapter we present our novel low-cost spatio-temporal tennis data collection system to obtain the player and ball locations relative to the court in real-time. In Section 1.1 we showed that some highly complex spatio-temporal tennis data collection systems already exist. However, their cost and complexity limits the use of these technologies to professional players and elite venues. This is why we developed a system more accessible to the general public. Taking this into account, the principal requirements of our technology are:

- **Accurate:** the accurate localisation of the ball and player location is essential for our system. Previous work in spatio-temporal tennis analysis described in Chapter 3 use clustering techniques or space discretisation that leaves space for a margin of error without loss of performance. We determine the acceptable error threshold based on expert umpires' performance. Figure 4.1 shows the umpire errors of challenged line calls according to Hawk-Eye data from 1 473 challenges during 15 tournaments between 2006 and 2007. The maximum error is 10cm. It is important to note that umpires at major tournaments (where Hawk-Eye is available) are highly qualified and exact ball location is easier to determine when bouncing next to the line, compared to bounces occurring at locations without nearby landmarks. Therefore we expect the error in the ball and player localisation from less expert manual annotators to be even higher in the general case.

- **Real-time:** many applications of this system would notably benefit from collecting and processing the information in real-time. For example, using it for line calling or providing feedback to players during training.

- **Unobtrusive:** after discussing with tennis coaches, we concluded that most tennis players prefer unobtrusive systems that will not interfere with their performance. This rules out the possibility of using wearable devices and racket sensors in our work.

- **Portable:** for our system to be accessible to the general public it must be able to be transported to and installed in any court quickly and easily.

- **Low cost:** existing technologies like Hawk-Eye and Playsight Smart-Court (see Section 1.1) are limited to major tournaments or a small number of exclusive tennis clubs due to their cost. We want to build a system which is more affordable and accessible to the general public.

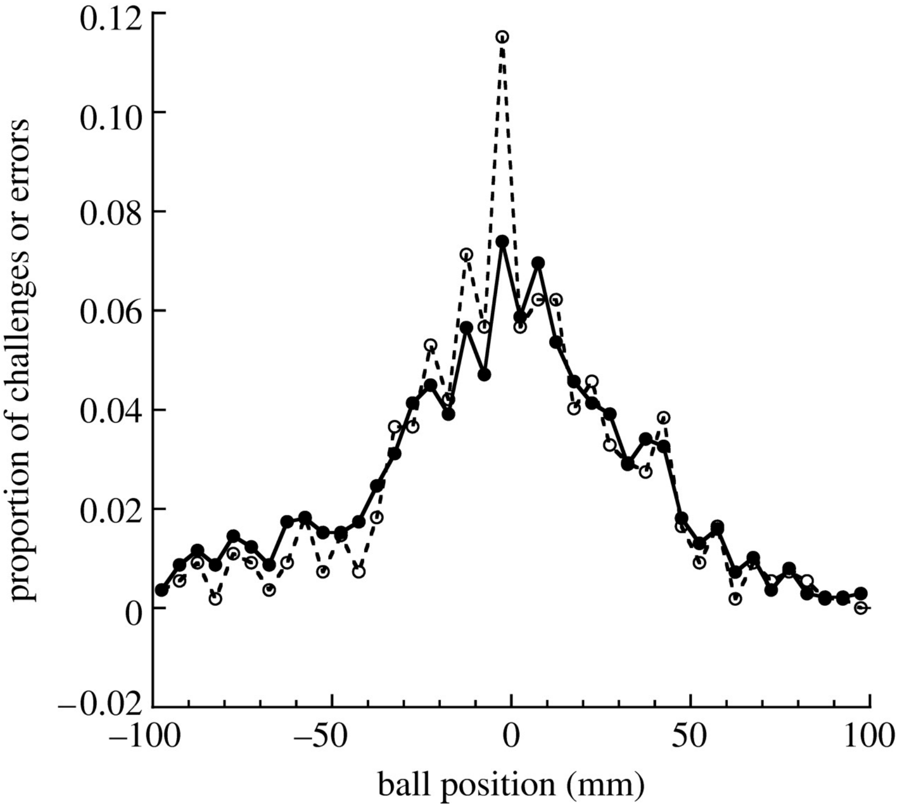


Figure 4.1: Umpire errors of challenged line calls according to Hawk-Eye [113]. Filled circles represent the proportion of challenges at each ball position and open circles are the proportion of line judge errors.

Decisions over the hardware used, software components and the development of algorithms for the different components of our detection system are framed by these constraints and the trade-offs between them. The rest of the chapter focuses on the main components of our system:

1. Hardware and Software Components – Section 4.2
2. Court Detection – Section 4.3.
3. 2D Player Detection – Section 4.4.
4. 2D Ball Detection – Section 4.5.
5. 3D Ball and Player Position (from 2D Data) – Section 4.6

4.2 System Components

4.2.1 Hardware

A clear image of the ball from at least two cameras is required at all times in order to accurately infer its 3D position. To guarantee this, we decided that the minimum number of cameras required is four, one at each corner of the court, as shown in Figure 4.2. The systems also requires a computer connected to the cameras to process the frames in real-time and store the recordings for subsequent analysis. Finally, a mechanism to synchronise frame capture from the four cameras is also necessary to ensure the accuracy of the 3D location calculation.

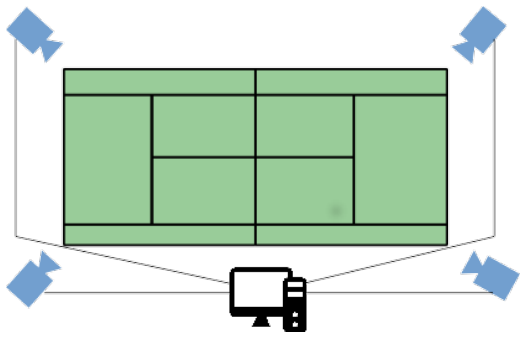


Figure 4.2: Configuration of the detection system hardware.

More specifically, the characteristics of the equipment used for our data collection system presented in this thesis are as follows. The camera model chosen is the Basler Ace acA 1300-60gm, a high-resolution (1280×1024 pixels) and high frame rate (60 fps) monochrome camera. High-resolution and frame rate are important to minimise motion blur, and we chose monochrome cameras to increase computational speed, considering that colour is not an essential attribute for object detection in our setting. We have complemented the cameras with low distortion Kowa LMVZ4411 lenses. The cameras are connected to a four-port Gigabit Ethernet card through Gigabit Ethernet cables that have a dual function: to deliver the camera capture at high speed to the computer and provide Power over Ethernet to the cameras, reducing cabling requirements. The computer used for processing is an HP Z620 Desktop Workstation with a 4 core processor[1], a fast 400GB PCI-E SSD and a GTX 980 NVIDIA graphics card. The system is low-cost, unobtrusive and portable, fulfilling three of the system constraints listed earlier.

[1]Intel Xeon(R) CPU E5-1630 v3 at 3.70GHz.

4.2.2 Software

Our software is composed of three layers as shown in Figure 4.3: frame capture, frame processing and display.

Frame Capture The Pylon Camera Software Suite was used to interface with the cameras and set properties such as exposure. An external 60 Hz signal generator was built to produce synchronised frame capture signals for the four cameras.

Frame Processing The computer vision part of the project is implemented in C++ using the OpenCV library [43] with an object-oriented design so that the system can easily be extended for new features. It is organised as a multi-threaded application in which player and ball are detected independently and in parallel for each frame (Sections 4.4 and 4.5 respectively) before the information is combined to obtain their 3D positions (Section 4.6). By processing the frames using eight parallel threads, the processing speed increases, and we are able to run our detection algorithms in real-time.

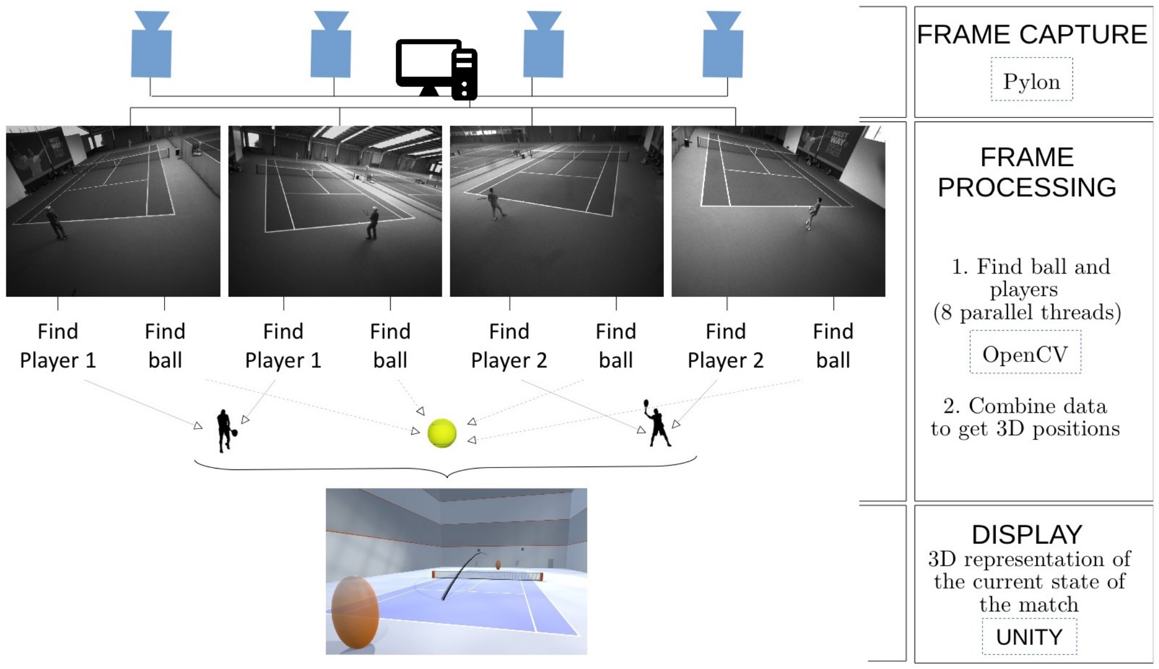


Figure 4.3: Software components of our system.

Display The data obtained from the frame processing module is fed into the data display module, which presents a 3D virtual environment where the court, player and ball positions are shown. It is a user-friendly display (screen-shot shown at the bottom of Figure 4.3) designed in Unity, which facilitates cross-platform portability.

4.3 Court Detection

Detecting the court lines is a necessary step for our system in order to represent the position of the players and ball within a meaningful coordinate system. Also, since the dimensions of the court in the real-world are known, as shown in Figure 2.1, identifying the court on the frame allows finding the extrinsic camera parameters (explained in Section 4.6.1) and the mapping of frame coordinates to real-world coordinates, as will be described in Section 4.6. Therefore, court detection is a crucial step for the vision system presented here, since any error translates to errors in the real-world coordinates of the player and ball. The main difficulties in detecting the court are:

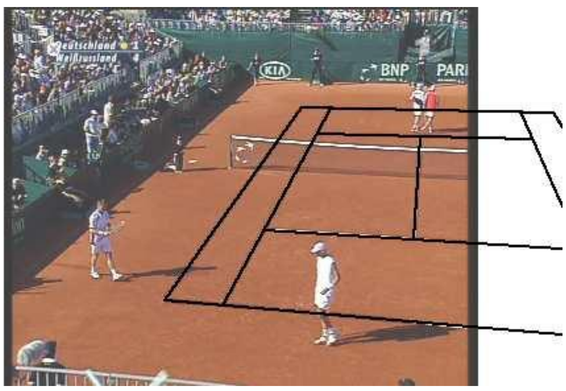
- Lines can be blurry, especially on grass or clay surfaces.
- Lines can be partially occluded, generally by players.
- Non-court lines can be detected, for example when there are grandstands.
- The court may be only partially visible, with part of it out of the frame.
- When some court lines are not detected, or extra lines are detected, finding the correspondences between lines on the frame and specific court lines (e.g. baseline) is difficult.

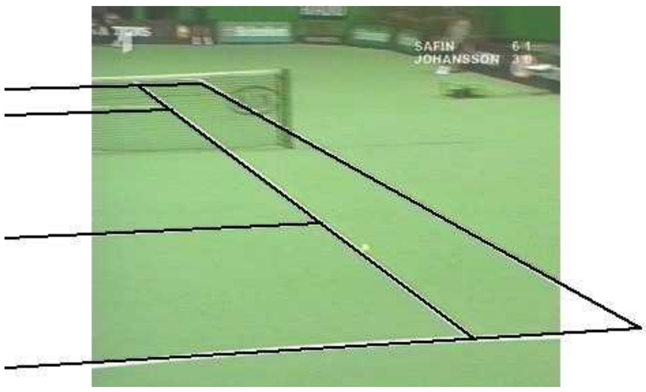
Our work on court detection was first developed for tennis broadcast videos, which are monocular, RGB and non-static. In this setting, all of the challenges mentioned above occur, and we developed an algorithm able to address them.

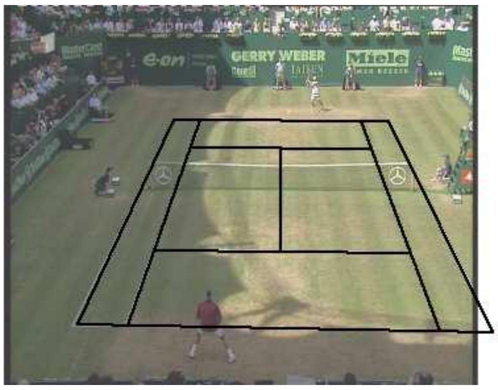
4.3.1 Related Work

Previous work on court detection and tracking has proven successful. In 2003, an algorithm robust against occlusions, shadows and partial views of the court was developed [114], shown in Figure 4.4. In 2010, another piece of work was presented achieving success in court detection in videos from the Australian Open (10 135 frames), French Open (5 026 frames), Wimbledon (7 667 frames) and US Open (8 327 frames), with an accuracy of 99%, 98%, 96% and 99% respectively [115]. The approach used by both studies is based on three main steps:

1. Extract white pixels.
2. Detect lines, using RANSAC[2] [115; 116] or Hough Transform [114].
3. Find correspondences between lines in the image and specific court lines, e.g. baseline.


(a) Partial view of the court.


(b) Partial view of the court.


(c) Scene with strong shadow.

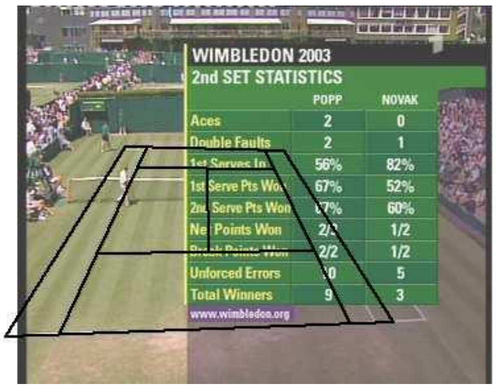

(d) Scene with large occlusion.

Figure 4.4: Tennis court detection on different surfaces, from [114].

[2]Random sample consensus.

4.3.2 Court Detection: Our Approach

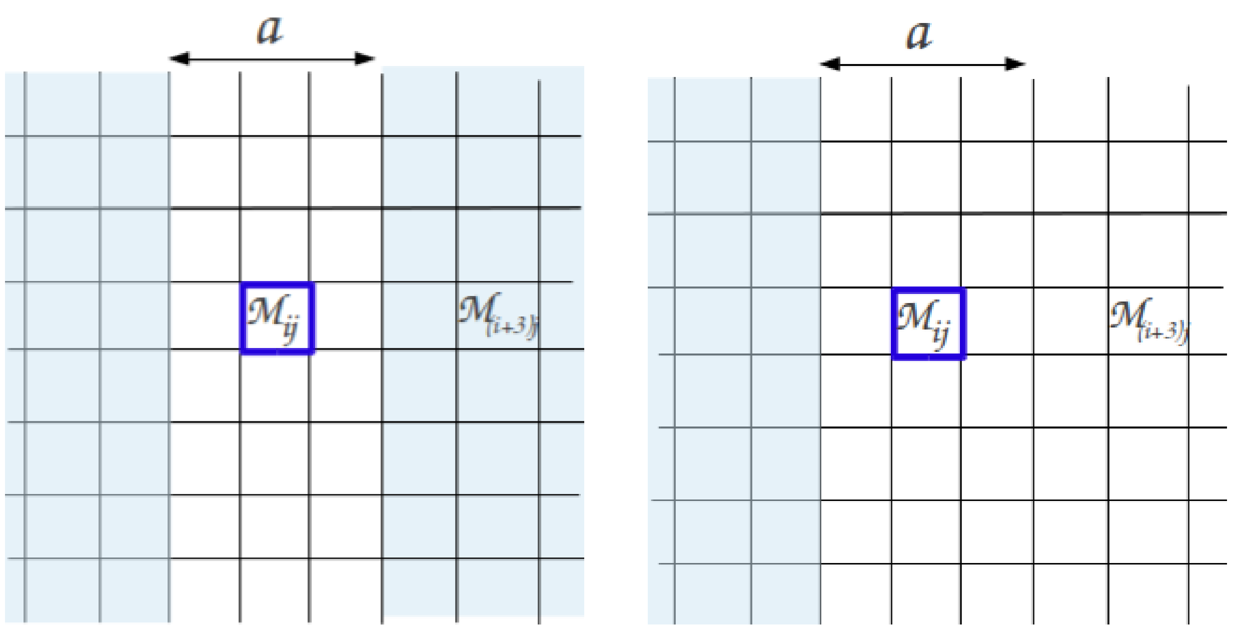
Following the three steps listed above, this section describes our implementation of court detection. We also include two additional components relative to existing work: net detection and a court detection confidence computation.

Extract Court Pixels

This is the first step in detecting the tennis court. It is fundamental to find the correct lines and to reduce the search space for line fitting algorithms, which are more computationally expensive than pixel selection. There is a trade-off between being too restrictive and omitting court pixels or being too permissive and including pixels belonging to the players, net or background. Court pixels are selected based on these two characteristics:

- Colour: We know that the lines of tennis courts are always white. However, because of illumination, video bandwidth and other factors, what we perceive as white in the videos is not pure white. For this reason, pixels with an intensity value higher than 200 in a monochrome image of the court (the maximum pixel intensity value that can be reached is 255) are selected.
- Belong to a line: For each pixel selected based on colour, we verify that its neighbouring pixels, within a certain distance τ either vertically or horizontally, have a considerably darker colour. τ is determined based on the approximate width of the tennis court lines and a threshold th for the intensity difference is selected. As shown in Figure 4.5a, if M_{ij} (pixel in the i^{th} column and j^{th} row in image M) belongs to a vertical line, then $M_{ij} - M_{(i+\tau)j} > th$ and $M_{ij} - M_{(i-\tau)j} > th$. However, this is not the case in Figure 4.5b. The same approach is used for horizontal lines, with $M_{ij} - M_{i(j+\tau)} > th$ and $M_{ij} - M_{i(j-\tau)} > th$.

Each element of the source image is analysed based on these features, and court pixels are set to white, as illustrated in the image of Figure 4.6.



(a) The pixel belongs to a line. (b) The pixel does not belong to a line.

Figure 4.5: Schematic representation to detect whether a pixel belongs to a line, with $\tau = 3$.

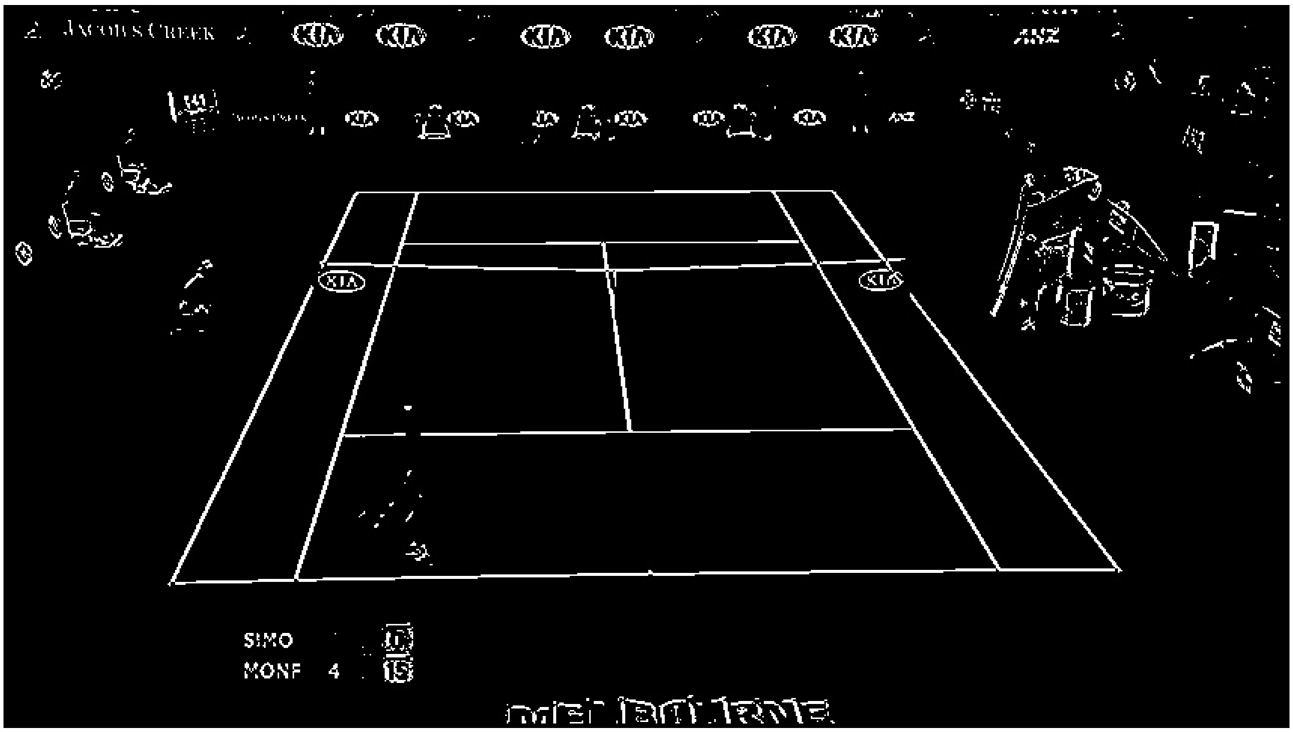


Figure 4.6: Image resulting from court pixel extraction.

Court Lines

Once the court pixels are extracted, lines are detected through the following steps:

1. Line detection: This is achieved by applying the probabilistic Hough Transform algorithm (explained in Section 2.5.3) to the court pixels image, like the one shown in Figure 4.6.
2. Selected lines classification: The lines that have been detected are classified into horizontal and vertical lines on the left and right-hand-side of the image separately. Next, the lines are sorted within each of the categories: horizontal lines are ordered by y-coordinate from top to bottom (left to right if the y coordinate is the same), left vertical lines are ordered by x-coordinate from right to left, and right vertical lines from left to right (from top to bottom if the x-coordinate is the same). A more detailed description of this process is provided in Appendix B.
3. Merging of selected lines: For each group of lines, the ones that are considered to be duplicates are merged. Appendix C contains a comprehensive explanation of the technique.

Court Correspondences

Now that a number of line candidates have been obtained, the next step consists in finding correspondences between lines in the image and specific court lines, e.g. the baseline.

For each possible combination of a pair of horizontal lines and a pair of vertical lines, we calculate the four intersections between horizontal and vertical lines[3]. Then, we try to find the correspondences between these four points and four points of court line intersections for eleven different court configurations, shown in Appendix F. For example, for configuration 1 in Appendix F, we will assume that the four points found on the frame are the four outermost corners of the court. Based on that, we will infer where the rest of the court lines are in the frame and calculate how accurate this hypothesis is. In this way the accuracy of each of the 11 court configurations is calculated.

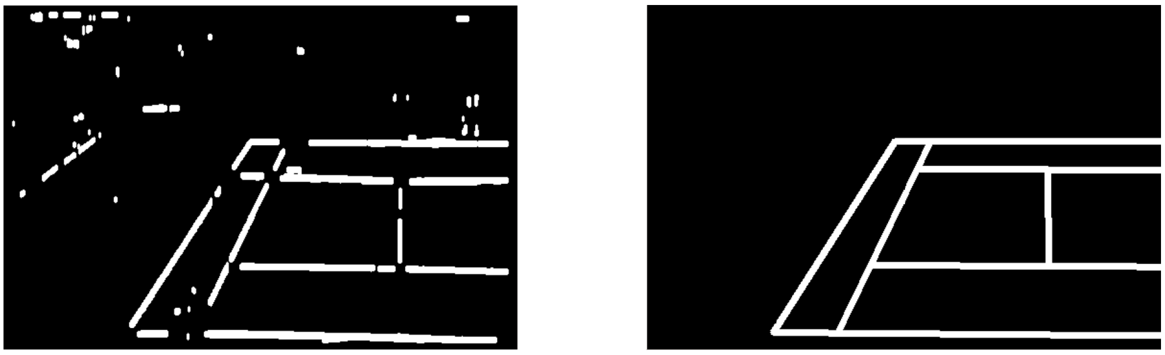
[3]The middle line of the court is excluded in finding court correspondences as it is generally not a white line, since the net is used to mark the middle of the court.

Court Accuracy

To measure how accurately a possible court configuration matches the court in the current frame, an image is generated where all court lines are projected onto a frame based on the current hypothesis (which contains only four lines), as shown in Figure 4.7b and explained in detail in Appendix E. Then we perform a bitwise *AND* operation between the projected court (Figure 4.7b) and the court pixels (Figure 4.7a). The accuracy is obtained by applying the following criteria to the resulting frame:

$$\begin{cases} +1 & \text{For each pixel of the projected court that matches a court pixel of the frame.} \\ -0.5 & \text{For each pixel of the projected court that does not match a court pixel of the frame.} \\ 0 & \text{For each pixel of the projected court that lies outside of the frame.} \end{cases}$$

Finally, the hypothesis with highest accuracy (amongst the 11 court configurations) is selected. This approach allows detecting the court event from a partial view, as shown in Figure 4.8.



(a) Frame court pixels. (b) Court model after transformation.

Figure 4.7: The two images necessary to measure the accuracy of the court projection.

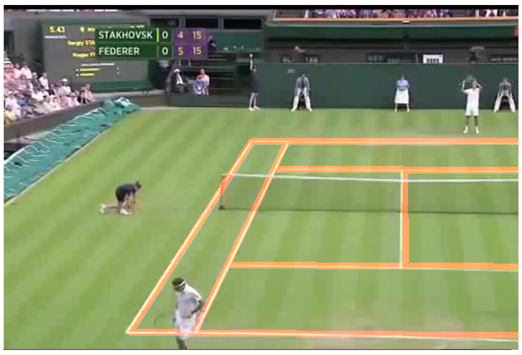


Figure 4.8: Court detection (in orange) from a partial view.

Court tracking

If the court was found in the previous frame, rather than repeating the protocol for court detection, the court is tracked based on its previous location. Court tracking consists in finding the new position of the four corners found when searching for the court, and this is done by separately tracking the four court lines that intersect at these corners. The algorithm for tracking a line consists of selecting 100 points[4] equally spaced along the line and finding the neighbouring pixel in any direction in the new frame with the highest intensity, provided it is above the intensity threshold used in court detection. If more than 50 points are found (half of the sampled points), the new court line is the one resulting to fitting them via a least-squares method. Otherwise, the court is searched again.

Location of the Net

The court dimensions and location of the centre line of the court are known from the court detection. The frame is scanned bottom to top from this line for a certain distance, set to the 10% of the court length[5]. Similar to the tracking algorithm, 100 points in the middle line are selected. For each of them, the pixel with the highest intensity among the 100 pixels above it is selected and the line best fitting these points is taken as the net, displayed in blue in Figure 4.9.

Evaluation

Finally, we devised an evaluation step in the court detection and tracking algorithms. This step consists in subtracting the source image (real image of the court) from the projected court. The projected court is a binary image which has only white (on the court) and black (background) pixels, with values 1 and 0 respectively. Therefore: $\mathit{WhitePixel} - \mathit{BlackPixel} = \mathit{WhitePixel}$, $\mathit{WhitePixel} - \mathit{WhitePixel} = \mathit{BlackPixel}$, and black pixels are always left unchanged since their value is 0 already, meaning that $\mathit{BlackPixel} - \mathit{BlackPixel} = \mathit{BlackPixel}$

[4]Adding more points would also work but would slow down the algorithm.

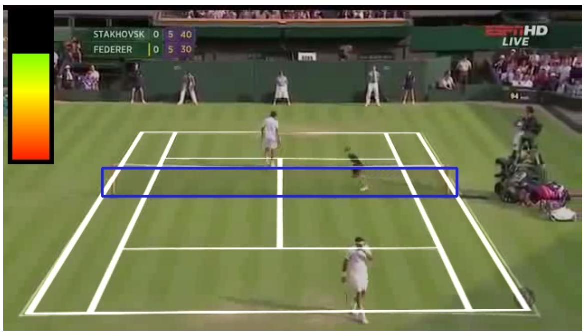
[5]Calculated using the perspective transformation of the court lines.

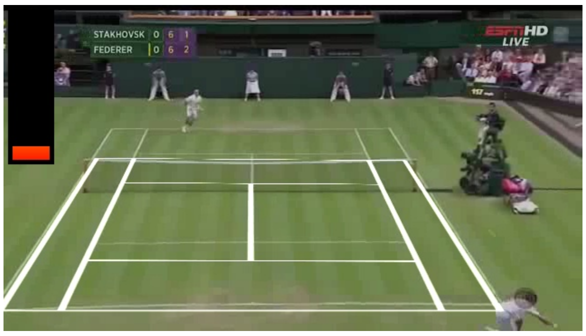
and *BlackPixel* − *WhitePixel* = *BlackPixel*. In the resulting frame, pixels with value 1 correspond to matching pixels (in the projected and detected court) and 0 pixels correspond to non-matching ones. The subtraction will lead to an empty matrix (all pixels black) if all the pixels of the source image that are in the same position as the projected court are white, resulting in the highest accuracy score. The accuracy score has a maximum value of 100 and is calculated as 100 minus the percentage of non-zero pixels in the subtracted image over the total number of pixels in the projected court image. Accordingly, a higher number of white pixels in the subtracted image will lead to lower accuracy score.

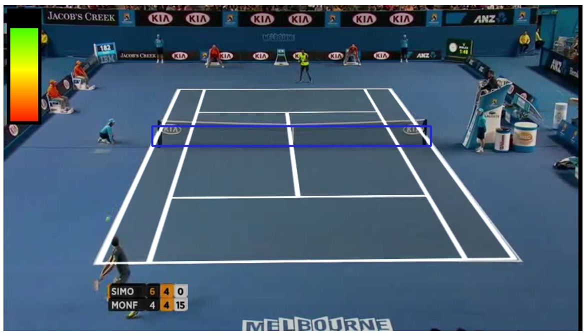
However, as previously mentioned the pixels in the source image should be pure white, and this differs from what we perceive as white in the image. In addition to this, the width of the lines of the projected and the real court are different. To overcome these limitations, we added a preprocessing stage to the source image and generated a new image by applying a modified thresholding operation. For every pixel in a monochrome representation of the image of origin that has an intensity value above 200 (out of a maximum value of 255), the pixel at the same location in the new image (initially all black) and its neighbours within a radius of 5 pixels are set to white, to account for court line thickness. After this preprocessing of the source image, the subtraction to the projected court is performed. Figure 4.9 shows examples of court projections and their accuracy bar. It can be seen that the bar truly represents the accuracy of the court fitting.

4.3.3 Results

Court detection was evaluated using our own metric, the accuracy score explained in Section 4.3.2, due to the lack of a unified protocol for the evaluation of this task. Our court detection algorithm had an accuracy score higher than 70% in 12 out of the 15 tested cases which included low-resolution and high-definition images of courts with occlusions or partial view in all kinds of surfaces (grass, clay and hard). The three cases where the algorithm failed corresponded to grass surfaces with the marked lines worn out, but it was successful in another 3 grass surfaces. For the isolated cases in which the court detection had an accuracy score


(a) Accurate court projection (68%) on grass surface.


(b) Inaccurate court projection (11%) on grass surface.


(c) Accurate court projection (86%) on hard surface.

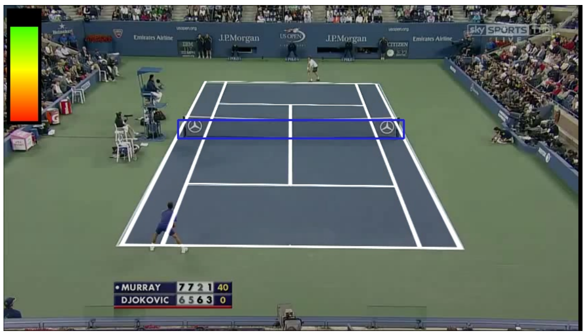

(d) Accurate court projection (86%) on hard surface.

Figure 4.9: Display of the court projection and its accuracy.

lower than 70%, the use of an accuracy threshold allowed to reject that fit as too poor and not continue on to player and ball detection for those cases until a good fit was found.

4.4 Player Detection

A broad range of techniques have been studied to solve this task. They will be reviewed in the next section, but it is important to note that the characteristics of the video are crucial to select the approach that is best suited. As mentioned above, our previous work addressed player detection in monocular RGB videos, where the camera was moving [117]. This was developed with the idea of using it with broadcast videos. However, inferring the 3D location of objects from monocular videos is extremely challenging and an ongoing area of research. For this reason, we decided to build our system with four cameras as shown in Figure 4.2 and, in this setting, our objective is to detect only one player per camera – the one closest to it

– resulting in a two-to-one camera to player ratio. The biggest challenges in detecting tennis players in real-time from monochrome fixed cameras are:

- Moving objects in the background (e.g. people playing on an adjacent court or spectators).
- Missing colour information, as our cameras are monochrome.
- Player representation: it is often hard to detect specific landmarks (e.g. players' feet), but specific coordinates are needed to synchronise player detections from different views.

In other situations, additional challenges include:

- Moving camera: this causes movements in the background which makes it more difficult to detect moving objects such as the ball or players and track them.
- Use of broadcast videos: they often contain additional information such as score displays, and the player and court might not be present in some frames (e.g. footage of spectators, player close-ups).

4.4.1 Related Work

Table 4.1 summarises the most relevant work in tennis player detection over recent years by showing the approach used and the characteristics of the footage analysed. The techniques referred in the table are summarised below:

- **Dominant Colour Detection** (DCD): the application of this technique to player detection was introduced in [118]. It consists in calculating the mean value μ and variance σ^2 for each RGB channel (e.g. μ_r and σ_r^2 are the mean and variance of the red channel) in the area inside of the court and the area immediately outside of it ("background"). Players ("foreground") are detected by selecting the pixels that are very different from these values. For example, a given pixel p with value r in its red channel is considered to be in the background if $|r - \mu_r| < 3\sigma_r^2$, or if the same condition is satisfied for the other colour channels.

- **Frame difference** (FD): In [119] players are detected by detecting salient/foreground objects through the subtraction of consecutive frames. Camera movement is compensated by using the court lines as reference. This approach has the drawback that if the player is not moving much, his/her contour might not be accurately detected.

The results from the pieces of research presented in Table 4.1 are promising but cannot be directly applied to our setting in which we do not have RGB information and need to coordinate views from multiple cameras. In addition to this, most publications only present a qualitative analysis of the results and do not provide information on the running time of their algorithms. This makes it difficult to make decisions about algorithm design, and it is also problematic for comparing them to our work. Nonetheless, and as will be shown in the next section, our approach not only uses knowledge uncovered by this previous work but also includes some new ideas to improve accuracy and execution speed in our setting.

4.4.2 Player Detection: Our Approach

Our design choices are motivated by two constraints: accuracy and processing speed, since we want our system to be real-time. Based on these constraints and the existing literature described above, we decided to use a combination of background subtraction and frame differencing for the player detection. Our approach consists of two main steps: background reconstruction and player detection itself.

Background reconstruction: generating the background image

We describe the background image of a video as the features that do not move or change over time (or do so very slowly) and the intuition is to select the pixels that are stable through time as the background. The background image is initialized to 0 (black image), as shown in

[6]Only the most challenging videos were tested for these results and details on testing methodology are absent.

[7]Player location defined as the player's bounding box, and considered correct if at least 70% of the player's body is actually in the bounding box.

[8]In [120] the authors combine DCD with ED, for which they use the Sobel operator [122].

Paper	Multiple camera	Cameras movement	RGB	Detection method	RT	Acc
Vinyes et al., 2015 [11]	✓	-	-	BS	✓	–
Vinyes, 2013 [117]	-	✓	✓	FD	✓	Q
Dang et al., 2010 [115]	-	✓	✓	FD	✓	86.6%[6]
Jiang et al., 2008 [120]	-	✓	✓	DCD + ED	N/A	Q
Han et al., 2007 [118]	-	✓	✓	DCD	N/A	95%[7]
Miyamori and Iisaku, 2000 [121]	-	✓	✓	FD	N/A	–
Sudhir et al., 1998 [119]	-	✓	✓	FD	N/A	Q

Table 4.1: Previous work in player detection. Detection methods are: BS – Background subtraction (described in the next section), DCD – Dominant Colour Detection, ED – Edge Detection[8], FD – Frame difference, RT – Real Time, Acc – Accuracy, Q – only qualitative results provided.

Figure 4.10. Then, each frame is divided into a grid of 8 pixels $\times$ 64 pixels blocks, because people are usually taller than they are wide. Then, blocks at the same position are compared between consecutive frames. The similarity between two blocks is estimated by calculating their pixel-wise intensity difference. If 98% of pixels have an absolute intensity difference lower than 3, the block is considered stable and incorporated into the background image. Otherwise, the block is not included in the background. This process is iteratively applied to new frames until all blocks in the background image are filled, meaning that a complete background image has been obtained. The process is applied to each camera separately, leading to one complete background image per camera.

During this process, player detection is not performed, but the process is fast enough (in the order of ms) for the delay to be negligible (we expect that at the beginning of a match/training session players are still getting into position). Selected steps from this process are shown in Figure 4.10, and it can be seen how the initially empty background image progressively incorporates background blocks until a complete background image is generated. A complete version is shown in Appendix D. After the background is generated, it gets updated with every new frame captured to incorporate changes to the background (e.g. illumination); this is done through a weighted sum of the current background image with a 0.99 weight and the incoming frame with a 0.01 weight. The incoming frame has a very low weight in order to avoid incorporating non-background changes (e.g. a player moving) and to include changes that are more stable over time (e.g. changes in illumination).

Player detection

Once the background image is generated, player detection can be achieved as illustrated in Figure 4.11. First, foreground pixels in the current frame are detected by calculating the absolute difference with the background image. Then, noise is removed by applying a binary thresholding operation, from which pixels with very low intensity (lower than 10) are discarded. Once this is complete, player detection takes place by selecting the foreground pixels within a region of interest (ROI), corresponding to the area in which the player can be located. For the frames in which the prior player position is unknown, the ROI for each player corresponds to their half court and detection is performed in this area. For the frames in which the prior positions are known, detection is carried out only in areas within a given distance d of the player, with d defined according to the maximum speed of a player's movement. The average size of a player in our videos is 50 pixels $\times$ 200 pixels. Our estimated maximum distance covered by a player between two consecutive frames in the x direction (generally corresponding to movement to the left or right) is 35 pixels, 70% of the player's width. In the y direction (generally corresponding to movement up and down or forward and backwards), we estimate that the maximum distance a player can move between two consecutive frames is 60 pixels, which represents 30% of the player's height, since lateral movements are generally of wider range than vertical movements and forward/backwards movements appear smaller than lateral movements due to the perspec-

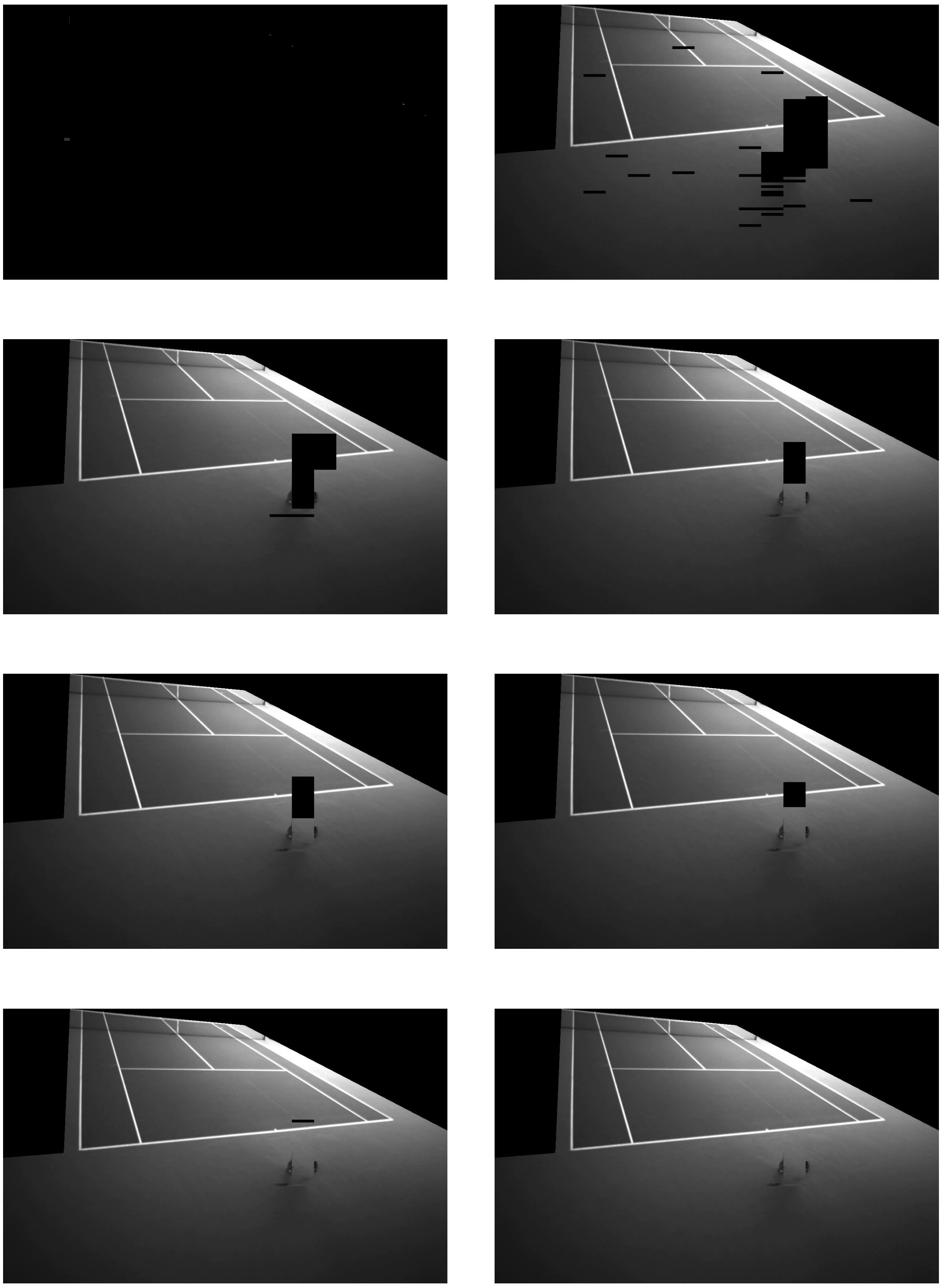
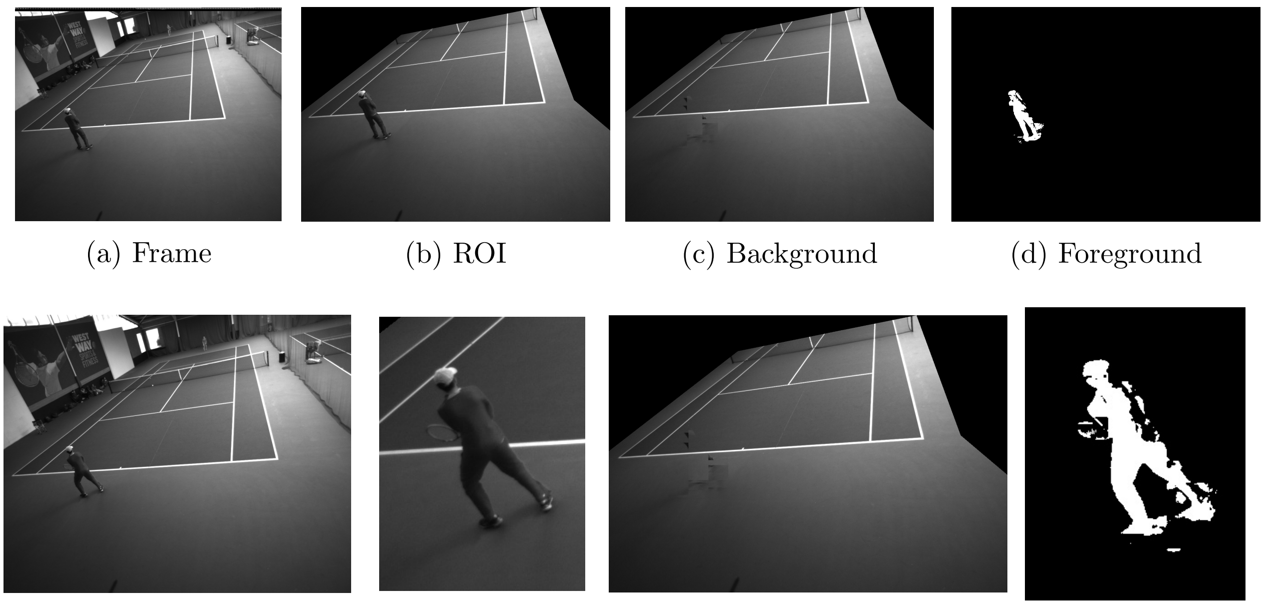


Figure 4.10: Background generation example. Initially the background image is empty and it progressively incorporates background blocks until a complete background image is generated.



(a) Frame (b) ROI (c) Background (d) Foreground

(e) Frame (f) ROI (g) Background (h) Foreground

Figure 4.11: Player detection at the first iteration (a to d) and at a later stage (e to h). ROI is Region of Interest.

tive. Once the collection of foreground pixels within the appropriate ROI is obtained, contours[9] are extracted using the OpenCV implementation *findContours* based on [123] and the contour with the largest area is selected. The precise location of the player is represented as the centre of mass $(\bar{x}, \bar{y})$ of the area within the selected contour and is calculated as follows:

$$\bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}} \tag{4.1}$$

where m are the spatial moments:

$$m_{pq} = \sum_{x,y} I_{x,y} \times x^p \times y^q \tag{4.2}$$

and $I_{x,y}$ is the intensity of the pixel with coordinates x and y in the input image.

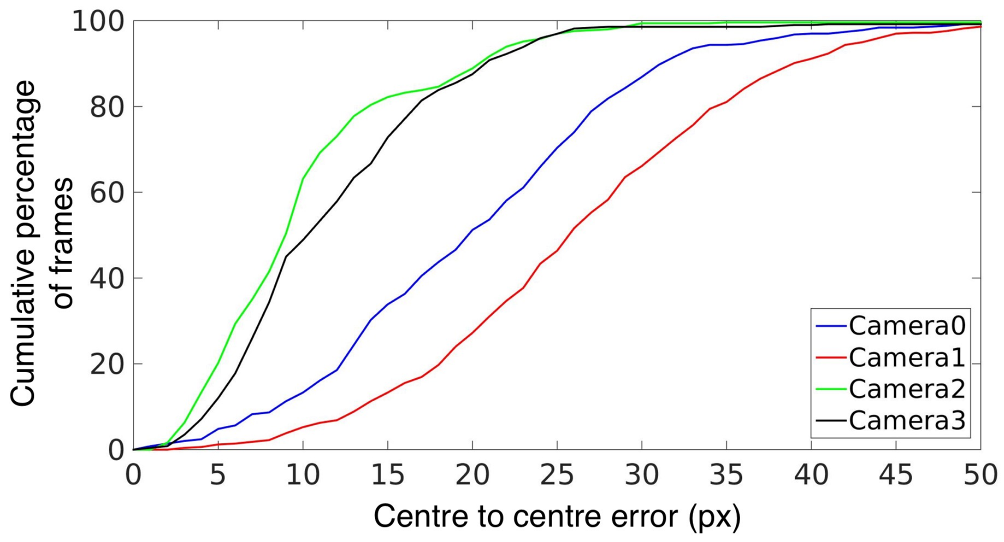
[9]Collection of foreground pixels that represent the boundary of an object.

4.4.3 Results

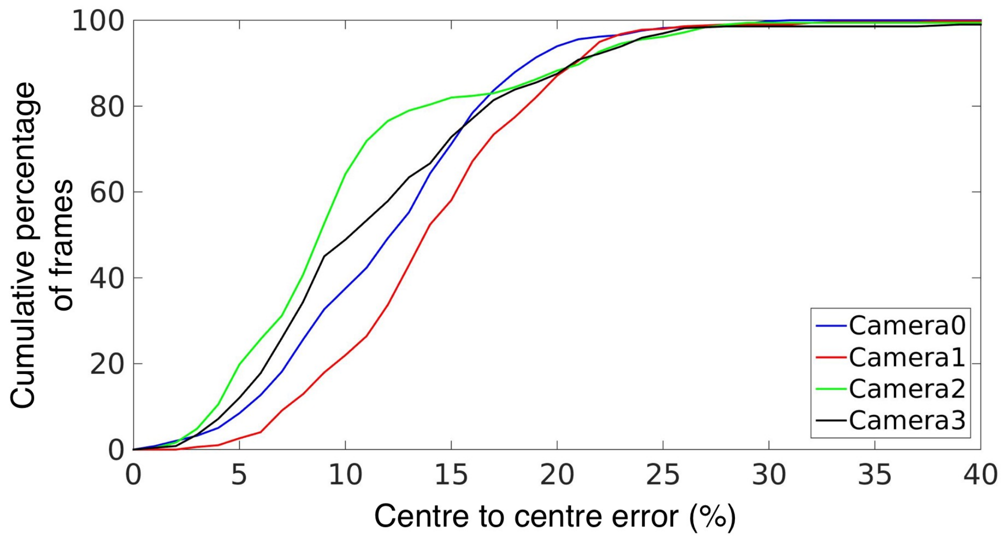
The player detection results are 2D coordinates that represent the estimated centre of mass of the area within the selected player contour on the frame. The ground truth player data is manually labelled and stored as the coordinates defining a rectangle surrounding the player. The results presented in this chapter (also for the ball and inferred 3D locations) are the analysis of 5 342 frames from 12 videos in 3 different scenarios (4 videos for each scenario, one per camera). Videos can be accessed at: `https://www.doc.ic.ac.uk/sv212-tennis/`.

The automatically detected centre of mass is within the ground truth bounding box in all of the analysed frames. However, to further evaluate our results we also calculated the detection error as the Euclidean distance between the centre of the ground truth bounding box and the detected centre of mass. Figure 4.12a shows that the errors never exceed 40 pixels.

Considering that the bounding box of the player contour in our videos is up to 50 pixels width by 200 pixels height and that we are comparing centre of mass (in experiments) to the centre of the bounding box (in ground truth) we consider our results to be of good quality. In addition to this, we decided to also represent the error as a relative measure with respect to the size of the bounding box. We noticed that a given error distance of x pixels represents a larger mistake if the bounding box of the player is smaller. To reflect this, the error was re-scaled according to the size of the ground-truth bounding box by taking its ratio in the height and width dimensions. Figure 4.12b shows these results for each camera, and we can see that for 80% of the frames the error is less than 20% of the bounding box, and the error never exceeds 30%. It is also interesting to note that the results are more stable between the different cameras in Figure 4.12b than in Figure 4.12a. This is because the player is closer to some of the cameras than others. For example, camera 1 has a bigger error in terms of pixels than the other cameras. This is probably because the player is very close to it, causing the distance between the detected centre of mass and the manually labelled centre of the bounding box to be larger in absolute terms, but when the error is scaled to the player size, it seems to be closer to the results from the other cameras.



(a) As a number of pixels.



(b) As a percentage of its bounding box.

Figure 4.12: Accuracy of player detection.

4.5 Ball Detection

After having discussed our player detection algorithm, this section focuses on tennis ball detection. Our objective is to detect the ball from at least two cameras at any point in time in order to be able to infer its 3D location. Our system was designed with four cameras so that when the ball is very far away from a camera or occluded by a player, it will still be visible by at least two other cameras. The main challenges in this task are the following:

- **Ball size:** the ball is very small compared to the court and players dimensions. This makes it harder to detect. For example, noise pixels in the frame differencing (as used to detect players) might form structures of similar size to the ball.
- **Ball speed:** even if the ball shape is a sphere, its appearance may be that of an ellipse with variable parameters when travelling at high speed (this effect is called motion blur).
- **Occlusions:** the ball might be occluded by a player, and its image may also merge with that of the racket when it is being hit.

Tracking a player only differed from detecting a player by reducing the region of the image that was processed. We knew that the player should be close to its previous location but the direction in which she/he was moving was not deterministic (i.e. a player might decide to change direction at any time). By contrast, the tennis ball follows the laws of physics and some trajectories are not possible (e.g. the ball will not change trajectory without impacting with another object). This has a strong effect on the tracking algorithms as we will see later. Additional challenges coming from the settings of our system are the same as in Section 4.4 (i.e. moving objects in the background and lack of colour information since we use monochrome cameras).

4.5.1 Related Work

Table 4.2 summarises previous work on ball detection. Most successful approaches start with the extraction of foreground pixels and their clustering into entities, by pixel proximity. This results in a number of ball candidates since, as we just mentioned, the small dimensions of the ball mean that they may get confused with noise. The most complicated part in the ball detection comes at this stage when candidates must be classified and the best one selected – this is where approaches differ. As can be seen in Table 4.2, some algorithms focus on selecting the best candidate based on the ball features alone [124; 125], while others also look at possible ball trajectories [126–128], taking advantage of the fact that ball trajectory can be modelled using the laws of physics.

Ball features

Selecting the best ball candidate can be done using different features. Colour, size and shape are the most common and Table 4.2 reveals which features are used for each of the algorithms presented. In [124], the best candidate is selected solely on colour, which might lead to inaccuracies as the colour of the tennis ball in an image can be affected by the colour of neighbouring pixels due to the camera resolution. This is why also looking at shape similarity seems a good idea. Finding which ball candidate has a shape most similar to a tennis ball can be approached in many different ways. In [126; 128] the authors only look at the height to width ratio, and it seems to give good results in their work. [127] takes a more sophisticated approach. First, an ellipse is fitted to each blob of pixels with the potential to be the tennis ball using the least squares criterion. Then, an SVM is used to classify them as tennis ball shape or other shape based on the following features of the fitted ellipse: (1) Ellipse centre coordinates, (2) Major and minor axes of the ellipse, (3) Mean of HSV[10] channels within the ellipse and (4) Gradient vectors[11] from a number of points in the ellipse contour.

Ball trajectory

In 2004, Yu et al. incorporated a new dimension into ball detection [128]. Besides looking at whether a candidate looked like a ball, they also took into account whether its motion was consistent with the trajectory of a ball. They used a Kalman Filter (more detail in Appendix G) to model tennis ball trajectories and found out which candidates contributed to plausible trajectories. On the same lines, a new approach was presented in [127], this time using a particle filter. This is a methodology that is used to solve problems in signal processing and Bayesian inference. The system state is defined as the ball position and velocity, and state transitions are modelled from the laws of physics with added noise. A more detailed description of the method is given in Appendix H. Most recently, a new approach was presented where the authors worked with the following constraint: the ball should follow a semi-linear trajectory[12] at a speed within

[10]Hue, Saturation, Value colour space
[11]Directional change of pixel intensity
[12]Piecewise linear trajectory.

some specified boundaries [126]. Instead of modelling entire ball trajectories, a ball candidate is added to a trajectory, if it follows a locally linear path. This assumes a non-spinning ball, and to model spin a more complex model would be needed.

Comparison

It is hard to compare the different techniques shown in Table 4.2 for two main reasons. First, there is no benchmark dataset or standardised method to calculate tracking accuracy in this domain, and each paper presents their results differently:

- In [124] the authors only provide qualitative results through visual inspection.
- In [128] results are presented as a percentage of the frames in which the ball was detected or tracked (average of 77%), but they do not offer a quantitative measure of the error distance in ball tracking.
- In [127] the authors analyse their results through visual inspection only, and qualify their results as 'excellent' for 69 out of 70 shots.

Second, some of the work presented has a goal that requires ball tracking but is different from it (e.g. detecting high-level events), and their results are only analysed for the final task [125; 126]. However, based on our visual examination of the images shown in the above-mentioned publications, we consider that all the algorithms that we have discussed produced qualitatively acceptable results.

Applicability to our setting

To see which of the above-presented techniques could be used in our work there is a major issue: the cited work does not provide information on running time, and it is crucial that our system runs in real-time. However, amongst the work shown in Table 4.2, we consider [125; 126] to be closer to our problem because they do not use colour information and our videos are

Paper	Ball features				Trajectory	Results
	Colour	Size	Shape	Extra		
Pingali et al., 1998 [124]	✓	-	-	-	N/A	Visual inspection, qualitative analysis.
Yu et al., 2004 [128]	✓	✓	✓	-	Kalman Filter	Results presented as the number of frames in which the ball was detected.
Yan et al., 2005 [127]	✓	-	✓	-	Particle Filter	Only qualitative results through visual inspection.
O'Conaire et al., 2009 [126]	-	✓	✓	Ball candidate is far away from the other candidates	Semi-linear path and speed constraint	The objective is to detect events and results only provided for this task.
Teachabarikiti et al., 2010 [125]	-	✓	-	Location	N/A	The objective is to detect events and results only provided for this task.

Table 4.2: Previous work in ball detection. [128] and [126] use the width/height ratio to classify the shape and [129] uses ellipse fitting characteristics.

monochrome. In addition to this, it has been recently shown that tracking using Kalman Filters, Particle Filters and Random Forests perform similarly, and Random Forests perform better for high noise measurements [130], which is our case. For these reasons, our approach involves using Random Forests to detect the tennis ball based on ball appearance and trajectory, as will be described in more detail in the next section.

4.5.2 Ball Detection: Our Approach

Our approach to ball detection is presented here, and it can be seen to have some common features with previous work and also contribute new ideas. First, we describe in more detail how foreground pixels are detected and clustered into ball candidates. Then we propose a novel approach for selecting the best candidate. We employ Random Forests to learn to identify ball shape (using the seven Hu moments, described later) and ball trajectory.

Extract foreground pixels

The aim of this stage is to extract the pixels that have changed from the previous frame: these pixels are intended to represent motion. Given a frame at time t, the pixels representing motion are stored in a matrix that will be referred to as *motion matrix*. As illustrated in Figure 4.13, it is computed through the following steps:

1. *first motion*: Calculate the absolute pixel intensity difference between frames $t-1$ and frame t and threshold the image, in order to keep only the pixels in the highest decile of intensity change. This frame will contain the moving objects which include the current ball position at time t and the ball position at $t-1$.

2. *second motion*: Similar to the previous step, calculate the absolute pixel intensity difference between frames $t+1$ and frame $t-1$. In this case the frame will contain the ball position at $t+1$ and at $t-1$.

3. In the *motion matrix*, set the pixels as in *first motion* and then set to 0 those pixels which are non-zero pixels in *second motion*.

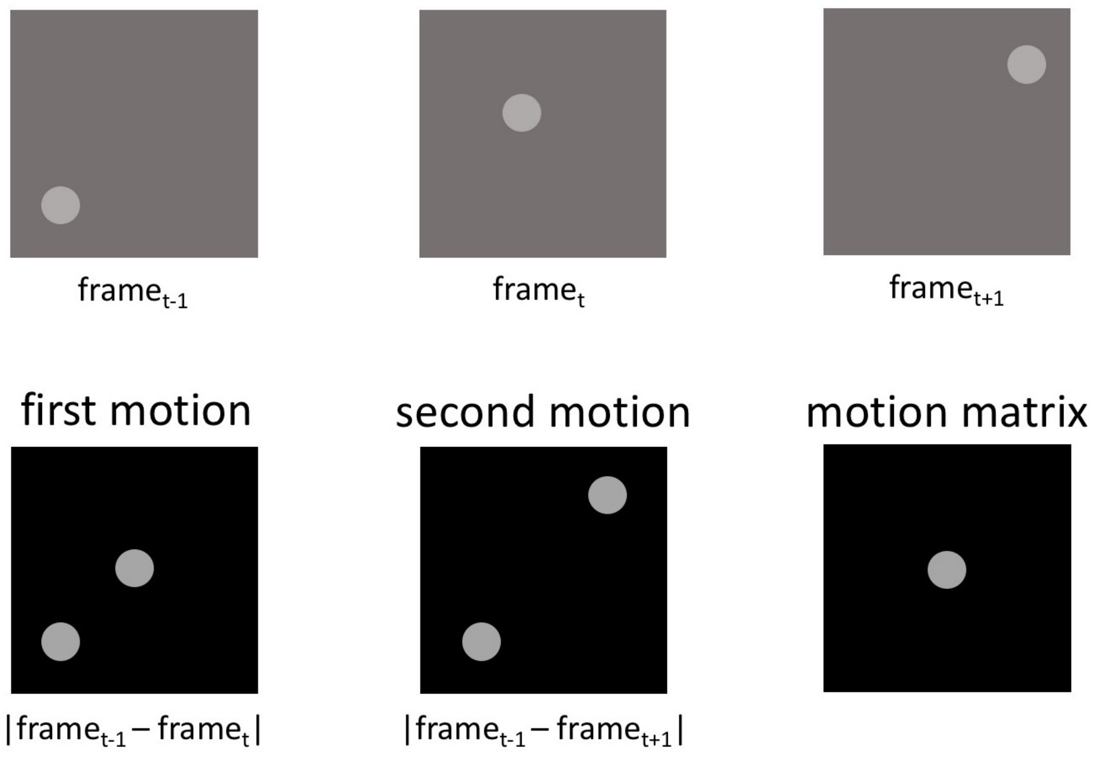


Figure 4.13: Diagram of motion extraction for ball detection, inspired by [131].

Finding and selecting candidates

Next, contours of ball candidates are detected on the *motion matrix* using the OpenCV implementation *findContours*, based on [123]. Then the best candidate is selected by assigning a score to each potential ball contour using two Random Forest classifiers, explained in Section 2.6.2. Each of the Random Forests has a maximum number of 500 trees with a maximum depth of 15 nodes and were trained using six videos each.

The first classifier is trained to recognise a ball contour shape based on the contours' Hu moments and the contour area, using 26 530 candidates. The seven Hu moments of a contour are descriptors invariant to image scale and rotation. Following from Equations (4.1) and (4.2) and using the same notation, the seven Hu moments are calculated as follows (from [9]):

$$\begin{aligned}
hu[0] &= \eta_{20} + \eta_{02} \\
hu[1] &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
hu[2] &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
hu[3] &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
hu[4] &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12}) + [(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\
&\quad + (3\eta_{21} - \eta 03)(\eta_{21} + \eta 03)[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
hu[5] &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
hu[6] &= (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03}) + [3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
&\quad - (\eta_{30} - \eta 12)(\eta_{21} + \eta 03)[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned} \tag{4.3}$$

where η are the normalised central moments:

$$\eta_{ji} = \frac{\mu_{ji}}{m_{00}^{(i+j)/2+1}} \tag{4.4}$$

m_{00} is defined in Equation (4.2) and $\mu_{i,j}$ are the central moments:

$$\mu_{ij} = \sum_{x,y} I_{x,y} \times (x - \bar{x})^i \times (y - \bar{y})^j \tag{4.5}$$

The second classifier is trained on the ball trajectory, using 1 863 trajectories from 6 videos. The trajectory is defined by the quadratic function that is fitted to the x and y coordinates and the features used for training are the average error of the fit to the trajectory, the ball candidates' average score (based on their shape) and standard deviation (based on their shape) and the number of previous candidates fitting that trajectory.

4.5.3 Results

The ball detection generates a set of 2D coordinates which are compared to the 5 342 manually annotated frames from 12 videos (as in Section 4.4.3), labelled with the coordinates of the centre of the tennis ball. Figure 4.14 shows the error in ball detection as the Euclidean distance between the detected point and the ground truth for every frame in which a ball candidate was obtained. Each curve in Figure 4.14 represents frames captured by a different camera. It can be observed that for each camera the error is less than 2 pixels for more than 75% of the frames and less than 6 pixels for almost all frames (Camera 1 does not even converge at 6 pixels because it has an outlier, which is at a distance of 51 pixels). Considering that the ball has a diameter of up to 10 pixels depending on its proximity to the camera, we find these results to be highly satisfactory.

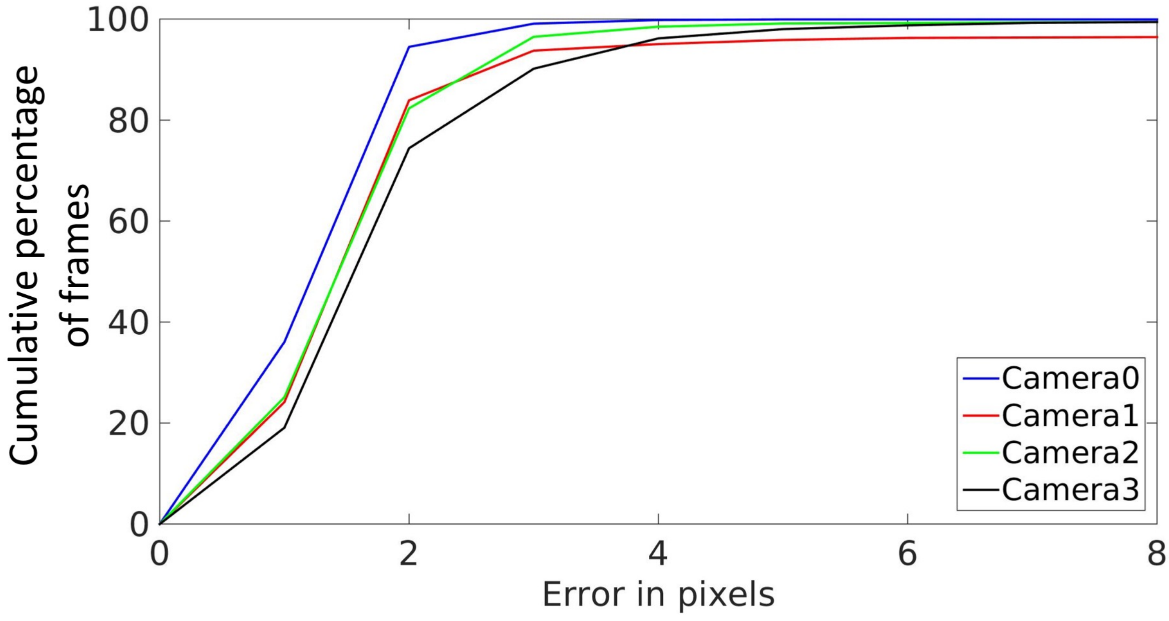


Figure 4.14: Accuracy of ball detection.

4.6 Triangulation

Ball and player detection in 2D are performed in parallel for the four cameras, and these results must be integrated in order to obtain the 3D location of ball and player relative to the court. This process is called triangulation and corresponds to mapping 2D frame coordinates to 3D coordinates in the real world. To address this, we first need to be able to build a camera model describing the mapping from any 3D point in the real world to 2D frame coordinates. This section will start reviewing some theory for building a camera model in Section 4.6.1 and doing the necessary camera calibration in Section 4.6.2. From these, our approach to triangulation is explained in Section 4.6.3. Finally, Section 4.6.4 will discuss the results obtained and reveal how small inaccuracies in 2D detection translate to 3D space. We note that there is a need to synchronise the cameras with respect to a common clock so that triangulation is performed on time-synchronised frame data.

4.6.1 Camera Model

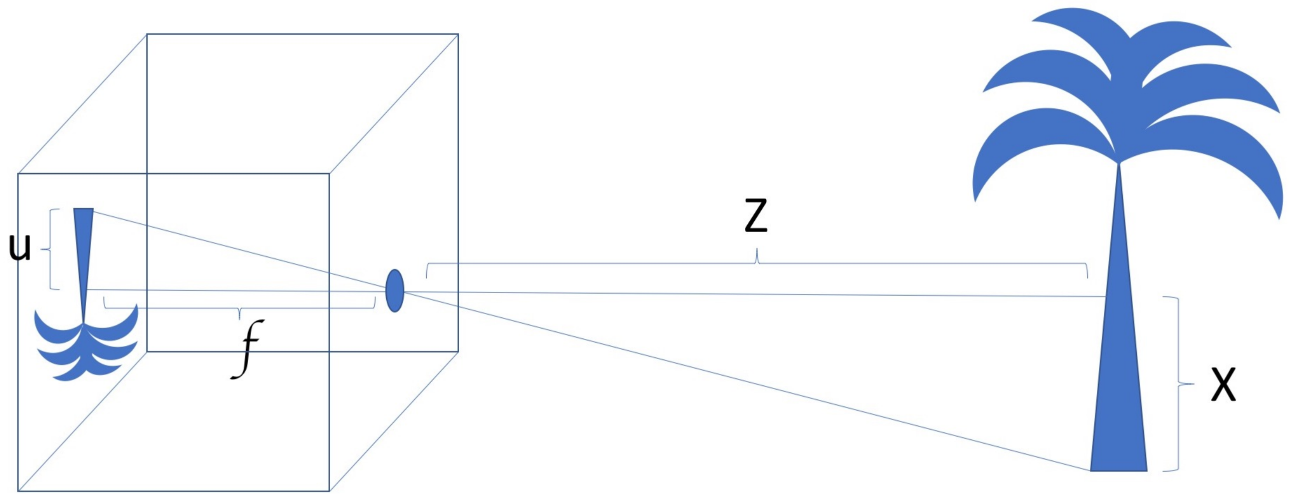


Figure 4.15: Diagram of a pinhole camera model.

The simplest camera model is that of a pinhole camera. As shown in Figure 4.15, the camera has a small aperture that allows only a few rays to reach the image plane. The distance between the pinhole and the image plane is the focal length f. The 3D coordinates of the object in the real world are labelled X, Y and Z (object height, width and distance from camera, respectively) and the 2D corresponding location in the image plane are u and v (image height and width, respectively). With this and by the property of similar triangles (shown in Figure 4.15 for f, u, Z and X), we have:

$$\frac{f}{Z} = \frac{u}{X} = \frac{v}{Y} \tag{4.6}$$

and therefore

$$u = \frac{fX}{Z} \quad \text{and} \quad v = \frac{fY}{Z} \tag{4.7}$$

which can be written as

$$\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = M_{int} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \text{where} \quad M_{int} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

by using the homogeneous coordinates (u', v', w') for (u, v) for which $u = \frac{u'}{w'}$ and $v = \frac{v'}{w'}$. If the intersection of the Z axis and image frame does not correspond to the origin of the image coordinate system, a translation should be incorporated to M_{int}, giving:

$$M_{int} = \begin{bmatrix} f & 0 & c_u \\ 0 & f & c_v \\ 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

where c_u and c_v are the 2D translation parameters in the u and v axis directions respectively. M_{int} is known as the intrinsic camera matrix expressing the camera intrinsic parameters [34].

The above camera model in Equation (4.8) assumes that the camera is located at the origin in the real-world and that it is pointing in the exact direction of the Z axis, but that is not generally the case. The location of the camera in the real-world is defined by the extrinsic camera parameters M_{ext}, whose values are its rotation R and 3D translation t relative to the origin:

$$M_{ext} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{33} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \quad \text{where} \quad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{33} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad \text{and} \quad t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{4.10}$$

By knowing these intrinsic and extrinsic camera parameters, it is possible to model the mapping from any point in the real-world to frame coordinates through the following equation:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = M_{int}\ M_{ext} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.11}$$

where X, Y, Z are the 3D coordinates of a point in the real world, u, v, are its coordinates in the frame in pixels and M_{int} and M_{ext} are the camera intrinsic and extrinsic parameters respectively [34].

4.6.2 Calibration

Finding the intrinsic and/or extrinsic camera parameters is performed through the process of camera calibration.

Intrinsic camera parameters

Intrinsic camera parameters can be estimated by varying M_{ext} (i.e. moving the camera) and knowing the correspondences of 3D coordinates in the real-world and 2D coordinates in the image frame. In this situation, M_{int} can be inferred as it is the only constant parameter in Equation (4.11). The same task can be achieved by taking multiple views of a planar surface. A common approach is to use a 2D chessboard image, where the dimensions of squares and their corner positions in the real-world are known, and their 2D location in the frame can easily be detected automatically because of the structure of the pattern. This is the approach that we adopted in our project, using the OpenCV implementation based on [132; 133]. Intrinsic parameters only need to be calculated once, before using the system for the first time, since they are inherent to the camera and do not change. This can be done off-line and does not represent any real-time overhead for our system.

Extrinsic camera parameters

Once the intrinsic camera parameters, M_{int} are known, M_{ext} can be calculated by using Equation (4.11) again, provided that correspondences between points in the frame and points in the real world are known. In our setting, the tennis court dimensions are known (see Section 2.1). We define the real-world coordinate system with the origin at the centre of the court and express intersections of the court lines in this newly defined coordinate system. When the cameras are installed in the court, M_{ext} is found by minimising the sum of squared distances between the coordinates of twelve points (court line intersections) in the frame and the projected points. Calculating M_{ext} needs to be done only once every time the cameras are installed in the court. It can be done before starting the match or training session without adding any overhead to the system when in-play.

4.6.3 Triangulation

Now that the camera's intrinsic and extrinsic parameters are known, triangulation can be performed. Triangulation is referred to the process of finding the 3D coordinates of a point, given its known position in images taken from at least two different views. As shown in Figure 4.16, a given point a in a 2D image corresponds to a ray R_a in 3D space. This ray can be defined using the camera parameters M_{int} and M_{ext} from Equation (4.10). R_a is a vector relative to an origin t (from M_{ext}) and defined as:

$$R_a = (M_{int}\ R)^{-1} \begin{bmatrix} a_x \\ a_y \\ 1 \end{bmatrix} \tag{4.12}$$

where R is the rotation matrix shown in Equation (4.10). Therefore, if we only have one view of an object (only one 2D image), we know that the object lies on that 3D ray in the real-world, but we lack sufficient information to know its exact location. However, if we know the location of an object in two 2D images taken from different angles (a and a' in Figure 4.16) it is possible

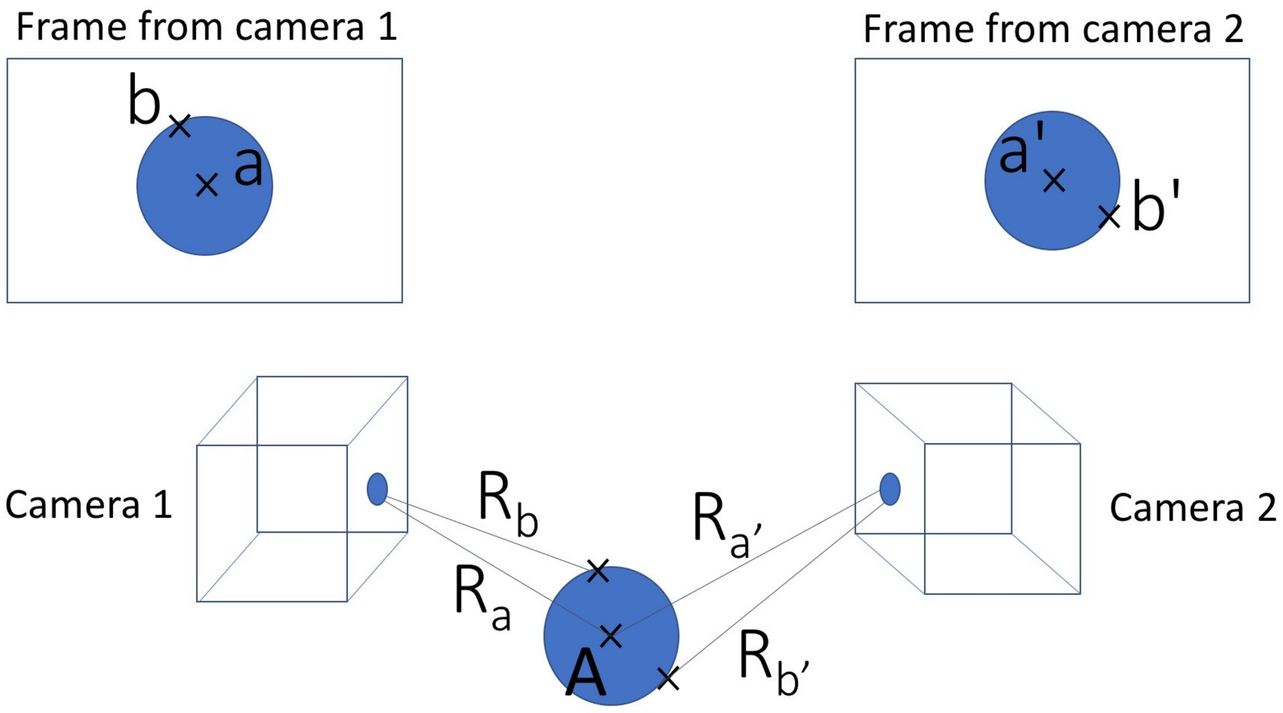


Figure 4.16: Triangulation.

to find the 3D coordinates of that point by calculating the intersection of their rays (R_a and $R_{a'}$ in Figure 4.16). Unfortunately, the rays (R_b and $R_{b'}$) might not intersect due to noise in the camera calibration stage or in detecting the 2D coordinates (e.g. if we detect b and b' instead of a and a'). Finding the correct 3D coordinates thus becomes a challenging problem for which different techniques can be applied (direct linear transformation [134], polynomial [135] etc.) We have employed the mid-point method [136] because of its efficiency and the speed requirement of this project (otherwise, more sophisticated methods such as polynomial triangulation could be used). The mid-point method is simple and the 3D point A is the best fit solution to:

$$min[d(R_b, A)^2 + d(R_{b'}, A)^2] \tag{4.13}$$

where $d(R, A)$ is the Euclidean distance between R and A.

4.6.4 3D data

The results for the 2D ball and player detection experiments can be accurately analysed because ground truth data was manually annotated in the videos. However, this is not the case for 3D data as this requires additional equipment that was not available to us (e.g. RFID sensors or lasers). To assess the accuracy of the 3D ball location inferred by our system, we analysed our results in two ways: qualitatively and quantitatively. First, we projected the detected 3D ball locations onto a 3D viewer (as shown in Figure 4.3) and replayed the recorded sequences. We visually inspected our 3D reconstruction and ensured that it was consistent with the recorded video sequences. To assess our results also quantitatively, we calculated the 3D location of the ball from the annotated 2D ground truth data and compared it to our inferred 3D locations, calculated from the automatically detected 2D ball coordinates. Figure 4.17 shows these results with the error defined as the Euclidean distance between the 3D locations calculated from the ground truth and our automatically calculated 3D locations. As can be seen in Figure 4.17, 80% of the 5 342 frames (mentioned in Section 4.4.3) had an error smaller than 10cm. It is important to note that even though the 2D ball location might not be obtained for each camera for every frame, 3D data is still produced for every single frame as it integrates data from all cameras and estimates the values for frames in which the ball was not detected.

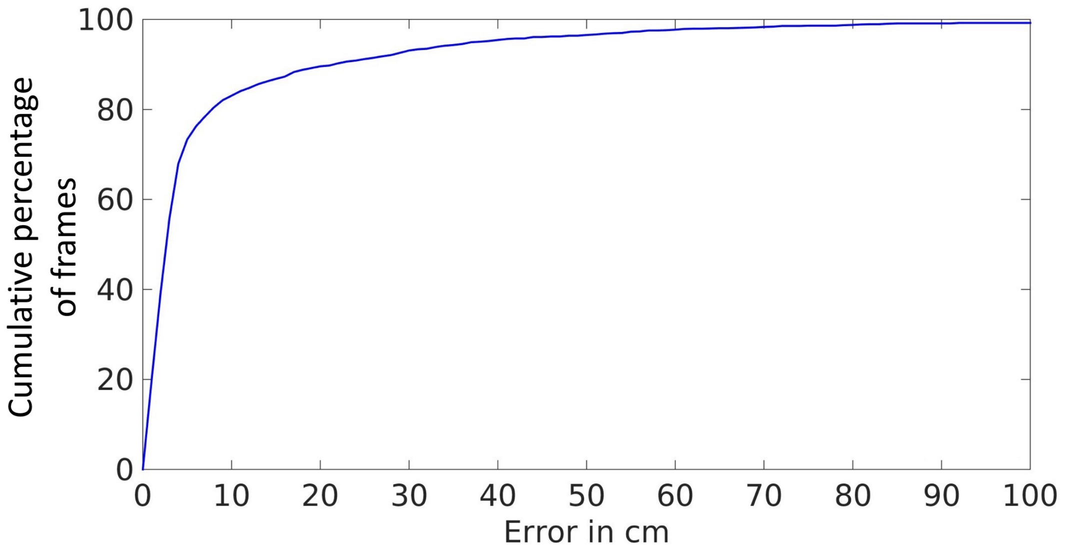


Figure 4.17: Accuracy of 3D ball detection in cm.

The same analysis was run for the player 3D location, and the error is always less than 25cm and less than 6cm for over 80% of the frames, as shown in Figure 4.18. It is interesting to note how Player 1 is detected with higher accuracy than Player 2. A posterior review of the videos revealed that there was a small focus error in one of the cameras, which was closer to Player 2.

Keeping in mind that the player dimensions are typically approximately 50cm $\times$170cm $\times$20cm and the ball diameter is in the order of 7cm, we consider our results to be of good quality.

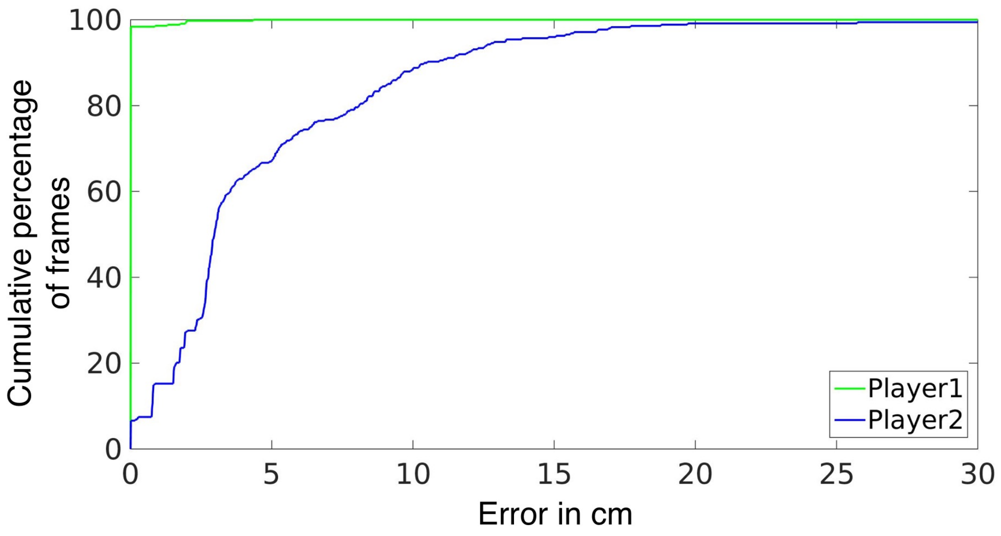


Figure 4.18: Accuracy of 3D player detection in cm.

4.7 Processing Speed

From the beginning of this chapter we emphasised the importance of our system to run in real-time. This means that the video processing time should be shorter than the video duration. Our cameras are 60fps, and our processing rate should be faster than 240fps as the system is comprised of 4 cameras. We randomly selected six videos and ran our system four times for each video. The average frame processing rate of the algorithm was 244.27$\pm$6.28fps[13] in total, when using the hardware described in Section 4.2.1. The speed of our system greatly benefits from the thread parallelism explained in Section 4.2.2; it runs more than three times faster than when the system is run in serial mode (72fps).

[13]Mean $\pm$ one standard deviation.

4.8 Conclusion

The main contribution of the work presented in this chapter is the design and implementation of a system able to collect 3D spatio-temporal data from a tennis match or training session in real-time from affordable commodity hardware, fulfilling the criteria listed in Section 4.1. We combined an adaptive background generation for detecting the players through background subtraction with Machine Learning techniques for the ball detection. Ball candidates were obtained through frame differencing and finding contours. Then, Random Forest classifiers were used to select the best candidates based on ball shape and trajectory. Finally, a mid-point triangulation method was used to calculate the 3D coordinates of the player and ball from their 2D locations in the frame.

We were able to locate the tennis ball with an error lower than 2 pixels for over 75% of the frames and not exceeding 6 pixels for most frames. The centre of mass of the player is always detected within its ground-truth bounding box and a maximum displacement error of 20% of the bounding box size for most frames. In 3D, both the ball and player were detected within an error of up to 10cm for most frames. In addition to this, the design of the system is modular, with the capability of incorporating improvements both in terms of speed and accuracy.

Another contribution from this chapter is the survey of related work and comparison of state-of-the-art research in ball and player detection, which has proven to be challenging due to the lack of benchmark datasets and standardised accuracy measures. We also placed our results within this context. Finally, with this present work, we also offered a quantitative analysis of our results. We introduced novel metrics for the objective quantitative evaluation of court, ball and player detection in tennis. We hope this will motivate other researchers in the area of extracting spatio-temporal sports data from videos to use similar objective quantitative metrics.

Chapter 5

Thinking Like a Coach: Data Analysis and Visualization

In Chapter 3 we have seen how spatio-temporal data analysis can ultimately lead to gaining new insights in sports analysis and prediction, addressing the Modelling Layer of our framework from Figure 1.11. Then, Chapter 4 focused on the Vision Layer and presented a new unobtrusive and portable system able to collect spatio-temporal tennis data in real-time using commodity hardware. The data that we collected are the positions of the centre of mass of the players and that of the ball during training sessions where the two players are a coach and a student. In this and the next chapter, we will concentrate on the Classification Layer, which lies in between the other two and deals with obtaining higher level tactical information. In this layer, we want to go a step further and not only extract the information that has been shown to be valuable in Chapter 3, such as ball landing positions, but obtain additional information that we believe to be relevant in analysing the play patterns, strengths and weaknesses of players, such as shot height and timing, or the type of action being played. Specifically, the aim of this chapter is to show that data collected from our affordable, accessible technology can also lead to an interesting analysis of the game, even if the data contains some noise, and to provide visualisations of this analysis since these are essential to have a real impact in sports. This chapter is based on our paper published at the Large Scale Sports Analytics workshop of the ACM SIGKDD conference in 2016 [12].

5.1 Tennis Data Visualisation

Good data visualisation can often make the important features of data stand out and lead to an improved analysis of the data [137]. Investigating different ways in which to present the data goes hand in hand with a deeper analysis of it and may require inferring additional information, for example, calculating player movement speed from player location sampled at two different time points. Sports is highly suitable for producing visualisations because the relevant objects are known (generally the players and ball) and it is easy to define the boundaries of the space in which they appear, i.e. courts with standard dimensions. The visualisation and analysis of spatio-temporal data in sports has three primary attributes, reviewed in [138]:

- Responsive: the user makes queries for specific game events or statistics.
- Exploratory: data is displayed to help the viewer get insights into the game.
- Informative: data is presented to enhance spectators' experience.

In this work, we focus on the exploratory aspect of analysis and visualisation, where the challenge resides in clearly presenting the data in a manner that contributes to the final purpose, in our case understanding players' strategies, strengths and weaknesses.

5.2 State-Of-The-Art Tennis Data Visualisation

State-of-the-art tennis visualisations include Hawk-Eye [139] and LucentVision [1; 140]. Hawk-Eye displays trajectories of the player and ball, as well as ball landing positions. In LucentVision, users can make queries that are based on:

- **Score:** display data in relation to the score. For example, show winning shots for points that have been won by a given player.
- **Statistics:** display information about specific statistics of the game. For example, present the fastest serves or shots.

- **Space:** show events relative to some spatial constraints. For example, display the ball landing positions from net approach shots.

The visualisations resulting from the queries are 3D visualisations (e.g. reconstruction of ball trajectories) or court maps (e.g. heatmaps of player position), as shown in Figure 5.1.

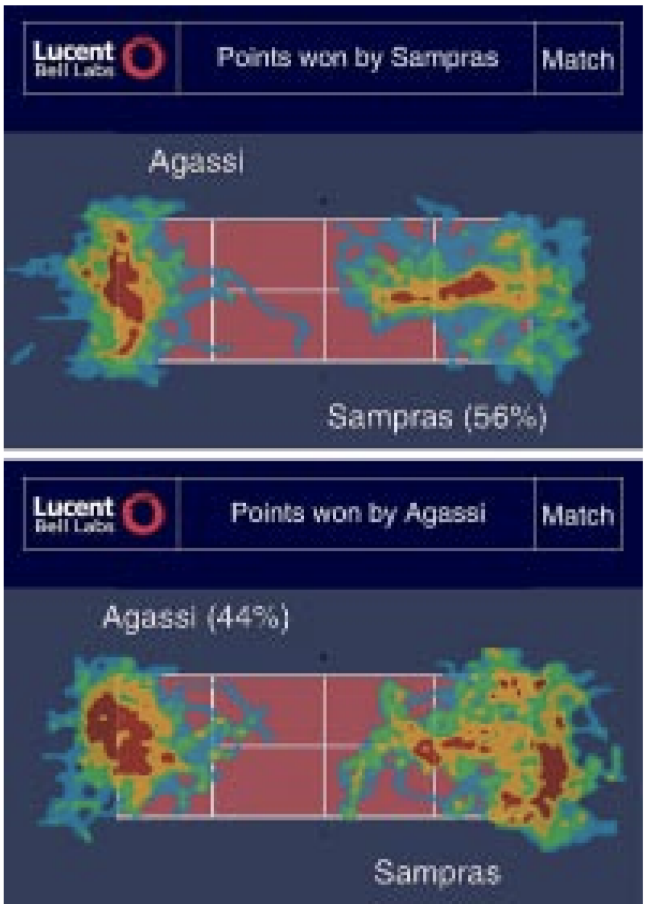


Figure 5.1: LucentVision heatmap from score query, from [1].

Besides these commercial products, another interesting tennis visualisation was presented by Damien Demaj [141]. With his background in cartography, he applied geospatial data analysis techniques to spatio-temporal tennis data. For example, his paper offers visualisations with feature overlay showing simultaneously serve impact locations, how important the serve was (according to the current score) and its success. In this paper, this visualisation is applied to the London Olympics Gold Medal match where Andy Murray defeated Roger Federer, leading to the conclusion that “Federer served with more spatial variation during the match. Murray, however, served with greater spatial variation at key points during the match”.

5.3 Objectives and Contributions

This chapter has three main objectives. First, provide a visualisation of our results to reflect the ball trajectories and player movement profiles using novel forms of presenting the data in addition to traditional ones; this is shown in Section 5.5. Secondly, combine ball and player information to detect when the ball is being hit by each of the players, as discussed in Section 5.6. Thirdly, automatically generate an overview of a match/training session with information inferred from the previously collected data, as described in Section 5.7. To better accomplish our objectives and overcome noisy and missing ball data, we incorporated a post-processing step in our data collection system to refine the automatically extracted ball trajectories, which will be described first, in Section 5.4. Interestingly, the analysis and visualisations shown in this chapter are applied to data extracted from our own data collection system presented in Chapter 4.

5.4 Data post-processing

For this chapter, the data from 55 videos of length between 100 and 1 600 frames each obtained using the system presented in Chapter 4 is analysed. It is important to note that the 2D and 3D data obtained from these videos contain some noise due to inaccuracies in the detection or phenomena such as occlusions. As previously mentioned, an objective of this work is to show how this can be overcome to achieve a useful analysis, even when data is collected from an affordable system.

Each element in the ball data corpus contains the 3D coordinates of the detected ball position and its associated frame number. To reduce noise and fill-in missing ball trajectory data, a post-processing step had to be added to our system. It consists of two stages: first, outliers are removed, and then missing points are filled in by interpolation.

5.4.1 Remove Noise

From the spatial distance between two consecutive 3D ball coordinates and their time difference (from the frame number), outliers are detected and removed. These correspond to objects moving unrealistically fast, at a speed that is above a threshold based on the maximal velocity that is realistically achievable by the tennis ball during a tennis match, i.e. 270km/h[1].

5.4.2 Interpolation of Missing Points

After outliers have been eliminated, the next step is to interpolate missing points using the law of physics applied to ball trajectory. Interpolation is a mathematical method for generating new data points based on a discrete set of known data points. The known data points are used to find the interpolation function p and this function is used to generate the new data points.

Finding the interpolation function p for our data is achieved by modelling the ball trajectory. Assuming that the tennis ball is not spinning, its trajectory during a match is composed of one or more consecutive parabolas, each of which starts when the ball bounces or is hit by a player. This applies to data in the x, y and z dimension separately across time (frame number). To interpolate missing values in the ball trajectory, mostly due to the removed outliers, we need to identify the frame number of data points representing the beginning or end of a parabola and calculate the parabola parameters. A data point from the ball trajectory data is considered to be at the beginning/end of a parabola if:

- It has the minimal height above the ground relative to the two previous and two following ball positions.
- It has the minimal/maximal y coordinate (along the length of the court) relative to the two previous and two following ball positions.
- It is at the beginning or end of the data stream.

[1]From `https://en.wikipedia.org/wiki/Fastest_recorded_tennis_serves`

- It has a gap of more than 4 frames or 0.5m in the height dimension with respect to the next ball trajectory data point.

Once the limits of the parabolas are known, a least squares polynomial fit is calculated for each parabola to obtain p. A polynomial fitting of degree 2 (quadratic fitting) can be expressed as follows:

$$p(x_i) = a_2 x_i^2 + a_1 x_i + a_0 \tag{5.1}$$

where a_n is the coefficient of the n^{th} degree term and x_i are the data points. From Equation (5.1), we can obtain the following system of linear equations for n points:

$$\begin{bmatrix} x_0^2 & x_0 & 1 \\ x_1^2 & x_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n^2 & x_n & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_n \end{bmatrix} \tag{5.2}$$

In our case, we know X and z and want to find the a coefficients. This corresponds to solving a linear system in which X and z are constant and a is the variable. For clarity of our description, the coefficients X and z are renamed to A and b respectively and the variable a to x. With this, the equation $Xa = z$ from Equation (5.2) becomes the classical expression of a linear system:

$$Ax = b \tag{5.3}$$

A linear least square fitting to Equation (5.3) is solved using the QR decomposition method, obtained with Givens rotations. This is described in detail in Appendix I. Once the coefficients A of the distinct parabolas are known, existing points are adjusted using the parabola coefficients. Finally, cubic spline interpolation is used to estimate missing values.

5.5 Player and Ball Location Visualisation

After eliminating outliers and interpolating missing values from the 3D ball positions, these and the 3D player positions having been extracted previously (cf. Chapter 4), the next step is analysing the data. The rest of the chapter deals with our approach for analysing and visually representing this data. This section approaches this objective by using a visual representation of the data to reveal important information about the locations of relevant objects. Heatmaps are a graphical representation particularly suited to represent location data. They can be used to visualise the frequency of occurence of a player or ball being in a particular area of the tennis court. This is done by creating a matrix with the scaled court dimensions (in discrete form) and using colour to represent the frequency of an object being in a particular area. The heatmaps that we present are in matrix form, and there are a number of different ways in which the tennis court can be divided. [142] discusses three ways in which a basketball court can be divided, and two of these can also be applied to tennis:

- **Cartesian:** the court is divided into equally-sized squares.
- **Zoned:** the court is divided into strategically significant areas.

Once the court is divided into smaller areas, we represent the time spent by the player or ball in that area through a colour reflecting its normalised frequency.

5.5.1 Player Position Profiles

Heatmaps to visualise movement patterns of tennis players can be used to assess their performance and compare patterns of movement between them, as shown in [143]. In this section, we visualise player positions and discuss their interpretation with regard to players' style and performance. Our analysis is restricted to the players in our videos, but we believe that the work presented in this section can be extended to a larger corpus of data to allow comparison between more players.

Cartesian View

Figure 5.2 is a heatmap of the player's positions across all the videos analysed for a Cartesian division of the court. The court and its immediate surrounding area are divided into a grid of equally sized squares of size corresponding to 20 cm $\times$20 cm and the percentage of time spent by the players in each of these squares is shown. The figure shows that players spend most of their time in the centre behind the baseline, occasionally coming closer to the net. This pattern is in accordance to what we observed in the videos and what could be expected to be the most common areas for a tennis player to be in a training scenario and for the serve.

Zoned Heatmap View

There are a number of ways in which the tennis court can be divided to reflect strategically important locations, and in our work, we look at two different zone divisions. The first is described in [144] (shown in Figure 5.3) and each half of the court is divided into three zones:

- Defensive zone: this is the zone furthest away from the net.
- Neutral zone: this is the middle zone, and it is sometimes called the transition zone.
- Offensive zone: this is the zone closest to the net.

The pattern in Figure 5.3 shows that players spend most of their time in the defensive zone (98.92% for the player on the left and 74.97% for the one on the right), as would be expected in a training session where offensive shots are less likely. We can also see that the player on the right, who was the one being coached, spent some time in the offensive zone (17.57%), while the coach did less (5.02%).

The second court division is described by Ron Waite, a professional tennis coach, in his column on Tennisserver [145]. The different areas, starting from the zone furthest away from the net and moving towards to middle of the court are:

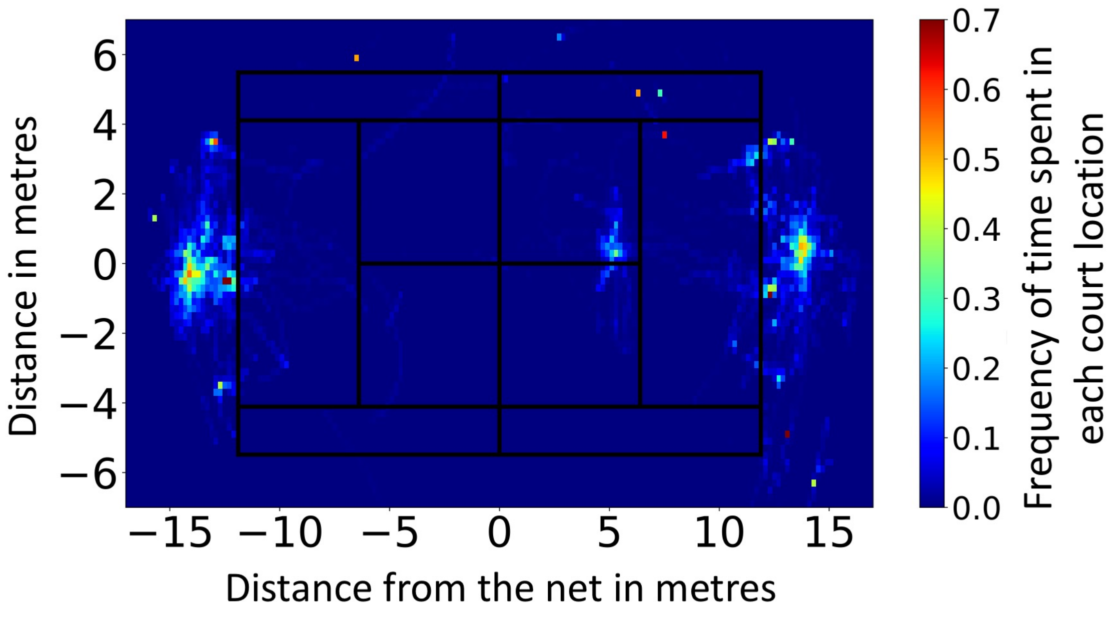


Figure 5.2: Cartesian heatmap of player position.

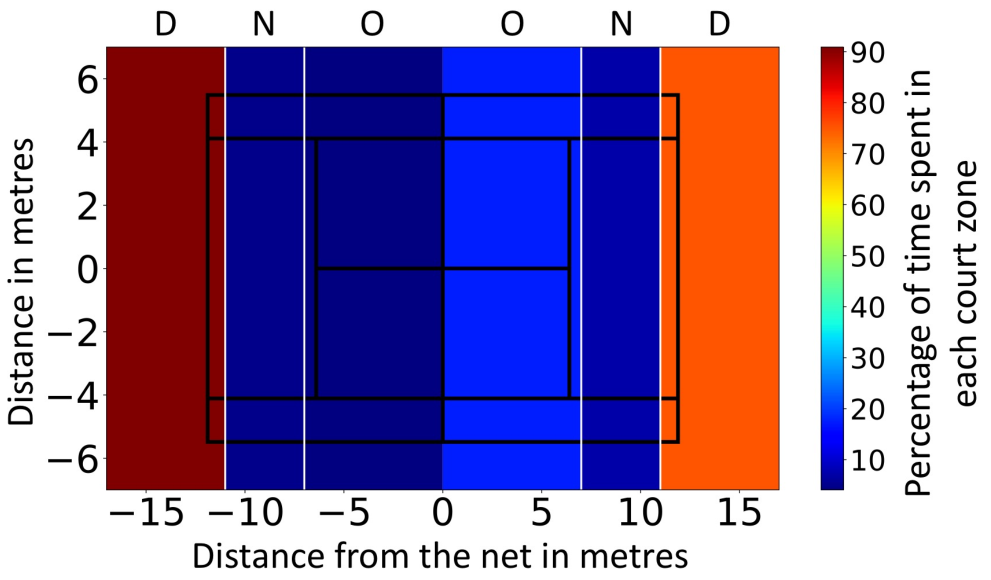


Figure 5.3: Zoned heatmap of player position, using the zones described in [144]. D - Defensive zone, N - Neutral zone and O - Offensive zone.

- Defensive zone: here the player is far behind the baseline and is mainly trying to hit the ball back without aspiring to win the point.

- Rally zone: this is where most groundstrokes are hit.

- Offensive zone: stepping closer to the net puts players in an offensive position where the range of trajectories that can be produced is larger, and the opponent has less time to react than for shots from the two previously mentioned zones.
- No man's land: this is the zone between the service line and baseline. This zone is where players are generally thought to be most vulnerable.
- Put away zone: this is the area from which put-aways (to win the rally) are most likely to occur. A put-away is a skilful shot that ends the point because the opponent cannot reach the ball.
- Volley zone: this zone is the closest to the net, optimal for playing volleys.

Figure 5.4 shows the players' patterns of position in the newly defined court zones and the specific values are reported in Table 5.1. As expected in a training session, both players spend most of their time in the rally zone and the next most visited zone is the offensive zone. As above, we can see the player on the right (the student) spends more time in offensive areas such as the putaway zone and volley area.

Area	Occupation % left player (coach)	Occupation % right player (student)
Defensive zone	1.18	1.83
Rally zone	69.14	51.40
Offensive zone	22.77	25.02
No man's land	2.60	1.95
Put away zone	2.38	13.33
Volley zone	1.93	6.46

Table 5.1: Percentage frequency of players' occupation of each court zone from Figure 5.4.

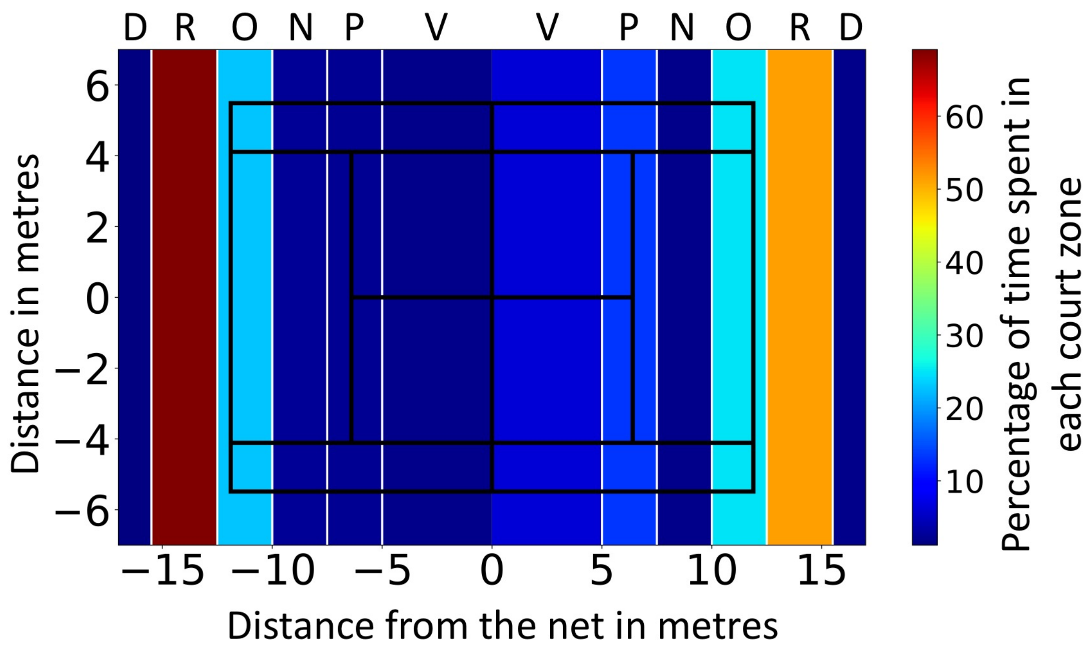


Figure 5.4: Zoned heatmap of player position, using the zones defined in [145]. D - Defensive zone, R - Rally zone, O - Offensive zone, N - No man's land, P - Put away zone and V - Volley zone.

5.5.2 Tennis Ball Positions

In regards to the visualisation of tennis ball locations, we have also represented it in the two different ways introduced in Section 5.5: Cartesian view and zoned heatmap view.

Cartesian View

The heatmaps to represent ball locations in a Cartesian view are designed as for the players (Section 5.5.1). Ball location is represented in two different ways as the frequency of:

- Ball position: Figure 5.5 is a heatmap of the areas traversed by the tennis ball.
- Ball landing positions: the visualisation presented in Figure 5.6 takes into account the height dimension of the ball location, and the frequency map represents the positions in which the height is approximately 0, being the locations where the ball hits the court.

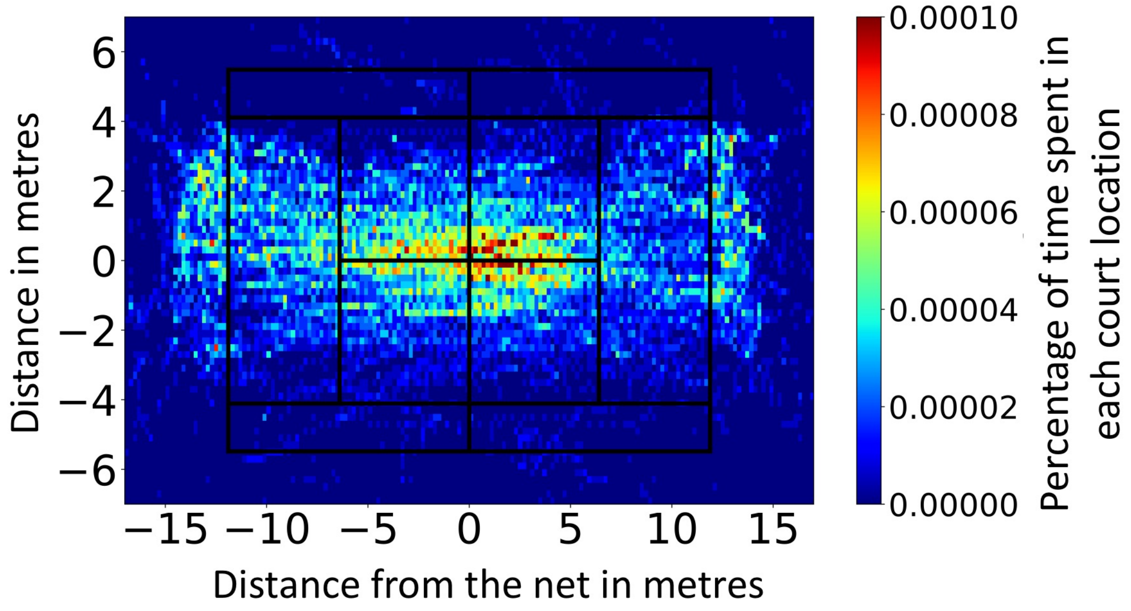


Figure 5.5: Cartesian heatmap of ball position.

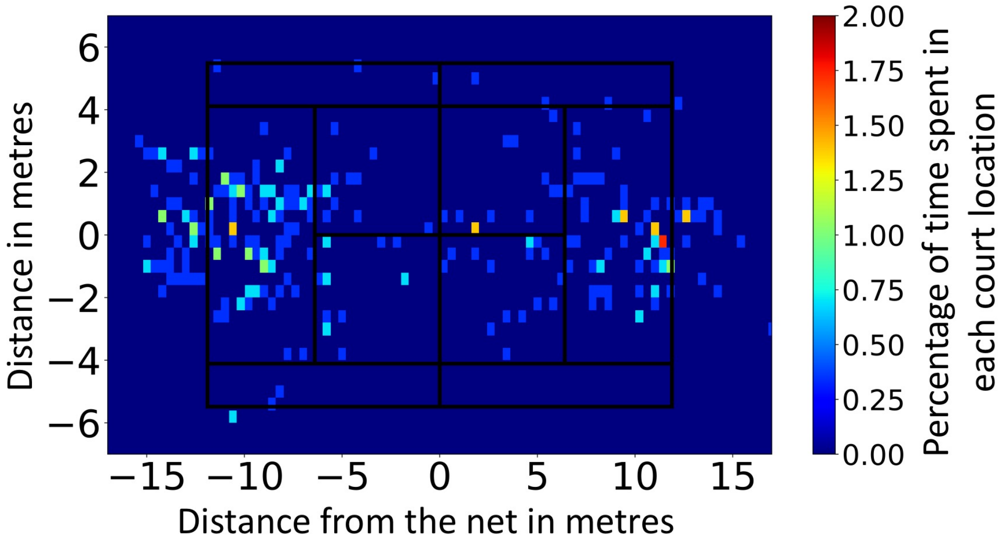


Figure 5.6: Cartesian heatmap of ball landing positions.

Figure 5.5 shows that the most common position of the ball is near the centre of the court. As seen in the videos, the ball travels at different angles but crosses the net close to the middle in most cases and diverges to a less localised area when getting further away from the net.

The first observation from Figure 5.6 is that the player on the right is making more errors (hitting the ball outside of the court) and the landing points of his shots are more spread out. In contrast to this, the player on the left is hitting more precise deep shots, generally to the centre of the court. This pattern fits the observations from the recordings in which the player on the left is coaching the one on the right.

Zoned Heatmap View

The court zones that are used to analyse the tennis ball landing positions are different from the ones used for the players' positions. For instance, it is common for a player to stand behind the baseline, but a ball landing in that area is an error. The court zones for ball landing positions, as described in [144], are as follows:

- Grind zone: This starts at the baseline and extends for 3.9 m towards the net. When a ball lands in this zone, the most effective return shot is a deep return.
- Torment zone: This is the middle zone that is at a distance of 3.9 m from the net and from the baseline. The shots in response to landings in this zone can be building shots[2] or attacks[3].
- Obliterate zone: This zone goes from the end of the torment zone up to the net. An offensive response, such as a volley, is generally expected as a return from these shots.

Figure 5.7 shows the ball landing positions in relation to the zones defined above and Table 5.2 shows the same results in more detail. Both players send most shots to the grind zone, as would be expected for non-competitive rallies, which was generally the case in the videos that we recorded. Aligned with this, both players also send the least number of shots to the obliterate zone, which corresponds to the most offensive actions. Finally, this diagram (Figure 5.7) also shows more clearly what was observed in Figure 5.6: the player on the right is committing more errors than the one on the left.

[2]Shots that create chances for a player to attack.
[3]Shots that create chances for a player to win the point.

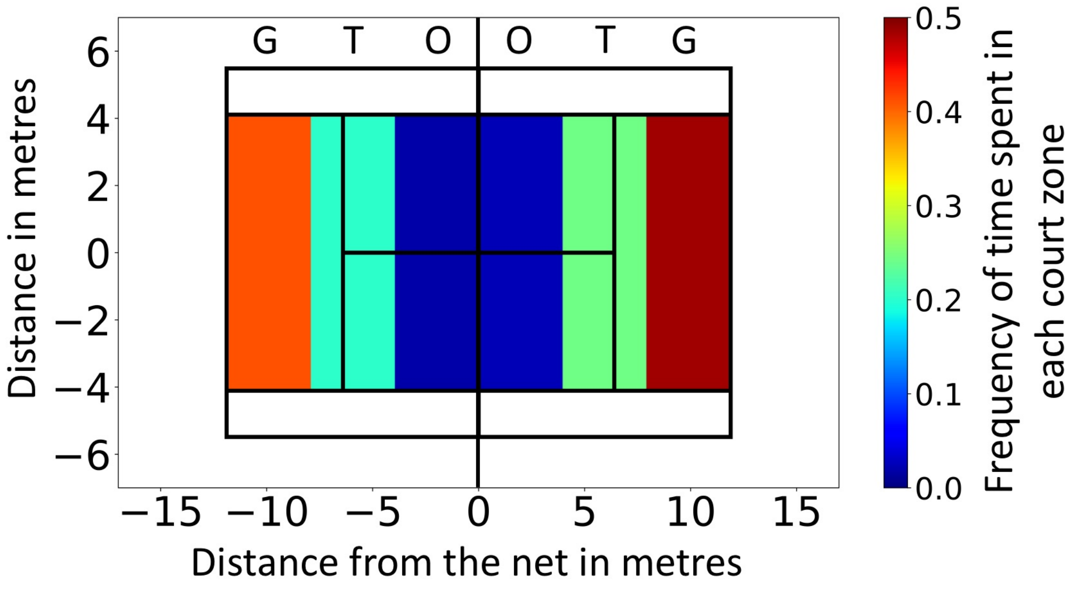


Figure 5.7: Zoned heatmap of ball landing positions. G - Grind zone, T - Torment zone and O - Obliterate zone.

Area	Left player shot	Right player shot
Grind zone	48.48	41.14
Torment zone	24.24	20.25
Obliterate zone	2.53	1.90
Out	27.27	34.18

Table 5.2: Percentage frequencies of ball landing position in court zones, the columns represent shots initiated by each player, from Figure 5.7.

Air Zones

The air zones of a shot reflect the height of the tennis ball as it passes the net and different air zones should be aimed for by a player depending on his/her current position. As a general rule of thumb, the player making the stroke should aim at a shot with a height above the net of 0.6m–0.9m when inside of the court, 0.9m–1.5m if near the baseline and 2.4m–3.0m if the player is positioned behind the baseline [144].

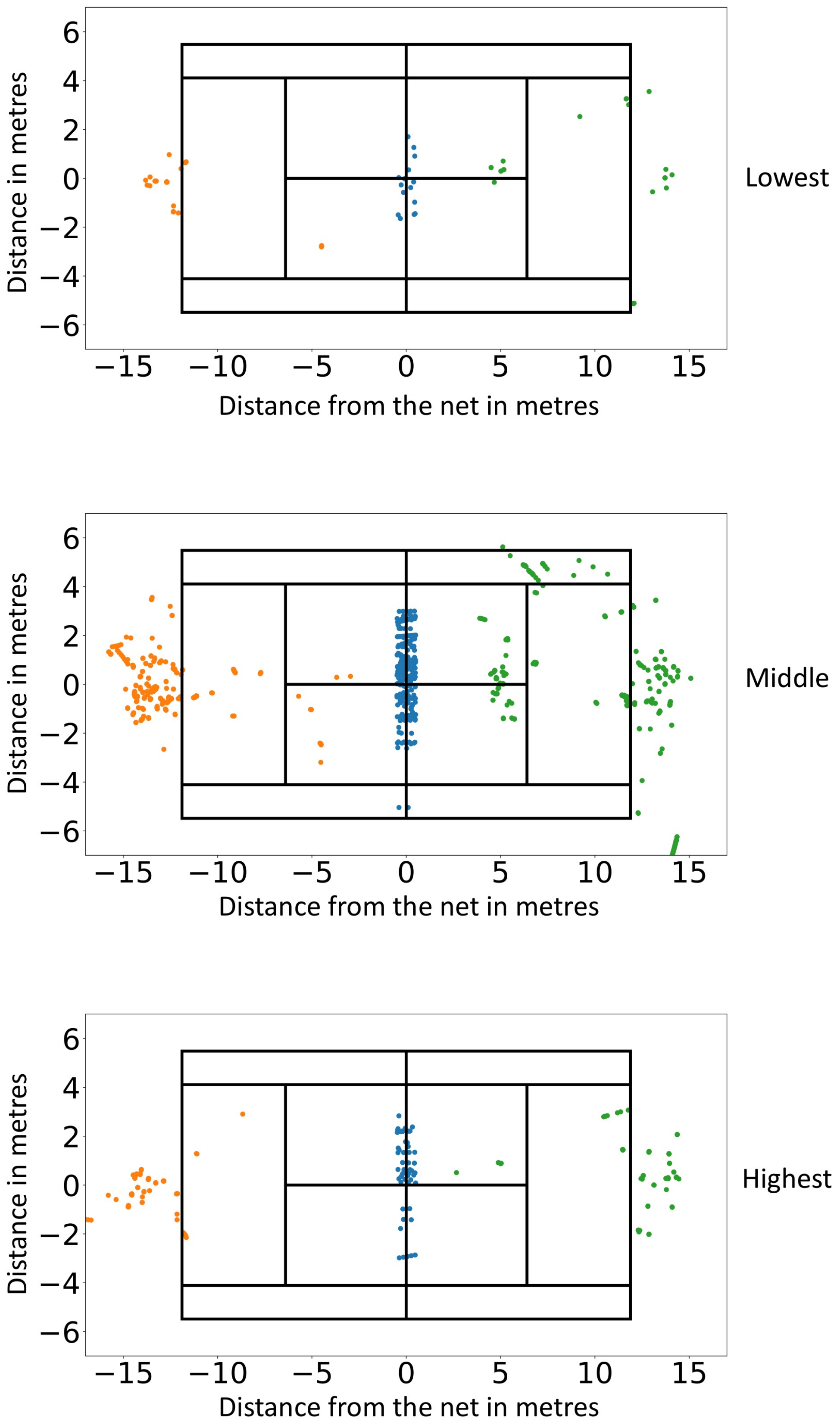


Figure 5.8: Ball and player positions for the ball crossing the net in air zones 1, 2 and 3, from top to bottom. Coach in orange, student in green.

Figure 5.8 shows three diagrams, one for each air zone, with the location of the players and the ball when the latter is crossing the net. According to the description given above, the players should be inside the court, near the baseline and behind the baseline for air zones 1, 2, and 3 respectively (from top to bottom in the figure). We can see that the player in orange follows this trend and, as the ball goes to higher air zones, he moves further away from the middle of the court, but this is less clear for the green player. Interestingly, the player in orange, who follows closer the theory of court zones, is the coach, probably because he is more experienced than the one in green (the student). From this, we see that visualisations of the kind shown in Figure 5.8 can be an indicator of the expertise of a player.

5.6 Visualising Players and Tennis Ball Combined

So far, we have looked at general location patterns for the ball and players. In this section, the data from the players is combined with that of the tennis ball in order to take the analysis a step further and detect stroke events.

5.6.1 Detect Events

To find out when the ball is hit by a player we calculate the distance between that player and the ball. This is calculated as their 3D Euclidean distance such that the distance between two points a and b, such that $a = (a_x, a_y, a_z)^T$ and $b = (b_x, b_y, b_z)^T$, is:

$$\sqrt{(a_x - b_x)^2 + (a_y - b_y)^2 + (a_z - b_z)^2} \tag{5.4}$$

When plotting the Euclidean 3D distance between the ball and a player over time, values close to zero in the y component correspond to the ball being close to the current player, and they can be interpreted as the frames in which he/she is attempting to hit the ball. Similarly, maximal y values can be seen as the ball being hit by the other player. Figure 5.9 is a scatter plot of the player–ball 3D distance over time for one of the players using our automatic video detection from Chapter 4. This figure shows the results for one of the videos, but other videos gave a similar output. Once we know the player–ball 3D distance, stroke events are found by selecting the local maxima and minima from this data. These are shown in Figure 5.9 with red and green circles for the two players respectively and the frame number at which they occur is shown in the horizontal coordinate.

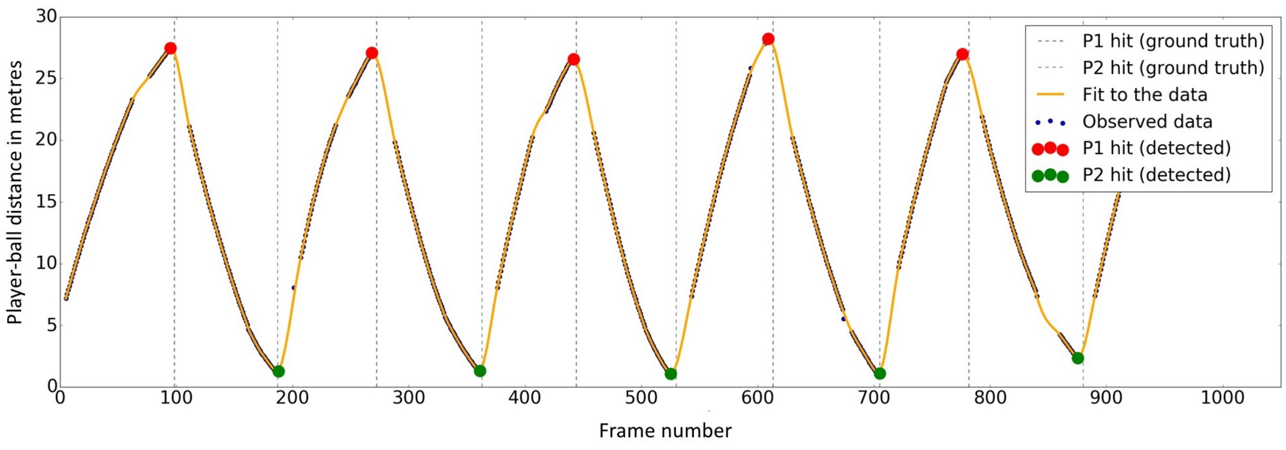


Figure 5.9: Distance between the player and tennis ball, measured from player 2 (P2).

5.6.2 Evaluation

To evaluate these results, we manually recorded the frame numbers at which the ball is hit by each of the players for one rally; these are represented by the vertical lines in Figure 5.9. Tables 5.3 and 5.4 show the comparison between the ground truth data and our automatic action detection(based on maxima and minima in the distance) for each player. We can see that in the current example, the error in the time at which an action occurs is between 0 and 5 frames, corresponding to a maximal error of 83ms in real time, with an average error of 53ms.

5.7 Extracting Player Spatio-Temporal Statistics

Finally, we have computed a number of player statistics to describe a training session or match and enable the comparison between players' performance. This collection of statistical data can be obtained automatically in our system and is very valuable for understanding the players'

Stroke number	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
Measured time (in frames)	95	268	442	609	776
Real time (in frames)	98	272	444	613	781
Error (in frames)	3	4	2	5	5
Error (in ms)	50	66	33	83	83

Table 5.3: Player 1 stroke detection for one rally.

Stroke number	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}
Measured time (in frames)	188	361	525	705	875
Real time (in frames)	187	363	530	705	880
Error (in frames)	1	2	5	0	5
Error (in ms)	16	33	83	0	83

Table 5.4: Player 2 stroke detection for one rally.

trends, strengths and weaknesses and comparing them with other players. Table 5.5 presents this information for a sample of 10 of the videos analysed and shows:

- The average player speed when moving on the tennis court. This is calculated by dividing the total distance travelled by that player during the training session (including the time during which the player was stationary) by its total duration. The distance travelled by the players is calculated as the 2D Euclidean distance where the distance between two points a and b is:

$$\sqrt{(a_x - b_x)^2 + (a_y - b_y)^2} \tag{5.5}$$

- The percentage of court surface covered by each player. To represent the area covered by a player, we divide the court and its immediate surrounding area into a grid of equally sized squares of size corresponding to 40 cm × 40 cm on the court (similar to Figure 5.2)

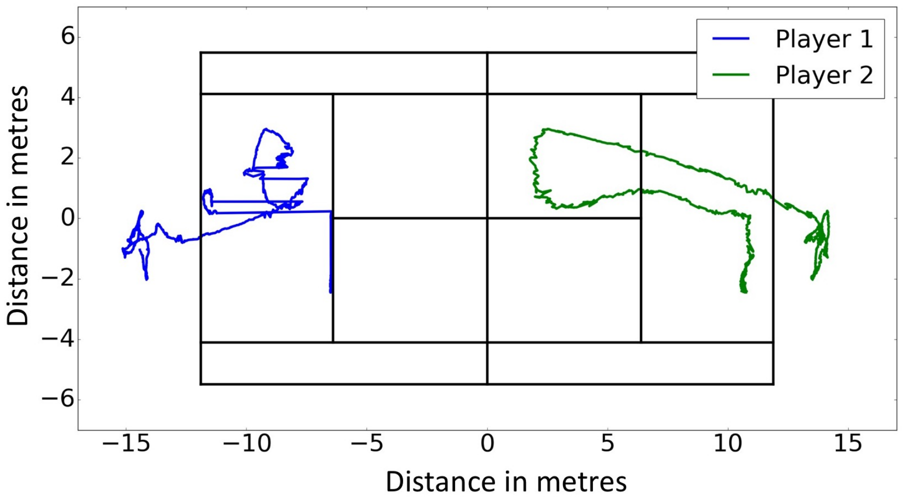


Figure 5.10: Players' trajectories during one rally of 16 shots. Player 1 is the coach and Player 2 the student.

and calculate the percentage of squares in the grid where the player has been at least once.

- The average speed of the strokes initiated by each player, which is calculated as for the player speed, as the total distance over the total duration. Our experiments did not include serves.

As an example of the information provided by Table 5.5 we can see that in video 77, even though both players have a similar average speed, Player 2 (the student) covers twice as much court area as Player 1 (the coach). This is illustrated in Figure 5.10, which shows the trajectories produced by each player in that video, and where we can see that Player 2 covered a larger court area by entering the service box.

5.8 Conclusions

This work, in conjunction with our system, presented in Chapter 4, presents an end-to-end pipeline for the spatio-temporal analysis of a tennis match/training session using a low-cost data collection system. The first contribution of this chapter is the pre-processing of tennis ball location data. The affordability of our technology from Chapter 4 has the disadvantage of

Video	Average speed		Area coverage[a]		Stroke average speed		Video	Number of
number	P1	P2	P1	P2	P1	P2	length	strokes
	(in metres/s)	(in metres/s)	(in %)	(in %)	(in metres/s)	(in metres/s)	(in s)	
25	0.8	1.7	1.1	1.4	21.1	17.4	14.2	10[b]
27	1.1	1.7	1.2	1.6	30.5	29.2	8.2	8
36	11.5	1.2	6.0	1.1	16.8	33.0	14.2	12
40	24.2	0.9	6.6	0.8	31.2	29.8	11.2	10
44	0.9	1.0	1.6	1.5	21.6	22.7	13.7	5[c]
53	0.7	0.4	0.4	0.2	30.3	40.9	6.6	3
64	1.1	3.0	2.7	7.4	39.5	31.7	21.7	14
68	0.8	1.3	1.8	2.2	25.6	30.0	19.4	12
70	2.1	0.9	5.1	2.4	29.3	31.3	23.5	15
77	1.7	1.9	5.8	10.8	28.2	25.3	28.4	16

Table 5.5: General statistics for 10 videos. P1 is player 1 (the coach) and P2 is player 2 (the student).

[a]i.e. Proportion of court surface.
[b]2 rallies
[c]3 rallies

including more outliers or missing data, when compared to highly sophisticated systems such as Hawk-Eye. To overcome this limitation, a method for removing outliers and filling in missing data is presented, which involves quadratic curve fitting and value interpolation.

Another contribution of this chapter is to show that the data that has been collected with a low-cost system can be used to gain insights into a match/training session and provide feedback to the players, coaches or audience through statistical information and an appealing and informative visualisation. More specifically, the contributions of the work presented here are the visualisation and analysis of:

- The player position heatmap in both a Cartesian view and in terms of tactically significant areas.
- The ball locations heatmap in both a Cartesian view and in terms of tactically significant areas, both in the flat 2D representation of the court and air zones separately.
- The extraction of high-level information such as the timing of the strokes of each of the players, the number of shots in a rally, the average ball speed or area covered by each player.

This is a contribution to the field in the sense that we present the data in novel ways with both the different court zones and aerial zones, reflecting strategically significant areas. It is accompanied by an analysis showing the insights that can be derived from the visualisations, which include an indication of the performance level of each player. For instance, in our analysis we are able to detect differences between the performance of the coach and the student.

Chapter 6

Learning the Technique: Machine Learning for Action Recognition

We started in Chapter 3 by showing different approaches to exploiting spatio-temporal tennis data to gain insights into patterns of tennis play and strategies and to predict future events, targeting the Modelling Layer of our framework in Figure 1.11. We then presented a system able to extract spatio-temporal tennis data in real-time in Chapter 4, addressing the Vision Layer of Figure 1.11. This was followed by Chapter 5, which focused on the Classification Layer, and showed how this data could be analysed and visualised to bring insights into a game/training session, generate high-level data and detect events such as the exact timing of a shot being played. The current chapter also concentrates on the Classification Layer of our framework, and the objective is to recognise the action being played from video data. We believe that recognizing which type of shot is being played and the quality of that shot would enhance the prediction and analysis capability of state-of-the-art modelling approaches. Another benefit can come from providing players with feedback on their strongest and weakest shots to guide them to where to put more effort while training. This chapter is based on our paper presented in 2017 at the CVPR Workshop on Computer Vision in Sports [13].

6.1 Challenges

The primary objective of this chapter is to be able to recognise the action being played from low-definition RGB videos. The first challenge that arises is defining what an action is and define the list of possible actions in the tennis domain. When we started this project, our plan was to use our own recorded videos and manually select and label clips of actions[1]. Creating our own dataset brought up a number of issues, revealing challenges in defining an action:

- Action delimitation: It is complicated to determine the exact moment at which an action starts or ends.
- Granularity: tennis shots can be classified into 'serve', 'backhand return', etc. but they can each also be further divided into four phases: 'preparation', 'swing', 'impact' and 'balance and recovery' [2]. Decisions have to be made about the granularity of the classes.
- Area selection: In a real-world environment, the player might be moving while performing an action. From this, two options are possible: always select an area of the same size with the player in the middle (i.e. a moving window in which the background will be changing) or select a larger area and keep it stable throughout the action, but the player will not always be at the centre of that area.

Beyond these, action recognition, in general, is a highly researched area [39; 146–148] in which some of the main challenges are [149; 150]:

- Environment: people wearing loose clothes, occlusions or changes in the environment such as lighting conditions, challenge action recognition. Also, the same action seen from different angles can have a very different appearance.
- Action definition: human actions are very diverse and include simple actions like walking or more complex action involving interactions with other people or objects. Even defining

[1]We later decided to use a publicly available dataset for the reasons explained in Section 6.3.1

[2]From Tennis Australia at `https://www.tennis.com.au/wp-content/uploads/2010/08/Stroke-and-Tactical-Fundamentals.pdf` , although other phases can be defined.

an action and determining its exact starting and end point is challenging, as highlighted above.

- Variability: the same action can be performed in different ways: slow or fast, long or short, in a large or short range (e.g. taking bigger steps whilst walking). Generalising action recognition is challenging.

6.2 Objectives

The objectives of this chapter are to achieve the following goals within the context of tennis:

- Recognize fine grained actions. Most sports action recognition tasks are based on detecting which sport is being played. However, we think that in order for action labels to be useful in more nuanced applications like professional training, these must be of fine-grained actions, rather than just a coarse-grained classification.

- Find a dataset appropriate for investigating the previous objective. We believe that it is very important to present the results of our work in the context of a publicly available dataset in order to enable future research to be compared to our method.

- Perform the action recognition directly from low-resolution RGB videos of actions played in-the-wild[3]. For our approach to be useful within the context of the framework presented in Figure 1.11 we need our system to be unobtrusive and affordable. This rules out the possibility of detecting actions from different types of data such as special depth cameras or wearable sensors.

- Develop a technique that is generalizable and can be applied to action recognition in contexts beyond tennis.

Section 6.3 will review previous work in the area of action recognition, firstly in the context of sports and secondly in more general contexts, revealing challenges of the previously listed

[3]Non-controlled environment, with moving background and/or changes in illumination.

objectives. Our approach will be explained in Section 6.4. The experiments will be presented next by introducing the dataset that we have selected for our experiments in Section 6.5 and then discussing the results in Section 6.6.

6.3 Previous work

6.3.1 Action Recognition in Sports

Even though vision-based action recognition is a highly researched area, its application to sports is still limited, mirroring the lack of benchmark datasets for this problem. Some of the most popular sports action datasets include UCF-Sport [151; 152] or more recently the Sports-1M dataset [148]. These datasets contain videos from many different sports, and their labels describe which sport is being played. In contrast to these, we are interested in detecting finer-grained actions, such as specific tennis shots (e.g. serve, backhand and forehand) or even the stroke sub-type (e.g. flat serve). This task accentuates the imbalance between a high intra-class variability and a low inter-class variability, bringing an additional challenge when compared to coarser action recognition.

Some research has been done in the area of tennis action recognition. In [153; 154] the authors present a video descriptor based on optical flow and classify actions into 'left-swing' and 'right-swing' with a support vector machine. In [155] tennis actions are classified into 'non-hit', 'hit' and 'serve'. Unfortunately, the videos used in these experiments are not publicly available, ([155] uses the ACASVA Actions Dataset [156] which provides features and labels, but not the RGB videos). For our work, we wanted to evaluate our methods using a publicly available dataset so that future research can be compared to our methods.

Amongst the many existing datasets for action recognition, we found one that conveniently suited our objective of fine-grained action recognition in tennis: THETIS [10]. Presented in 2013, THETIS is a complete dataset of fine-grained tennis actions comprising footage from 55 different subjects performing 12 distinct tennis shots (listed in Section 6.5.1) multiple times.

The videos are RGB, low-definition, monocular and shot in-the-wild, with dynamic backgrounds and occlusions. Our objective is to build a model able to classify actions in the videos into the 12 fine-grained action classes from raw footage, without the need for pre-processing (e.g. silhouette detection) and with the ability to generalise to other tasks. For this reason, we are interested in exploring deep learning techniques instead of more traditional approaches based on hand-crafted features.

6.3.2 Action Recognition in General

Given the limited work in the area of fine-grained action recognition for sports, we extended our review of the literature into existing approaches for action recognition in general with the expectation that the ideas from these could also be relevant to our problem domain. This section will present the most relevant state-of-the-art approaches to video-based action recognition. Action recognition is a very large area of research, and it encompasses problems from a broad range of scenarios. Their characteristics affect dramatically the choice of technique that is best suited to solve the problem. These are some of the variations that may occur:

- Action type: coarse or fine-grained (e.g. ‘person playing tennis’ vs ‘person doing flat service in tennis’).
- Scene setting: actions recorded in an experimental setting or in-the-wild. The latter may contain changes in illumination, occlusions or a moving background.
- Video properties: monocular or multi-view, static or moving camera, image resolution.

Amongst the approaches to video-based action recognition, two main categories can be drawn: classifiers based on hand-crafted features and deep neural networks.

Approaches Based on Hand-Crafted Features

Classifiers based on hand-crafted features are the most well-established approach. They generally involve two main steps: feature extraction and classification. Extraction of hand-crafted features is based on domain-knowledge and some of the most popular techniques include Histogram of Oriented Gradients (HOG) [38], Harris detector [146], Motion Boundary Histograms (MBH) [157] or Cuboid detector [158]. There are a number of approaches that use these hand-crafted features to build feature vectors that are suitable for action recognition [159–162]. We have investigated two of them in more in depth.

First, in [159], mid-level motion features are obtained as follows:

- Calculate the optical flow, shown in Figure 6.1c and explained in Section 2.5.1.
- Split the optical flow into 4 components: vertical and horizontal components and positive and negative values.
- Apply a Gaussian blur (low-pass filter) and normalize the result for each optical flow component previously split (Figures 6.1d to 6.1g). This last step is performed to reduce noise and position differences between two different clips of players performing the same action.

Actions are compared via a frame-by-frame cross-correlation technique resulting in a correlation matrix between actions. Action recognition is obtained via a nearest neighbour approach. We preformed a test of this technique by comparing the actions shown in Figures 6.2 and 6.3. The result of the comparison is shown in Figure 6.4b (Figure 6.4a shown as a baseline) and the algorithm is finding it difficult to generalize, i.e. Figure 6.4b compares two backhand actions, but the correlation matrix shown no similarity except for the first three frames.

Second, we also looked at kinematic features as in [160] which are also based on optical flow and include:

- Divergence: a local measure of the optical flow direction reflecting whether it is more directed outwards or inwards. This corresponds to whether the arrows in Figure 6.1c are predominantly pointing towards the center of the image or towards its boundaries. Using the formulas from [160], the divergence $f^1(\mathbf{x}, t_i)$ of the pixel at location $\mathbf{x}$ at frame t_i can be formalised as:

$$f^1(\mathbf{x}, t_i) = \frac{\delta u(\mathbf{x}, t_i)}{\delta x} + \frac{\delta v(\mathbf{x}, t_i)}{\delta y} \tag{6.1}$$

where $\frac{\delta u(\mathbf{x},t_i)}{\delta x}$ and $\frac{\delta v(\mathbf{x},t_i)}{\delta y}$ are the partial derivatives of the u and v components of the optical flow in the x and y directions at time t_i.

- Vorticity: describes the local "spinning" motion near some point. Using the notation from above, the vorticity $f^2(\mathbf{x}, t_i)$ is calculated as:

$$f^2(\mathbf{x}, t_i) = \frac{\delta v(\mathbf{x}, t_i)}{\delta x} - \frac{\delta u(\mathbf{x}, t_i)}{\delta y} \tag{6.2}$$

- The symmetry ($f^3(t_i)$ and $f^4(t_i)$) and asymmetry ($f^5(t_i)$ and $f^6(t_i)$) in the image, which are calculated as follows.

$$f^3(t_i) = u(t_i) + u(t_i)^T \tag{6.3}$$

$$f^4(t_i) = v(t_i) + v(t_i)^T \tag{6.4}$$

$$f^5(t_i) = u(t_i) - u(t_i)^T \tag{6.5}$$

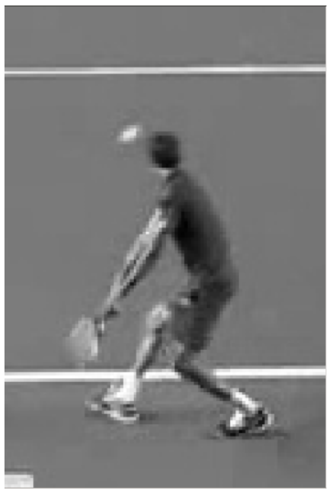
$$f^6(t_i) = v(t_i) - v(t_i)^T \tag{6.6}$$

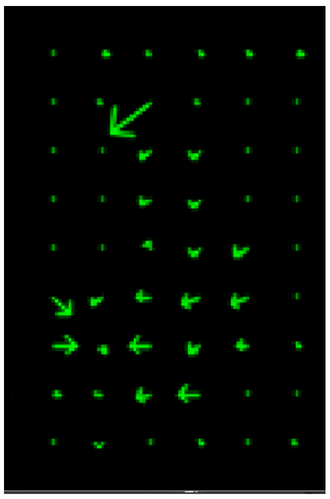
where $u(t_i)$ and $v(t_i)$ are the u and v components of the optical flow at time t_i.

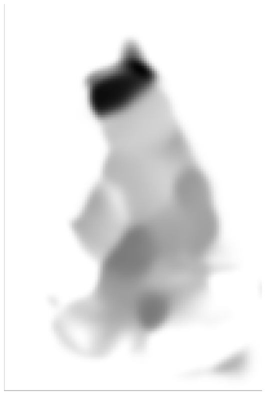
Similar to the previous technique, using kinematic features for fine-grained tennis action recognition proved ineffective.

Literature in the field shows that these classifiers built on top of hand-crafted features have achieved impressive results [161; 163–166], making their success undeniable. However the major drawback in using them is that their selection can be problem-dependent and difficult to generalize. Therefore, even if these techniques are very effective for some problems, their performance might decrease seriously for other problems, as we have just seen to be the case for fine-grained tennis actions.

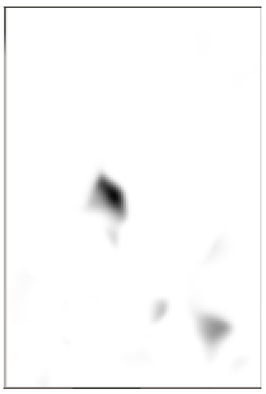
(a) Previous frame.


(b) Current frame.


(c) Optical flow.


(d) F_{x-}

(e) F_{x+}


(f) F_{y-}

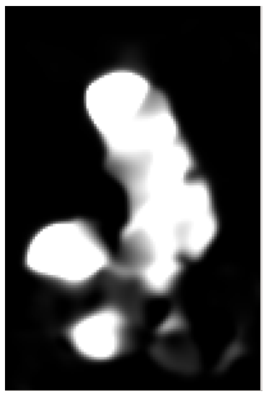

(g) F_{y+}

Figure 6.1: Construction of the low-level features.

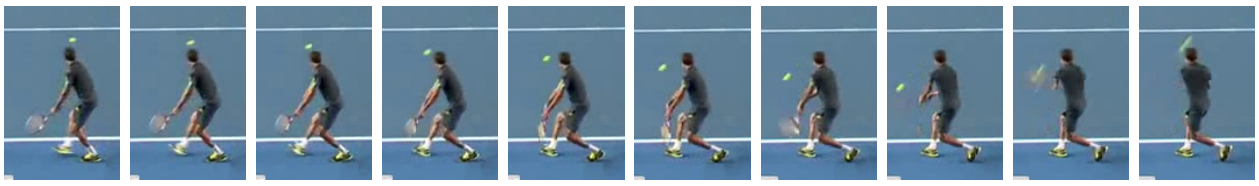

(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 6.2: Action 8.

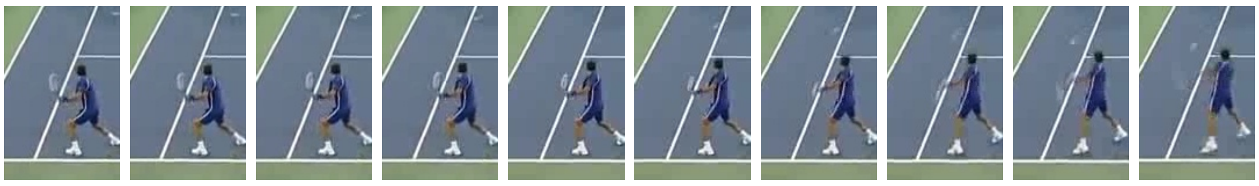
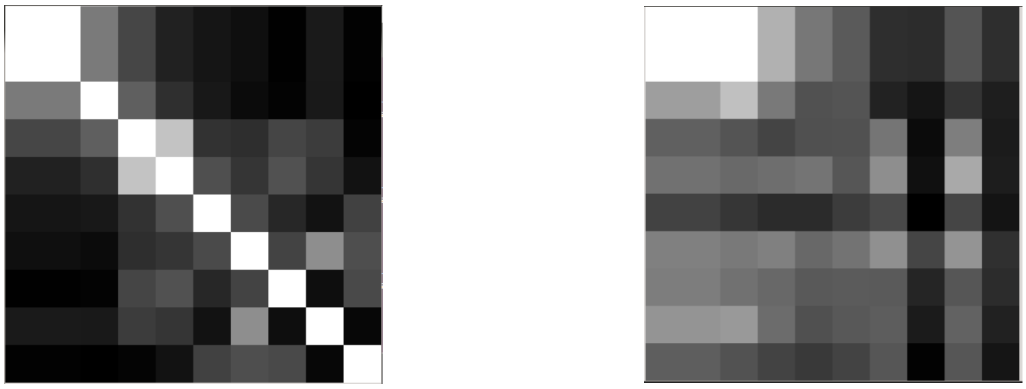

(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 6.3: Action 9.



(a) Action 8 vs itself (b) Action 8 vs 9

Figure 6.4: Frame-by-frame comparison of two actions using low level features from our own experiments.

Deep Architectures

Learned features are obtained through the performance of a machine learning task. For example, a neural network that learns to classify labeled images will contain in its hidden layers a representation of the input data that can be used as features to represent such data. These learned features have the potential of detecting structures that are more semantically meaningful and of being more generalizable than hand-crafted features [167]. In recent years, learned features have gained popularity and they have been shown to be extremely powerful in the field of still image understanding. In particular, Cpnvolutional Neural Networks (CNNs) have exceeded any other state-of-the-art method in the domain of image classification and have consistently won object recognition competitions since 2012 [64; 65; 168].

Driven by these achievements, attention has been brought to the application of deep neural networks to video classification. Unfortunately, their application to video processing has been proven to be more challenging and their success in this task cannot yet be compared to that for still images. This can be attributed to two main limitations. First, video data is more complex than still images because of the temporal dependencies, requiring models to learn more complex structures [148]. Second, the availability of substantial datasets is lower in comparison to those for still images. For instance, video classification benchmark datasets – such as KTH [149], Weizmann [169; 170], UCF Sports Action [151; 152], UCF-50 [171], HMDB-51 [172] – contain a smaller number of classes.

Some progress has been made in overcoming these issues and applying deep networks to action recognition. In 2011, one of the first succesful extensions of CNNs to video sequences was presented [173]. The authors extend a traditional 2D CNN to 3D, incorporating the time domain, to learn features, and then use long short-term memory units (LSTMs) for classification (cf. Section 6.4.2). Their results improve upon other deep learning approaches and their performance is competitive with hand-crafted based classifiers. Their experiments also show the benefits of using LSTMs in comparison to traditional Recurrent Neural Networks (RNNs, cf. Section 6.4.2). In [148], an end-to-end CNN video classifier is presented and evaluated in the Sports-1M dataset. They investigated different approaches to incorporating the time dimension, by fusing the information across the time domain earlier or later in the network. Interestingly, their best model performs similarly to their single-frame model, opening the question of whether these features are capturing any motion information at all for the classification task. In [174] a two-stream CNN is presented, with a spatial stream that works on single frames and a temporal stream that utilizes optical flow. Their results outperform these presented in [148] and are competitive with state-of-the-art hand-crafted models. All of these models bring insights to the application of deep learning to videos but also highlight the difficulty of transferring their potential from still images to video sequences.

6.4 Action Recognition in Tennis: Our Approach

Being aware of the challenges and advances in the field of action recogntion, this section presents our approach. It is a novel approach, and to our knowledge, it represents the first application of deep neural networks for the action recognition of fine grained sports actions (tennis in our case). The proposed algorithm extracts features from each frame individually by using the well-known convolutional neural network (CNN) named *Inception* [168; 175], pre-trained on an independent general image dataset and without fine-tuning. The resulting sequences of features are then fed to a deep neural network consisting of three stacked long short-term memory units (LSTMs) [69], a particular type of recurrent neural network (RNN). Our action classification algorithm is composed of two main steps: first, feature extraction using the *Inception* neural network (see Section 6.4.1) and second, classification through a deep LSTM network (see Section 6.4.2).

6.4.1 Feature Extraction

Inception is a well-known deep CNN architecture from [168; 175]. It was first introduced in 2015, obtaining the best results in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC '14) – an image classification challenge of 1 000 categories containing about 1.2 million images for training. *Inception* is a network 22 layers deep consisting of traditional convolutional layers stacked in the lower layers and 'Inception modules' stacked in higher layers. Every 'Inception module' concatenates the output of the following operations performed on its inputs (which are the results from the previous layers): 1×1 convolution, 1×1 convolution followed by 3×3 convolution, 1×1 convolution followed by 5×5 convolution and 3×3 max pooling followed by a 1×1 convolution.

Motivated by its performance in general image classification tasks, we chose *Inception* as our feature extraction algorithm. In ILSVRC '14, the network was able to attach coarse-grained labels to still images such as 'person playing tennis', but we wondered whether the features in the later layers also carried information about a person's posture or features discriminative for action recognition in videos. Interestingly, the *Inception* architecture was designed to optimize computational resources so that inference can be done in settings such as a mobile phone. We find this to be critically important for many action recognition applications.

In our work, *Inception* is used for feature extraction as shown in Figure 6.5. Each video clip, whose duration ranges from 90 to 150 frames, is cropped to the first 100 frames[4] due to memory constrains and to ensure all videos are of same length (alternate frames for HMDB[5], where videos are longer). At each frame, the previously mentioned network is applied to make predictions and the 2 048 features resulting from the previous-to-last layer are stacked into a $2\,048 \times 100$ representation of the video. For videos shorter than 100 frames, we employ zero-

[4]Which corresponds to about 2 seconds, a reasonable duration for a tennis stroke action.
[5]HMDB: a large human motion database, described in Section 6.7.1 [172].

padding (i.e. adding zeros in the final frames). These are then presented to our classification network to learn to label them with the appropriate tennis shot type.

Different to many applications in which the last layer is retrained for the specific problem, we decided not to retrain the network using THETIS. First, we wanted to see how applicable the features learned from ImageNet were in a different context. Second, given the small size of the THETIS dataset compared to the datasets normally used in deep architectures, we wanted as far as possible to avoid over-fitting by fine-tuning the network to this particular dataset.

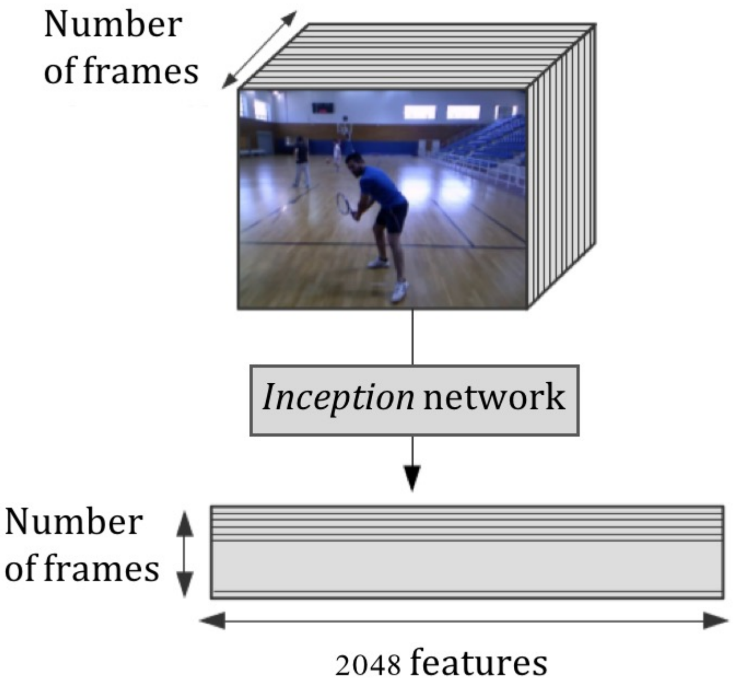


Figure 6.5: Feature extraction pipeline diagram.

6.4.2 Deep LSTM for Action Classification

Video-based action recognition requires the modelling of long-term time dependencies on highly complex data (images). We have seen that LSTMs are very suitable to learn these long-term dependencies (cf. Section 2.6.1). By stacking LSTMs on top of each other, it becomes possible to learn high-level structures in high-dimensional data such as images. Each layer uses as input the output of the previous layer, creating a hierarchical representation of the input data, where higher layers feature more abstract and complex representations of the data. In [69], the authors showed that deep LSTMs greatly improved performance in speech recognition compared to one-layer LSTMs. For these reasons we decided to use a deep LSTM network.

The detailed architecture of our network is shown in Figure 6.6. It has three stacked LSTM layers, empirically found to give best results, with the constrain of trying to minimize the number of layers to prevent overfitting. Each of these LSTMs layers has 90 hidden units and a softmax function is applied to the last layer to obtain the predicted classification output. In learning, our implementation has the following characteristics:

- Cost function: we employ the cross-entropy cost function $E(\mathbf{w})$ as a function of the weights $\mathbf{w}$ calculated as the average of all cross-entropies in the sample [41]:

$$E(\mathbf{w}) = -\frac{1}{N}\sum_{k=1}^{K}\sum_{m=1}^{M}\{j_{m,k}\log(y_{m,k})\} \tag{6.7}$$

  with K is the number of classes (k is the class), M_k number of examples for class k (M for all classes add up to N, the total number of samples), j is a binary indicator of whether the example was correctly classfied as k and y is the predicted probability of label k. This cost function is the one most commonly used in deep neural networks [47].

- Parameter regularization: L2 regularization is applied to reduce overfitting. It is the most commonly used [47] and corresponds to adding a regularizer term to the cross-entropy function resulting in:

$$E(\mathbf{w}) = -\frac{1}{N}\sum_{k=1}^{K}\sum_{m=1}^{M}\{j_{m,k}\log(y_{m,k})\} + \frac{\lambda}{2}\sum_{w}||w||^2 \tag{6.8}$$

  In our implementation, L2 is scaled by a λ value of 0.003, which was found empirically after trying a range of values between 0.001 to 0.005.

- Optimization algorithm: the Adam gradient-based optimization algorithm is used to perform gradient descent [176] and optimize the network. This algorithm is efficient([176; 177]) and robust to the value of hyperparameters (e.g. λ in Equation (6.8)).

We employ exponential decay of the learning rate, with a starting learning rate of 0.001 (0.005 for HMDB[6]) and decaying with a base of 0.96 every 100 000 steps. During training, the accuracy of prediction for the validation set is calculated every 10 steps to select the best model and the parameters for the best results stored. We manually tuned the hyperparameters of our network, which is common practice in deep learning [47] and we used *Tensorflow* for our implementation [178]. The code is available on: `https://github.com/silviav12/LSTM_action_recognition`.

6.5 Experimental Setup

6.5.1 Experimental Dataset: THETIS

Our experiments were conducted on the THETIS dataset [10]. It contains 1 980 monocular RGB videos of 12 tennis actions performed three times by 55 different players (31 amateurs and 24 experienced). Actions are performed using a tennis racket but there is no tennis ball in the videos. The 12 actions are:

- backhand (two hands)
- backhand
- backhand (slice)
- backhand (volley)
- forehand (flat)
- forehand (open stance)
- forehand (slice)
- forehand (volley)
- service (flat)
- service (kick)
- service (slice)
- smash

We only used the RGB videos from the dataset[7]. Videos of this dataset contain moving background and the video sequences vary in length. Figure 6.7 shows a sample of frames from the dataset. To our knowledge, only two publications make use of THETIS in action recognition experiments and there are no published results on the RGB videos alone. [10] presents the dataset and experiments accompanying it. They perform action recognition using state-of-the-art algorithms applied to 2D and 3D skeleton data. They achieve a one-vs-all average classification

[6]HMDB: a large human motion database, described in Section 6.7.1 [172].

[7]That is, not including data such as depth, skeleton 2D and 3D and silhouettes, which were are also provided.

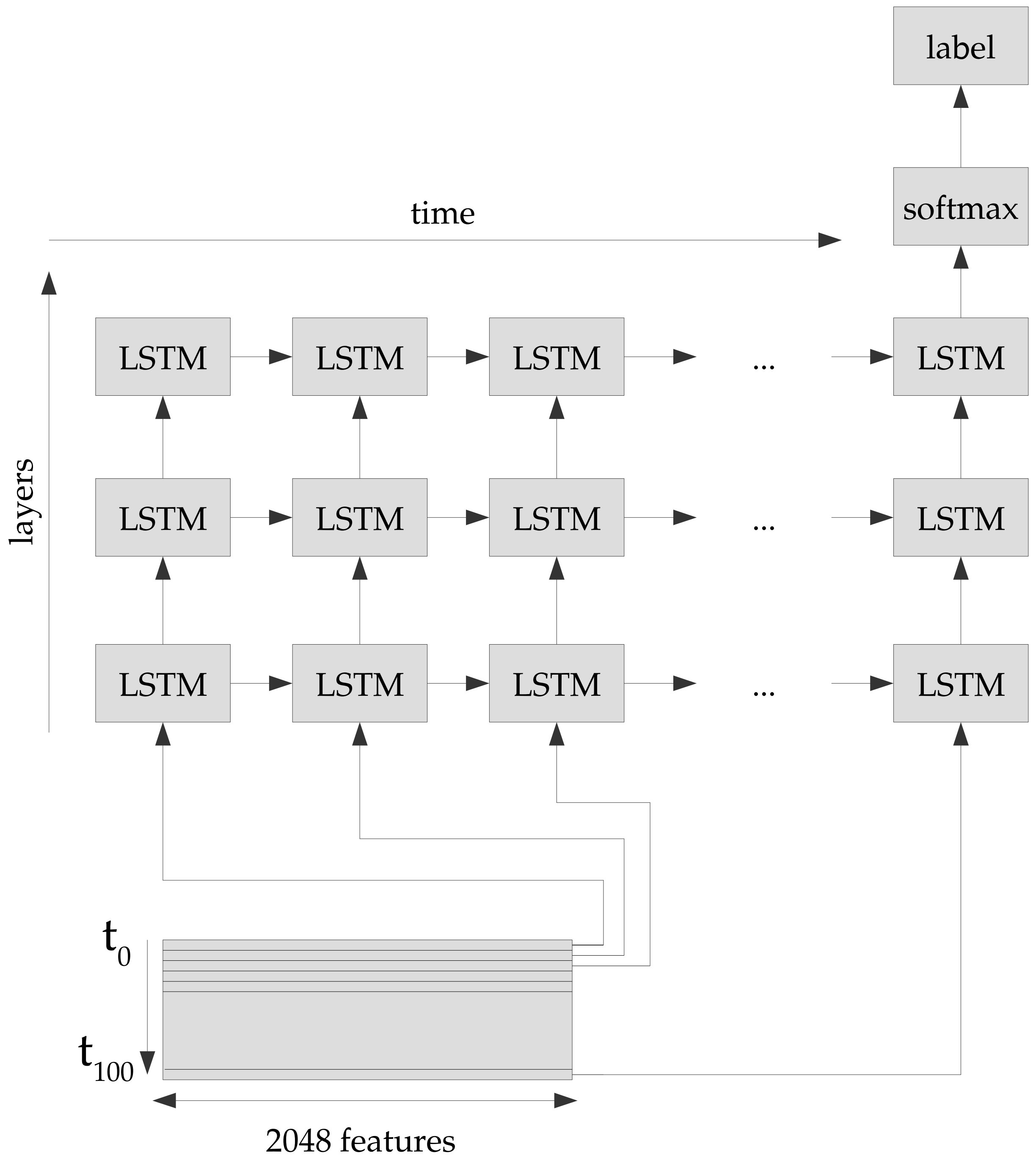


Figure 6.6: Architecture of our classification neural network. At the bottom is the input to the network, a sequence of 2 048 features per frame for 100 frames. This is the unrolled version of the recurrent network with time going from left to right, and input is processed in this direction. The input is also processed through 3 LSTMs layers, upwards, with 90 hidden units at each layer. At the end, input is processed through a softmax layer to obtain the predicted label.

accuracy of 60.23% and 54.40% respectively, compared to a 92.99% accuracy when applied to the well-known KTH dataset [149], showing how challenging the THETIS dataset is. In [179], experiments are performed using silhouette data achieving an accuracy of 86%.

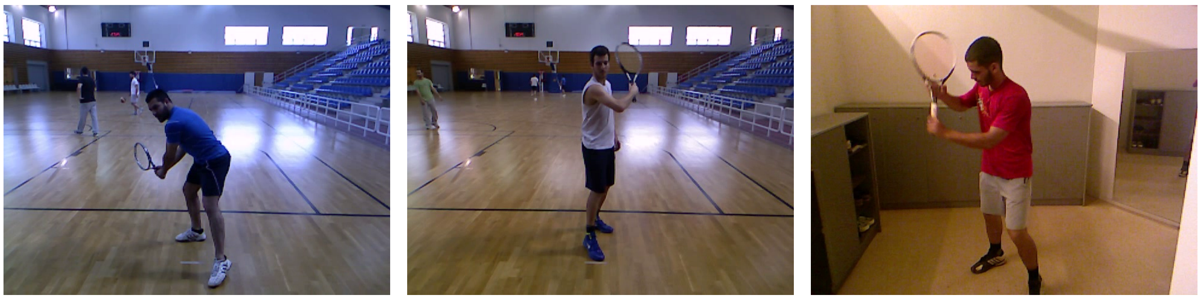


Figure 6.7: Samples from THETIS dataset.

6.5.2 Evaluation

As recommended by the authors of the dataset [10], we performed a leave-one-out cross-validation procedure. For each experiment, all the videos from a specific subject are selected as the test set, videos from five other subjects (randomly selected) are kept as validation set and the rest are used to train the network. This procedure is repeated three times for each experiment. For the evaluation, we show a normalized confusion matrix of the results averaged between all subjects and across the three repetitions of the experiment and provide the accuracy and F1 scores[8], to assess the precision and recall.

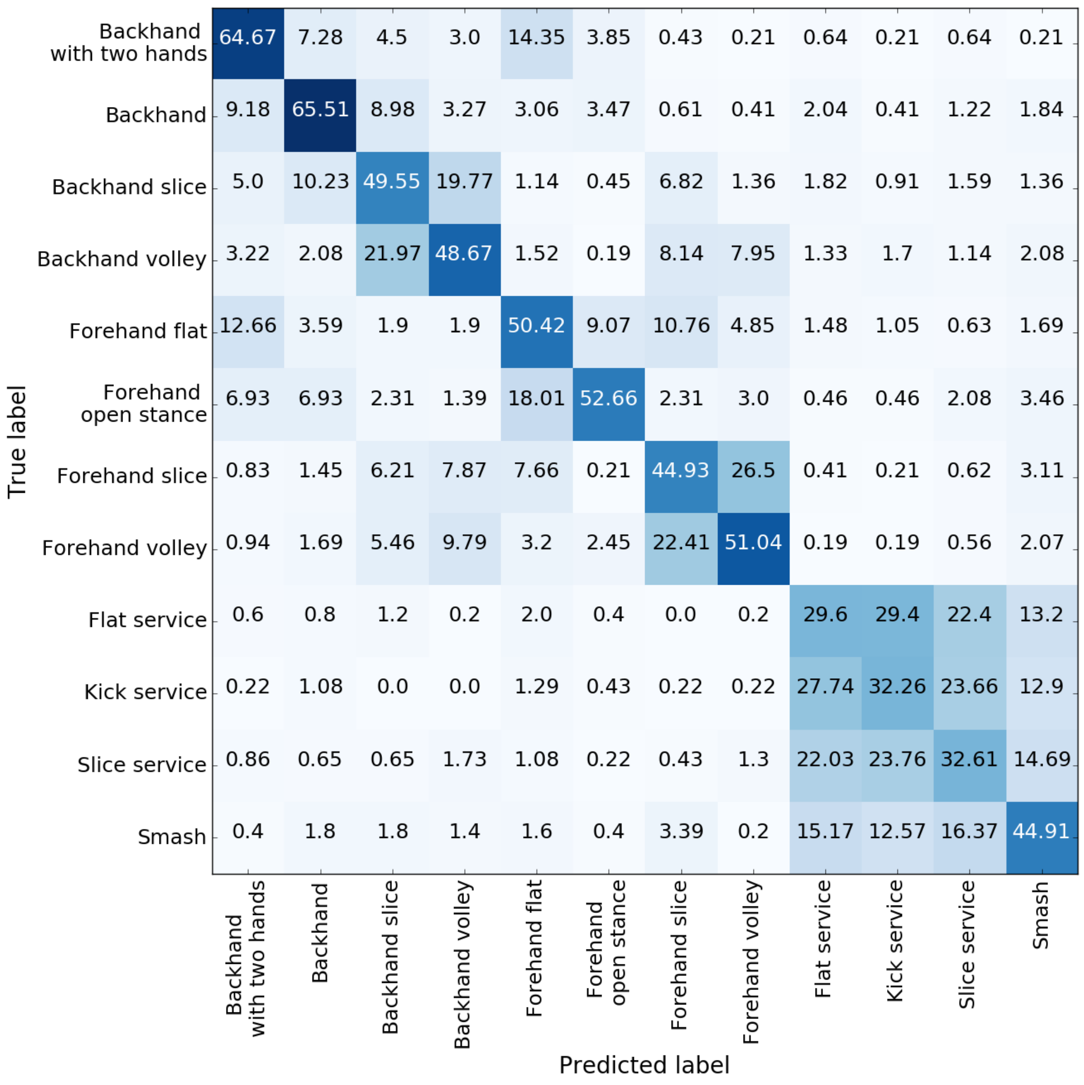
6.6 Results

6.6.1 Action Classification

The first experiment consists of classifying the videos into the correct class, amongst the 12 actions from the THETIS dataset. Figure 6.8 shows the confusion matrix of this experiment. The average prediction accuracy is 47.22%, with an F1 score of 47.05%. Figure 6.8 shows that

[8] $F1 = 2 \times (precision * recall)/(precision + recall)$, in our experiments, F1 score is represented as a percentage, and the highest possible score is 100.

for each of the 12 actions, most videos are labeled with the correct shot type and some actions, such as backhand with two hands and backhand have an accuracy over 60%. By looking at the results in more detail, one can realize that most errors are interpretable. For instance, the network makes errors in discriminating between the different types of serve or smash. Videos in the THETIS dataset do not contain the tennis ball, and this could explain why smash and serve are often confused. Another source of confusion are slices and volleys, both in backhand and forehand; these two actions are also similar for a human observer. Again, one main difference between volley and slice is that in the former the ball is hit before bouncing. In [10], the authors perform similar experiments but use depth videos and 3D skeletons rather than raw image and obtain 60% and 54.4% accuracy, respectively. It is reasonable to assume that classifying raw footage brings additional challenges and we consider that our results are satisfactory.

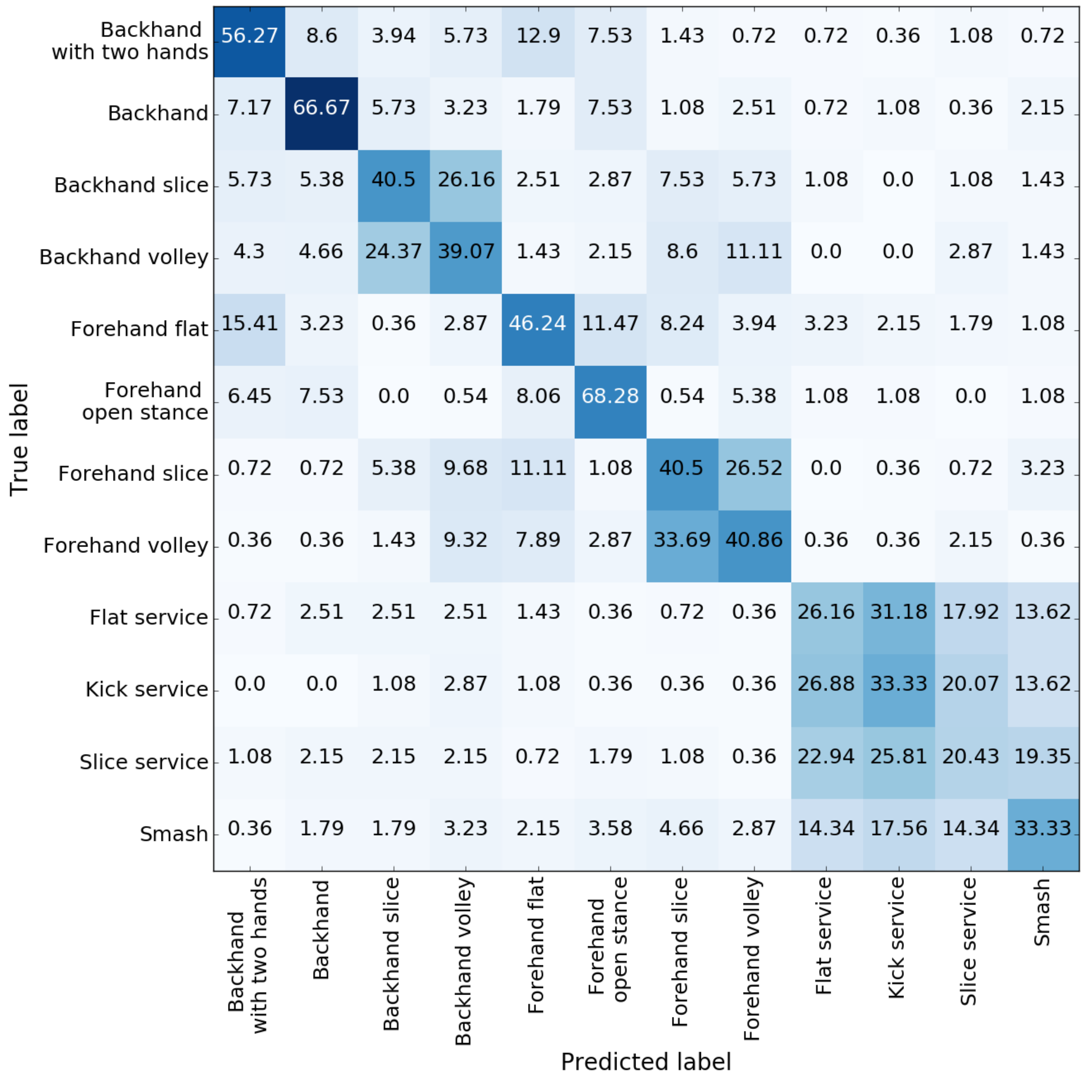


	Backhand with two hands	Backhand	Backhand slice	Backhand volley	Forehand flat	Forehand open stance	Forehand slice	Forehand volley	Flat service	Kick service	Slice service	Smash
Backhand with two hands	64.67	7.28	4.5	3.0	14.35	3.85	0.43	0.21	0.64	0.21	0.64	0.21
Backhand	9.18	65.51	8.98	3.27	3.06	3.47	0.61	0.41	2.04	0.41	1.22	1.84
Backhand slice	5.0	10.23	49.55	19.77	1.14	0.45	6.82	1.36	1.82	0.91	1.59	1.36
Backhand volley	3.22	2.08	21.97	48.67	1.52	0.19	8.14	7.95	1.33	1.7	1.14	2.08
Forehand flat	12.66	3.59	1.9	1.9	50.42	9.07	10.76	4.85	1.48	1.05	0.63	1.69
Forehand open stance	6.93	6.93	2.31	1.39	18.01	52.66	2.31	3.0	0.46	0.46	2.08	3.46
Forehand slice	0.83	1.45	6.21	7.87	7.66	0.21	44.93	26.5	0.41	0.21	0.62	3.11
Forehand volley	0.94	1.69	5.46	9.79	3.2	2.45	22.41	51.04	0.19	0.19	0.56	2.07
Flat service	0.6	0.8	1.2	0.2	2.0	0.4	0.0	0.2	29.6	29.4	22.4	13.2
Kick service	0.22	1.08	0.0	0.0	1.29	0.43	0.22	0.22	27.74	32.26	23.66	12.9
Slice service	0.86	0.65	0.65	1.73	1.08	0.22	0.43	1.3	22.03	23.76	32.61	14.69
Smash	0.4	1.8	1.8	1.4	1.6	0.4	3.39	0.2	15.17	12.57	16.37	44.91

Figure 6.8: Confusion matrix of our model applied to the THETIS dataset (all players).

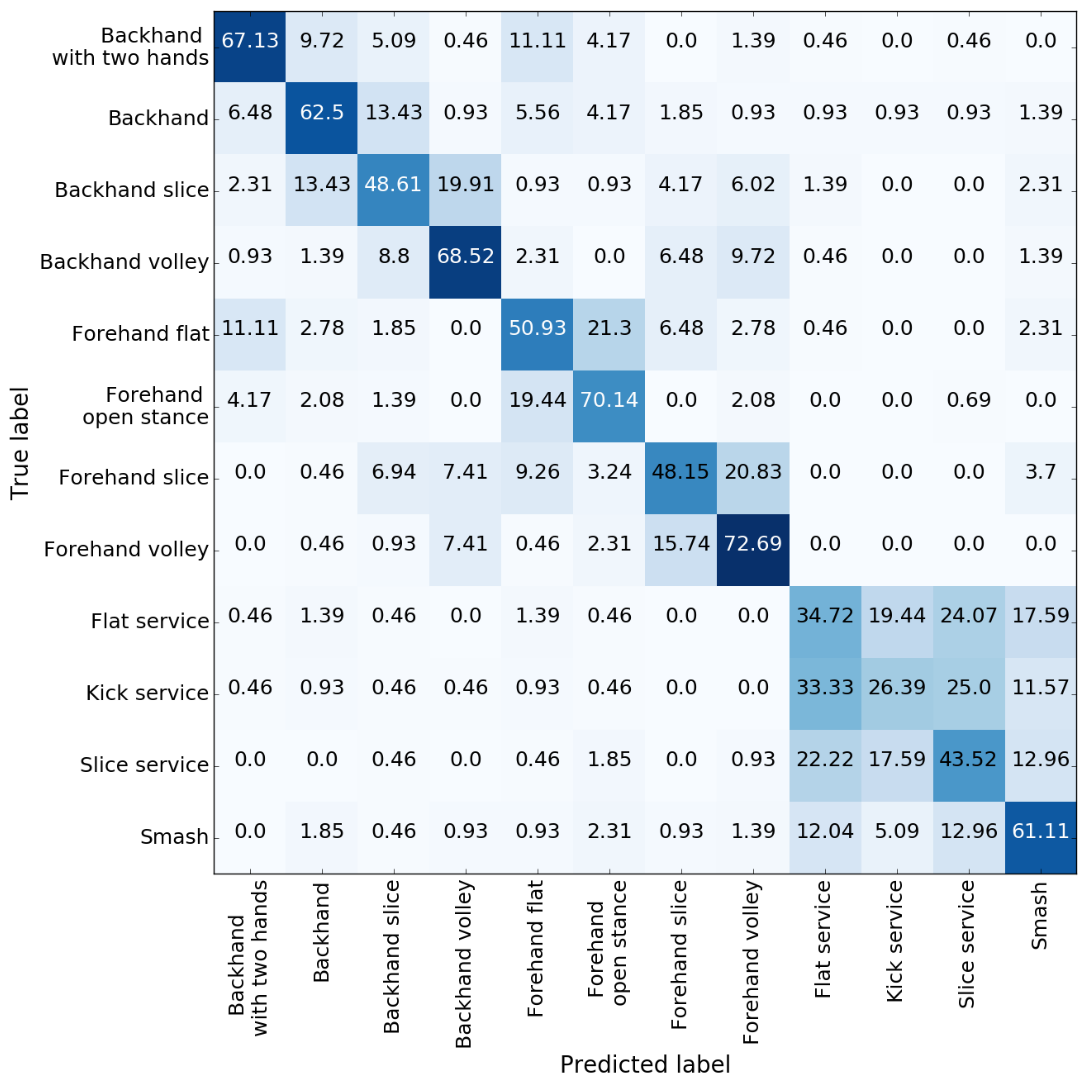
6.6.2 Expertise Detection

As described in Section 6.5.1, the THETIS dataset contains shots performed by 31 amateur and 24 experienced players. To assess how our network's classification accuracy was affected by the expertise of a player, we performed two different experiments. First, we calculated the prediction accuracy for each group of players separately, when trained on the entire dataset (with the same leave-one out cross validation procedure). Interestingly, the accuracy of our model is higher for professional players (54.09%) than amateurs (41.90%). Figures 6.9 and 6.10 show these results in detail for amateur and professional players respectively. One possible explanation is that professionals have a better technique making their shots more distinct and the biggest difference, as can be seen in the figures, is within the different types of serve.



	Backhand with two hands	Backhand	Backhand slice	Backhand volley	Forehand flat	Forehand open stance	Forehand slice	Forehand volley	Flat service	Kick service	Slice service	Smash
Backhand with two hands	56.27	8.6	3.94	5.73	12.9	7.53	1.43	0.72	0.72	0.36	1.08	0.72
Backhand	7.17	66.67	5.73	3.23	1.79	7.53	1.08	2.51	0.72	1.08	0.36	2.15
Backhand slice	5.73	5.38	40.5	26.16	2.51	2.87	7.53	5.73	1.08	0.0	1.08	1.43
Backhand volley	4.3	4.66	24.37	39.07	1.43	2.15	8.6	11.11	0.0	0.0	2.87	1.43
Forehand flat	15.41	3.23	0.36	2.87	46.24	11.47	8.24	3.94	3.23	2.15	1.79	1.08
Forehand open stance	6.45	7.53	0.0	0.54	8.06	68.28	0.54	5.38	1.08	1.08	0.0	1.08
Forehand slice	0.72	0.72	5.38	9.68	11.11	1.08	40.5	26.52	0.0	0.36	0.72	3.23
Forehand volley	0.36	0.36	1.43	9.32	7.89	2.87	33.69	40.86	0.36	0.36	2.15	0.36
Flat service	0.72	2.51	2.51	2.51	1.43	0.36	0.72	0.36	26.16	31.18	17.92	13.62
Kick service	0.0	0.0	1.08	2.87	1.08	0.36	0.36	0.36	26.88	33.33	20.07	13.62
Slice service	1.08	2.15	2.15	2.15	0.72	1.79	1.08	0.36	22.94	25.81	20.43	19.35
Smash	0.36	1.79	1.79	3.23	2.15	3.58	4.66	2.87	14.34	17.56	14.34	33.33

Figure 6.9: Confusion matrix of our model applied to the THETIS dataset (amateur players).



True label \ Predicted label	Backhand with two hands	Backhand	Backhand slice	Backhand volley	Forehand flat	Forehand open stance	Forehand slice	Forehand volley	Flat service	Kick service	Slice service	Smash
Backhand with two hands	67.13	9.72	5.09	0.46	11.11	4.17	0.0	1.39	0.46	0.0	0.46	0.0
Backhand	6.48	62.5	13.43	0.93	5.56	4.17	1.85	0.93	0.93	0.93	0.93	1.39
Backhand slice	2.31	13.43	48.61	19.91	0.93	0.93	4.17	6.02	1.39	0.0	0.0	2.31
Backhand volley	0.93	1.39	8.8	68.52	2.31	0.0	6.48	9.72	0.46	0.0	0.0	1.39
Forehand flat	11.11	2.78	1.85	0.0	50.93	21.3	6.48	2.78	0.46	0.0	0.0	2.31
Forehand open stance	4.17	2.08	1.39	0.0	19.44	70.14	0.0	2.08	0.0	0.0	0.69	0.0
Forehand slice	0.0	0.46	6.94	7.41	9.26	3.24	48.15	20.83	0.0	0.0	0.0	3.7
Forehand volley	0.0	0.46	0.93	7.41	0.46	2.31	15.74	72.69	0.0	0.0	0.0	0.0
Flat service	0.46	1.39	0.46	0.0	1.39	0.46	0.0	0.0	34.72	19.44	24.07	17.59
Kick service	0.46	0.93	0.46	0.46	0.93	0.46	0.0	0.0	33.33	26.39	25.0	11.57
Slice service	0.0	0.0	0.46	0.0	0.46	1.85	0.0	0.93	22.22	17.59	43.52	12.96
Smash	0.0	1.85	0.46	0.93	0.93	2.31	0.93	1.39	12.04	5.09	12.96	61.11

Figure 6.10: Confusion matrix of our model applied to the THETIS dataset (professionals).

To further investigate how the network was affected by the players' expertise, we compared the network's performance when trained using only amateur players, only professionals and on a mixed set of players. In order for the results to be comparable, the number of examples used for training should be similar. For this reason, we could not use previous results for the mixed set of players since they are calculated using twice the amount of data compared to the data that can be used when training on only amateurs or experts. To solve the issue, we re-ran the experiments selecting the test and validation sets as before but randomly selecting only 25 examples for training. Taking into account that, we used videos from 5 players for validation and 1 for testing in all experiments, the final training sets contain videos from 25 players for the amateurs and mixed groups and 18 for professionals.

Players in training set	Players in test set	Accuracy
mixed	mixed	39.65% (47.22%)
mixed	amateur	37.02% (41.90%)
mixed	professional	43.06% (54.09%)
amateur	amateur	37.70%
professional	professional	45.00%

Table 6.1: Accuracy of classification when training with amateurs, professionals or a mixed population, using a training set of 25 players for amateurs and mixed population and 18 players for professionals. This was done to balance the number of examples in training sets of all populations. In brackets, results of training with 49 players are shown.

Results are summarized in Table 6.1. As expected, the first observation is that when training with a mixed group of players but using only 25 players for training, the classification accuracy for all groups of players is lower than when using the entire dataset (training with 49 players). When training with a reduced number of examples the accuracy is 39.65%, 37.02% and 43.06% for mixed players, amateurs and professionals respectively versus 47.22% , 41.90% and 54.09%. These results are consistent with previous experiments since classification of videos of professional players achieves the highest accuracy. Interestingly, this is further increased when only professionals are used for training – accuracy increases from 43.06% to 45.00%. Classification of videos from amateur players also benefits marginally from training with only amateur players, increasing the classification accuracy from 37.02% to 37.70%. These results suggest that the network could be learning different features depending on the level of expertise of the players. It may be that features that help to best discriminate between different actions in amateur players are different to those helpful in identifying different shots played by professionals. Another interesting observation from these experiments is that the quality of the training data is important, but in our particular example, size of the training set is even more important as best results are achieved when using the entire dataset.

6.6.3 From Fine-Grained Actions to Stroke Types

Observing the results from Figure 6.8, we noticed that our learning algorithm was not only able to identify the 12 fine-grained actions from the THETIS dataset but, when making mistakes, these were generally between actions that can be grouped into the same type of tennis stroke. For instance, even though a slice service is confused with a flat or kick service, the network is still recognizing that it is a serve. For this reason, we looked at the accuracy of the prediction when grouping classes into more general stroke types, as shown in Table 6.2. We still consider these actions to be fine-grained as they are within the domain of tennis actions and fine-grained when compared to general action recognition (e.g. detecting which sport is being played).

For this, we used the results from the first experiments of action classification and grouped the labels into the 4 main action types in Table 6.2: backhand, forehand, service and smash. The results from this are shown in Figure 6.11. In this setting, the action classification accuracy reaches 76.92% with and F1 score of 76.90%. As can be seen from the Figure 6.8, the mean accuracy is brought down by the smash detection. In fact, most errors come from a confusion between smash and serve. These are quite similar in terms of body movement and the main differences are the state of the game (serve is played at the beginning of a point), player position in the court and ball trajectory before hitting the ball. THETIS videos do not contain the tennis ball, which could help in discriminating between the two actions. Also, in real-world applications we might expect to know the players position or state of the game, helping to further discriminate between the two actions.

Having obtained these results, we wondered how predictions would compare if we trained on the main stroke types directly. For this, we grouped smash and serves into the same category as they are very similar in terms of body movement and it helps balancing the classes. Table 6.3 shows the results by category, and we can see that training for the specific task, which is detecting one of the three main strokes in this case, produces better classification results than training for finer-grained actions and then regrouping the actions into their more general categories. The classification accuracy improves from 84.10% to 88.16% for players of mixed abilities, from

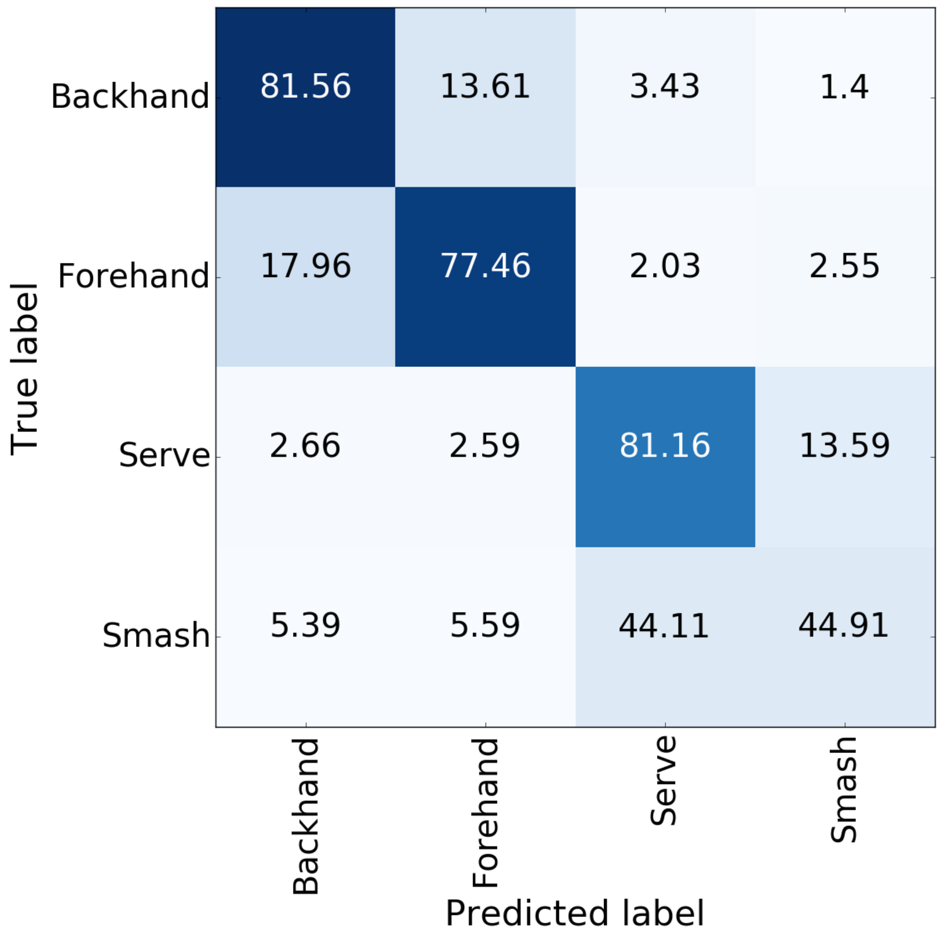


Figure 6.11: Confusion matrix of our model on the THETIS dataset, grouped classes.

81.23% to 84.33% for amateurs and from 87.82% to 89.42% for professionals, when trained using the entire dataset.

Two main observations can be derived from this. First, the network performs best when trained for the specific task on which it is evaluated and, second, the features relevant to discriminate between stroke type and between finer-grained actions might be different.

6.7 Applications Beyond Tennis

6.7.1 HMDB Dataset

To show the applicability of the implemented LSTM Neural Network (NN) to general action recognition tasks, we also show the results of experiments performed on the HMDB

Action group	Actions
Backhand	backhand (with two hands)
	backhand
	backhand (slice)
	backhand (volley)
Forehand	forehand (flat)
	forehand (open stance)
	forehand (slice)
	forehand (volley)
Service	service (flat)
	service (kick)
	service (slice)
Smash	smash

Table 6.2: Fine-grained actions grouped into stroke types.

Players tested	Trained actions	Accuracy
all	all	84.10%
all	3	88.16%
amateur	all	81.23%
amateur	3	84.33%
professional	all	87.82%
professional	3	89.42%

Table 6.3: Accuracy of classification when training with fine-grained actions and then regrouping vs training directly with the three main stroke classes.

dataset [172]. This contains 6 849 videos from 51 actions that range from facial actions such as smiling to body movements like climbing, horse riding or hand shaking. Videos are extracted mostly from movies, but also from other datasets, and they can be considered in-the-wild.

6.7.2 Experiments

In our experiments we use the three different splits of the data (into training and testing sets) as provided by the original authors [172]. For each split, the training set contain 3 570 examples, which we randomly divide into training and validation sets with 70% and 30% of the data respectively.

6.7.3 Results

To investigate the ability of generalization of our network, we evaluated it on the 51-calss HMDB dataset. We achieved an accuracy of 43.19% in terms of identifying the correct action class and an F1 score of 42.48%. In Table 6.4, our performance in HMDB is compared to existing models that, like ourselves, use RGB data exclusively. For instance, we do not include: the two-stream ConvNet of [174] (59.4%) which uses optical flow information, models in [180] that use Fisher Vectors (53.3%) and a combination of HOG, HOF and MBH (60.1%). The results presented here show that our network has the potential to be applied to other tasks, further supporting that it could be applicable to other sports.

Model	HMDB-51 accuracy
Spatial stream ConvNet [174]	40.5%
Soft attention model [181]	41.3%
Vinyes et al. [13]	43.2%
Composite LSTM [182]	44.1%

Table 6.4: HMDB-51 classification accuracy by state-of-the-art models from RGB data alone.

6.8 Conclusion

In this work we have presented a 3-layered LSTM network able to classify fine-grained tennis actions. It uncovered a number of interesting points. First, our network achieved good results by using features extracted through the application of the *Inception* neural network, trained on an independent dataset and without the need of fine-tuning. This suggests that the hidden layers of *Inception* are a robust data representation with the potential to be transferable to multiple tasks and domains. Second, the network's classification errors were interpretable, suggesting it was learning semantically meaningful information. Endorsing this idea, the network performed better when trained with only amateur or only professional players rather than a mixed population. It is possible that it learned different features when looking at amateurs and professional players and it would be interesting to investigate this further. Third, the network performed better for professional players than amateurs, when trained on a mixed population. A possible cause is that professionals have a better technique that makes their strokes more distinct. In the future, we would like to consider whether this can be exploited to assess a player's expertise. Fourth, the same network architecture was able to classify the three main stroke types with an 88.16% accuracy, and it performed better than when an indirect inference was made from finer-grained actions. This further supports the robustness and transferability of the *Inception*features. Finally, we also showed how the proposed approach can be applied to general action recognition tasks, by evaluating it on the HMDB dataset. With this work we also wish to motivate the exploration of applying deep neural networks in the sports domain and the use and production of benchmark datasets in sports action recognition.

Our main contribution is the presentation of the first deep Neural Network for fine-grained action recognition in tennis, and one of the first ones for fine-grained action recognition in general. Our approach achieves 88.16% accuracy for classifying backhands, forehands and serves and 47.22% accuracy in classifying 12 finer-grained actions. We also show that the network is learning semantically meaningful information as most errors are interpretable and its performance is different for expert and amateur players.

Chapter 7

Post-Match Conference: Conclusions and Future Work

7.1 Summary of Thesis Achievements

7.1.1 Framework

We have presented a three-layered framework for the spatio-temporal analysis of tennis (cf. Figure 1.11) consisting of a Vision, Classification and Modelling Layer with algorithmic innovation and implementation in the vision and classification layers. Within this context, we have shown that innovation in the areas Computer Vision and Machine Learning has enabled the development of an affordable, real-time and unobtrusive system for spatio-temporal data collection in tennis and its subsequent analysis.

A principal benefit of our framework is that even though our layers are interconnected, their implementations are independent of each other. As a consequence, each layer can be modified and enhanced without the need to adapt the other layers. For instance, the design of our framework takes into account the rapid development of technology, and as video capturing devices improve, accuracy within the Vision Layer will also improve, enhancing the performance of the framework as a whole. In this direction, we will start a project using a mobile phone in the Vision Layer instead of the technology based on four cameras described in this thesis.

7.1.2 Vision Layer

The development of techniques for spatio-temporal data collection in tennis from videos or film dates back more than 50 years. The work that we presented for the Vision Layer of our framework is a low-cost, portable and unobtrusive data collection system for tennis. We also provide qualitative and quantitative analysis of the results in terms of novel error metrics. We combined Computer Vision and Machine Learning techniques for the player and ball detection by the generation of an adaptive background image and selection of ball candidates based on shape and trajectory using random forest classifiers.

The proposed spatio-temporal data collection system uses four cameras and commodity hardware to detect the court lines, track the players and tennis ball on the frame and obtain their 3D position in the real world. The ball and player inferred 3D locations had an error lower than 10 cm and 6 cm respectively for 80% of the frames. We consider these results to be of good quality, especially considering the size of the player and ball, 50 cm $\times$170 cm $\times$20 cm and 7 cm respectively, and the human error rate for estimating the ball position, which goes up to 10 cm even for experts at major tournaments (approximately the tennis ball diameter).

The work on the Vision Layer included graphs showing the detection error at the pixel level for 2D data and at the cm level for 3D. We also presented results for the player as the percentage of his/her size to normalise the error and reflect its significance with respect to the players' dimensions in the frame. This accuracy presentation is novel in the sports domain, and we believe that encouraging other researchers to present results in a similar manner would enhance the ability to compare different techniques.

The lack of a unified evaluation system complicates the comparison of different approaches. We approached this challenge by presenting tables comparing different techniques for ball and player detection and show what, to our knowledge, is the first attempt to compare different Computer Vision enabled systems for tennis spatio-temporal data extraction.

7.1.3 Classification Layer

Data Analysis

We presented visualisations of the data obtained from the Vision Layer as both a Cartesian and a zoned heatmap, accompanied by an analysis of the visualised data. We also automatically extracted high-level information such as stroke timing and area covered by each player. Accomplishing a full pipeline from data collection to visualisation showed the level of analysis that can be obtained from the system introduced in the Vision Layer. We also presented a method for pre-processing tennis ball trajectory data using quadratic fitting and value interpolation.

Action Recognition

We developed and trained a novel deep neural network for domain-specific action recognition, specifically in tennis. We proposed a 3-layered LSTM network trained on features extracted from videos exploiting well-known *Inception* neural network trained on an independent dataset and without fine-tuning. To our knowledge, this is the first application of neural networks to the problem of fine-grained tennis action recognition, and one of the first applications to fine-grained action recognition in general. We evaluated our approach on the challenging THETIS dataset, which contains low-definition RGB videos of fine-grained tennis actions, and obtained promising results of 47.22% accuracy in one against all classification of 12 fine-grained actions and 88.16% accuracy for classifying backhands, forehands and serves.

With our work, which is applied to a publicly available dataset, we aim at encouraging other researchers to use publicly available datasets, facilitating the comparison of different approaches. We believe that this would represent a step forward from current work in sports fine-grained action recognition, which is performed on datasets that are not normally accessible to the public.

In addition to this, we showed that the neural network can be fine-tuned to professional or amateur players, reflecting that it is learning features differentiating how the two different groups play. Finally, we also showed that our neural network can be used in applications beyond tennis.

7.1.4 Modelling Layer

In the Modelling Layer, we surveyed different methods that incorporate visual information for the analysis and prediction of tennis matches. We did not find published records of comprehensive reviews encompassing the different approaches that use spatio-temporal data for the analysis of tennis. Considering that such a piece of work is essential to take the field forward, we critically reviewed current research and provided a sense of the evolution in the field. Following this, we also evaluated the limitations of state-of-the-art approaches in spatio-temporal tennis analysis, pointed out the main causes for these limitations and motivated how the work of this thesis addresses them.

7.2 Applications

Applications in Tennis

The work presented in this thesis can be applied to tennis in multiple directions:

- Enhancing Training: Understanding the characteristics of shots, movement patterns of players and the type of action being played enables an in-depth personalised analysis of player performance and tactics. Access to personalised feedback can help in designing training sessions and tracking the progress of players. For example, it is possible to assess which type of shot a player must invest more training effort into. The support in the form of data visualisations reinforces the validity and trust of a player in this advice. The feedback can be in real-time, having an immediate impact on the learning process of the player.

- In-Play Analysis and Prediction: In-Play analysis (compared to aggregated analysis) enables players and coaches to better assess the performance of tennis players and their opponents on the day and assess which tactics are working best. Prediction that incorporates spatio-temporal data can evolve as the match progresses and help players in adjusting their tactics during a match.

- Line Call System: The framework proposed can be used as an automated referee for line calls.

- Access to a Professional Experience: Combining the applications mentioned above in an integrated training, in-play analysis and line call system opens up the possibility for amateur players to have an experience similar to that of professional players in major tournaments, where Hawk-Eye is available, but with additional feedback, for example, by taking into account the type of shot being played.

- Augment Spectators' Experience: The automatic extraction of high-level information such as the area covered by a player or type of shot and its visualisation can significantly benefit broadcast by providing additional information to the spectators. Augmenting the spectator's experience is crucial to keep fans engaged and to attract a new generation of fans to the sport.

Appplications to Other Sports

This framework can also be applied to other sports. The application to other racket sports such as badminton or table tennis requires modifications to the Vision Layer and adjustments in the Classification Layer, for example, to define the types of shot for these sports. The application of our framework can also be extended to team sports by incorporating multi-player tracking in the Vision Layer and including a module to analyse interactions between players.

7.3 Current Limitations

7.3.1 Benchmark Datasets

A major challenge that we encountered during our research was the lack of benchmark datasets. Other domains such as face recognition, image captioning or speech understanding have benchmark datasets that allow researchers to compare their work, find out which techniques work best and, in this way, advance the field. The reader might have noticed – especially in Chapters 4 and 6 – how difficult it is to compare existing work in problems such as tennis ball and player detection or fine-grained action recognition. With our work, we wanted to point out this issue and encourage the development and use of benchmark datasets in the field of sports.

7.3.2 Evaluation Protocols

The difficulty in comparing existing work in object tracking and action recognition in tennis has been highlighted above. Besides the lack of benchmark datasets, another major contributor to this challenge is the absence of unified evaluation protocols. As we have seen in Chapters 4 and 6, many researchers present only a qualitative evaluation and, when quantitative results are presented, they describe the percentage of frames in which the detection was 'correct'. However, the exact definition of 'correct' is rarely provided. We believe in the need for developing objective quantitative evaluations of algorithms for visual data extraction in sports.

7.3.3 The Gap Between Research and Application

Progress is being made in the application of technology to tennis, becoming an integral part of major tournaments with the Hawk-Eye line call system. However, the integration of technology in training and amateur tournaments is still lacking. From our experience, there is a need for enhancing communication between professionals in sports and Computer Science researchers. Tennis coaches have the domain knowledge necessary to develop training programs from our data analysis. During our project we have had discussions with a number of professional tennis coaches, which has greatly helped in refining our approach to data analysis and visualisation.

7.4 Future Work

Player detection

It would be extremely beneficial to detect specific landmarks in players' bodies such as their feet from visual data to have a more accurate representation of their position, trajectory and allow the possibility of counting the number of steps. Recognizing different parts of the human body in the player from visual data, as detecting the joints could lead to a biomechanical analysis and provide feedback about their technique.

Analysis of Results

We would also like to further improve the quantitative analysis of our results by measuring the real-world 3D position of the ball and players. This could be done with additional equipment such as sensors. In the same direction, research in the field would significantly improve if a benchmark dataset with ground truth was published.

Data Analysis

The data analysis presented by our Classification Layer could be improved by including more data from a higher number of different players and full tennis matches, rather than training sessions. By including more players, we would be able to perform the same analysis in a player-specific manner and be able to compare different players' patterns to gain insights into their strengths and weaknesses. By analysing data from a match, it would be possible to investigate tactics, and find patterns from an event-specific perspective (e.g. patterns in the first or second serve) and analyse their effects on the likelihood of winning a point.

General Framework

Future work in extending our framework includes:

- The application to other sports, which would require additional modules for interaction between players in team sports.
- The incorporation of additional types of multimedia data, such as audio signals, which have been shown to be very powerful in detecting different types of stroke in tennis [23; 183].
- The extension of data visualisations by applying approaches used in other domains, as shown in [141; 184] and making the system interactive (e.g. allowing the user to select different views or levels of granularity).

Appendix A

ATP 58 Tennis Tactics Patterns

In 1996, the United States Tennis Association published 58 tennis tactics patterns [97] (AC = advantage court, DC = deuce court, CC = crosscourt, T = zone of the court formed by the intersection of the service line and centre service line):

Net play (8 patterns):

- volley to opponent's weakness (if ball is above the net)
- volley deep down the line (if ball is below the net)
- volley deep CC (after your approach, if ball is above the net)
- serve wide and volley to the open court
- serve to the "T" and volley behind opponent or to a weakness
- angle your overhead away (if lob is short)
- aim your overhead CC (if lob is deep)
- let ball bounce and aim CC (high lobs)

Groundstroke (8 patterns):

- attack a short ball down the line (rally CC)
- attach a short ball CC (rally CC)
- hit a severe angle (rally CC to get short, wide ball)
- drive inside-out through the court (from a ball down the middle)
- drive inside-out off the court (from a ball in left half of the court)
- hit looping drives to opponent's backhand (when driven deep)
- attach a short ball (exchange sliced back-

hands)

- high and deep down the middle (against deep shots in the middle)

Serve & return (18 patterns):

- serve wide to open the court (DC)
- serve wide to open the court (AC)
- serve to the "T" to reduce the angles (DC)
- serve to the "T" to reduce the angles (AC)
- serve at the body to jam the receiver
- return deep CC (DC against a wide serve)
- return deep CC (AC against a wide serve)
- return deep down the line (DC against an extreme wide serve)
- return deep down the line (AC against an extreme wide serve)
- return deep down the middle (DC against a serve to the "T")
- return deep down the middle (AC against a serve to the "T")
- return deep down the line (DC against a serve to the "T")
- return deep down the line (AC against a serve to the "T")
- hit a forcing shot down the line (against a short, weak serve)
- hit a forcing shot CC (against a short, weak serve)
- chip or drive down the line and come to the net (against a short, weak serve)
- return low at the servers feet against a serve-and-volleyer
- return down the line (against a serve-and-volleyer)

Midcourt (8 patterns):

- drive hard and flat down the line (if ball is above the net)
- drive cross court for a winner (if ball is above the net)
- slice down the line (if ball is below the net)
- drop shot down the line (if ball is below the net)
- approach down the middle (from a ball in the middle)
- use an inside-out forehand approach (from a ball in the middle)

- use a looping topspin approach (from a deep, high bouncing shot)
- move in and hit an approach volley (from a looped shot)

Defensive play (16 patterns):

- Pass down the line (against a deep CC approach)
- Pass CC (against a deep CC approach)
- pass CC (against a moderate down-the-line approach)
- use a two-shot pass (against a moderate down the line approach)
- use a two-shot pass (against an approach up the middle)
- use a two-shot pass inside out (against an approach up the middle)
- overpower your opponent (against a weak approach up the middle)
- use two shots to pass, first CC, then down the line (against a deep sliced approach to backhand)
- use two shots to pass, first down the line, then down the opposite line (against a deep sliced approach to backhand)
- use two shots to pass (against a deep sliced approach to forehand)
- use two shots to pass, first down the line, then down the opposite line (against a deep sliced approach to forehand)
- use a two-shot pass (against a short, sliced approach)
- drive at opponents right hip (against a short, weak volley)
- drive and then lob (against an approach to your backhand)
- drive and then lob CC (against an approach to your backhand)
- hit a high defensive lob (against a deep approach shot)

Appendix B

Classification of Lines for Court Detection

As described in Section 4.3.2, the lines detected in the image are classified into horizontal and vertical lines on the left and right-hand-side of the image separately to perform court detection. The algorithm used to detect lines defines them in terms of the coordinates of two points on the line. A line is considered horizontal if the absolute difference d_y in the y axis coordinates of the two points that define it is lower than their absolute difference d_x in the x axis, as shown in Figure B.1. The rest of lines are considered to be vertical.

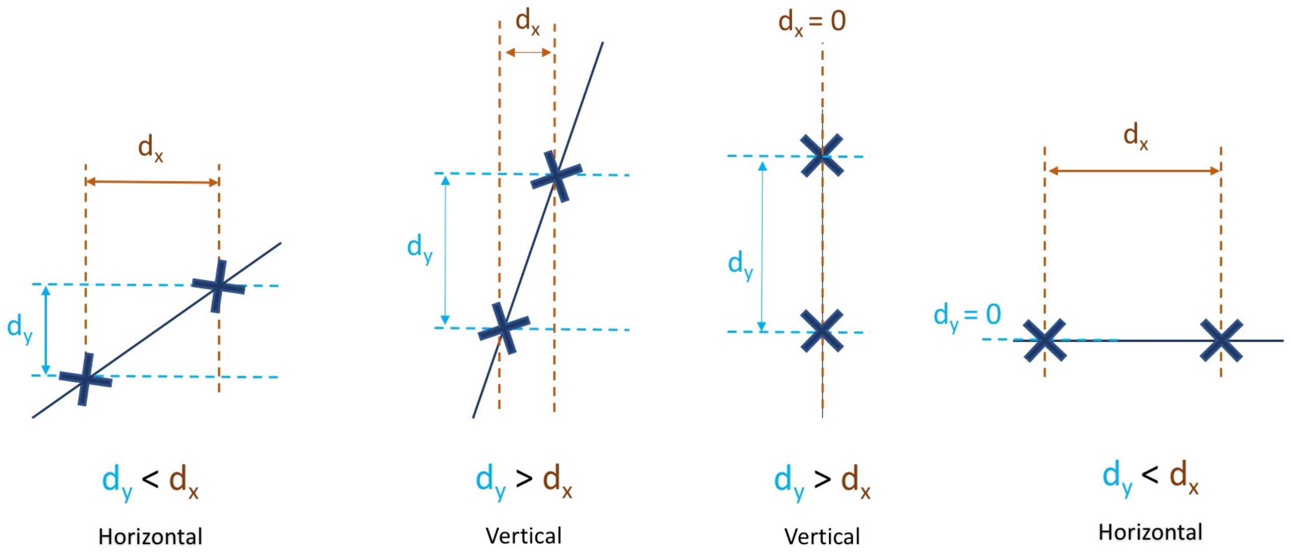


Figure B.1: Classification into vertical and horizontal lines.

Vertical lines are further classified into left-vertical and right-vertical lines with the aim to separate the lines that could be the right and left lines of the court. To achieve this, a margin of 1/3rd of the frame width is selected relative to its centre, as shown in green in Figure B.2, and lines within this margin are ignored. If the court occupies less than a 3rd of the frame, it is safe to assume that action is not taking place, because broadcasts provide closer views of the court when in-play. If only a partial view of the court is available, all lines will be classified into the same group, as the partial view implies that only vertical lines on either side (left or right) are visible, with the exception of the centre service line that will be classified as being on the other side. Figure B.2 illustrates an example of line classification with horizontal, left-vertical and right-vertical shown in magenta, blue and yellow respectively.

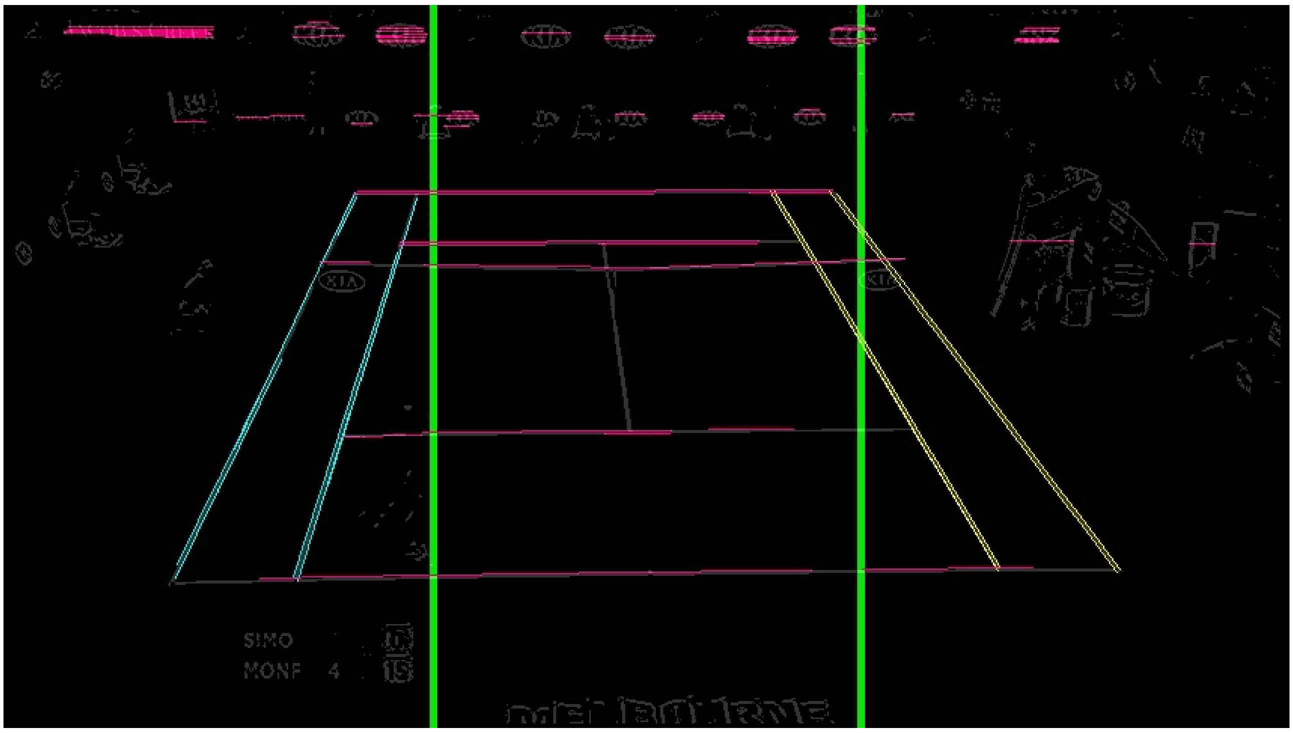


Figure B.2: Image resulting from lines extraction and classification into horizontal (magenta), left-vertical (blue) and right-vertical (yellow) lines.

Appendix C

Merging Lines for Court Detection

As described in Section 4.3.2, lines that are considered to be duplicated within each group of lines (classified into horizontal and vertical lines as explained in Appendix B) are merged. The merging step iterates through the sorted lines and, if appropriate, merges them resulting in a vector of merged lines.

As shown in Figure C.1, for a pair of horizontal lines L and R, we first check which of the two is the leftmost (L in this example). Then we calculate the difference d in the y-coordinates of the rightmost point of L (L_r) and the leftmost point of R (R_l). If d is below the threshold of 10 pixels (corresponding to twice the court line width), the lines are merged. The new line M is defined by the leftmost point of L (L_l) and the rightmost point of R (R_r).

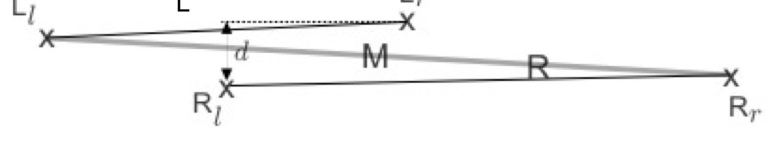


Figure C.1: Schematic for the merging of horizontal lines

As shown in Figure C.2, for a pair of vertical lines T and B, their slope and the distance d between the x-coordinates at their intersection with a fixed horizontal line y is compared. As above, if d is below the threshold of 10 pixels, lines are merged. The new line M is defined by points with highest and lowest y-coordinate of both lines (T_t and B_b respectively).

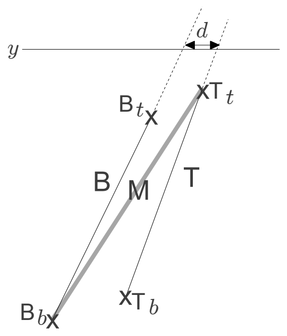


Figure C.2: Schematic for the merging of vertical lines.

The merging of horizontal lines has an additional step. Based on our empirical observations, the vertical lines are very well detected, and it is not common to have outliers. However, many horizontal lines are detected in addition to the court lines, but most of them are located outside of the court. Thus, horizontal lines are considered for merging if they fall within the highest and lowest y-coordinates reached by any of the vertical lines (we call the distance between these two y coordinates h), with a certain margin of error, shown by the green lines in Figure C.3. The error margin in the area below the court is set to 1/15th of h and to 2/15th of h in the area above it, because detection in the top part of the frame is more challenging as objects appear smaller and court lines thinner. After selection and merging of the lines found (Figure B.2), the result is Figure C.3. Note that even if it is not visible in the figure, many lines are duplicated in Figure B.2.

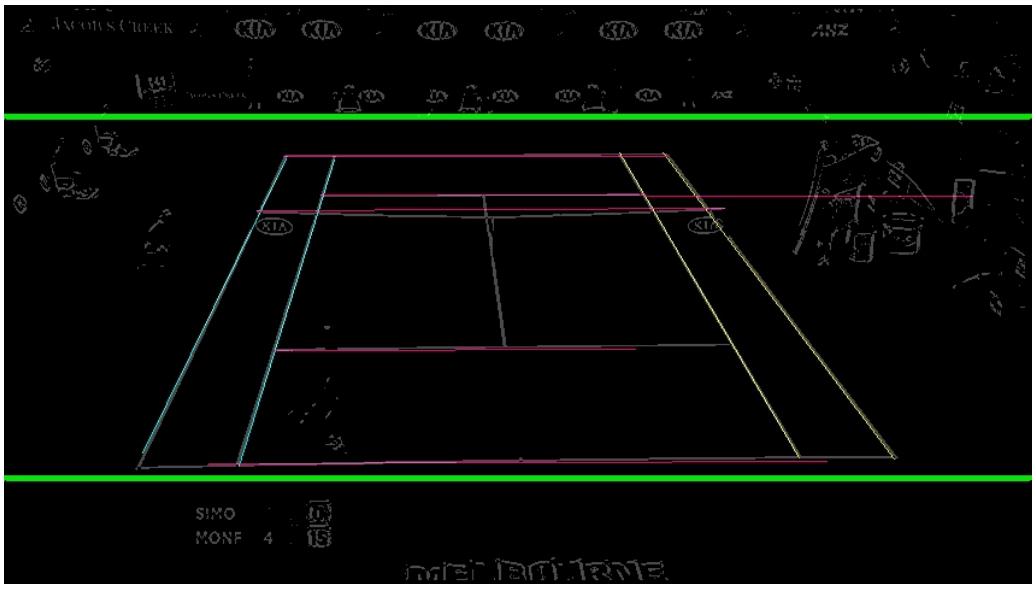


Figure C.3: Image resulting from lines detection, classification and merging.

Appendix D

Background Generation Example

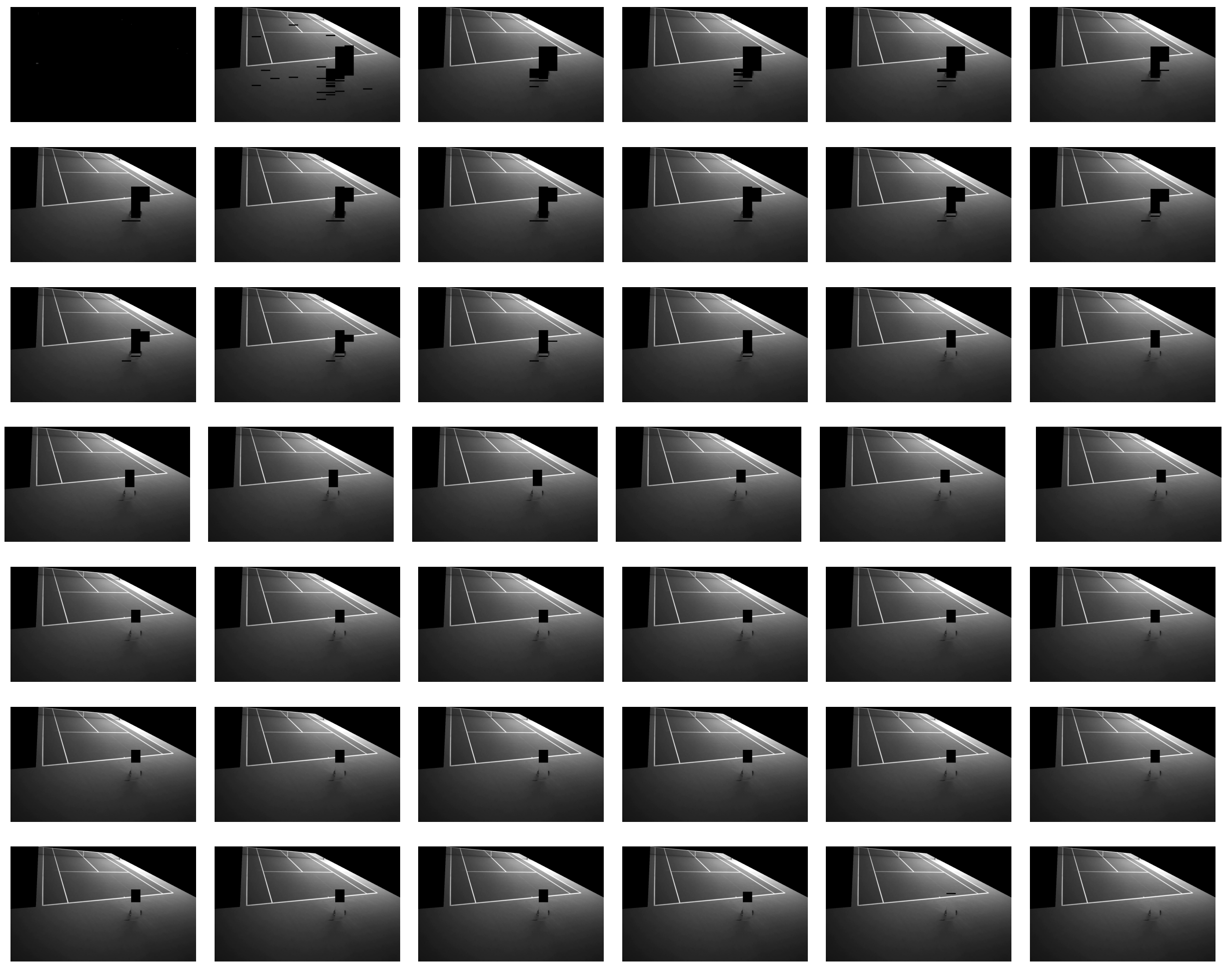


Figure D.1: Background generation example. Initially the background image is empty and it progressively incorporates background blocks until a complete background image is generated. Described in Section 4.4.2.

Appendix E

Tennis Court Projection

Section 4.3.2 and Figure 4.7 show how a model of the court lines is projected onto a frame based on a hypothesis of four point correspondences. This is achieved by creating a binary image (black and white) to model a tennis court with its standardised dimensions, shown in Figure 2.1, and with court lines in white. It is shown in black in Figure E.1 for clarity but white is used to simulate the real colour, which will be useful for calculating the court projection accuracy Section 4.3.2. As shown in the example of Figure E.1, the homography matrix H is calculated via the perspective transformation[1] of points a, b, c and d in the model and the corresponding a', b', c' and d' in the image. Then, the projection of the court lines onto the image is a simple warp of the tennis court model using H and resulting in Figure E.2).

[1]Using the OpenCV implementation [43].

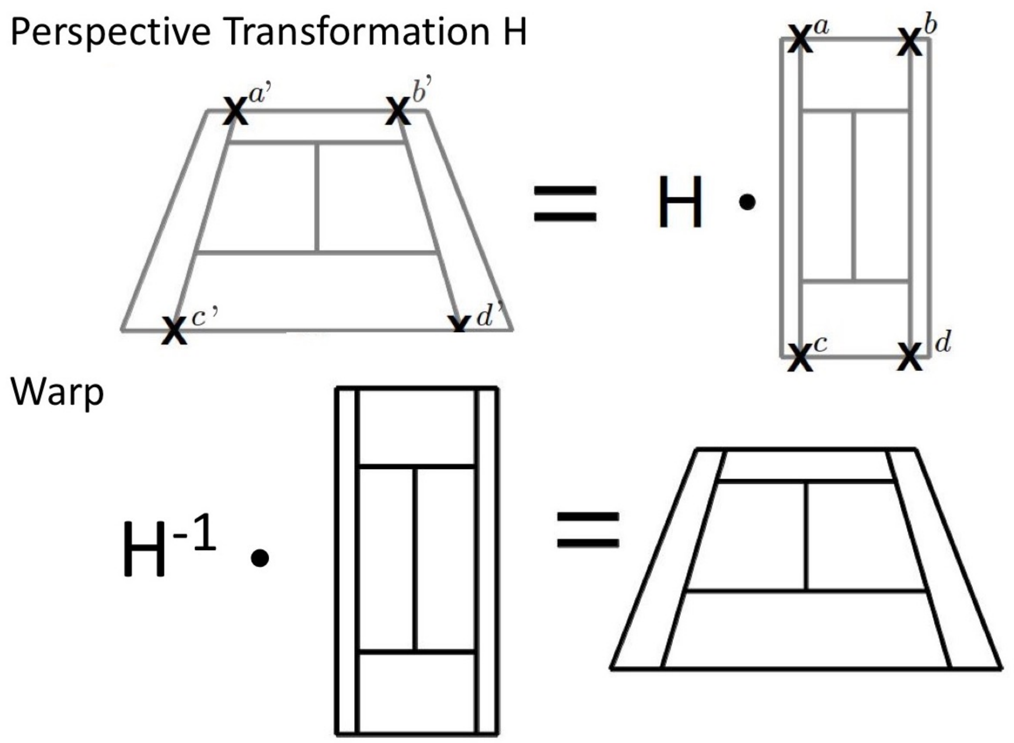


Figure E.1: Court homography.

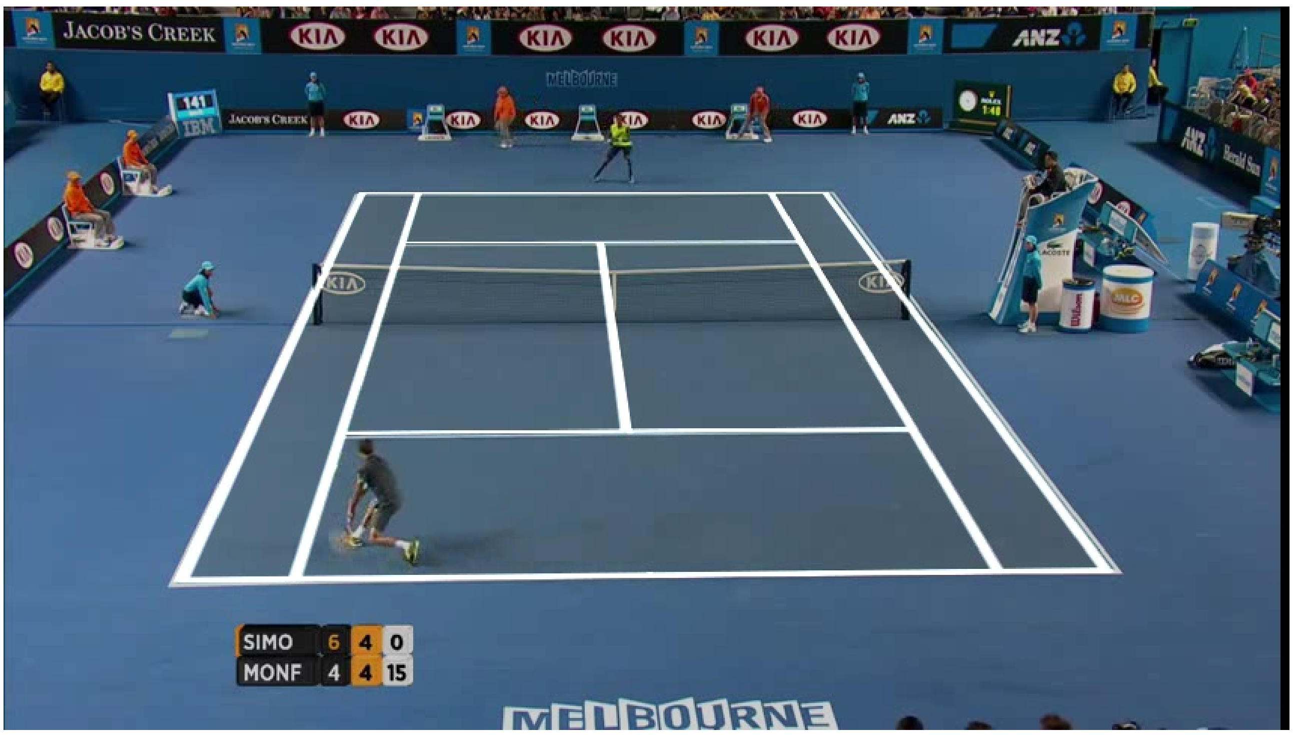


Figure E.2: Court lines projected onto the video frame.

Appendix F

Tennis Court Configurations

This figure shows eleven diagrams of a tennis court. In Section 4.3.2 we explained how different correspondences between 4 points of the court in the frame and a model of the court are tested. For each configuration, the four points in the frame are associated with the four red circles.

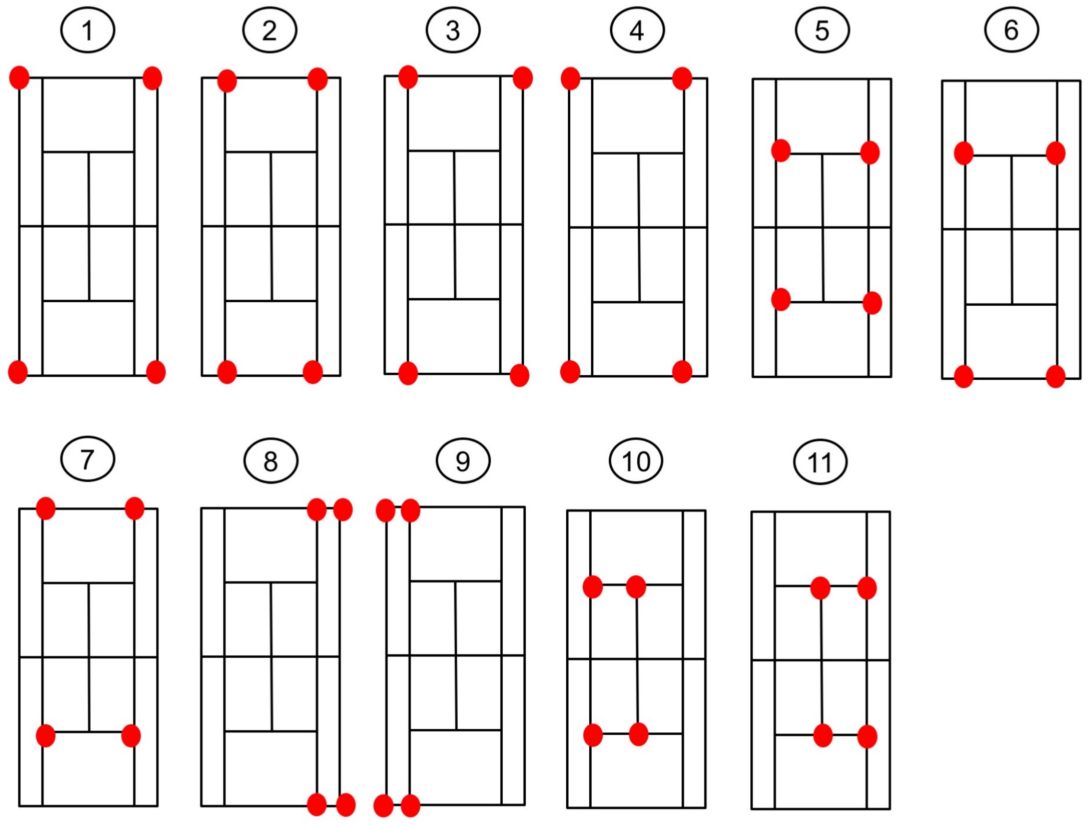


Figure F.1: Court configurations.

Appendix G

Kalman Filter

Named after Rudolf Emil Kálmán, this iterative algorithm is based on Bayesian inference and estimates unknown variables from a set of observations, which are assumed to contain Gaussian noise [89]. It performs two steps at each iteration:

- **Prediction:** estimate the state of the system $\hat{\mathbf{x}}_{t|t-1}$ at time t.
- **Update:** upon getting the observation at time t the best state estimate $\hat{\mathbf{x}}_{t|t}$ is calculated by multiplying the probability density function (pdf) of the prediction estimate and of the current observation. Both of these are Gaussian distributions and therefore, the resulting pdf of the best estimate $\hat{\mathbf{x}}_{t|t}$ is also a Gaussian distribution. The relative weight of the prediction estimate and current observation depends on their uncertainty and is controlled by a variable called the Kalman Gain, Equation (G.3).

We will use a simple example to explain Kalman filters: a tennis ball trajectory and its observation using a computer vision application to detect its pixel coordinates in video frames. The system state $\mathbf{x}_t$ at time t will be the 3D coordinates of the ball position in the real world and its velocity. The estimate of the state at time t will be called $\hat{\mathbf{x}}_t$ and the observed ball position with noise $\mathbf{z}_t$ (e.g. if it was obtained through a computer vision algorithm that is not perfect). The covariance matrices associated with the observation noise and the prediction $\hat{\mathbf{x}}_{t|t-1}$ of an unknown true state $\mathbf{x}_t$ are $\mathbf{R}_t$ and $\mathbf{P}_t$ respectively. Using this notation the prediction step is calculated as follows [185]:

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t\hat{\mathbf{x}}_{t-1|t-1} + \mathbf{B}_t\mathbf{u}_t \tag{G.1}$$

where $\mathbf{F}$ is the state transition matrix for the modelled system (ball trajectory), $\mathbf{B}$ is the control input matrix (to model the effect of $\mathbf{u}$) and $\mathbf{u}$ is the vector of control input parameters (e.g. car motion affected by the brake) and:

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t\mathbf{P}_{t|t-1}\mathbf{F}_t^T + \mathbf{Q}_t \tag{G.2}$$

where $\mathbf{Q}$ is the noise covariance matrix of the control inputs.

The Kalman Gain $\mathbf{K}_t$ used in the update stage is calculated as follows:

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^T(\mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^T + \mathbf{R}_t)^{-1} \tag{G.3}$$

where $\mathbf{H}$ is the transformation matrix, which maps the state vector parameters to the observation domain (e.g. real-world 3D position and velocity to pixel coordinates on the frame).

A high value for K means that state measurements are more accurate and the estimations more unstable. Therefore, new estimates will be more influenced by the measurements than by the previous estimates. This is easy to see in the following equation for calculating the state estimation in the update step [185]:

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}_t\hat{\mathbf{x}}_{t|t-1}) \tag{G.4}$$

Finally, when the new estimation is obtained, its error is estimated as follows:

$$\mathbf{P}_t = \mathbf{P}_{t|t-1} - \mathbf{K}_t\mathbf{H}_t\mathbf{P}_{t|t-1} \tag{G.5}$$

A major advantage of this algorithm is that it does not need to have all the observations in order to calculate an estimate of the real value. This makes it very suitable for on-line applications, such a ball tracking. At each iteration, a new data point is fed in, and the estimation gets closer to the real value.

In practice, the Kalman filter is particularly suited to object tracking in videos [186]. As described in Section 4.5.1, the Kalman filter can be used in tennis ball tracking by defining the search window for the next ball candidate centered at $\hat{\mathbf{x}}_{t|t-1}$. When the next ball candidate is detected, the Kalman filter is updated with this observation, as shown in Equation (G.4). If the next ball candidate is not found (e.g. due to occlusion) the predicted state is used to continue with the tracking. More details about the use of Kalman filter in sports for ball tracking can be found in [187–191].

Appendix H

Particle Filter

A Particle Filter is a sampling method used to solve problems in signal processing and to perform tractable Bayesian inference [192; 193]. The latter is the case in ball tracking algorithms, as mentioned in Section 4.5.1.

The Particle Filter sampling method sequentially estimates states given observations as follows:

1. A set of particles represents the posterior distribution of a stochastic process given noisy or partial observations. Particles have weights according to their likelihood.
2. Re-sampling is performed based on the particles' weights, but new samples are given equal weights.
3. New particles are generated with weights based on the posterior distribution of a stochastic process given the particles from the previous step.
4. Start the process again.

[128] describes in more detail how this is applied to ball tracking.

Appendix I

Givens Rotations for QR Decomposition

In Section 5.4.2 we mention that Givens rotations are used for the calculation of the QR decomposition to solve Equation (5.3): $Ax = b$.

The QR decomposition of A can be used to solve the previous equation. This means that

$$A = QR \tag{I.1}$$

with the constraints that Q is an orthogonal matrix and R is an upper triangular matrix[1]. From this:

$$QRx = b \tag{I.2}$$

therefore:

$$Q^T QRx = Q^T b \tag{I.3}$$

and

$$Rx = Q^T b \tag{I.4}$$

because Q is an orthogonal matrix, which means that $Q^T Q = I$. Equation (I.4) is straightforward to solve by back substitution since R is an upper triangular matrix. This is the reason for using QR decomposition of A to solve Equation (5.3).

[1]Upper triangular matrix: every element below the diagonal is zero.

The QR decomposition of A is obtained through Givens rotations, which can be used to zero out isolated elements of any matrix. A Givens rotation $G(i, j, \theta)$ can be represented as a matrix where:

$$g_{kk} = 1 \text{ for k} \neq \text{i, j}, \quad g_{kk} = c \text{ for k = i, j}, \quad g_{ji} = -g_{ji} = s \tag{I.5}$$

$c = \cos\,\theta$ and $s = \sin\,\theta$ for a rotation of θ radians in the (i, j) plane.

To calculate the QR decomposition of a matrix A, R can be obtained by applying Givens rotations that will zero out each element below the diagonal in turn. To zero out the element $A_{i,j}$, a Givens rotation matrix $G_1(i, j, \theta)$ defined as in Equation (I.5) with:

$$c = a/r \text{ and } \quad s = -b/r \tag{I.6}$$

with

$$r = \sqrt{a^2 + b^2} \text{ given } \quad a = A_{i-1,j} \text{ and } \quad b = A_{i,j} \tag{I.7}$$

The multiplication of $G_1(i, j, \theta)$ by A (with A on the right) results in A_2, with the element at the position (i, j) set to zero. The same process is applied to A_2 by applying a Givens rotation matrix $G_2(i, j, \theta)$ to zero out one element from A_2 . This process is repeated until all elements below the diagonal are zero. At this point, A is the R matrix, and Q corresponds to the product of Givens rotations matrices ($G1$, $G2$ etc.). More formally:

$$\text{If} \quad A = QR \text{ then } \quad Q^T A = Q^T QR = R \tag{I.8}$$

$$\text{then} \quad Q^T A = G_m \ldots G_3 G_2 G_1 A \tag{I.9}$$

$$\text{so} \quad Q = (G_m \ldots G_3 G_2 G_1)^T = G_1^T G_2^T G_3^T \ldots G_m^T \tag{I.10}$$

Bibliography

[1] G. Pingali, Y. Jean, A. Opalach, and I. Carlbom, “LucentVision: Converting Real World Events into Multimedia Experiences,” in *IEEE International Conference on Multimedia and Expo*, vol. 3, pp. 1433–1436, IEEE, 2000.

[2] M. Lewis, *Moneyball: The Art of Winning an Unfair Game.* WW Norton & Company, 2004.

[3] L. Steinberg, “Changing The Game: The Rise of Sports Analytics,” in *Forbes*, April 2015.

[4] Y. Baodong, “Hawk-Eye Technology using Tennis Match,” *Computer Modelling and New Technologies*, vol. 18, no. 12C, pp. 400–402, 2014.

[5] C. Shachar, E. Khazanov, and B. Yoram, “Smart Court System,” 2014.

[6] X. Wei, P. Lucey, S. Morgan, and S. Sridharan, “Predicting Shot Locations in Tennis Using Spatiotemporal Data,” (Hobart, Tasmania, Australia), pp. 1–8, IEEE, November 2013.

[7] G. Hunter and K. Zienowicz, “Can Markov Models Accurately Simulate Lawn Tennis Rallies ?,” in *2nd I.M.A. International Conference on Mathematics in Sport*, IEEE, 2009.

[8] D. Spanias, *Professional Tennis: Quantitative Models and Ranking Algorithms.* PhD thesis, Imperial College London, London, UK, 2015.

[9] M.-K. Hu, “Visual Pattern Recognition by Moment Invariants,” *IRE Transactions on Information Theory*, vol. 8, no. 2, pp. 179–187, 1962.

[10] S. Gourgari, G. Goudelis, K. Karpouzis, and S. Kollias, "THETIS: Three Dimensional Tennis Shots a Human Action Dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (Workshops)*, pp. 676–681, IEEE, 2013.

[11] S. Vinyes Mora, G. Barnett, C. da Costa-Luis, J. Garcia Maegli, M. Ingram, J. Neale, and W. J. Knottenbelt, "A Low-Cost Spatiotemporal Data Collection System for Tennis," in *Proceedings of the 5th Mathematics in Sport Conference*, (Loughborough, UK), June 2015.

[12] S. Vinyes Mora and W. J. Knottenbelt, "Spatio-Temporal Analysis of Tennis Matches," in *Proceedings of the 3rd ACM KKD Workshop on Large Scale Sports Analytics*, (San Francisco, California, USA), August 2016.

[13] S. Vinyes Mora and W. J. Knottenbelt, "Deep Learning for Domain-Specific Action Recognition in Tennis," in *Proceedings of the 3rd IEEE International CVPR Workshop on Computer Vision in Sports*, (Honolulu, Hawaii), July 2017.

[14] S. A. Kovalchik, "Searching for the GOAT of Tennis Win Prediction," *Journal of Quantitative Analysis in Sports*, vol. 12, no. 3, pp. 127–138, 2016.

[15] J. G. Kemeny and J. L. Snell, *Finite Markov Chains.* van Nostrand Princeton, NJ, 1960.

[16] T. J. Barnett and S. R. Clarke, "Using Microsoft Excel to Model a Tennis Match," in *6th Conference on Mathematics and Computers in Sport*, pp. 63–68, Bond University, Queensland, Australia, 2002.

[17] T. Barnett, A. Brown, and S. Clarke, "Developing a Model that Reflects Outcomes of Tennis Matches," in *Proceedings of the 8th Australasian Conference on Mathematics and Computers in Sport, Coolangatta, Queensland*, pp. 178–188, 2006.

[18] A. J. O'Malley, "Probability Formulas and Statistical Analysis in Tennis," *Journal of Quantitative Analysis in Sports*, vol. 4, no. 2, p. 15, 2008.

[19] F. J. G. M. Klaassen and J. R. Magnus, “Are Points in Tennis Independent and Identically Distributed? Evidence from a Dynamic Binary Panel Data Model,” *Journal of the American Statistical Association*, vol. 96, no. 454, pp. 500–509, 2001.

[20] T. Barnett and S. R. Clarke, “Combining Player Statistics to Predict Outcomes of Tennis Matches,” *IMA Journal of Management Mathematics*, vol. 16, no. 2, pp. 113–120, 2005.

[21] W. J. Knottenbelt, D. Spanias, and A. M. Madurska, “A Common-Opponent Stochastic Model for Predicting the Outcome of Professional Tennis Matches,” *Computers & Mathematics with Applications*, vol. 64, no. 12, pp. 3820–3827, 2012.

[22] B. Goldsack, *A Novel Impact-Based Model for Predicting the Outcome of Professional Singles Tennis Matches.* MSc Thesis, Imperial College London, 2013.

[23] G. Hunter, A. Shihab, and K. Zienowicz, “Modelling Tennis Rallies using Information From Both Video and Audio Signals,” in *International Conference on Mathematics in Sport*, 2007.

[24] X. Wei, P. Lucey, S. Morgan, P. Carr, M. Reid, and S. Sridharan, “Predicting Serves in Tennis using Style Priors,” in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 2207–2215, ACM, 2015.

[25] X. Wei, P. Lucey, S. Morgan, M. Reid, and S. Sridharan, ““The Thin Edge of the Wedge”: Accurately Predicting Shot Outcomes in Tennis using Style and Context Priors,” *Proceedings of the MIT Sloan Sports Analytics Conference*, 2016.

[26] X. Wei, P. Lucey, S. Morgan, and S. Sridharan, “Forecasting the Next Shot Location in Tennis Using Fine-Grained Spatio-Temporal Tracking Data,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 11, pp. 2988–2997, 2016.

[27] I. Biederman, “Recognition-by-Components: A Theory of Human Image Understanding,” *Psychological R eview*, vol. 94, no. 2, p. 115, 1987.

[28] D. H. Ballard, G. E. Hinton, T. J. Sejnowski, *et al.*, “Parallel Visual Computation,” *Nature*, vol. 306, no. 5938, pp. 21–26, 1983.

[29] J. J. DiCarlo, D. Zoccolan, and N. C. Rust, "How Does the Brain Solve Visual Object Recognition?," *Neuron*, vol. 73, no. 3, pp. 415–434, 2012.

[30] D. H. Hubel and T. N. Wiesel, "Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex," *The Journal of Physiology*, vol. 160, no. 1, pp. 106–154, 1962.

[31] D. G. Lowe, "Object Recognition from Local Scale-Invariant Features," in *The Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157, IEEE, 1999.

[32] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields," *arXiv preprint arXiv:1611.08050*, 2016.

[33] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and Tell: A Neural Image Caption Generator," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164, 2015.

[34] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer Science & Business Media, 2010.

[35] J. Herault, *Biologically Inspired Computer Vision: Fundamentals and Applications*. John Wiley & Sons, 2015.

[36] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proceedings of the International Joint Conference on Artificial Intelligence*, Vancouver, BC, Canada, 1981.

[37] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *Alvey Vision Conference*, vol. 15, pp. 10–5244, Manchester, UK, 1988.

[38] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, IEEE, 2005.

[39] R. Poppe, "A Survey on Vision-Based Human Action Recognition," *Image and Vision Computing*, vol. 28, no. 6, pp. 976–990, 2010.

[40] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[41] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer, 2006.

[42] F. Masci, "Line Detection by Hough Transformation." Available at `http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/HoughTrans_lines_09.pdf`, 2009.

[43] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[44] J. Matas, C. Galambos, and J. Kittler, "Robust Detection of Lines Using the Progressive Probabilistic Hough Transform," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 119–137, 2000.

[45] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, vol. 3, pp. 210–229, July 1959.

[46] T. M. Mitchell, *Machine Learning.* McGraw-Hill, Boston, MA, 1997.

[47] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT press, 2016.

[48] E. Alpaydin, *Introduction to Machine Learning.* MIT press, 2014.

[49] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1. MIT Press, Cambridge, MA, USA, 1998.

[50] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The Arcade Learning Environment: An Evaluation Platform for General Agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[51] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, "Achieving Human Parity in Conversational Speech Recognition," *arXiv preprint arXiv:1610.05256*, 2016.

[52] B. Li, T. Sainath, A. Narayanan, J. Caroselli, M. Bacchiani, A. Misra, I. Shafran, H. Sak, G. Pundak, K. Chin, *et al.*, "Acoustic Modeling for Google Home," *Proceddings of Interspeech*, 2017.

[53] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using Collaborative Filtering to Weave an Information Tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[54] P. Resnick and H. R. Varian, "Recommender Systems," *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[55] G. A. Wiggins, "Searching for Computational Creativity," *New Generation Computing*, vol. 24, no. 3, pp. 209–222, 2006.

[56] A. Elgammal, B. Liu, M. Elhoseiny, and M. Mazzone, "CAN: Creative Adversarial Networks, Generating "Art" by Learning About Styles and Deviating from Style Norms," *arXiv preprint arXiv:1706.07068*, 2017.

[57] L. A. Gatys, A. S. Ecker, and M. Bethge, "A Neural Algorithm of Artistic Style," *arXiv preprint arXiv:1508.06576*, 2015.

[58] F. Rosenblatt, "Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms," tech. rep., Cornell Aeronautical Lab Inc Buffalo, NY, 1961.

[59] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1," ch. Learning Internal Representations by Error Propagation, pp. 318–362, Cambridge, MA, USA: MIT Press, 1986.

[60] E. R. Kandel, J. H. Schwartz, T. M. Jessell, S. A. Siegelbaum, A. J. Hudspeth, *et al.*, *Principles of Neural Science*, vol. 4. McGraw-Hill, New York, 2000.

[61] F. Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain," *Psychological Review*, vol. 65, no. 6, p. 386, 1958.

[62] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

[63] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[64] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[65] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, *et al.*, "Imagenet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[66] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[67] J. L. Elman, "Finding Structure in Time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[68] A. Graves *et al.*, *Supervised Sequence Labelling with Recurrent Neural Networks*, vol. 385 of *Studies in Computational Intelligence.* Springer, 2012.

[69] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks," in *Proceedings of the IEEE Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, IEEE, 2013.

[70] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[71] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[72] W. Zaremba, I. Sutskever, and O. Vinyals, “Recurrent Neural Network Regularization,” *arXiv preprint arXiv:1409.2329*, 2014.

[73] L. Breiman, “Random Forests,” *UC Berkeley TR567*, 1999.

[74] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[75] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees.* CRC press, 1984.

[76] Louppe, Gilles, *Understanding Random Forests.* PhD Thesis, University of Liege, 2014.

[77] L. Breiman, “Manual on Setting up, Using and Understanding Random Forest,” *V4. 0, University of California Berkeley, Statistics Department, Berkeley*, 2003.

[78] S. Lloyd, “Least Squares Quantization in PCM,” *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[79] A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum Likelihood from Incomplete Data Via the EM Algorithm,” *Journal of the Royal Statistical Society. Series B (methodological)*, pp. 1–38, 1977.

[80] V. Vapnik, *The Nature of Statistical Learning Theory.* Springer Science & Business Media, 1995.

[81] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*, vol. 1. MIT Press, Cambridge, MA, USA, 2006.

[82] K. P. Bennett and O. L. Mangasarian, “Robust Linear Programming Discrimination of Two Linearly Inseparable Sets,” *Optimization Methods and Software*, vol. 1, no. 1, pp. 23–34, 1992.

[83] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[84] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support Vector Regression Machines," in *Advances in Neural Information Processing Systems*, pp. 155–161, 1997.

[85] V. N. Vapnik and V. Vapnik, *Statistical Learning Theory*, vol. 1. Wiley New York, 1998.

[86] C.-W. Hsu and C.-J. Lin, "A Comparison of Methods for Multiclass Support Vector Machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[87] T. Dean and K. Kanazawa, "A Model for Reasoning about Persistence and Causation," *Computational intelligence*, vol. 5, no. 2, pp. 142–150, 1989.

[88] K. P. Murphy, *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD Thesis, University of California, Berkeley, 2002.

[89] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[90] R. J. Elliott, L. Aggoun, and J. B. Moore, *Hidden Markov Models: Estimation and Control.* Springer Science & Business Media, 2008.

[91] G. J. McLachlan and K. E. Basford, *Mixture Models: Inference and Applications to Clustering*, vol. 84. Marcel Dekker, 1988.

[92] C. Aviles, N. Benguigui, E. Beaudoin, and F. Godart, "Developing Early Perception and Getting Ready for Action on the Return of Serve," *International Tennis Federation Coaching & Sport Science Review*, vol. 28, no. 6, p. 8, 2002.

[93] R. N. Singer, J. H. Cauraugh, D. Chen, G. M. Steinberg, and S. G. Frehlich, "Visual Search, Anticipation, and Reactive Comparisons Between Highly-Skilled and Beginning Tennis Players," *Journal of Applied Sport Psychology*, vol. 8, no. 1, pp. 9–26, 1996.

[94] L. S. Overney, O. Blanke, and M. H. Herzog, "Enhanced Temporal but not Attentional Processing in Expert Tennis Players," *PLoS One*, vol. 3, no. 6, p. e2380, 2008.

[95] H. Collins and R. Evans, "You Cannot Be Serious! Public Understanding of Technology with Special Reference to Hawk-Eye," *Public Understanding of Science*, vol. 17, no. 3, pp. 283–308, 2008.

[96] B. Dyer, "The Controversy of Sports Technology: a Systematic Review," *SpringerPlus*, vol. 4, no. 1, p. 524, 2015.

[97] United States Tennis Association, *Tennis Tactics: Winning Patterns of Play.* Human Kinetics, 1996.

[98] S. Intille and A. Bobick, "Representation and Visual Recognition of Complex, Multi-Agent Actions using Belief Networks," *Proceedings of the CVPR Workshop on Interpretation of Visual Motion*, 1998.

[99] S. S. Intille and A. F. Bobick, "A Framework for Recognizing Multi-Agent Action from Visual Evidence," in *Proceedings of the Innovative Applications of Artificial Intelligence Conference*, pp. 518–525, American Association for Artificial Intelligence, 1999.

[100] W. E. L. Grimson, D. P. Huttenlocher, *et al.*, *Object Recognition by Computer: the Role of Geometric Constraints.* MIT Press, 1990.

[101] W. E. L. Grimson and T. Lozano-Perez, "Localizing Overlapping Parts by Searching the Interpretation Tree," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-9, no. 4, pp. 469–482, 1987.

[102] A. Stevenson, *Oxford Dictionary of English.* Oxford University Press, 2010.

[103] M. Crespo and M. Reid, *Coaching Beginner and Intermediate Tennis Players: A Manual of the ITF Coaching Programme.* International Tennis Federation, 2009.

[104] M. Crespo and M. Reid, *What Research Tells Us About Strategy and Tactics.* International Tennis Federation, 2001.

[105] J. R. Wang and N. Parameswaran, "Analyzing Tennis Tactics from Broadcasting Tennis Video Clips," in *International Multimedia Modelling Conference*, pp. 102–106, IEEE, 2005.

[106] W.-T. Chu and W.-H. Tsai, “Modeling Spatio-Temporal Relationships Between Moving Objects for Event Tactics Analysis in Tennis Videos,” *Multimedia Tools and Applications*, vol. 50, no. 1, pp. 149–171, 2010.

[107] L. Bergroth, H. Hakonen, and T. Raita, “A Survey of Longest Common Subsequence Algorithms,” in *Seventh International Symposium on String Processing and Information Retrieval*, pp. 39–48, IEEE, 2000.

[108] A. Terroba, W. Kosters, J. Varona, and C. S. Manresa-Yee, “Finding Optimal Strategies in Tennis from Video Sequences,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 27, no. 06, 2013.

[109] A. Terroba, W. A. Kosters, and J. K. Vis, “Tactical Analysis Modeling through Data Mining,” in *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval*, pp. 176–181, 2010.

[110] J. K. Vis, W. A. Kosters, and A. Terroba, “Tennis Patterns: Player, Match and Beyond,” in *22nd Benelux Conference on Artificial Intelligence, Luxembourg*, pp. 25–26, 2010.

[111] X. Wei, P. Lucey, S. Morgan, and S. Sridharan, “Sweet-Spot: Using Spatiotemporal Data to Discover and Predict Shots in Tennis,” in *MIT Sloan Sports Analytics Conference*, 2013.

[112] P. G. O’Donoghue and E. Brown, “The Importance of Service in Grand Slam Singles Tennis,” *International Journal of Performance Analysis in Sport*, vol. 8, no. 3, pp. 70–78, 2008.

[113] G. Mather, “Perceptual Uncertainty and Line-Call Challenges in Professional Tennis,” *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 275, no. 1643, pp. 1645–1651, 2008.

[114] D. Farin, S. Krabbe, and W. Effelsberg, “Robust Camera Calibration for Sport Videos Using Court Models,” *Proceedings of SPIE*, vol. 5307, pp. 80–91, 2003.

[115] B. Dang, A. Tran, T. Dinh, and T. Dinh, "A Real Time Player Tracking System for Broadcast Tennis Video," *Intelligent Information and Database Systems*, vol. 5991, pp. 105–113, 2010.

[116] O. Chum, T. Werner, and J. Matas, "Epipolar Geometry Estimation via RANSAC Benefits from the Oriented Epipolar Constraint," in *Proceedings of the International Conference on Pattern Recognition*, vol. 1, pp. 112—-115 Vol.1, 2004.

[117] S. Vinyes Mora, *Video-Based Analysis of Professional Tennis Matches.* MSc Thesis, Imperial College London, 2013.

[118] J. Han, D. Farin, *et al.*, "Generic 3D Modeling for Content Analysis of Court-Net Sports Sequences," in *Proceedings of the 13th International Conference on Multimedia Modeling*, pp. 279–288, Springer, 2007.

[119] G. Sudhir, J. C.-M. Lee, and A. K. Jain, "Automatic Classification of Tennis Video for High-Level Content-Based Retrieval," in *Proceedings of the IEEE International Workshop on Content-Based Access of Image and Video Database*, pp. 81—-90, 1998.

[120] Y.-C. Jiang, K.-T. Lai, C.-H. Hsieh, and M.-F. Lai, "Player Detection and Tracking in Broadcast Tennis Video," in *Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology*, pp. 759–770, Springer-Verlag, 2008.

[121] H. Miyamori and S.-I. Iisaku, "Video Annotation for Content-Based Retrieval Using Human Behavior Analysis and Domain Knowledge," *IEEE International Conference on Automatic Face and Gesture Recognition*, pp. 320–325, 2000.

[122] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (2nd Ed).* Prentice Hall, 2002.

[123] S. Suzuki, "Topological Structural Analysis of Digitized Binary Images by Border Following," *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46, 1985.

[124] G. S. Pingali, Y. Jean, and I. Carlbom, "Real Time Tracking for Enhanced Tennis Broadcasts," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 260–265, 1998.

[125] K. Teachabarikiti, T. H. Chalidabhongse, and A. Thammano, "Players Tracking and Ball Detection for an Automatic Tennis Video Annotation," in *11th International Conference on Control Automation Robotics Vision*, (Singapore), pp. 2461–2494, 2010.

[126] C. O'Conaire, P. Kelly, D. Connaghan, and N. E. O'Connor, "Tennissense: A Platform for Extracting Semantic Information from Multi-Camera Tennis Data," in *Proceedings of the 16th International Conference on Digital Signal Processing*, pp. 1–6, IEEE, 2009.

[127] F. Yan, W. Christmas, and J. Kittler, "A Tennis Ball Tracking Algorithm for Automatic Annotation of Tennis Match," *Procedings of the British Machine Vision Conference 2005*, pp. 67.1–67.10, 2005.

[128] X. Yu, C.-H. Sim, J. R. Wang, and L. F. Cheong, "A Trajectory-Based Ball Detection and Tracking Algorithm in Broadcast Tennis Video," in *International Conference on Image Processing*, vol. 2, pp. 1049–1052, IEEE, 2004.

[129] F. Yan, A. Kostin, W. Christmas, and J. Kittler, "A Novel Data Association Algorithm for Object Tracking in Clutter with Application to Tennis Video Analysis," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 634–641, IEEE, 2006.

[130] K. Thormann, F. Sigges, and M. Baum, "Learning an object tracker with a random forest and simulated measurements," in *Information Fusion (Fusion), 2017 20th International Conference on*, pp. 1–4, IEEE, 2017.

[131] George Barnett, Casper da Costa-Luis, Jose Garcia Maegli, Martin Ingram, James Neale, *The Intelligent Tennis Court.* MSc Group Project, Imperial College London, 2013.

[132] J.-Y. Bouguet, "Complete camera calibration toolbox for matlab (1999)." Available at `http://www.vision.caltech.edu/bouguetj`.

[133] Z. Zhang, “A Flexible New Technique for Camera Calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[134] Y. Abdel-Aziz, “Direct Linear Transformation from Comparator Coordinates in Close-Range Photogrammetry,” in *ASP Symposium on Close Range Photogrammetry.*, pp. 1–19, Falls Church (VA). American Society of Photogrammetry, 1971.

[135] R. I. Hartley and P. Sturm, “Triangulation,” *Computer Vision and Image Understanding*, vol. 68, no. 2, pp. 146–157, 1997.

[136] P. A. Beardsley, A. Zisserman, and D. W. Murray, “Navigation Using Affine Structure from Motion,” in *European Conference on Computer Vision*, pp. 85–96, Springer, 1994.

[137] J. Bertin, “Semiology of Graphics: Diagrams, Networks, Maps,” *Madison, WI: The University of Wisconsin Press*, 1983.

[138] X. Yu and D. Farin, “Current and Emerging Topics in Sports Video Processing,” in *IEEE International Conference on Multimedia and Expo*, pp. 526–529, IEEE, 2005.

[139] N. Owens, C. Harris, and C. Stennett, “Hawk-Eye Tennis System,” in *International Conference on Visual Information Engineering*, pp. 182–185, Institution of Electrical Engineers, 2003.

[140] G. Pingali, A. Opalach, Y. Jean, and I. Carlbom, “Visualization of Sports Using Motion Trajectories: Providing Insights into Performance, Style, and Strategy,” in *Proceedings of the Conference on Visualization*, pp. 75–82, IEEE Computer Society, 2001.

[141] D. Demaj, “Geovisualizing Spatio-Temporal Patterns in Tennis: An Alternative Approach to Post-Match Analysis.” Available at `http://gamesetmap.com/`, 2013.

[142] J. Gudmundsson and M. Horton, “Spatio-Temporal Analysis of Team Sports,” *ACM Computing Surveys*, vol. 50, no. 2, p. 22, 2017.

[143] G. McGhee, “Using Maps and Data Vis to Understand Tennis.” National Geographic Data Points Series. Available at `http://news.nationalgeographic.com/2015/09/150906-data-points-tennis-tracking/`, 2016.

[144] F. Giampaolo and J. Levey, *Championship Tennis*. Human Kinetics, 2013.

[145] R. Waite, "Playing in the Zones." Available at `http://www.tennisserver.com/`, 2015.

[146] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning Realistic Human Actions from Movies," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3222–3229, IEEE, 2008.

[147] H. Wang and C. Schmid, "Action Recognition with Improved Trajectories," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3551–3558, 2013.

[148] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-Scale Video Classification with Convolutional Neural Networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, IEEE, 2014.

[149] C. Schuldt, I. Laptev, and B. Caputo, "Recognizing Human Actions: A Local SVM Approach," in *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 3, pp. 32–36, IEEE, 2004.

[150] X. Baró, J. Gonzalez, J. Fabian, M. A. Bautista, M. Oliu, H. Jair Escalante, I. Guyon, and S. Escalera, "Chalearn Looking at People 2015 Challenges: Action Spotting and Cultural Event Recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (Workshops)*, pp. 1–9, 2015.

[151] M. D. Rodriguez, J. Ahmed, and M. Shah, "Action MACH: A Spatio-Temporal Maximum Average Correlation Height Filter for Action Recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3001–3008, IEEE, 2008.

[152] K. Soomro and A. R. Zamir, "Action Recognition in Realistic Sports Videos," in *Computer Vision in Sports*, pp. 181–208, Springer, 2014.

[153] G. Zhu, C. Xu, W. Gao, and Q. Huang, "Action Recognition in Broadcast Tennis Video Using Optical Flow and Support Vector Machine," in *European Conference on Computer Vision*, pp. 89–98, Springer, 2006.

[154] G. Zhu, C. Xu, Q. Huang, W. Gao, and L. Xing, "Player Action Recognition in Broadcast Tennis Video with Applications to Semantic Analysis of Sports Game," in *Proceedings of the 14th ACM International Conference on Multimedia*, pp. 431–440, ACM, 2006.

[155] N. FarajiDavar, T. De Campos, J. Kittler, and F. Yan, "Transductive Transfer Learning for Action Recognition in Tennis Games," in *IEEE International Conference on Computer Vision (Workshops)*, pp. 1548–1553, IEEE, 2011.

[156] T. De Campos, M. Barnard, K. Mikolajczyk, J. Kittler, F. Yan, W. Christmas, and D. Windridge, "An Evaluation of Bags-of-Words and Spatio-Temporal Shapes for Action Recognition," in *IEEE Workshop on Applications of Computer Vision (WACV)*, pp. 344–351, IEEE, 2011.

[157] N. Dalal, B. Triggs, and C. Schmid, "Human Detection Using Oriented Histograms of Flow and Appearance," in *European Conference on Computer Vision*, pp. 428–441, Springer, 2006.

[158] P. Dollar, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior Recognition via Sparse Spatio-Temporal Features," in *2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pp. 65–72, IEEE, 2005.

[159] A. A. Efros, A. C. Berg, G. Mori, and J. Malik, "Recognizing Action at a Distance," in *Proceeding of the IEEE International Conference on Computer Vision.*, pp. 726–733, 2003.

[160] S. Ali and M. Shah, "Human Action Recognition in Videos using Kinematic Features and Multiple Instance Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 2, pp. 288–303, 2010.

[161] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu, "Action Recognition by Dense Trajectories," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3169–3176, IEEE, 2011.

[162] X. Peng, L. Wang, X. Wang, and Y. Qiao, "Bag of Visual Words and Fusion Methods for

Action Recognition: Comprehensive Study and Good Practice," *Computer Vision and Image Understanding*, vol. 150, pp. 109–125, 2016.

[163] P. Matikainen, M. Hebert, and R. Sukthankar, "Trajectons: Action Recognition Through the Motion Analysis of Tracked Features," in *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pp. 514–521, IEEE, 2009.

[164] R. Messing, C. Pal, and H. Kautz, "Activity Recognition Using the Velocity Histories of Tracked keypoints," in *Computer Vision, 2009 IEEE 12th International Conference on*, pp. 104–111, IEEE, 2009.

[165] J. Sun, X. Wu, S. Yan, L.-F. Cheong, T.-S. Chua, and J. Li, "Hierarchical Spatio-Temporal Context Modeling for Action Recognition," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 2004–2011, IEEE, 2009.

[166] X. Xiao, D. Xu, and W. Wan, "Overview: Video Recognition from Handcrafted Method to Deep Learning Method," in *International Conference on Audio, Language and Image Processing (ICALIP)*, pp. 646–651, IEEE, 2016.

[167] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A Deep Convolutional Activation Feature for Generic Visual Recognition," in *International Conference on Machine Learning*, pp. 647–655, 2014.

[168] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper with Convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, IEEE, 2015.

[169] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri, "Actions as Space-Time Shapes," in *IEEE International Conference on Computer Vision*, vol. 2, pp. 1395–1402, IEEE, 2005.

[170] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as Space-Time Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 12, pp. 2247–2253, 2007.

[171] K. K. Reddy and M. Shah, “Recognizing 50 Human Action Categories of Web Videos,” *Machine Vision and Applications*, vol. 24, no. 5, pp. 971–981, 2013.

[172] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “HMDB: A Large Video Database for Human Motion Recognition,” in *Proceedings of the IEEE Conference on Computer Vision*, pp. 2556–2563, IEEE, 2011.

[173] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, “Sequential Deep Learning for Human Action Recognition,” in *International Workshop on Human Behavior Understanding*, vol. 7065 of *Lecture Notes in Computer Science*, (Berlin, Heidelberg), pp. 29–39, Springer, 2011.

[174] K. Simonyan and A. Zisserman, “Two-Stream Convolutional Networks for Action Recognition in Videos,” in *Advances in Neural Information Processing Systems*, pp. 568–576, 2014.

[175] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[176] D. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[177] S. Ruder, “An Overview of Gradient Descent Optimization Algorithms,” *arXiv preprint arXiv:1609.04747*, 2016.

[178] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, *et al.*, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.” Software available from tensorflow.org, 2015.

[179] J. Vainstein, J. F. Manera, P. Negri, C. Delrieux, and A. Maguitman, “Modeling Video Activity with Dynamic Phrases and its Application to Action Recognition in Tennis Videos,” in *Iberoamerican Congress on Pattern Recognition*, pp. 909–916, Springer, 2014.

[180] H. Wang, D. Oneata, J. Verbeek, and C. Schmid, “A Robust and Efficient Video Representation for Action Recognition,” *International Journal of Computer Vision*, vol. 119, no. 3, pp. 219–238, 2016.

[181] S. Sharma, R. Kiros, and R. Salakhutdinov, “Action Recognition using Visual Attention,” *arXiv preprint arXiv:1511.04119*, 2015.

[182] N. Srivastava, E. Mansimov, and R. Salakhutdinov, “Unsupervised Learning of Video Representations using LSTMs.,” in *International Conference on Machine Learning*, pp. 843–852, 2015.

[183] K. Zienowicz, G. Hunter, and A. Shihab, “The Use of Spectrographic Template Matching to Identify and Classify Salient Sound Events in Tennis Matches,” *Proceedings of the Institute of Acoustics (UK)*, vol. 30, no. 2, pp. 171–179, 2008.

[184] Y. Du, C. Ma, C. Wu, X. Xu, Y. Guo, Y. Zhou, and J. Li, “A Visual Analytics Approach for Station-Based Air Quality Data,” *Sensors*, vol. 17, no. 1, p. 30, 2016.

[185] R. Faragher, “Understanding the Basis of the Kalman Filter via a Simple and Intuitive Derivation [lecture notes],” *IEEE Signal processing magazine*, vol. 29, no. 5, pp. 128–132, 2012.

[186] E. V. Cuevas, D. Zaldivar, and R. Rojas, “Kalman Filter for Vision Tracking.” Available at `http://www.diss.fu-berlin.de/docs/servlets/MCRFileNodeServlet/FUDOCS_derivate_000000000473/2005_12.pdf`, 2005.

[187] X. Yu, H. W. Leong, C. Xu, and Q. Tian, “Trajectory-Based Ball Detection and Tracking in Broadcast Soccer Video,” *IEEE Transactions on Multimedia*, vol. 8, no. 6, pp. 1164–1178, 2006.

[188] X. Yu, Q. Tian, and K. W. Wan, “A Novel Ball Detection Framework for Real Soccer Video,” in *Proceedings of the IEEE International Conference on Multimedia and Expo*, vol. 2, pp. 265–268, 2003.

[189] X. Yu, C. Xu, Q. Tian, and H. W. Leong, “A Ball Tracking Framework for Broadcast Soccer Video,” in *Proceedings of the IEEE International Conference on Multimedia and Expo*, vol. 2, pp. 273–276, 2003.

[190] X. Yu, C. Xu, H. W. Leong, Q. Tian, Q. Tang, and K. W. Wan, “Trajectory-Based Ball Detection and Tracking with Applications to Semantic Analysis of Broadcast Soccer Video,” *Proceedings of the 11th ACM international conference on Multimedia*, p. 11, 2003.

[191] X. Yu, N. Jiang, and A. E. Luang, “Trajectory-Based Ball Detection and Tracking in Broadcast Soccer Video with the Aid of Camera Motion Recovery,” *IEEE International Conference on Multimedia and Expo*, pp. 1543–1546, 2007.

[192] N. J. Gordon, D. J. Salmond, and A. F. Smith, “Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation,” *IEE Proceedings F (Radar and Signal Processing)*, vol. 140, no. 2, pp. 107–113, 1993.

[193] P. Del Moral, “Non-Linear Filtering: Interacting Particle Resolution,” *Markov Processes and Related Fields*, vol. 2, no. 4, pp. 555–581, 1996.