Investigation into tracking football players from single viewpoint video sequences

Michael Davis

Bachelor of Science in Computer Science with Honours
The University of Bath
April 2008
This dissertation may be made available for consultation within the University Library and may be photocopied or lent to other libraries for the purposes of consultation.

Signed:
Investigation into tracking football players from single viewpoint video sequences

Submitted by: Michael Davis

COPYRIGHT

Attention is drawn to the fact that copyright of this dissertation rests with its author. The Intellectual Property Rights of the products produced as part of the project belong to the University of Bath (see http://www.bath.ac.uk/ordinances/#intelprop).
This copy of the dissertation has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and that no quotation from the dissertation and no information derived from it may be published without the prior written consent of the author.

Declaration

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Bachelor of Science in the Department of Computer Science. No portion of the work in this dissertation has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

Signed:
Abstract

Analysis of sports footage is used extensively by broadcasters and sport scientists. This dissertation develops a system to track multiple players through a football scene. Football represents a complex domain for tracking applications. Challenges include, non linear motion, congested scenes and frequent occlusions. Standard broadcast footage, complete with moving cameras, on screen graphics and scene cuts is used. Multiple techniques are implemented and evaluated for each component of the system. In particular, player identification and tracking algorithms are investigated. An investigation into behaviour analysis derives useful information from tracking data.
Contents

1 Introduction 1
   1.1 Motivation for Analysis of Footage 1
   1.2 Problem Statement 2
      1.2.1 Primary Objective 2
      1.2.2 Secondary Objectives 3
   1.3 Measuring Success 4
   1.4 Project Overview 5

2 Literature Survey 7
   2.1 Introduction 7
   2.2 Pitch Segmentation 8
      2.2.1 The Football Domain 8
      2.2.2 Colour Models 10
      2.2.3 Histogram Segmentation 11
      2.2.4 Background Subtraction 12
      2.2.5 Probabilistic Classifiers 13
      2.2.6 Thresholding 15
      2.2.7 Connected Components 16
   2.3 Object Recognition 17
      2.3.1 Line Identification 18
CONTENTS

3.7 Test Plan ................................................. 44

4 Background Segmentation and Object Recognition .......... 47
  4.1 Background Segmentation ........................................ 47
    4.1.1 Background Subtraction .................................. 47
    4.1.2 Statistical Pitch Models .................................. 49
    4.1.3 Thresholding .............................................. 55
    4.1.4 Morphological Operators .................................. 59
  4.2 Object Classification ......................................... 60
    4.2.1 Field Identification ....................................... 60
    4.2.2 Line Identification ....................................... 61
    4.2.3 Player Classification ...................................... 62
    4.2.4 Team Classification ....................................... 69

5 Tracking, Camera Models and Behaviour Analysis ............ 72
  5.1 Camera Motion .............................................. 72
  5.2 Nearest Neighbour .......................................... 76
    5.2.1 Evaluation and Discussion ................................ 76
  5.3 Kalman Filters .............................................. 78
    5.3.1 Evaluation and Discussion ................................ 82
  5.4 Condensation ............................................... 83
    5.4.1 Algorithm ............................................... 84
    5.4.2 State Transformations .................................... 85
    5.4.3 Multiple Player Tracking ................................ 87
    5.4.4 Fitness Function .......................................... 88
    5.4.5 Initialising and Terminating Tracks ....................... 90
    5.4.6 Evaluation and Discussion ................................ 91
  5.5 Behaviour Analysis ........................................... 92
### CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5.1 Attacking Team</td>
<td>92</td>
</tr>
<tr>
<td>5.5.2 Unmarked Players</td>
<td>94</td>
</tr>
<tr>
<td>5.6 Scene Detection</td>
<td>95</td>
</tr>
<tr>
<td>6 Results</td>
<td>97</td>
</tr>
<tr>
<td>6.1 Test Plan</td>
<td>97</td>
</tr>
<tr>
<td>6.2 Full System Test Results</td>
<td>98</td>
</tr>
<tr>
<td>6.3 Camera Movement</td>
<td>101</td>
</tr>
<tr>
<td>6.4 Congestion</td>
<td>102</td>
</tr>
<tr>
<td>6.5 Camera Angle</td>
<td>104</td>
</tr>
<tr>
<td>6.6 Lighting / Weather Conditions</td>
<td>108</td>
</tr>
<tr>
<td>6.7 Team Colours</td>
<td>110</td>
</tr>
<tr>
<td>6.8 Additional Applications</td>
<td>112</td>
</tr>
<tr>
<td>6.8.1 Tracking Rugby Players</td>
<td>112</td>
</tr>
<tr>
<td>6.8.2 Tracking Pedestrians</td>
<td>114</td>
</tr>
<tr>
<td>7 Conclusions &amp; Future Work</td>
<td>118</td>
</tr>
<tr>
<td>7.1 Evaluation of results</td>
<td>118</td>
</tr>
<tr>
<td>7.1.1 Condensation</td>
<td>119</td>
</tr>
<tr>
<td>7.1.2 Track Visualisation</td>
<td>121</td>
</tr>
<tr>
<td>7.1.3 Player Initialisation</td>
<td>121</td>
</tr>
<tr>
<td>7.1.4 Player Identification</td>
<td>123</td>
</tr>
<tr>
<td>7.1.5 Team Classifications</td>
<td>124</td>
</tr>
<tr>
<td>7.2 Delivered System vs. Objectives</td>
<td>125</td>
</tr>
<tr>
<td>7.2.1 Primary Objective</td>
<td>125</td>
</tr>
<tr>
<td>7.2.2 Secondary Objectives</td>
<td>125</td>
</tr>
<tr>
<td>7.3 Extensions to Functionality</td>
<td>130</td>
</tr>
<tr>
<td>7.3.1 Multiple Pass Tracking</td>
<td>130</td>
</tr>
</tbody>
</table>
CONTENTS

7.3.2 Terminating Bad Tracks ............................................. 130
7.3.3 Camera to Pitch Plane Transformations ......................... 131

A Project Plan .................................................................... 136
Acknowledgements

I would like to express my most sincere gratitude to my supervisor Dr John Collomosse for all his help, support and guidance throughout the project. This project would not have been possible without his ideas and feedback. Many thanks to my parents for their support throughout and without whom this document could not have been printed. I would also like to thank everyone who helped in my seemingly eternal quest to source suitable sports footage.
Chapter 1

Introduction

Accurate and reliable tracking of moving objects through a video sequence is an active, open area of computer vision research. In recent years a vast amount of literature has been published exploring many different methods for achieving the most accurate and reliable tracking. From this research, tracking systems, such as [32][27], have been successfully deployed to track objects such as vehicles and people in real world scenarios.

This project investigates tracking multiple players through a football scene. Various tracking algorithms are implemented and evaluated to produce accurate and reliable tracking of players through the video sequence. Information about players’ movements and interactions has a range of applications in sport science and sports broadcasting (Section 1.1).

There are many interesting aspects of the football domain that make player tracking a complex and challenging area of computer vision research. Such a system must be capable of tracking multiple objects (up to 23) through a large, congested, rapidly changing scene where movements are likely to be non linear and there are frequent occlusions.

1.1 Motivation for Analysis of Footage

A more scientific approach to sport has been possible in recent years due to developments in the field of sport science. Players, managers and coaches are using developments in sports technology to obtain novel, previously immeasurable, metrics, which can be analysed and used to improve athletes’ performance.

In football, the important metrics are how players move and how they move with respect to
the movements of the opposing team. From this data, scientists and managers can obtain useful metrics such as players’ speed, agility, strategy, positioning and fitness. These are used to identify areas to be developed and training is tailored accordingly. For example, this data could identify players who need to work on their fitness by comparing their speed at the start and end of a match or highlight players who were continually out of position. This has led to a demand for the ability to reliably and accurately track players’ positions and movement on the football field.

Some commercial systems, such as Prozone [25], can provide this data. These are expensive to setup and run as they require a great deal of hardware and manual marking of player positions. This means the information is only available to top class professional teams. The application of similar scientific methods and practises can have the same beneficial effect on performance for smaller, amateur clubs and pupils learning the game in schools. A tracking system capable of collecting performance data for smaller teams requires the production of a simpler, more cost effective system.

Football is enjoyed by both sport participators and spectators, with large numbers of people watching football at matches and on television. Precise knowledge of players’ positions and movements have been used by the BBC and other football broadcasters to augment video footage with addition information, provide statistics on the match and even show clips from synthesised viewpoints [4]. This enhances the viewers experience by providing more information about the game than can be obtained by viewing the flat footage. Researchers have also used this information to produce 3D animations of scenes [5]. Such animations could be sent over the Internet and to phones with lower bandwidth and computational requirements than encoded video.

1.2 Problem Statement

1.2.1 Primary Objective

The principle objective of this project is to build a system capable of accurately and reliably tracking multiple players through a football scene from a monocular viewpoint video sequence. The system should take a standard video sequence of football as input and output the location of each player in every frame through the sequence.
1.2.2 Secondary Objectives

- **Quantifiable Metrics.** To achieve the primary goal of accurately and reliably tracking these notions must be formalised into a precise and quantifiable metrics. Results can measured against these scales to demonstrate a tracker is indeed reliable and accurate.

- **Evaluating different approaches.** Another objective throughout the project is to find the ‘best’ methods by investigating and evaluating a number of the methods highlighted in Section 2 and using the quantifiable metrics to draw direct comparisons and conclusions about relative benefits.

- **No user input.** The primary objective specifies only a video sequence as required input. Many of the systems and techniques documented in Section 2 require user input in building models off line or supplying large training data sets. This system aims to minimise user input by developing self calibrating models or automatically bootstrapping models from the image sequence data.

- **Classifying tracked players into teams.** To satisfy this objective the system must output a team classification for each player in addition to the track. This information may also be used to improve the performance of the tracking.

- **A camera model** to compensate for camera movements through the image sequence. This can be used to improve the performance of the tracking. It may also be used to transform the track into the current camera plane for plotting the players path with respect to the current scene.

- **Generalising the system to work for other sports and tracking applications.** This project is an investigation into computer vision methods for tracking, applied to the football domain. Although football specific information may be used to improve the performance of the tracker, generalising to other tracking applications demonstrates the methods are not specific to the football domain.

- **Scene classification** and automatic detection of scene changes to index and annotate footage. The primary objective considers input from a single scene, using a long range, zoomed out viewpoint. This extension would allow the tracker to be applied to extended sequences of play, such as a whole match. Scene detection could be used to reinitialise tracking when there is a cut and scene classification could identify only long range shots suitable for tracking.
CHAPTER 1. INTRODUCTION

- **Behaviour Analysis** to determine some basic information about players movements and tactics from the raw player track data. This information can help managers and coaches analyse performance and identify strengths and weaknesses in their teams. Such analysis can also provide interesting statistics about the game to spectators.

- **Tracking the ball.** This is more complex than tracking the players as the ball is smaller and travelling faster. Outputting the path of the ball would be very useful if the information were to be fed into match analysis software as it would enable metrics such as time on ball and number of passes to be computed.

- **Identifying individual players.** Player identification would enable the system to relate tracks in different scenes to the same player. Player profiles could be computed based on tracking information throughout the match.

Many solutions use multiple cameras to better triangulate players positions and resolve occlusions [34] or fixed cameras to provide a constant projection from the camera plane to the ground plane [23]. This system concentrates on tracking multiple players from a monocular viewpoint. Although multiple cameras can yield better accuracy, various tracking algorithms and occlusion resolution techniques are explored to produce comparable results using a monocular viewpoint. Any improvements to the one camera model can be applied directly to multiple camera systems. A simple system, using a single camera, could also have applications in the amateur game, allowing more teams to benefit from the types of performance metrics available to big clubs.

A full set of requirements for the system are defined in Section 3.4.

1.3 Measuring Success

Different tracking algorithms are known to have particular strengths and weaknesses. Some tracking algorithms perform well when objects move with a predictable first order motion model and others perform well in cases of occlusion. To determine which tracking algorithm performs best in the football domain it is necessary to draw direct comparisons between methods. To do this, quantifiable metrics are defined to measure accuracy and reliability of a tracker.

One such method, for evaluating tracking algorithms, measures the deviation of an algorithms’ output from a manually marked ground truth. The tracker which achieves the smallest deviations will be the best at accurately tracking the players. This can be run for
several image sequences (with differing independent variables such as light levels, players in scene and amount of occlusions). An algorithm that can maintain accurate tracking in varying conditions will be reliable.

1.4 Project Overview

This section provides an overview of the dissertation, summarising the content and contribution of each chapter toward the principle goal of player tracking.

Chapter 1 – Introduction
This chapter outlines the aims and motivations of the project. Describing the challenges of tracking from video sequences and outlining applications for tracking football players.

Chapter 2 – Literature Survey
This provides a comprehensive survey of relevant literature. It identifies existing methods used for football player tracking and methods that could be adapted to the football domain. Trends, gaps and challenges documented in the literature are identified and related to the project.

Chapter 3 – System Specification
This chapter defines a more formal set of requirements for the system and details a high level design based on systems identified in Chapter 2.

Chapter 4 – Background segmentation and Object Recognition
This chapter investigates segmenting pitch and crowd regions from each frame and identifying line and player objects from the remaining foreground regions. It also details methods for team classification on the player objects.

Chapter 5 – Tracking
This chapter tracks the player objects from Chapter 4 through the image sequence. Nearest neighbour, kalman filters and condensation trackers are implemented and evaluated. A method for modelling camera movement between frames is developed and integrated into the tracking algorithms. Investigations into scene detection and behaviour analysis are conducted.

Chapter 6 – Results
In this chapter, the intermediate results and experiments from Chapters 4 & 5 are combined with the tests described in Section 3.7. The success of all the methods investigated are compared and evaluated to determine the best method for tracking players.
Chapter 7 – Conclusions and Future Work
Detailed discussion of the results, drawing conclusions about the success of each method. Areas for improvements and extensions for future work are identified.
Chapter 2

Literature Survey

2.1 Introduction

This chapter identifies areas of research relevant to multiple object tracking from video sequences. It evaluates methods and algorithms previously applied to the sports tracking domain, general computer vision methods and novel techniques from similar problem domains that could be adapted to the football tracking problem. The algorithms implemented in later chapters are based on the ideas and insights gained from this survey.

Consideration of the major challenges in the football tracking domain are identified and different solutions are evaluated. The major challenges, identified in the literature, are:

- Identifying players in occlusions, Sections 2.3.3 & 2.4
- Maintaining accurate tracking on an image sequence from a moving camera, Section 2.5
- Player recognition, classification into teams and identification, Section 2.3.3
- Designing a Feature Space to accurately identify the paths of each player, Section 2.4.1

Computer vision systems are, in general, modular. This chapter looks at each stage of the tracking process:

- Image processing and segmentation to remove noise and pitch regions of the scene. Section 2.2
• Fitting shape models to the extracted foreground region to identify pitch markings and players. Section 2.3

• Filtering and tracking methods to accurately and reliably track the multiple player objects through the image sequence. Section 2.4

This chapter also considers types of footage used in football broadcasts. In particular, how a non static camera will affect tracking and methods for automatic scene detection and classification (Section 2.5).

2.2 Pitch Segmentation

Applying low level filtering to the raw image sequence can filter noise and parts of the image not relevant to the tracking process, reducing the operational space for classification and tracking. Removing the pitch will identify player regions to track. Segmentation is a very common approach to tracking applications, used on numerous previous occasions, for example [34] [10]. There are many well researched and evaluated methods for achieving good background segmentations. The choice of a best method is dependent on the characteristics of the scene and the trade off between quality of results and the complexity of the algorithm.

2.2.1 The Football Domain

There are many aspects of the football domain which are relevant to the pitch segmentation task. Football can be considered a large, cluttered, rapidly changing scene where movements are likely to be non linear and there are frequent occlusions. Football games take place in any weather conditions and many different lighting conditions, including high intensity flood lighting. The 90 minute duration allows lighting and weather conditions to vary through the game. Large stadiums can cast shadows across regions of the pitch such that lighting conditions may not be constant in a given frame, as shown in Figure 2.1. Dearden et al [8] note, footage from cameras situated in the stands will be subject to an amount of camera jitter due to the movements of spectators.

There are some useful features of the football domain which are relevant to the segmentation task. Football is popular as a spectator sport, care is taken to ensure players kits are distinguishable from the pitch and the kit of the opposing team. Grass is always the most
Figure 2.1: A standard football ground. The shadow of the stand and the striped pitch mean the pitch consists of several shades of green.

dominant colour in the scene and this colour does not appear in non grass regions of the scene.

A discontinuity in existing football player tracking systems is whether pitch markings (such as the white lines) should be filtered as pitch or preserved in the filtered image. Some applications define the background as; areas which do not move or move independently of the salient objects. In this definition, pitch markings will be filtered as background. Other systems make use of pitch markings [20], using lines as fixed reference points. These can be used for modelling camera movement and computing camera to pitch plane transformations.

All the methods presented below represent a bottom up approach to segmentation. Each pixel is classified by measuring it against a background model. The following section considers different background models. There are two general considerations applicable to the following models:

Model duration

It is possible to construct one background model and use this throughout the sequence. Section 2.2.1 describes how conditions may vary over the duration of the game. If conditions vary significantly from those in which the model was trained it could lead to misclassification. Alternatively a background model could be computed on a per frame basis. This would be more reactive to changes in conditions however this adds extra complexity through building additional models. Some models are adaptive, it may be possible to update the model without rebuilding it. This allows the model to evolve over time, adapting to dif-
ferent conditions. Stauffer and Grimson [27] note “most researchers have abandoned non-adaptive methods of backgrounding because of the need for manual initialisation. Without re-initialisation, errors in the background accumulate over time, making this method useful only in highly-supervised, short-term tracking applications without significant changes in the scene”.

**Grass vs Pixel Model**

It is possible to build a spatially independent pitch model, representing a grass pixel. Each pixel is measured against the pitch model to produce a segmentation. Alternatively it is possible to build a model for each pixels’ distribution over time and measure each pixel against it’s distribution model. Many scenarios, such as a street scene, have noisy backgrounds. Here a per pixel model is essential as a pixel in background state may correspond to either black tarmac or a white road marking. A per pixel model has greater complexity in building extra models and is more susceptible to camera jitter as a slight movement would cause background pixels to be misaligned with their models. In the soccer domain the background is predominantly pitch and thus the distribution models will be very similar for the majority of the pixels.

Ideally our background classification should be given as a confidence value to emphasise features rather than provide a binary mask. This will allow the higher level, “more informed” processes to determine if a pixel is background, based on the segmentation’s confidence figure and additional probabilistic information it may have.

It is important that the background segmentation process extracts precise non pitch regions. The shape of these regions is exploited in player classification.

### 2.2.2 Colour Models

The background region is represented by a consistent, distinct colour. This motivates the use colour information to segment grass pixels. The choice of colour representation is important for accurate segmentation. Most video is represented as a Red x Green x Blue (RGB) triple. HSV is generally more appropriate in vision applications as perceptually similar colours are ‘closer’ in this colour space and intensity is de-coupled from colour. This makes it less sensitive to illumination effects.

Many other colour models are explored in the literature. A CIE lab colour space attempts to create a *perceptually uniform* space, such that a fixed change in any value is of equal visual importance. It is designed to be easily computable from a standard RGB or HSV
Another possible improvement is proposed by Vanderbroucke et al [30]. They demonstrate a novel approach to improve the results of colour image segmentation. An adapted hybrid colour space is created, with neither psycho-visual nor physical colour significance but such that colour difference between the pixel classes is best discriminated. To build this colour space it is necessary to construct learning samples, representative of the background and non background pixels. Their results demonstrate an improvement on segmentation using RGB and HSV when applied to the football domain [30]. This approach would introduce extra costs in complexity of computation and implementation. It is an option if results using a simple colour model prove insufficient.

2.2.3 Histogram Segmentation

The simplest method commonly used to model the pitch region is a Histogram. This involves quantising the colour space into a finite number of discrete bins. Constructing a histogram calculates the frequency of each colour bin in the image. Section 2.2.1 notes the image is predominantly pitch and the pitch is a consistent green. This will create a clear density peak at this green colour, as seen in Figure 2.2. A pitch model can be defined as any pixel within a variance of the density peak. A binary pitch mask can be obtained by iterating through each pixel and setting any pixel within the tolerance value of the peak as pitch. This method is applied to many football tracking works, for example [26], [35], [8].

The first consideration with this method is bin size. Too many bins will spread the emphasised peak such that variance and noise will create many smaller, less distinct peaks.
Too few bins and the density peak will be smoothed away. This model assumes the pitch consists of a single predominant colour, within the range of one or two buckets. Football pitches can be cut such that the pitch has two different shades of green and shadows from players and the stands can introduce further shades (Figure 2.1). Figure 2.2 clearly demonstrates a bimodal distribution. In this case, the hsv colour space reduced the difference between shades of grass but the shaded and un-shaded regions still cause two distinct peaks. Both peaks are close enough to fall within the pitch tolerance in this example but a larger contrast between these shades of green could cause only a section of the pitch to be segmented.

2.2.4 Background Subtraction

Background subtraction is a simple image segmentation technique commonly used in vision applications, for example [18] [22]. This utilises the video sequence, comparing each frame to its predecessors to build and maintain a temporal, time averaged background. Moving features can be emphasised by subtracting the current frame from the background. This provides a fast, efficient segmentation of moving objects from a stationary background. A time averaged background value \( B \) for a pixel \((x, y)\) at time \(t\) can be described as:

\[
B(x, y, t) = \frac{1}{t} \sum_{i=1}^{t} I(x, y, i) \tag{2.1}
\]

This can also be computed incrementally to build the desired adaptive model:

\[
B(x, y, t) = \frac{(t - 1)}{t} B(x, y, t - 1) + \frac{1}{t} I(x, y, i) \tag{2.2}
\]

This should allow the model to adapt to background changes over time. When conditions change, this model will average the background over the sum of both conditions. A better model implements ‘forgetting’, using a moving window average or exponential forgetting. The weighting of each frame going back in time reduces exponentially. This can be expressed by the equation:

\[
B(x, y, t) = (1 - \alpha)B(x, y, t - 1) + \alpha I(x, y, i) \tag{2.3}
\]

Where \(0 < \alpha < 1\). This is the general equation of a standard moving average FIR filter. Once a background model is constructed for a pixel \((x, y)\) it can be classified at time \(t\) as:
CHAPTER 2. LITERATURE SURVEY

\[ \text{bitMask}(x, y, t) = \begin{cases} 
1 & \text{if } |I(x,y,t) - B(x,y,t-1)| \leq B_t \\
0 & \text{otherwise} 
\end{cases} \]  \tag{2.4}

Where \( B_t \) is the threshold for pixel variance in the background (Section 2.2.6).

One serious problem with this method arises when trying to identify slow moving or stationary objects, such as the goal keeper, who can remain stationary for significant periods of time. Using the background model above would cause slow moving objects to get integrated into the background. An improvement to this model, demonstrated in [18], is to classify the pixels as moving or non moving first, then update the background with only those pixels marked as background. Friedman and Russel [10] report this method to be ‘reasonably successful’ but it still has problems with very slow and stationary objects being classified as not moving and getting integrated into the background.

Camera jitter and moving cameras are also a problem for this method. Pixels get misaligned with their background models, causing inaccuracies. This method is not suitable for extracting pitch markings as only moving objects are segmented. If this method is not suitable for accurately segmenting the players it may still be useful for identifying regions of grass, necessary when bootstrapping any grass model.

2.2.5 Probabilistic Classifiers

A more refined approach to background subtraction can use probabilistic information, on the distribution of the pixel values, to form a more robust, statistical background model. A pixel’s value is a colour triple. It has been established this is a good property for distinguishing grass pixels (Section 2.2.1). Other tracking applications, such as pfinder [32], have used multi-scale models, including shape and spatial properties in addition to colour.

In most tracking applications, for example [10], [32], a model is constructed for each pixel. This is necessary in a scene with a noisy background. Each pixel could have its own unique distribution, dependent on its spatial location in the scene. As the football background is grass, background pixel distributions across the scene will be consistent (within a small variance). This motivates using a single model to represent all grass pixels. This is common in football applications, for example [34]. A single pitch model will have lower complexity and will be more robust to camera movements as pixels are measured against a spatially independent model.
A simple model can be obtained by fitting an Eigenmodel to the pitch pixel distribution in the chosen colour space. We can determine a likelihood of a pixel being background by calculating its Mahalanobis distance (Section 4.1.2) from the Eigenmodel. A lower Mahalanobis distance is more likely to be background. Eigenmodels assume the distribution forms a unimodal, Gaussian distribution in the feature space.

Figure 2.2 demonstrates how shadows can cause a multi-modal distribution of pixel values. Using an Eigenmodel to represent this distribution could cause misclassification. To model such multi-modal distributions a Gaussian Mixture Model is required. This uses a number of Gaussian components to accurately describe the distribution. Most implementations use an Expectation Maximisation algorithm to decide the number of components and their locations. An incremental version of EM, suitable for real time applications is presented here [10]. Eigenmodels, GMMs and methods for classifying pixels using these models are considered in more detail in Section 4.1.2. Although it is possible to add additional Gaussians to a mixture model, as the system should be re-initialised without user intervention a simpler approach will rebuild the model if there are substantial changes in background conditions.

Although Gaussian mixtures add complexity to building the model and classifying the pixels most researchers, for example [23], use a Gaussian mixture model rather than a single Gaussian as the model produces a better representation of non Gaussian distributions. A common problem associated with probabilistic, density-based modelling, as noted by [26], is how many Gaussians to use, known as the model order selection problem. Too few will over generalise the data, causing non background pixels to be incorporated into the background model and too many Gaussians can over-fit the model such that the benefits of the probabilistic approach are lost. This choice is further complicated by the computational cost of additional Gaussian components. Russell et al [10] use three Gaussians in their vehicle tracker, corresponding to three well defined states, road, shadow and vehicle. Stauffer and Grimson [27] use 3-5 Gaussians, based on the memory and computational power available.

This method can filter pitch markings by including them in the background distribution sample or not filter markings by excluding them from the pitch sample. This method also has no requirement that the pitch must consist of a single predominant colour. The hypothesis only requires players to be suitably distinct from the background in the chosen colour space. This will allow the system to be used in a wider range of tracking applications (Section 1.2 Extension 3).
2.2.6 Thresholding

All the background subtraction methods require some threshold value as a boundary between pitch and non pitch classification. This parameter is key to an accurate segmentation. The simplest threshold is a constant, set by the user on start-up. This has the lowest complexity but does not allow the threshold to adapt to changing conditions and removes a level of automation from the system.

When using the histogram method (Section 2.2.3), the valley points either side of the peak could be used as the threshold. This can be updated dynamically to compensate for varying conditions. Care must be taken to compensate for bimodal distributions, as in Figure 2.2. Otsu [24] demonstrate a thresholding technique based on a multi-modal histogram.

Another common method uses a one dimensional variant of the \textit{k-means clustering algorithm} to generate a threshold value. Some initial threshold \(T\) is chosen (randomly or by some other method), the image is segmented using this threshold and a new threshold \(T'\) is calculated where:

\[
T' = \frac{\text{mean(Foreground)} + \text{mean(Background)}}{2}
\]  

(2.5)

The algorithm is repeated setting \(T = T'\) until \(T\) and \(T'\) converge. This will produce a better threshold for accurately segmenting the player objects. Re-running the algorithm per frame (or on key frames) will create a dynamic threshold. This method adds extra complexity to the system as multiple segmentations are computed per frame.

Changes in lighting across the scene, caused by shadows cast by the stands (Figure 2.1) could cause the optimal threshold value to vary through the image. In this case better segmentation can be achieved using a different threshold value for each pixel, based only on variances in the local environment. A local threshold technique, such as a \textit{dynamic sliding window}, defines a local environment as a window around each pixel. A threshold is computed for each pixel based on the variation within it’s local environment. This is more reactive to local variances and not affected by larger variances elsewhere in the image. There is still a choice of method for selecting a threshold value within the window. Common approaches select a threshold based on variation from the mean or take the difference between the minimum and maximum values in the window. More advanced thresholding could use one of the techniques above within the local window. Local thresholding increases the complexity by computing extra thresholds but can have notable improvements on the accuracy of the segmentation if there is variation in the optimal threshold value through
The success of segmentation and thresholding techniques can be evaluated by counting the percentage of pixels incorrectly classified as background and the number of pixels incorrectly classified as foreground. The increase in accuracy from using a local threshold can be measured by comparing the number of incorrect classifications with a global threshold method.

Should the above segmentation techniques prove insufficient, extensions could experiment with methods that fuse several complementary segmentation techniques, as demonstrated by Nixon et al [1]. If smaller objects, such as the lines, get overlooked as they do not differ from the grass by as much as the player objects, Utsumi et al [29] demonstrate a method of combining the result from a colour filter with a local edge property, using a fuzzy function.

2.2.7 Connected Components

Pitch segmentation aims to remove the pitch pixels, leaving player, line and ball regions. Following the classification, foreground (non-pitch) pixels must be collected together to form labelled regions or connected components. These objects are carried forward to the next phase to be classified player, line or background.

The pitch classification methods discussed above are all examples of low level image processing techniques, performing classifications at a pixel level. Collecting the classified pixels together to form higher level connected components is a bottom up approach to classification. The pitch classifier makes no use of global scene information such as ‘non pitch
classifications appear together in player regions’. Each pixel is classified separately with no
collection of spacial properties or surrounding pixels. In noisy scenes this may result
in player regions getting fragmented, not forming the desired connected component. Figure 2.3(d) demonstrates players fragmented following thresholding. Further processing is
required to generate the desired connected components.

Needham [23] uses a probabilistic relaxation to connect fragmented regions. Given a fore-
ground probability $p(\text{fore})$, from the segmentation process, $p(\text{fore})$ is iteratively updated
by:
\[
p(\text{fore}) = p(\text{fore}) + \delta \quad \text{if median value of neighbouring pixels} > 0.5
\]
\[
p(\text{fore}) = p(\text{fore}) - \delta \quad \text{otherwise}
\]
(2.6)
He reports applying the process 3 times with $\delta=0.2$ produced well segmented foreground
regions.

Dearden et al [8] and Bebie [5] use the morphological operators erosion and dilation for
reducing fragmentation. These create a new value for each pixel based on the values of
pixels in a surrounding window. Figure 2.3(a-c) demonstrates an erosion and dilation op-
erator applied to a simple image. This is similar to the probabilistic relaxation method.
Here the shape and weightings of the window are defined by a structuring element. Using
dilation with the appropriate structuring element can join connected components, remov-
ing fragmentation. Figure 2.3(e) demonstrates how fragmentation can be resolved using
dilation followed by erosion. Erosion can fragment connected components and remove small
connected components. This can be used to filter individual misclassified pitch pixels and
break minor occlusions. As these operators are not the inverses of each other, a combina-
tion of dilation and erosion can be used to break small occlusions, filter small misclassified
grass regions and resolve fragmented player regions. Dearden et al [8] warn erosion can
potentially remove useful player information, in particular, when player regions are small
due to a highly zoom camera.

2.3 Object Recognition

Background segmentation only enhances features of the image, it does not unambiguously
identify objects within it. Figure 2.4 demonstrates a typical result from an image segmen-
tation process. The pitch region, player objects and line information must be identified
before the tracking algorithm can be applied.
2.3.1 Line Identification

Many football tracking applications make use of line information in a scene. Pitch markings represent features in a relatively feature sparse scene. These can be used to compute camera motion models and create camera to the pitch plane projections. A classical Hough transform technique \cite{13} is the most common method for identifying lines in an image, for example \cite{17} \cite{9}.

The Hough transform locates imperfect instances of geometric forms within an image. It operates on a binary image, such as the foreground mask. To locate straight lines they must be considered in their polar representation:

\[ r = x \cos(\theta) + y \sin(\theta) \] (2.7)

This has 2 free parameters \( r \) and \( \theta \). This two dimensional parameter space is known as the Hough space. Each point in the Hough space uniquely maps to a line in the image space. In a discrete parameter space this represents a finite number of candidate lines in the image space. For any point \( (x, y) \) in the image space there will be a finite number of candidate lines that pass through that point. The first step of the algorithm assigns a scalar weight to every location in the Hough space, known as an ‘accumulator array’. The algorithm iterates through each ‘line pixel’ in the binary image, casting a vote in the accumulator of every candidate line that passes through this pixel. The evidence for a line in the scene is the number of votes in the corresponding accumulator. The second step of the algorithm searches for maxima in the accumulator array, corresponding to the lines in the image.

Lines are likely to be discontinuous due minor inaccuracies in the image segmentation and obstructions by players. As the Hough transform is a global method it examines lines over the whole image, disregarding noise and discontinuities that may distort a bottom up line building approach. Global analysis also finds all lines in a single pass. Although noise and artefacts tend to be disregarded, removing player regions will reduce the complexity by
reducing the number of pixels to iterate through. Reducing noise in the Hough space also emphasises the lines. If the segmented image is not sufficient for detecting the lines, low level filtering techniques such as thinning and edge detection can be applied to enhance the line detail.

2.3.2 Pitch Identification

By identifying the pitch region of the scene non pitch areas, such as advertising boards and the crowd, can be removed. Although these parts of the image do contain useful information, this is not relevant to the primary goal of tracking. This is simpler and more accurate than segmenting these areas using methods similar to the pitch extraction. Unlike the pitch, these areas are moving, noisy and contain much variance. A common method for pitch identification is to group the pitch regions of the image and calculate the convex hull. This is sufficient as a football pitch is convex (from all camera angles) and there are algorithms to compute convex hulls with complexity $O(n \log n)$. This is used in several applications for example [8].

If lines have been identified in the image they could be used to segment the pitch region. Naidoo et al [22] demonstrate a method for identifying the field using pitch markings.
CHAPTER 2. LITERATURE SURVEY

2.3.3 Player Identification

A simple method commonly used for identifying player objects looks for interconnected regions that meet some basic heuristics. A simple heuristic used by Utsumi et al [29] asserts: A player is any region where:

\[ 15 \leq h \leq 50 \]  
\[ 6 \leq w \leq 45 \]  
\[ w \leq h \]

where \( h \) and \( w \) represent the height and width of a region’s bounding box respectively. Yoon et al [35] use a set of heuristics based on size, shape and colour of the connected regions.

The major difficulty in identifying players is occlusions, when two or more player regions
become connected. A connected component approach will classify all occluded players as a single player. A common approach to this problem is to use multiple cameras and hope that there is an un-occluded view from another camera. Figure 2.6(a) demonstrates the worst case of occlusion, one player is totally occluded behind another. From this viewpoint it is impossible to disambiguate these players. In partial occlusions, such as Figure 2.6(c), a simple approach is to use morphological operators to erode close regions in the hope they break apart [8]. In worse cases of occlusion the tracking process must resolve the occlusion, maintaining tracking without the player region evidence then re-associating with the correct player region once the occlusion is resolved. It is important that occluded regions (regions containing more than one player) are still classified as player regions to provide the tracker with maximal information for occlusion resolution.

The success of player classification can be evaluated using precision recall statistics. Recall refers to the number of players classified correctly and precision is related to the number of non player regions misclassified as players. Precise definitions are discussed in Section 4.2.3. The best method will be a trade-off between precision and recall. The best trade off will be dependant on the tracking algorithm. Some tracking algorithms may be robust against missing players or particularly susceptible to noise from misclassifications.

Shape Properties

A more advanced player shape model would allow active identification and assertions that an object is a player, rather than relying on heuristics. A common method for classifying objects after segmentation is to use global shape properties and image moments. These are weighted averages (moments) of the connected components’ pixels that describe particular properties of the region. Basic properties include the area, orientation and major/minor axes. More complex properties are defined as functions of these moments describing properties such as roundness and eccentricity. A combination of these shape properties can be used to define a feature space in which to model player shape. A feature space can be
constructed from various shape properties and a probabilistic model, such as a *Gaussian Mixture Model*, constructed from a training set of player regions. Player classification can be achieved by comparing shape properties of each connected component to the model. Figure 2.7 demonstrates the range of shapes a player can form during a sequence of play and Figure 2.6(b) & (c) demonstrate how occlusion can vary the shape of a connected component. If this method is to produce accurate player classification, the player model must be robust to these variations in shape. Section 4.2.3 provides further discussion on shape models.

Another method for identifying features by shape is the *Generalised Hough Transform*. This is similar to the Hough Transform used for identifying lines (Section 2.3.1). The parametric equation is replaced by a look-up table defining the relationship (angle and distance) between some reference point (on the shape) and the shapes’ boundary points. As with shape property methods, this method will need to be robust to the variation in player shapes. This method will also be susceptible to scale and rotation, particularly during occlusions. Adding parameters for scale and rotation will significantly increase the complexity of this method.

**Active Contours**

Yoon et al [35] and Koichi et al [29] use a nearest Nearest Neighbour tracking system (Section 2.4.2). A player object is represented by a simple centroid co-ordinate and a bounding box. More advanced trackers trace points through a high dimensional feature space. A more advanced, higher dimensional, representation of a player that includes shape and colour information can improve the accuracy of the tracker, producing better predictions of players states’ and better occlusion resolution.

Contour models are a common method for identifying objects’ shapes in a segmented image. A flexible contour model such as a *Point Distribution Model* (PDM) can represent the mean geometry and variation of a shape from a number of training images. A set of landmark points are used to identify common points through the training set. To use a PDM, objects must be normalised, aligning scale, rotation and translation and there must be a mechanism to identify landmark points. A new connected component is classified by correcting scale and rotation, identifying landmark points and measuring the deviation from the mean models’ geometry. *Principal Component Analysis* (PCA) can be applied to identify the principle modes of variation in the shape model. The model can be projected into just these dimensions, significantly reducing the dimensionality.
CHAPTER 2. LITERATURE SURVEY

Active Shape Models (ASM) can be used to model player shapes from the PDM. If this method is adopted, the principle model components can be included in the player state feature space. This increases the data available to the tracker for occlusion reasoning. PDM’s and other active contour models have been applied to various tracking problems such as pedestrian tracking [3] however (as noted by Needham [23]) there is far greater variation in the possible shapes of a football player (Figure 2.7). This will make it hard to find a set of landmark points that accurately represent a player without also incorrectly representing a number of implausible shapes. Active Appearance Models are one solution to the high variation in shape. AAMs match an image to a model using information within the image region.

Needham [23] experiments with contour based shape models, employing a B-Spline model [3]. He observes this model ‘struggled’ to find the players arms and legs and contributed only a ‘small (and rather disappointing) improvement’. He progresses to use a template matching approach. A set of templates are created from a set of player regions by taking a feature vector of all the pixels within the normalised player region. The template feature vectors are clustered into an optimum set of templates, using PCA and k-means algorithms, to form a player model. A new object is evaluated against each template and the closest match is used. His experiments demonstrate this is a more robust model in example scenes.

2.3.4 Team Classification

Player classification into teams is a secondary objective of the system. Several systems have a mechanism for this. Players kits are designed such that players and spectators can distinguish between opposing teams. We can exploit this to classify players teams. There will be a maximum of 5 different kits on the pitch, 2 team kits, 2 goalkeeper kits and the referee’s kit. A simple, commonly used approach uses a voting system [22]. This method learns the 5 kit colours before the game. For each player region it iterates through each pixel. Any pixel within a threshold of a kit colour gets a vote for that kit. A player is classified as the kit with the highest number of votes. The number of votes (and the ratio between the classes) provide a confidence for the classification. This method requires user input in initialising colours and careful choice of threshold values, especially when teams have similar kits.

Another common algorithm uses histograms to represent the five colours [34]. Unlike the grass histogram used in image segmentation (Section 2.2.3), multi coloured kits, different coloured shorts, players’ skin and hair make for noisier, less distinct histograms. Yoon
et al\cite{35} use two histograms for shorts and shirts. This can have problems if the shorts classification does not match the shirt classification.

Huang and Hilton\cite{14} apply a clustering algorithm and a Gaussian Mixture Model to the foreground mask to segment the distinct colours of the team kits. Players’ teams can be classified using their *Mahalanobis Distance* to the model. This model can be built online from the frame sequence, alleviating the need for training on start-up. There is a choice whether to build one model, using a method like *k-means* to cluster the points into five sets, or whether to use a model per team kit.

The accuracy of player classification methods can be evaluated from the percentage of incorrect classifications across a variety of matches with different team kits. These methods are not specific to the football domain, they could be adapted to more general tracking applications where all tracked objects correspond to one of a finite set of distinctive (possibly noisy) colours.

### 2.3.5 Player Identification

Another secondary objective of the system is to identify individual players in a scene. This is a strengthening of the differentiable players requirement for occlusion resolution. To uniquely identify a player will require the team classification (Section 2.3.4) and analysis of the number on the players’ shirt, except in the trivial cases of referee and goal keepers where the individual kits will be sufficient for identification. Clearly the players’ number may not be visible in every frame, for example if a player is facing the camera. As the tracking algorithm will connect the objects between frames it is only necessary to identify the number in one frame to identify a player throughout the track.
To obtain the number from a player's shirt, an OCR method must be applied. OCR on typed documents, with regular fonts, is generally considered a solved problem, achieving very high success rates. OCR applied to more general problems such as handwriting, number plates, and football shirts achieves lower success rates and is an open area of research. There are many challenges in applying OCR to football shirts. The resolution and aspect ratio may not allow sufficient resolution for distinguishing the characters. Players' position will cause each shirt to be a different size and perspective and as the shirt is not a rigid body, the numbers can get distorted (Figure 2.8). Our scenario does have the luxury of a smaller dictionary, as the characters are only digits, not a full alphabet.

Ko et al. [21] propose a simple method for applying OCR to licence plates. They use contour and shape models to identify individual characters, then the vertical axis is found and a feature vector is created with respect to the vertical axis. This feature vector can be classified by matching it to a set of template models. This scenario shares many of the same challenges as the football shirts, they demonstrate it remains robust with heavy distortion and strong perspective views.

2.4 Tracking

Once player objects have been isolated, the next consideration is how to track them. Players will be successfully tracked if every player object in one frame is associated with its corresponding object in successive frames. Tracking football players has some complexities over standard tracking problems. There are multiple objects to track, all moving at different speeds with non-linear motion. The number of objects will vary as players enter and leave the field of view and there must be a strategy to track players through occlusions as image segmentation and object recognition cannot be relied upon to resolve occlusions in a monocular viewpoint model (Section 2.3.3).

The success of these methods can be evaluated by measuring the deviation of the tracks they generate from a ground truth track, manually marked on each frame. The algorithm which achieves the smallest deviations will be the best at accurately tracking the players. An algorithm that can maintain accurate tracking on a number of video sequences, with varying conditions, will be reliable. Considering sequences with varying levels of occlusion can test an algorithm’s occlusion resolution.
2.4.1 Feature Spaces

A tracking algorithm relates points in a feature space. A player is represented as a point in the $n$ dimensional feature space. The tracking algorithm relates points, corresponding to the players’ varying state over time. This forms a set of curves in the feature space, representing players’ trajectories. An occlusion occurs when one players’ state maps to the same space as (or within the covariance of) another point in the feature space. The choice of feature space is very important. Including features that will distinguish players can avoid occlusions.

The simplest feature space is a two-dimensional $(x, y)$ co-ordinate space. The players centroids can be plotted and the tracking algorithm will track changes in the objects’ positions. There is a consideration whether to track the players in the camera plane or the ground plane. Xu et al [34] transform co-ordinates to the ground plane before tracking. This can be beneficial as it can resolve dynamic occlusions, where players are occluded in the camera plane, despite being far from each other on the ground plane. This method relies on an accurate transformation to the ground plane. Xu uses multiple static cameras to define a fixed transformation to the ground plane and provide high resolution of players locations. Inaccurate transformations, as a result of a moving monocular viewpoint and the additional complexity may counteract the benefits of ground plane tracking in this system.

Xu et al [34] include bounding box co-ordinates of the player object in their feature space when using a Kalman filter. This gives the Kalman filter more information about a players’ state, reducing the impact of any error in centroid measurements.

Dearden et al [8] use colour and velocity information in their feature space. This helps resolve occlusions between players travelling in different directions or from opposite teams. Player representations remain un-occluded in the higher dimensional feature space. Needham [23] improve a basic feature space by including a shape descriptor. This is achieved by convolving a scaled bounding box with a set of pre-computed player shape kernels to produce a high dimensional feature vector. This supplies the tracker with more information about the player shape, improving the accuracy and occlusion resolution. If a PDM or similar contour based player model (Section 2.3.3) is used, this can be included in the player representation to enhance the tracker and better resolve occluded players.

Although the dimensionality of the feature space will have an impact on the complexity of the tracking algorithm, Principle Component Analysis (Section 2.3.3) can be applied and points projected into the space of principle variation, reducing the dimensionality of the feature space whilst maintaining the distinguishing component vectors of the player objects.
for resolving occlusions.

2.4.2 Nearest Neighbour

Nearest Neighbour is the simplest mechanism employed for player tracking. Given the previous state and a set of candidate states, nearest neighbour associates the previous state to the ‘nearest’ of the candidate states. Nearest is considered to be the minimum Euclidean distance in the feature space. This is used in several applications, for example [14] [35]. This can be scaled to track multiple players by considering a set of previous states. Each is associated to the nearest of the candidate states. Additional objects in the old frame can be considered players that have left the scene. Additional objects in the new frame can be considered players that have entered the scene. A heuristic, defining the maximum displacement of a player between frames, should be added to reduce the chances of a player, exiting the scene in one area, being associated with a player entering the scene in another area. During an occlusion, the nearest object in the next frame may not be the same player. This method is sufficient for tracking objects in a sparse scene when players do not occlude, however this method cannot reliably track players through occlusions or continue tracking once the occlusion has passed. Although both [14] and [35] enhance the basic model to make some use of colour and shape in their feature space, Yoon et al [35] identify “an algorithm that can deal with more complex problems, such as occlusion” as an improvement for future study.

2.4.3 Probabilistic Tracking

Section 2.2.5 demonstrated the application of a probabilistic framework to classify pitch and non pitch pixels. Tracking football players is fraught with uncertainty. The probabilistic approach can be extended to the dynamic tracking problem. The deterministic nearest neighbour approach generates a unique estimate of a players’ state \( X_t \) for time \( t \), based on the observed player information in the image sequence at time \( t \). In a probabilistic setting \( X_t \) is just one property, typically the mean or mode, of a probability distribution.

We can express probability for a player state \( X_t \), at time \( t \) in terms of the observed data \( r_f \) by the Posterior Density:

\[
P(X_t | r_f)
\]

(2.11)

The observations \( r_f \) are taken from the imperfect image segmentation and player identifi-
CHAPTER 2. LITERATURE SURVEY

cation. The likelihood for the hypothesised state \( X_t \) can be expressed:

\[
\text{Observational Density } p(r_f|X_t)
\]

Using Bayes’ formula for densities, the Posterior Density \( 2.11 \) can be written:

\[
\text{Posterior Density } P(X_t|r_f) = k \cdot p(r_f|X_t)p_0(X)
\]

where \( k \) is a normalisation constant and \( p_0(X) \) is a Prior Density function. In a tracking environment the prior knowledge of \( X \) is related to Posterior Density \( P(X_{t-1}|r_f) \). This formula expresses the probability of a state \( X_t \) in terms of prior knowledge from the previous state \( X_{t-1} \) and some imperfect observational data. This is the principle behind probabilistic tracking methods.

2.4.4 Kalman Filters

Kalman filters are a common method for tracking an objects’ state through a feature space. Kalman filters use a predict-correct cycle to estimate an objects state. The predict step computes a predicted state from the previous state. The correction step updates the prediction by ‘mixing’ it with evidence from the scene. ‘Mixing’ varies the weighting of the prediction and evidence in the Kalman state based on confidence values. This allows the tracker to compensate for noisy measurements. A confidence (or error) in the Kalman estimate is also computed at each step.

Prediction creates a predicted state \( X_t^+ \) from the previous state \( X_{t-1} \). The prediction is modelled by a state transition matrix, \( A \). A predicted error \( P_t^+ \) is computed from the state transition \( A \), the previous error \( P_{t-1} \) and an error term \( Q \). \( Q \) represents the difference between the state transition model, \( A \), and the actual state transition.

\[
X_t^+ = AX_{t-1}
\]
\[
P_t^+ = AP_{t-1}A^T + Q
\]

The correction stage calculates a weighting \( k \), known as the Kalman Gain. This defines the weighting of predicted state \( X_t^+ \) and measured state \( z \) in the new state estimate \( X_t \). \( k \) is based on the confidence in the two states. The state estimate \( X_t \) and error \( P_t \) is computed as follows.

\[
X_t = X_t^+ + k(z - cX_t^+)
\]
\[ P_t = (I - kc)P_t^- \] (2.17)

\( c \) is a “measurement vector” that projects just the state elements that are observed and measured from the image sequence. Further details on the Kalman filter are given in section 5.3.

Using the probabilistic approach of combining the actual measurement with a predicted state makes the Kalman filter more robust than the Nearest Neighbour method. For example, if there is no measured data \( z \) for a frame, such as in the case of occlusion, the tracker can continue to track using the predictions until the occlusion is resolved and measured data is available.

The Kalman Filter requires the tracked state transformation be described by a linear function. This is represented by the matrix \( A \) in Equations 2.14 & 2.15 above. In the football domain this corresponds to player movements obeying an approximation to first order motion. Abrupt changes in direction and velocity are not executed under first order motion. This may lead to errors in the predicted state and inaccuracies in tracking.

Extended Kalman Filters remove the linear functions restriction. A state transformation function need only be differentiable. A Non linear function cannot be represented by a matrix \( A \) thus it cannot be applied directly to the covariance in Equation 2.15. At each time step a Jacobian matrix of partial derivatives is computed to linearise the non linear transformation function with respect to the current estimate. This linearisation is known to lead to inaccuracies and loss of tracking if the initial estimate is wrong or the transformation is inaccurate. Despite this, the extended Kalman filter can produce better results where state transformations are non linear.

Kalman filters only maintain one hypothesis and corresponding confidence value for a players’ state. Interacting Multiple Model (IMM) is a powerful extension to the Kalman filter. This maintains a set of Kalman Filters to track multiple players and different hypotheses about players movements between frames.

Stauffer and Grimson [27] demonstrate using an IMM to successfully track both pedestrians and cars. Xu et al [34] use a simple Kalman filter to track players in their system. A players’ state is represented by its bounding box and centroid co-ordinates. They comment, “when more than two players are grouped in the same foreground region, the uncertainty in estimation is large”. Multiple hypothesis tracking is identified as a possible improvement.
Figure 2.9: *One dimensional example of Factored Sampling.* This figure shows a set of points $s^{(i)}$ (at the centres of the blobs). These are generated by sampling a prior density $p(x)$. Each sample is has a weight $\pi_i$ (represented by the area of the blob) calculated using the Observational Density $p(z|x = s^{(i)})$. The weighted point set is a representation of the posterior density $p(x|z)$. Source: Isard [15].

### 2.4.5 Particle Filters

*Condensation* (Conditional Density Propagation), also know as particle filtering [15], is another probabilistic tracking algorithm that has proved to be a powerful tool for image tracking. Condensation has been adapted for many applications due to its simplicity, generality and previous successes over a wide range of challenging applications. Unlike a Kalman filter, condensation makes no assumption or restriction that the probability of the distributions must be Gaussian. This allows multi-modal distributions that can maintain multiple weighted hypotheses. In the case of occlusions there will be several competing observations, corresponding to the players in the occlusion. This will tend toward a multi-modal non Gaussian distribution for the probability.

Condensation uses an iterative *factored sampling* technique to approximate the posterior density $P(X_t|Z_t)$ (Section 2.4.3). The objects’ state is maintained by a set of weighted samples. Condensation is an iterative algorithm, applying a four step cycle (sample, predict, evaluate and update) to obtain a new posterior density and estimated state from a previous state and an observation. The condensation algorithm, as it was presented in the original paper is given in Figure 2.10.

At each iteration, a weighted sampling of the previous sample set generates a new set of samples. A drift is applied to each sample state, moving and separating the particles in the feature space. This models the state transformation. A drift may be based on a transformation model and/or a random movement. In a state based condensation tracker multiple transformation models are defined and one is selected based on a probabilistic...
for Each time-step $t$ do
From the old sample set $\{s_i^{(n)}, \pi_i^{(n)}, c_i^{(n)}, n = 1...N\}$ at time-step $t - 1$, construct a ‘new’ sample-set $\{s_i^{(n)}, \pi_i^{(n)}, c_i^{(n)}, n = 1...N\}$ for time $t$.
for $n = 1$ to $N$ do
1. Select a sample $s_t^{(n)}$ as follows:
   a) generate a random number $r \in [0, 1]$, uniformly distributed.
   b) find, by binary sub-division, the smallest $j$ for which $c_{i-1}^{(j)} \geq r$.
   c) set $s_t^{(n)} = s_j$
2. Predict by sampling from $p(x_t|x_{t-1} = s_t^{(n)})$ to choose each $s_t^{(n)}$. For instance, in the case that the dynamics are governed by a linear stochastic differential equation, the new sample value may be generated as: $s_t^{(n)} = A s_t^{(n)} + B w_t^{(n)}$ where $w_t^{(n)}$ is a vector of standard normal random variates and $BB^T$ is the process noise covariance.
3. Measure and weight the new position in terms of the measured features $z_t$: \[ \pi_t^{(n)} = p(z_t|x_t = s_t^{(n)}) \] then normalise so that $\sum_n \pi_t^{(n)} = 1$ and store together with cumulative probability as $(s_t^{(n)}, \pi_t^{(n)}, c_t^{(n)})$ where $c_t^{(0)} = 0, c_t^{(n)} = c_t^{(n-1)} + \pi_t^{(n)} (n = 1, ..., N)$.
end for
Once the $N$ samples have been constructed: estimate, if desired, moments of the tracked position at time-step $t$ as $\epsilon[f(x_t)] = \sum_{n=1}^{N} \pi_t^{(n)} f(s_t^{(n)})$ obtaining, for instance, a mean position using $f(x) = x$.
end for

Figure 2.10: The Condensation Algorithm. Source: Isard [15].

weighting. The new samples are evaluated by applying a fitness function. This computes a weighting based on how well each samples’ state explains the evidence. An estimate for the objects state can be computed from the highest weighted sample or a weighted mean. Figure 2.9 demonstrates this particle transition through one iteration. Section 5.4 discusses how condensation is applied to a football tracking scenario.

Condensation provides a method for efficiently tracking an object in a cluttered scene. Instantiating multiple instances of condensation can lead to inaccuracies. Dearden et al report both trackers can trace the same player after an occlusion between players on the same team. Tracking footballers requires a multiple object tracker rather than multiple single object trackers. Dearden et al suggest information could be passed between the multiple condensation instances to rectify this. Needham describes an extension where multiple sets of players are maintained. Each set has different sampling probabilities. A fitness function is used to compare each sample set with the data from the image and find the set which best fits the measured data. Okuma et al describe a boosted particle filter which uses Adaboost and mixture particle filters to create a multiple object tracking
CHAPTER 2. LITERATURE SURVEY

system for ice hockey. This is more robust than the basic condensation algorithm and handles players entering and leaving the scene.

Football player tracking requires a robust, multiple object tracker capable of handling objects entering and leaving the scene. The simplicity and multiple hypothesis tracking make condensation an attractive tracking method.

2.5 Camera Behaviour

The core system is based on image sequences from a static monocular viewpoint. Transforming the system to work with a moving camera is identified as a secondary objective. Many different types of footage have been used in existing football tracking systems. This section considers the implications of extending the system to track players in a moving scene and modelling the camera behaviour to detect scene changes.

2.5.1 Moving Camera

In general, the main source of footage in a football game comes from the ‘spotter’ camera, located at the top of the stand. This has the ability to pan, tilt and zoom during a sequence of play and is prone to sudden movements [8]. To track players from this footage, an accurate model of the camera motion must be produced. This defines a plane to plane transformation to map points from the previous reference frame into the current reference frame. The camera model is applied to any measurements with respect to the previous frame. This cancels the effect of the moving camera.

To find a transformation between two planes, at least four common reference points must be matched. Cross correlation can match a common region in two frames. A template of a region in the first frame is created and matched against every pixel in the second. A Mean Square Error (MSE) is calculated to identify possible correspondences which best match the template. Alternatively a feature detection method such as Harris Corners or SIFT features could be used to identify prominent features in both frames. Cross correlation or feature detection will produce a set of possible correspondences. Valid correspondences will appear in both frames, these are explained by a valid camera model. These are called inliners. Some correspondences are not valid, these will not be explained by the camera model. These are called outliners. The RANSAC algorithm can be used to identify the trend and build a model from a set containing inliners and outliners. Section 5.1 describes the RANSAC algorithm. Feature sparse football scenes will make feature detection a
challenge.

Bebie and Bieri [5] create a transformation between two static cameras by matching three straight lines, marked manually by the user. Yoon et al. [35] match co-ordinates on the centre and side lines where both are visible. When these are not available they match points on the advertising board region surrounding the pitch.

Once common regions have been identified, transformations from one plane to another can be performed using an homography. The homography defines the geometrical relationship between two planes. Points (in homogeneous form) can be transformed from one plane into another through pre-multiplication by the homogeneous transformation matrix. Sullivan and Carlsson [28] demonstrate such a homography to map the scenes from four stationary cameras into a single view of the whole pitch. Iwase and Saito [16] use a homography to combine tracking information from different camera planes. These works differ from our system as their planes are generated by multiple cameras however this method can be adapted to a moving camera model. A panoramic scene can be produced using the homography to mosaic the image sequence.

Common points, identified in the image, could be considered points in a feature space. This allows tracking methods (Section 2.4) to be applied to the camera model. The tracker can be used to predict the locations of reference points in the next frame, adding probabilistic information into the camera model. Beetz [6] uses a Kalman filter (Section 2.4.4) to track the camera motion and restrict the search space when finding reference points in a new frame.

The success of a camera motion model can be measured by the error in the homography. This can be averaged over a moving camera sequence and should be tested against a number of camera movements, such as zooms, pans, sudden and fast movements. The success of tracking on a moving camera can be evaluated by comparing the accuracy of a tracker on similar moving and non moving sequences.

2.5.2 Optical Flow

Optical Flow represents the movement of image pixels from one frame to the next. It provides a model of all the motion between frames. This makes it relevant to modelling the camera motion and object tracking (Section 2.4). Optical flow computes a dense representation of the motion, modelling the movement of every pixel in the scene. Other motion models derive movement from selected points or image features. This is a fundamental dif-
ference between optical flow and other motion modelling techniques. A dense motion vector field is computed by associating every pixel with a velocity vector, $v_{i,j} = (u, v)$, defining the speed and direction of that pixel’s movement. To calculate the motion, similar regions between the two frames must be identified. There are many different types of optical flow algorithm. Galvin et al \cite{11} evaluate 8 different algorithms. Three possible algorithms are:

- **Block matching.** This is a cross correlation technique. Regions in the old frame are matched with regions in the new frame by means of template matching. Applying a cross correlation to regions of the homogeneous grass could result in many incorrect correlation’s and outliers, generating noisy results.

- **Gradient detection.** This is a more advanced optical flow algorithm. This matches common regions using gradient information. In sparse grass regions gradient information may be limited.

- **Frequency detection.** This applies a (Fast) Fourier Transform (FFT) algorithm to the images and matches regions based on frequency and phase information.

Optical flow makes assumptions that surfaces are well textured and the illumination is constant (even if it is not stationary). These assumptions will not hold in real world football scenes. This will lead to cumulative errors and loss of tracking over time \cite{11}. Galvin et al \cite{11} note, optical flow generally does not perform well in cases of occlusion and flat untextured regions, such as a football pitch.

Beetz et al \cite{6} use an optical flow system to model camera behaviour in scenes where the camera is too zoomed or moving too fast to identify line information. The noise and outliers are filtered using the RANSAC algorithm to identify the trend in the cross correlation. The filtered flow is used to calculate a predicted homography. An optical flow system could be used to model camera motion where no common regions can be identified.

### 2.5.3 Automatic Scene Detection

Sports video is arguably one of the most challenging domains for robust scene detection. In general, football footage only consists of a few types of shot, these include; long shots, from cameras in the stands, close-up player shots, out of field shots of the crowd and slow motion/reverse angle replays. These are shown in Figure 2.11.

The soccer field causes a strong colour correlation between shots. There is little change in the colour distribution in RGB or HSV space. One scene can contain a large amount of
sudden motion (both from the camera and the subjects). This can create variation in frame statistics equivalent to those in a cut. Football footage also contains both fast cuts and gradual transitions, such as wipes and dissolves. This makes football footage a challenge for standard scene detection algorithms.

In general, cut detection works on a two-phase-principle. Scoring takes each consecutive pair of frames and produces a probability that there is a scene cut. Thresholding is applied to identify scene cuts based on a certain threshold value.

By considering (fast) moving cameras, basic scoring methods such as *Sum of absolute differences* (SAD) cannot be used effectively. SAD sums the absolute of the difference in the value of each pixel in two consecutive frames. If the camera moves then the pixels will correspond to different parts of the image, yielding high variance and a scene cut detection. Mateer and Robinson’s scene detection tool SALSA \[19\] uses “a fast, high-accuracy, projective transform estimator” to model the camera’s motion. From this, sample grids of the same areas can be scored. If a precise model of the camera’s motion is already available (Section 2.5.1), this method could be applied with little additional cost in complexity.

Ekin et al \[9\] demonstrate a method for shot boundary detection in football using a combination of scoring factors to compensate for the complexities of the football domain. For a frame pair $i$ and $i-k$, they consider the difference in the percentage of grass pixels and the difference in the colour histogram. Choosing $k > 1$ detects both cuts and gradual fades. Different thresholds are used for off-field shots where the grass ratio is not applicable. They also demonstrate methods for slow-motion replay, goal and referee detection.

The success of scene detection methods can be evaluated using three measures. *Precision*, the probability a detected cut is a cut. *Recall*, the probability an existing cut will be
detected and $F_1$, a combined measure which is a high value if, and only if, both precision and recall are high values. A scene detector that detects too many cuts, i.e. all the scene changes are detected but additional cuts in the sequence are erroneously detected, will have a low precision, a high recall and a low $F_1$ value. A scene detector that only finds valid cuts but only finds 50% of them will have a high precision but a low recall and low $F_1$.

Another possible extension could investigate the use of more than one camera. Systems such as [14] [5] demonstrate this can produce more accurate tracking. The tracker can use the camera with the best view, i.e. that best resolves an occlusion, or apply the tracker to each viewpoint and combine the results. Results are generally combined using a graph representation [14] and shortest path algorithm. Using more than one camera would give a stereo view into the scene. From a stereo view the fundamental matrix can be estimated. Epipolar geometry could be used to obtain better measurements for players’ ground positions [16] or to generate a 3D model of the scene [5]. Using multiple cameras would introduce complexities in system setup and in synchronisation.

2.6 Conclusion

This review demonstrates tracking football players is an open, unsolved research topic. This chapter outlined algorithms and technologies that have been, or could be, applied to football player tracking and its related problems. Advantages, disadvantages and comparisons of the various techniques have been discussed with considerations of metrics for evaluation. The key algorithms and areas for further study are as follows:

- **Pitch Segmentation.** Several different methods were identified for removing pitch regions from the image (Section 2.2). Previous studies have shown all these methods produce good results. Section 4.1 experiments with background subtraction, Eigenmodels and GMM methods to achieve the best segmentation. This is measured using the evaluation methods discussed. Most existing systems use a global threshold. Section 4.1.3 experiments with the various threshold techniques from Section 2.2.6. Additional techniques to fuse complementary segmentation techniques and more advanced colour spaces were investigated as possible extensions if the investigated methods fail to give the desired results.

- **Object Recognition.** The Hough transform was presented as the common method for identifying the pitch markings. Edge detection and thinning techniques were proposed if the segmented image is insufficient for resolving line data. Most existing
systems use a naive heuristics method for identifying player regions. Contour models, shape models and techniques such as PCM’s were explored to build a more refined player object model. Section 4.2.3 explores methods for player recognition.

- Tracking. The predominant methods for tracking football players are Nearest Neighbour, Kalman filters and Condensation. Chapter 5 applies these algorithms to the football tracking application. Feature space is identified as fundamental to effective tracking. Different feature spaces are developed throughout this chapter. The trackers will be evaluated against manually plotted, ground truth trajectories. Occlusion resolution is explicitly tested.

- Extensions. Section 2.3.3 discusses strategies for classifying players into teams and OCR techniques that could be used to identify individual players. Player classification mechanisms are implemented in several football tracking systems. Section 4.2.4 documents a method for accurate team classification. Section 2.5 discusses the challenges of tracking with moving cameras and scene detection. Different methods for detecting and modelling the camera motion were discussed. Section 5.1 builds a camera motion model based on these techniques. Section 5.6 considers a method for scene classification. Optical flow and tracking algorithms were identified as possible enhancements.

Algorithms that are football specific and those which are more generic have been identified. The basic segmentation and tracking system are generic enough to apply in other tracking applications. Optimisations and enhancements, such as player shape models, may use more football domain specifics. These will restrict the trackers application outside the football domain.
Chapter 3

System Specification

3.1 Problem Decomposition

To effectively manage the time and risk factors within the problem (Section 1.2) the project must be organised into smaller more manageable sections. The problem is been decomposed into core functionality and a number of extensions as follows:

3.1.1 Core System

The core system must accurately and reliably track multiple players through a football scene from a monocular viewpoint video sequence, as defined in the primary objective (Section 1.2.1). This should use quantifiable metrics to determine accuracy and reliability and evaluate different methods for tracking.

3.1.2 Extensions

The following extensions to the core system have been determined from the secondary objectives of the problem description (Section 1.2.2). These have been classed High, Medium or Low priority based on the value they add to the tracking system:

1. Classifying the tracked players into teams. (H)
2. Building a camera model to compensate for camera movements through the image sequence. (H)
3. Generalising the system to work for other sports and tracking applications. (M)
4. Scene classification and automatic detection of scene changes to index and annotate footage. (M)

5. Behaviour Analysis to determine some basic information about players movements and tactics from the raw player track data. (M)

5. Tracking the ball. (L)

6. Identifying individual players. (L)

### 3.2 Project Development Strategy

The experimental, investigatory nature of this work lends itself toward an incremental, evolutionary software process model. An agile approach is taken to the evolutionary model where the progress and direction of investigation are governed by rigorous evaluation of the system at each iteration. The first objective will be to build a basic working model. Inline with this development the evaluation and testing frameworks must be developed such that a system is in place to test and evaluate each iteration, not just the final result. This early focus on testing and evaluation is required for the agile approach to the development. From this evaluation, flaws and causes of failure can be identified and used to drive changes and enhancements through the next iteration of the process. The evaluation routines can be applied at each iteration to produce quantifiable metrics to measure the improvements from a refinement. The results of this process are documented in the following three chapters. A full project plan for this process is given in Appendix A.

### 3.3 Use Cases

This section outlines two use cases detailing how the system may be used in real world scenarios.

**Scenario:** It is common for broadcasters to provide post match analysis. Experts select key sequences from a match and analyse what teams have done well or badly. Often this involves analysing the position and movements of one or more players through the sequence.

**Use:** Applying the tacking system to the footage should produce a video sequence with the players’ tracks marked onto the pitch. This will give a visual aid to help viewers comprehend the players’ movements. The user should be able to select which players are...
tracked as often not all the players movements are of interest. If the video sequence is from a moving camera, the system must compute a camera motion model to plot previous player positions with respect to the current reference frame.

**Scenario:** A system is developed to analyse player tactics and performance from their positions and movements through the match. Systems (such as the commercial Prozone application [25]) already exist to provide such analysis for professional clubs. Such a system would require details of players’ positions as input.

**Use:** The tacking data could be fed into such a system as input. The Prozone application obtains player data by manually marking it through a video sequence. Automated player tracking could improve this process. To satisfy this use case the tracks must have accuracy and reliability comparable to a manually marked track. If accuracy is not as good as manually marking, a confidence value must be included to allow the user to gauge accuracy and switch to manual marking where there is a loss of accuracy. Player identification is required to allow tracks from different sequences to be linked to the same player.

### 3.4 Requirements Analysis

This section outlines a more formal set of requirements for the tracking system.

#### 3.4.1 Functional requirements

1. The system must track multiple football players through a video sequence of a football match.
   
   (a) This should be output as a set of \((x,y)\) co-ordinates
   
   (b) This should also be visualised by augmenting the tracks to the current frame and creating a video to visualise the tracked objects’ evolution over time.

2. The user should be able to select players to track from the initial frame or alternatively select ‘track all’. If ‘track all’ is selected the system should automatically initialise tracks for all players in the initial scene.

3. The system should be able to initialise and terminate tracks to compensate for players entering and leaving the field of view.

   (a) Tracks should not be terminated immediately. This prevents tracks of players only temporarily leaving the screen being terminated and reinitialised.
CHAPTER 3. SYSTEM SPECIFICATION

4. The system should output extra data to quantify the accuracy and reliability of the track, measured against a ground truth.
   (a) This data should be in a visualisable form (for example graphed or augmented to the video).

5. The system requires a standard video sequence of a football scene as input.
   (a) The video sequence should be in a recognised, standard file format (avi).
   (b) This should be a single scene from a long range, zoomed out camera, such as that of the ‘spotter cam’. This ensures a suitable amount of information is present in the scene for tracking. This is not a huge restriction as a large proportion of broadcast footage is sourced from this viewpoint. Extension 4 defines an automatic scene detection system to weaken this requirement.
   (c) The video sequence could be created from a static or moving camera. A moving camera may pan, tilt and zoom. Performing both slow and rapid movements.
   (d) Footage may contain screen overlays and graphics appended by the broadcaster.

6. The system should have mechanisms to compensate for changes in light and shadows cast by the players.

7. Minimising user input is a secondary requirement. If user input is unavoidable, tools should be provided to enable a user to easily generate these inputs.

3.4.2 Non-functional requirements

8. The project should constitute 300 hours work and comply with all the universities rules and guidelines on project submission.

9. The system is not required to run in real time.

10. The system should run on a standard PC with no special hardware requirements.

11. The algorithms and design should be documented and discussed such that the results could be replicated (e.g. in an optimised language) in future work.

12. The system should track players accurately and reliably.
   (a) Different techniques should be evaluated to find an accurate and reliable method.
(b) Accuracy and reliability should be quantifiable by metrics to allow direct comparison between tracking techniques, as discussed in the project description §1.2.

(c) Each technique should be tested in different scenarios by varying independent variables such as lighting conditions and number of players in the scene.

(d) The system should also output confidence figures to distinguish where there is error due to uncertainty (maybe due to occlusion or players leaving/entering view) and where it is tracking incorrectly.

To satisfy the extensions

To satisfy the extensions discussed in Section §3.1.2 the following requirements are added. The extension each requirement relates to is given in brackets.

13. The system should classify players into teams (1).

   (a) This information should be output with the players tracks.
   (b) Team classification should be augmented to the video sequence with the track.

14. The system should accurately and reliably track objects on a video sequence from a moving camera(2).

   (a) The system must model pan, tilt, zoom, fast and slow movements.
   (b) This should be measured by the same metrics as static football tracking.
   (c) This camera model should also be used to transform a players track into the current reference frame for visualisation.
   (d) Sanity checking must be performed to prevent bad camera motion models breaking the tracking.

15. The system should accurately and reliably track objects in other sports and tracking scenarios (3).

   (a) This should be measured by the same metrics as football tracking.

16. The system should automatically detect cuts and scene changes (4).

   (a) The system must be able to kill and reinitialise tracking on cuts.
   (b) The system must be able to distinguish long shots, suitable for tracking, from un-trackable close ups and off-screen shots.
17. The system should perform analysis on the players’ tracks to extract what is happening from the scene. (5)

(a) The system should be able to determine which team is attacking

(b) The system should identify set pieces such as throw-ins, free kicks and corners.

(c) There is a vast amount of behaviour information that could be extracted from the scene. Other behaviour information could include how fast players are moving, which players are being marked and how good players are at finding space (losing markers). These are important metrics as they measure behaviours that will help teams win matches.

18. The system should accurately and reliably track the ball (6).

(a) This should be measured by the same metrics as player tracking.

19. The system should be able to identify individual players (7)

(a) This should be output with player tracks.

(b) This information should be used to relate tracks belonging to the same player.

3.5 Beyond the Scope

Producing an industrial standard software product or a system that can run in real time will be outside the scope of this project. The aim is to implement and experiment with computer vision techniques to track football players. It concentrates on evaluating methods and achieving functionality, using tools and languages that focus on rapid application development rather than efficiency. The algorithms and design should be documented, commented and discussed sufficiently that results could be replicated in an optimised way in future work.

There are also some interesting areas of research within the field of sports tracking which must remain as exciting extensions for future work due to the time restrictions and the additional complexity involved. Many researchers (e.g. [14][28]) use multiple cameras to improve accuracy, in-particular players positions and occlusion resolution. This is outside the scope of the system due to the complexity in synchronisation, calibration and sourcing footage. As discussed in Section 1.2 a simple single camera system could have applications in the amateur game, allowing more teams to benefit from the types of performance metrics available to big clubs.
CHAPTER 3. SYSTEM SPECIFICATION

3.6 Architectural Design

Computer vision projects are generally modular in nature. From the conclusions of Chapter 2 based on the systems and methods investigated in the literature survey the overall task can be decomposed into the following distinct modules. The modules and their interactions are shown in Figure 3.1.

**Pitch Segmentation** - Low level image processing to remove grass regions

**Object Detection** - Takes an image with pitch regions removed as input. Non-pitch background (crowd and advertising board regions) are segmented. In the pitch region, objects are collected into connected components and classified as lines, players or non players.

**Tracking** - Takes player objects over a sequence of frames, from object detection module, and tracks players’ paths through the sequence.

**Camera Motion Model** - Takes the current and previous frame as input and creates a model, in the form of a homography, for transforming points in the previous frame to the current reference frame. The output of this module is fed into the tracking and visualisation modules.

**Team Classification** - Takes the player objects from the object detection and classifies them into teams. The results of this are fed into the tracking and visualisation modules.

**Visualisation and Evaluation** - Takes the output of the tracker and visualises the tracks by plotting them on the frames and generating a video of the sequence with the tracks appended. This also outputs the team classifications and uses the camera motion model to plot the tracks with respect to the current frame. Evaluation runs the tests to evaluate the success of the tracking and graphs of the results.

3.7 Test Plan

The success of the investigation into the relative merits of different techniques is dependant on the ability to test and quantify the success of each method. To achieve this a test to quantifying the effectiveness of each module must be defined. Using these tests a new method can be evaluated by retesting the system and measuring the difference in that modules’ performance. Methods for evaluating the success of each module were considered in Chapter 2.
Classification problems such as pitch segmentation, object recognition and team classification can be evaluated using precision/recall statistics. These are discussed in Section 2.3.3 [4.2.3]. It was noted a trade-off must be made between precision and recall. The exact trade-off will depend on the robustness to the different types of errors in other areas of the system.

Camera models may be tested by manually marking the same set of points on two consec-
utive frames then using the camera model to locate the points in the second frame from
the first. The average error between the manually marked points and the computed points
in the second frame provide a measure of how accurate the model is.

The main tracking module can be tested using the ground truth metric discussed in Sec-
tion 1.3. This is similar to the camera motion evaluation, measuring the deviation from a
manually marked ground truth. In the case a tracker loses its player it will be reinitialised.
This means a second metric, number or player losses, is also very important consideration
in determining the reliability.

To demonstrate a method is robust, the above tests must be applied over a number of
scenes with differing independent variables such as light conditions, number of players and
levels of occlusion.
Chapter 4

Background Segmentation and Object Recognition

This chapter describes the first two components of the core tracking system. First a sub-system is developed to segment pitch regions from a frame. Remaining regions include players, pitch markings (lines), advertising boards, crowd and on-screen graphics. Second, a system classifies these regions as lines, players or background. Player information is fundamental to all the tracking algorithms. Detailed discussion of how line information is utilised is given in Section 4.2.2. In the following sections different methods are described, evaluated and extended to produce the most accurate segmentation and classification.

4.1 Background Segmentation

This section explores methods for segmenting pitch regions (grass pixels) from the scene. It is possible background may also include the crowd and advertising boards but the noisy signals and high variation in colour make it hard to define a model for segmenting only these regions. A simpler method, using a convex hull, to remove non pitch background is given in Section 4.2.1.

4.1.1 Background Subtraction

Background subtraction is identified in Section 2.2.4 as one common method used in tracking applications. Background pixels are identified as those which ‘significantly’ differ from
CHAPTER 4. BACKGROUND SEGMENTATION AND OBJECT RECOGNITION

Figure 4.1: Background subtraction in HSV space using a threshold of 0.1, setting foreground pixels black. (a) Applied to a standard sequence from a static camera. (b) Applied to a sequence from a moving viewpoint.

A time averaged background. Recall, from Section 2.2.4, a time averaged background, $B$, for a pixel $(x,y)$ at time $t$ can be described as:

$$B(x,y,t) = \frac{1}{t} \sum_{i=1}^{t} I(x,y,i)$$  \hspace{1cm} (4.1)

and a classification for a pixel $(x,y)$ at time $t$ can be made:

$$\text{classification}(x,y,t) = \begin{cases} 
1 & \text{if } |I(x,y,t) - B(x,y,t-1)| \leq B_t \\
0 & \text{otherwise}
\end{cases}$$  \hspace{1cm} (4.2)

Where $B_t$ is a threshold, defining how much a foreground pixel must differ from background. Setting $B_t=0.1$ is found to produce good results working in the HSV colour space.

Figure 4.1 demonstrates the results of applying this technique to two image sequences. When applied to a sequence from a static viewpoint (Figure 4.1(a)) this method has successfully filtered a high percentage of the pitch pixels. It has also filtered the on screen graphic as this remains constant throughout the sequence. Player regions, lines and the advertising board (top left) have been identified as foreground regions. There are a number of pitch pixels falsely classified as foreground across the frame, appearing as a black speckling on the image. There is also false-positive foreground classifications around the moving players. This is caused by players movements and shadows. This error in accurately identifying the precise player region could cause incorrect classifications in player classification.
CHAPTER 4. BACKGROUND SEGMENTATION AND OBJECT RECOGNITION

Foreground regions properties, such as shape and area, will be used to differentiate players from line and background regions.

Applied to a moving camera sequence (Figure 4.1(b)) this method still identifies a high percentage of the pitch pixels and the broadcaster’s overlays. There are more false-positive classifications of pitch as foreground as pixels no-longer align with their time averaged models. There is also some false-positive foreground classification around the lines. In Figure 4.1(b) the position of the centre circle over the last five frames has been classified as foreground. The error around players is also exaggerated. This level of false positive foreground classification could cause pitch regions to be identified as additional players. As these are close to valid players they are likely to cause occlusions and errors in tracking.

**Evaluation and Discussion**

On average, this method identifies approximately 97% of the grass pixels. This result demonstrates background subtraction is a feasible method for isolating pitch regions. Even when applied to a moving camera a high proportion of the pitch pixels (approximately 95%) are segmented correctly. However, there is still a large number of pitch pixels falsely classified as foreground. On a 288x384 image, 5% represents approximately 5000 pixels. These inaccuracies could affect the player identification and the location of the players’ centroid co-ordinates. This will affect accuracy and reliability in tracking. This method also requires at least one frame from which to build a background model and it is highly likely this initial frame will contain players.

Although background subtraction is not accurate enough for pitch segmentations it performs well at identifying a sample (approximately 97%) of the pitch pixels. In the following sections more advanced, statistical pitch models will be used to identify pitch pixels more accurately. Any pitch model must be initialised by fitting the model to some training data. This method provides a simple, low complexity technique for identifying the pitch pixels needed to train any pitch model. This satisfies the secondary objective by automatically bootstrapping the model and removing the need for user input.

### 4.1.2 Statistical Pitch Models

A simple model, based on the hypothesis ‘pitch pixels will not move between frames’ was shown to be insufficient for accurate pitch segmentation due to camera motion and variation in the values of grass pixels. To improve the accuracy of pitch segmentation a spatially
independent pitch model capable of compensating for the variation in grass pixels must be developed. The previous section also demonstrated how a pitch sample could be obtained using background subtraction. This will allow a statistical model to be automatically bootstrapped with no user interaction.

Figure 4.2 shows the distribution of a pitch sample (obtained through background subtraction) in HSV, RGB and CIE Lab colour spaces. The dense clustering of the data in the HSV space justifies the choice of HSV as a feature space in which to build the statistical model.

Section 2.2.5 describes how an Eigenmodel may be used to model the distribution of the pitch pixels in a feature space. The aim is to produce a measure of how well any given point, \( p' \) in the feature space fits the distribution of pitch pixels \( \{p_1, p_2, ..., p_n\} \). Figure 4.3(a) demonstrates the red vs green distribution of a pitch sample. The points \( P_a \) and \( P_b \) are the red vs green values of two pixels to classify. Although \( P_b \) is closer to the mean pitch colour, \( P_a \) is closer to the pitch model as it is less standard deviations from the distribution. Using an Eigenmodel and computing the Mahalanobis distance can express this.

To build an eigenmodel and compute the Mahalanobis distance, first a measure of the variation \( C \) is computed from the training data \( \{p_1, p_2, ..., p_n\} \) by computing a dot product of the mean centred distribution as follows:

\[
\mu = \frac{1}{n} \sum_{i=1}^{n} p_i
\]  

\[
C = \frac{1}{n} (P - \mu)(P - \mu)^T
\]
where $\mathbf{P}$ is a matrix of column vectors for the distribution $\mathbf{P} = [p_1 \ p_2 \ p_n]$. $\mathbf{C}$ is called the **covariance** matrix. As this is a square, symmetric matrix, eigen decomposition can be performed to compute a matrix of its eigenvectors $\mathbf{U} = [u_1 \ u_2 \ u_d]$ and a diagonal matrix $\mathbf{V}$ of corresponding Eigenvalues $\{\sigma_1, \sigma_2, ..., \sigma_d\}$. These eigenvectors form an orthonormal basis set for the feature space with the vectors denoting the principle directions of variation. Figure 4.3(b) demonstrates the eigenvectors $U_1$, $U_2$ for the example distribution. The eigenvalues represent the amount of variation in the direction of the corresponding eigenvector. In the example $\sigma_1 > \sigma_2$.

To measure the distance of a point from the distribution it is transformed into a space with the distribution centred at the origin such that a unit vector represents variation from the distribution. In this space the Euclidean Distance from the origin is a measure of the distance from the model.

A point $p'$ can be transformed into such a space using:
Subtracting the mean $\mu$ shifts the point into a space with the distribution centred at the origin. Pre-multiplying by $U^{-1}$ transforms the point into a space where the unit vectors are the principle directions of variation (the eigenvectors). Pre-multiplying by $V^{-1}$ scales the space to account for the amount of variation in each basis vector. Taking the magnitude of the resulting vector will find the Euclidean distance from the origin. Using $a \cdot a^T = |a|^2$ and $U^{-1} = U^T$ (as $U$ is orthonormal) the distance $d(p')$ can be computed from:

$$d(p')^2 = (p' - \mu)^T U V^{-1} U^T (p' - \mu)$$

This is the definition of the Mahalanobis distance. Figure 4.3(c) demonstrates an Eigen-model fitted to the example distribution. The Ellipse represents points one standard deviation from the model computed using the distance from model measure outlined above.

This method assumes the models’ point distribution in the feature space is Gaussian. In Figure 4.3(a) the pitch sample is a Gaussian distribution in Red/Green space. Figure 4.3(d) plots a different pitch sample in the red/green space. This sample is taken from a pitch with two shades of green and some regions in shade. These pixels do not form a Gaussian distribution in the colour space. Using a different colour space will have an impact on the shape of the distribution (as seen in Figure 4.2) however high contrasts between shaded and non shaded or several particularly distinct shades of green on the pitch can cause distributions to be non Gaussian, even in HSV space.

A Gaussian Mixture Model (GMM) is used to compensate for multi-modal distributions. A Gaussian mixture creates a set of Eigenmodels to best represent the non Gaussian distribution. Parameters for this model such as how many gaussians to use and where to fit them to best represent the distribution are computed using an iterative Expectation maximisation algorithm. This algorithm comprises of two steps:

- **Expectation Step**: Given a set of parameters (guessed initially then iteratively updated), expectation values, specifying how each data point fits the model are computed.

- **Maximisation Step**: Re-computes the parameters to maximise the expectation values across all the data points.

This is repeated until the model parameters converge. This is an approximation algorithm
as it is not guaranteed to find an optimal result. The use of random values in the initial prediction and maximisation step make this algorithm non deterministic. The complexity in computing these expectations and iterating through the process a number of times means this method has much higher complexity than a single Eigenmodel.

Figure 4.3(d) demonstrates a Gaussian mixture model fitted to the non Gaussian distribution. In this example the model uses three Eigenmodels to fit a model to the distribution. A simple method to classify pixels using a GMM computes the Mahalanobis distance to each component of the model and considers the minimum distance as the fitness to the model. A more sophisticated measure computes the Mahalanobis distance to each model and weights these distances based on the importance of each component in the model, computing a likeliness value for the point belonging to the model. We consider the minimum Mahalanobis Distance for pitch segmentation as we would like to classify pixels close to any part of the distribution as pitch. We do not want shade pixels to be adversely weighted because they are less frequent than un-shaded grass pixels across the scene.

**Evaluation and Discussion**

Figure 4.3 shows the results of using an Eigenmodel and GMM for pitch segmentation in two scenes from broadcast footage of two different football games. The Figure shows the Mahalanobis distances and classification using a manually chosen threshold. The models are computed from the same training data, identified automatically using the technique documented in Section 4.1.1.

In these images and across the range of test sequences, taken from broadcast footage, the grass pixels exhibit an approximation to a Gaussian distribution in HSV space. The results show both models have produced similar levels of success providing better segmentation than the background subtraction method and a segmentation sufficient for classification. In these examples the Eigenmodel demonstrates better results. The expanded players demonstrate the Eigenmodel performs better at recalling all the player pixels. The red circles (highlighting misclassifications) show the higher precision using an Eigenmodel. This is caused by the GMM over fitting the model to the data. As the distributions form good approximations to Gaussians they can be modelled using a single Eigenmodel. Using multiple Eigenmodels provides a better fit to the data however outlying pitch pixels are removed from the model. Components are also fitted to outlying pitch points closer to player regions in the feature space. This causes more misclassification of player pixels as pitch, resulting in more fragmentation and smaller player regions. This will have an impact
Figure 4.4: (a)&(e) Mahalanobis Distances for each pixel in a frame using an Eigenmodel. (d)&(g) Mahalanobis Distances for each pixel in a frame using a GMM. In these images dark pixels have a low Mahalanobis Distances, corresponding to pitch regions and light pixels have a high Mahalanobis Distances, corresponding to non pitch regions. (b)(d)(f)(h) Show Mahalanobis Distances from (a)(c)(e)(f) respectively, thresholded to classify pitch pixels, generating a binary pitch mask. Thresholded using a global threshold value chosen manually to provide suitable classification. Red circles demonstrate precision errors, misclassifying pitch pixels as non pitch. The blue boxes demonstrate recall errors, failing to identify non-pitch regions, causing player fragmentation. Players in yellow boxes provide a direct comparison between the two methods.

when applying morphological operators (Section 4.1.4) as small player regions will restrict the level of erosion that can be performed without removing useful player regions.

The GMM model has a considerably higher complexity than the Eigenmodel. It was found to be in-feasible to compute a GMM for every frame in reasonable time. Low complexity enables Eigenmodels to be re-computed per frame or at regular intervals. This makes the system more reactive to changes in conditions (such as lighting). The non deterministic, approximation algorithm for GMM also leads to occasional bad models. These may consist of too many components due to bad non deterministic choices or take a long time to compute.

These results required the user to specify a threshold. The following section considers automating this thresholding process. Even with a user defined threshold neither model produced a perfect segmentation of the pitch regions. Using just a colour model, some noisy
Figure 4.5: (top) Two frames classified using user defined thresholds that produced the best results. (bottom) The same frames classified using the threshold from the other. Red circles indicate player information lost in thresholding. No fixed value will produce good segmentation for all models.

Grass pixels will be misclassified non-pitch and some player pixels will fit the pitch model. The highlighted players in Figure 4.4 demonstrate how these errors in segmentation have caused some player regions to become fragmented. To produce a more accurate segmentation a spatial consideration must be added. This will use the hypothesis ‘non-pitch pixels cluster in player regions’ to remove isolated misclassified grass pixels and resolve any fragmentation. Using morphological operators to apply this spatial hypothesis is investigated after thresholding.

4.1.3 Thresholding

To classify a pixel ‘Pitch’ or ‘non Pitch’ from the Mahalanobis Distance there must be some ‘cut off’, defining the distance from the model at which a pixel is no longer pitch. In Figure 4.4 pitch pixels have been classified by manually selecting a threshold that produces good results. Ideally one might hope for a static threshold, for example three standard deviations, that produces the best segmentation given any model. Figure 4.5 demonstrates the best user defined threshold from one model applied to another frame with a different model. It is clear a static threshold is insufficient for producing good segmentations. A threshold must be set at-least on a per model basis. Manually selecting this threshold would remove a level of automation. This is specified as undesirable in the secondary objectives. Section 2.2.6 highlighted several methods for computing a suitable threshold value. Fig-
Figure 4.6 demonstrates using the mean, a one dimensional variant of k-means and a dynamic sliding window method to threshold two frames. A well chosen manual threshold is given for comparison. Approximate precision recall metrics are given in Figure 4.6(k). Recall measures the percentage of player pixels that are identified. Precision measures the percentage of positive classifications that are players. The precision recall metrics are an approximation as they are computed against a manually selecting good threshold, this may not be optimal. Further discussion on precision recall measurements is given in Section 4.2.3.

- **Mean**: Using the mean of the Mahalanobis Distance distribution provides a reasonable threshold in both scenes. Although this appears to work, it does not produce a consistent thresholded image. In the top frame the threshold is too low. Although it achieves high recall there is lower precision than the optimum threshold. This causes some pitch pixels to be misclassified. These are highlighted in red circles. In the second frame the threshold is too high. There is a high precision with fewer misclassifications but there is low recall causing player fragmentation. This is caused by the means’ dependance on the amount of pitch pixels in the scene, causing it to be unreliable across a range of footage.

- **Dynamic Sliding Window**: The dynamic sliding window computes a threshold based on a small window around each pixel rather than the Mahalanobis Distance distribution across the whole frame. Thresholds are not affected by larger variations elsewhere in the scene. This can be seen in the bottom frame where the area of misclassified pitch in the manually threshold image has been successfully classified. Although other methods have correctly classified this region, it is at a cost of a loss of recall elsewhere in the frame. Figure 4.6(k) demonstrates this method produces higher precision and recall than the global mean however it is not as accurate as the manually selected global threshold. Computing multiple thresholds also creates an additional cost in complexity.

- **K-Means** This method attempts to find a threshold that sections the data into two distinct clusters, using a one dimensional version of the k-means algorithm. This method obtains high precision, comparable with that of the manually selected threshold. There are less misclassifications than other methods. This is at a cost of lower recall. Many player pixels are misclassified as pitch, causing small fragmented player regions. These are highlighted in red circles. Low recall makes this method unsuitable as the small, highly fragmented regions will cause problems in the following phases.

Clearly these methods produce superior results to the fixed threshold (Figure 4.5) but the
**CHAPTER 4. BACKGROUND SEGMENTATION AND OBJECT RECOGNITION**

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>80%</td>
<td>99%</td>
</tr>
<tr>
<td>Dynamic Sliding Window</td>
<td>87%</td>
<td>87%</td>
</tr>
<tr>
<td>K-Means (k)</td>
<td>100%</td>
<td>79%</td>
</tr>
</tbody>
</table>

Figure 4.6: Three thresholding algorithms applied to two frames from different video sequences. A manually selected global threshold that produces a good segmentation is provided for comparison. Red Circles highlight inaccuracies in the segmentation caused by choice of threshold values. (k) Approximate Precision recall statistics for the different methods averaged over 15 frames from the two sequences.

classification is not as accurate as the optimal threshold.

Using the shape of the Histogram to determine suitable threshold values was explored in Section 2.2.6. Figure 4.7 shows two histograms of Mahalanobis distances for all the pixels in a frame. The peak at low Mahalanobis distances is due to the predominance of grass in the frames. Figure 4.7 demonstrates taking the right hand base of the peak as a threshold. Clearly anything left of the peak must be classified grass as it is closer to the pitch model than most pitch pixels. The results were seen to be comparable to a manually selected value. Precision recall values were computed against a manually chosen threshold. This results in misleading precision recall values as it is hard to establish whether the manual or histogram threshold produces the best segmentation. The high values do indicate this method produces more accurate segmentation than using the mean, k-means or dynamic sliding window methods.

The difficulty in this method is automatically establishing the 'base of the peak'. A trivial algorithm finds the maximum bin from the histogram then steps right through the bins with decreasing counts until a non decreasing bin (with count greater than the previous) is found. Figure 4.7(b) demonstrates a case where this algorithm would fail. There is a local
minima between the maximum value and the base of the peak distribution. Using only the trivial algorithm will cause the pitch pixels to the right of this minima to be misclassified as non-pitch. Reducing the number of bins will change the distribution of this graph, values will be rounded into larger bins having a smoothing effect. Reducing the number of bins too much would allow non-pitch pixels to be rounded into a bin with pitch pixels, causing misclassification.

Another solution imposes an heuristic, restricting which bins may be used as the threshold. An approximation to the number of grass pixels in the frame is known from background subtraction (Section 4.1.1). This identified a large proportion of pitch pixels (approximately 97%). The number of pixels in bins to the left of a candidate threshold value must be greater than the expected number of pitch pixels identified from background subtraction. This ensures a high proportion of the pitch pixels are always segmented, producing results comparable to the manually selected threshold value.

Using the histogram method may have implications when applying the tracker to other sports footage (Extension 3) as it is making use of the predominance of grass pixels in the football scene.
Figure 4.8: Morphological operators applied to a thresholded pitch segmentation. Misclassified non pitch pixels are removed and fragmented player regions are connected prior to the classification process. Structuring elements applied: (b) Horizontal Line, size 1. (c) Disk, size 1. (d) Disk, size 5. (e) Disk, size 2. (f) Dilation (Disk, size 5) followed by erosion (Disk, size 2).

4.1.4 Morphological Operators

In the following section non pitch pixels will be collected into connected component regions and classified using object recognition techniques. Although pixels are, in general, classified with a high level of accuracy, some erroneous classifications are inevitable. Pitch pixels may be misclassified as non-pitch, appearing as a speckling across the image. Player pixels, in particular skin tones, may get misclassified as pitch. In extreme cases player regions become fragmented, arms and legs are separated from bodies. This is shown in Figure 4.8(a). This is a common result when using low level techniques to classify at a pixel level (Section 2.2.7). Bebie[5] and Dearden[8] report similar results.

Morphological Operators erosion and dilation are applied to remove erroneous pitch regions and connect fragmented regions. Morphological operators use structuring elements. These define how a pixels’ new value is computed from pixel values in a surrounding window. Figure 4.8 demonstrates morphological operators and combinations of operators with different structuring elements applied to a thresholded image. Clearly the correct structuring elements and combination of operators is key to successfully removing pitch pixels and
connecting fragmentation.

Bebie [5] uses a dilation followed by an erosion to connect fragmented regions. Figure 4.8(f) demonstrates applying a dilation first, followed by an erosion. To remove the larger misclassified pitch regions the erosion has also re-fragmented the player regions. To filter erroneous pitch regions an erosion operator must be applied first. This must be small enough that small player regions are not erased. Figure 4.8(c) demonstrates an erosion with a disk structuring element size 1. The red circle highlights how vital player information is nearly lost. To combat this, Figure 4.8(b) uses a Horizontal line element. Although this does not remove all the misclassifications it does not erode as much useful player information. The horizontal line element is used to reflect the elongated shape of player regions. Figure 4.8(d) & (e) apply a dilation and erosion to the eroded image (b). This connects the fragmented regions. The dilation (d) must be larger than the original erosion (b) to connect the fragmented regions. Good results are produced using a disk shaped element.

From these results a successful combination for filtering misclassified grass pixels (without removing player information) and reconnecting fragmented player regions uses the operators:

- Erosion: Horizontal line element Size 1.
- Dilation: Disk Element (larger than initial erosion) Size 6.
- Erosion: Disk Element (smaller than dilation) Size 4.

4.2 Object Classification

The results of the previous section segment pitch regions from the scene. The remaining regions may include players, lines, crowd, on screen overlays and more. Player regions must be distinguished from other regions as these will form the input to any tracking algorithm. A player region (connected component) may consist of more than one player when there is an occlusion.

4.2.1 Field Identification

Players must occur within the pitch region. In some scenes (Figure 4.9) the pitch region may not extend to the whole frame. Identifying and segmenting only the pitch region reduces the operating space for tracking.
CHAPTER 4. BACKGROUND SEGMENTATION AND OBJECT RECOGNITION

Pitch segmentation has already classified all the pitch pixels. Computing a convex hull of these pixels generates a convex polygon encompassing the convex pitch. All regions outside this area may be classified as background and segmented. Figure 4.9 demonstrates the convex hull of a pitch region and the segmented image. Not applying the player classification to these background regions reduces the complexity and prevents false positive player classifications from crowd regions. Figure 4.9 (bottom) demonstrates this method may not extract all advertising board and crowd regions. Misclassification of grass pixels at the edge of the pitch create a non convex pitch region. Small amounts of crowd or advertising board may fall within the convex hull of this pitch region. This is not a major issue as these will be segmented at the player classification stage.

4.2.2 Line Identification

Following pitch segmentation, the remaining regions are predominantly lines, screen overlays and players. It is possible to apply a player classifier directly to the remaining regions. Figure 4.10(a) was generated by applying the player classification from Section 4.2.3 in the first iteration of the development cycle. Two players are occluded with line regions causing them to have irregular shapes and be misclassified as non-Players.

To prevent these misclassifications player-line occlusions must be resolved before player classification. Erosion in the pitch segmentation stage can resolve minor player-line occlusions. In section Section 4.1.4 some line information was removed when erosion was applied. Scaling this idea to remove larger lines requires larger, more aggressive erosions. Figure 4.8 (Section 4.1.4) demonstrated how larger erosions can lead to player fragmentation and loss.
CHAPTER 4. BACKGROUND SEGMENTATION AND OBJECT RECOGNITION

Figure 4.10: (a) Basic Player classification based on shape properties applied to connected components in frame. Non player regions have been masked out in black. Where players have occluded with lines the unusual composite shape has not been identified as a player. (b) Two further cases of misclassification due to line occlusions. Using a Hough transform to identify and remove the lines, these players have been correctly classified in the bottom images.

A Hough Transform is used to identify the main lines in each frame. Identified lines can be filtered from the frame prior to the player classification, breaking line-player occlusions. The area of the line connected component is tested at points along the line to avoid filtering players occluded with the line. A window is defined around a point on the line, if the connected component area within the window is greater than some threshold, the point is classified player. Empirical results found no line region had more than 200 pixels in a 10x10 window. All player regions have substantially larger areas. Figure 4.10(b) demonstrates 2 player-line occlusions resolved using this method. Once player regions are not occluded with lines regions they form more regular shapes and are correctly classified as players. When applied to a sequence where several players were occluded with lines player recall improved from 82% to 85%.

4.2.3 Player Classification

Section 2.3.3 considers different methods for classifying connected components as players. This section develops a method that achieves player classification with high accuracy. Figure 4.11 demonstrates a sample of regions to be classified.

The simplest approach to player recognition (Section 2.3.3) uses heuristics, based on area or bounding box dimensions. Variation in player sizes due to occlusions and the perspective camera view makes such an approach inaccurate. A more advanced player model is needed for accurate classification.
To compute the accuracy of player classification, quantifiable metrics to measure accuracy must be defined. Precision and Recall are common metrics used for classification. A confusion matrix defines the possible results of classification:

<table>
<thead>
<tr>
<th></th>
<th>actual positive</th>
<th>actual negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>classified positive</td>
<td>TP</td>
<td>FP</td>
</tr>
<tr>
<td>classified negative</td>
<td>FN</td>
<td>TN</td>
</tr>
</tbody>
</table>

From this matrix, precision and recall are defined as:

\[
Recall = \frac{TP}{TP + FN} \tag{4.7}
\]

\[
Precision = \frac{TP}{TP + FP} \tag{4.8}
\]

In the player classification task, TP (True Positive) results are player regions classified as players. FN (False Negative) results are player regions misclassified as non players. FP (False Positive) are non player regions classified as players.

A technique that classifies all regions as players has high recall but low precision. A classifier that returns no player regions would have high precision but no recall. These examples demonstrate it is trivial to generate an algorithm that produces good results for one measure. An optimal classifier would score highly at both measures. In practice there is a ‘trade-off’ between precision and recall. The best trade off is dependant on the particular tracking algorithm. A testing framework is developed to measure precision and recall accuracy of a classifier against a manually marked player segmentation, averaged over a number of frames from a video sequence.

Figure 4.11: A sample of player and non player regions to be classified.
CHAPTER 4. BACKGROUND SEGMENTATION AND OBJECT RECOGNITION

Although the size of a player varies across the scene, the shape of player regions is quite different to the shape of most non-player regions (the advertising board or onscreen graphic in Figure 4.11). A better shape model is built by considering more advanced shape properties. The following shape properties are considered in building a model to distinguish player regions:

- **Area**: Number of pixels in region.
- **Convex Area**: Number of pixel in convex image of shape.
- **Solidity**: \( \frac{\text{Area}}{\text{Convex Area}} \)
- **Major/Minor Axis**: Major/Minor axis of the ellipse with the same normalised second central moments as the region.
- **Orientation**: Angle between the \( x \)-axis and the Major axis.
- **Ellipsity**: \( \frac{\text{MinorAxis}}{\text{MajorAxis}} \)
- **Extent**: \( \frac{\text{Area}}{\text{AreaBoundingBox}} \)

From these properties we can create a feature space, representing the shape of a connected component. A subset of these properties form the dimensions of the space. The dimensionality of the space is the number of properties used. Every connected component maps to a point in this space by computing a vector of its shape properties. The aim is to find a feature space such that (a) Player regions cluster into a small number of areas in the space, (b) Non-player regions fall far from these clusters in the space. If such a feature space can be found an Eigenmodel or GMM can be fitted to the distribution to create a player model.

To identify such a space and train the model, training data, consisting of both player and non-player regions is required. Figure 4.12 shows a selection of the training data. This was selected over a number of frames to include a wide range of player shapes and a selection of non-player regions.

We can measure how ‘player’ a region is by computing the regions’ shape properties, mapping it into the feature space, then measuring the Mahalanobis distance to the player model. This can be thresholded, similar to pitch segmentation (Section 4.1.3), to produce a classification. Using a number of shape properties results in a high dimensional feature space. Principle Component Analysis (PCA) is be applied to reduce the dimensionality of the feature space to the dimensions of greatest variation.
CHAPTER 4. BACKGROUND SEGMENTATION AND OBJECT RECOGNITION

Many of the properties, such as area, are dependent on scale. Players scales differ across the frame due to the perspective camera angle. To combat this, a scale invariant feature space is also tested. All regions are scaled to a constant size and a second set of shape properties are computed, adding more possible dimensions for the feature space. When scaled, the shape of occluded player regions forms a better approximation to that of a player (as seen in Figure 4.11).

Evaluation and Discussion

Figure 4.13 demonstrates a sample of the candidate feature spaces tested (projected down into the two dimensions of principle variation for visualisation purposes). Different subsets of properties are tested to maximise clustering of player regions and find the best separation between player and non player regions.

- **Feature Space 1** produces a reasonable separation between players and non players, however player pixels are not well clustered. This is reflected in the accuracy of classification using this space. Figure 4.13(e) shows Feature space 1 has a high recall from good separation between players and non players however poorer precision due to a larger number of outlier player regions being misclassified.

- **Feature space 2** has a similar separation however introducing area and solidity properties to the feature space helps cluster player regions. This is reflected by similar recall but higher precision in classification, as seen in Figure 4.13(e).

- **Feature space 3** harnesses scale invariant properties to produce ‘tighter’ clustering (lower covariance). However there is lower precision than feature space 2 from more non player regions in the player regions. Despite better clustering there are a similar number of outlier player regions causing no improvements in precision, as reflected in Figure 4.13(e). The imprecision is caused by small line and grass regions forming...
‘player like’ shapes when scaled up to player size. These can be seen in Figure 4.11. There is lower covariance across the distribution as scaling approximates occluded players shapes to those of single player regions, as suggested above. This does not lead to improved recall as scaling also distorts other player regions, creating irregular shapes that form outlier points in the feature space, cancelling the precision benefit from better classification of occluded players.

- **Feature Space 4** also utilises scale invariant properties. This feature space achieves better recall than feature space 3 at a cost of a loss in precision.
CHAPTER 4. BACKGROUND SEGMENTATION AND OBJECT RECOGNITION

Figure 4.14: Precision vs Recall for player classification using different thresholds in the classifier. Increasing the threshold increases the recall at a cost of loss in precision. The trade off between precision and recall can be altered by varying the threshold.

Feature space 2 produced the best precision and recall measures. This feature space provides the most accurate classification from all those tested. As in pitch classification, the choice of threshold will have an effect on the precision and recall of any classifier. Figure 4.14 demonstrates how varying the threshold varies the precision and recall properties of the classifier. The optimum trade off between precision and recall is dependant on the particular tracking algorithm. Changing the threshold allows the classifier to be ‘tuned’ to best suit each tracker.

Alternative Approaches

There are many other approaches to player classification in the literature. In this section two more classifiers are evaluated.

Feature Vectors

The shape classifier uses a feature space that describes the shape of a connected component region. An alternative classifier constructs a high dimensional feature space from the grayscale pixel values of a connected component. A window is fitted around the region to be classified and a feature vector is generated from all the grayscale pixel values within the window. Pixels in the window but not within the region are set to a null value. PCA must be applied, using a sample of player and non player data to identify the vectors of maximum variation. The high dimensional space requires more training data than the lower dimensional shape model. A GMM or Eigenmodel can be fitted to the PCA’ed distribution and regions can be classified as in the shape based classifier.
Figure 4.15: (a) Scene classified using feature vector classification. The 2 black regions with green crosses are misclassified player regions. The unmasked onscreen graphic indicates this is incorrectly classified as a player region. (b) Training set plotted in three dimensions of principle variation for feature vectors’ PCA’d feature space. (c) Player classification using neural network classifier. Pink crosses represent regions classified as player.

Figure 4.15(b) shows the training data distribution plotted in the PCA’d feature space (projected down into the three dimensions of maximum variation for visualisation). Figure 4.15(a) applies this method to classify all the connected components in a frame. Grass, line and ball regions have been correctly classified non player however the onscreen graphic has been classified player and two player regions have been misclassified. Tested under the same conditions as the shape classifier this method achieved a precision of 83.44% and a recall of 80.68%. This is accuracy comparable to shape feature space 1.

**Neural Networks**

Another approach is to use a neural network to perform classification. A neural network consists of a number of simple processing elements (neurons) connected together to perform complex tasks such as classification. Neural networks mark a different approach to the classification problem. Other methods described above follow a bottom up approach, constructing high level player objects from low level shapes, in turn constructed from lower
level pixel classifications. Neural networks represent a top down approach. The network is trained from a sample of player data to build a player model. The network uses this model to actively search frames for evidence that explains this model. The network will take the grayscale values of pixels in a bounding box as input (as in the feature vector). Output shall be 1 for player regions and 0 for non player regions.

Inside the network, neurons are organised into layers. At each layer neurons take a weighted sum of outputs from the previous layer as input. Each neuron applies a Transfer Function to the input to generate it’s output. Neurons in the first layer take a weighted sum of the input vector as input. Output is a weighted sum of the final hidden layer. To generate a correct classification with high accuracy the number of layers, weightings for the sums, number of neurons and transfer functions must be carefully chosen.

As player classification is a complex pattern recognition task, a 2 layer network is used. Best results were obtained using 10 neurons in the each hidden layer. A logsig transfer function is chosen for all neurons as its output range is 0 to 1, making it easier to output a boolean value. To find the correct weightings that will generate the desired output the network is trained from a training data set. Training is done using a Back-propagation technique.

Initial experimentation with this method provides some promising results. Figure 4.15(c) shows pink crosses where the network has classified player regions. This method achieves a high recall rate, correctly classifying all player regions, however it also classifies a line, the onscreen graphic and frame edges as player regions. This is a low level of precision. Neural networks and their design is a vast, complex field. Discussions of how this result could be improved are found in Section 7.1.4

4.2.4 Team Classification

A secondary objective specified tracked players should be classified into their respective teams. Team information can be used in tracking (Chapter 5) to help resolve occlusions between players on opposing teams and in behaviour analysis (Section 5.5) to contrast the behaviours of opposing teams.

A players’ team is represented by the colour of his shirt and shorts. Teams always play in easily distinguishable colours to make team classification easy for other players, spectators and the referee. We capitalise on this, using the colour information in player regions to perform team classification.
As teams play in distinct colours it was hypothesised the distribution of player pixel colour values would cluster into two distinct regions in an appropriate colour space. These clusters would represent the colours of the two teams. An algorithm such as k-means could automatically identify the two clusters. An Eigenmodel fitted to each cluster would model the team colours. This would create a fully automated team classifier, automatically bootstrapped from scene information.

Figure 4.16 shows a sample of player pixel colour values (from two teams) plotted in HSV space. In figure 4.16(a) player pixels for team 1 form a bimodal colour distribution, represented by the two distinct clusters. This is caused by their two colour (red and white striped) shirts. It is possible to identify the three clusters in this distribution using the k-means algorithm however it is undecidable which clusters belong to each team. This
prevents a model being automatically bootstrapped from the scene data. A small amount of training data (less than 10 players per team) is required to derive the distribution of each team.

The existence of bimodal distributions also requires a GMM to accurately model the distribution. A training set is built using a simple tool that allows the user to select the players from each team across a number of frames. Only a small number of players need to be identified as each pixel within the player contributes toward building the distribution. For each team, the colour value of every player pixel, from its training set, is represented as a vector in HSV space, PCA is applied to this distribution then a GMM is built. There is one GMM to represent each team.

To test a region against a team model, each pixel in the region is measured against the teams’ GMM. The pixel is measured against each component of the teams GMM and the minimum Mahalanobis distance is considered as its fitness. A likelihood measure (Section 4.1.2) is not used as pixels corresponding to any component of the GMM should get low values, not just pixels corresponding to the the most predominant colour. A mean pixel fitness is computed to normalise with respect to the number of pixels in the region. Each candidate region is tested against each team model to give a confidence value for a region belonging to each team. A region is classed as the team which achieves the highest confidence.

Figure 4.16 shows the results of using this model to classify players in two scenes. The radius of the two semicircles demonstrate the confidence of a player being in each team. The classification is indicated by the colour of the cross above each player. Notice the referee scores a low confidence against both models. Although the confidence is low, the referee is still classified and this classification is consistent through the sequence. The referee is tracked with the same accuracy as other players. Section 7.1.5 outlines extending the team classification to include the referee and goal keepers.
Chapter 5

Tracking, Camera Models and Behaviour Analysis

This Chapter documents and evaluates the tracking algorithms implemented. Three different solutions to the tracking problem, Nearest Neighbour, Kalman Filters and Condensation are considered. These all meet the requirements for the tracking system, tracking multiple players and using the camera motion model to plot the previous track on the current frame as output. A system to compute a camera motion model is described. This chapter also investigates scene classification and behaviour analysis.

5.1 Camera Motion

Tracking is an inference problem. It aims to infer the movement of players from a sequence of static images. A basic tracking system will track the players movements through the sequence using image co-ordinates, with respect to a reference frame in the image plane. In the case of a static camera, players movement in the image reference frame can be translated into movements in real world co-ordinates using a simple static transformation. This motivates the use of static cameras in many football tracking projects. In a moving camera system there is no a simple, static correlation between movements with respect to frame co-ordinates and real world co-ordinates. A player running along the pitch, being tracked by the camera, will remain static with respect to the image reference frame.

This has implications for the football tracking system.
The requirements specify the system will mark a players’ track by plotting the players’ previous locations on the current frame. If there is camera movement between two frames the co-ordinates of a previous player location (with respect to the previous camera reference frame) will not correspond to the same point on the pitch in the new reference frame. This will produce inaccurate player tracks if camera movement is not modelled.

The kalman filter will make a prediction of a players location in the next frame, based on a motion model. This point will correspond to a location with respect to the current reference frame. This point must be translated into the new reference frame if it is to be compared to player data, described with respect to the new reference frame. Nearest Neighbour and condensation also make use of image co-ordinates from previous frames. These would also benefit from a camera motion model.

To compensate for camera motion, a transformation, mapping points from the previous camera reference frame into the current reference frame is required.

This transformation is an homography. A homography provides a translation from one plane to another. It is a 3x3 matrix applied to co-ordinates in homogeneous form. A homography can be computed, using linear algebra, from 4 points, matched in the two planes. These points should all lie on the same plane in the image.

To compute an homography a number of interest points must be identified in the two scenes. A subset of points on the same plane must be present in both frames. Points on the football field will all lie on the same plane. Interest points are identified using a Harris Corner Detection technique. Given a point \( f \), Harris Corner Detection computes the ‘cornerness’ of the point \( C_f \). This is computed from the Hessian Matrix, \( H \) of the point. The Hessian Matrix is a combination of the partial, second order derivatives of the point \( f \):

\[
H = \begin{pmatrix}
  f_{xx} & f_{xy} \\
  f_{yx} & f_{yy}
\end{pmatrix}
\]

From this \( C_f \) is computed as:

\[
C_f = \det(H) - \kappa \text{trace}^2(H)
\]

Good results were achieved setting \( \kappa = 0.04 \). This is a common value, reported in the literature to produce good results. Football scenes present a challenge for feature point detection as the predominant pitch creates a flat signal with high feature sparsity. Figure 5.1


Figure 5.1: (a) & (b) Points identified using Harris Corner Detection in two consecutive frames. (c) Using the histogram to map all points from previous frames into the current reference frame to create a mosaic. (d) Measure of the accuracy and number of errors in the camera model for different sequences. Computed by performing three camera model translations leading back to original reference frame and measuring the deviations for a set of equally spaced points (in pixels).

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Bad Transformations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static Camera</td>
<td>1.69%</td>
<td>0</td>
</tr>
<tr>
<td>Moving Camera 1</td>
<td>5.45%</td>
<td>2</td>
</tr>
<tr>
<td>Moving Camera 2</td>
<td>3.17%</td>
<td>1</td>
</tr>
<tr>
<td>Standard Lighting</td>
<td>0.81%</td>
<td>0</td>
</tr>
<tr>
<td>Low light, Torrential Rain</td>
<td>0.78%</td>
<td>1</td>
</tr>
<tr>
<td>Bright Sunlight, Shadows</td>
<td>1.42%</td>
<td>0</td>
</tr>
</tbody>
</table>

demonstrates the results of applying the Harris Corner Detector to two consecutive frames. This has produced a good result, identifying a number of points, on the same plane, that appear in both frames. These can be matched to compute the homography.

The RANSAC (RANdom SAmple Consensus) algorithm is applied to the set of Harris features to match points that occur in both frames and compute an homography. The set of points contains inliners and outliners. Inliners are points that are common in both frames and will be matched by applying the appropriate histogram transformation. Outliners are points that do not appear in both frames. These will not be explained, even when the correct histogram is discovered. Given a set containing both inliners and outliners the
RANSAC algorithm can approximate the model parameters that explain the inliners, in this case the histogram values. The RANSAC algorithm also allows for inliners to be subject to noise. The algorithm works by iteratively selecting a random subset of the original data as hypothesised inliners. A model is computed from these inliners and tested to measure the error in the model. The algorithm iterates a fixed number of times, refining the set of inliners or choosing a new set of inliners. Following these iterations the model generating the lowest error is returned. This generates a histogram defining the transformation between two frames.

Figure 5.1(c) provides a visualisation of the camera model. The transformation is applied to all pixels from previous frames. These are ‘stitched’ to the current frame to form a mosaic of the moving camera video sequence. Figure 5.1(d) shows the results of applying the camera motion model to a set of sequences with differing independent variables. The results were obtained by computing transformations from frame $i$ to $i + 1$, from $i + 1$ to $i + 2$ and from $i + 2$ back to $i$. A set of points, equally spaced across frame $i$ were transformed (using the camera models) into frame $i + 1$, then $i + 2$ and back to $i$. The distance (in pixels) between the original and transformed points represents the error in the camera model. This is averaged over 40 frames. The value represents average error for three applications of the camera model. A system of three applications is used as transforming from $i$ to $i + 1$ and back may lead to biased results as both transformations would be computed on the same set of feature points. The results show this method produces good results across a range of sequences. There is some minor inaccuracies for rapidly moving cameras. These are introduced by the larger transformations. Figure 5.1(c) is the mosaic from test ‘moving camera 1’. The inaccuracies can be observed where the pitch stripes do not all align perfectly.

In some frames, it is possible the algorithm will match points on moving features such as players, match points that do not occur in the same plane or fail to find enough features common to both frames to compute a homography accurately. This can lead to inaccurate transformations. Four such transformations can be observed in Figure 5.1(d). If this transformation is used in tracking algorithms it may lead to inaccuracy and loss of tracking. To catch such transformations a basic error check is applied. This checks for transformations corresponding to large movements in the vertical direction. The vertical direction is used as the camera is prone to large, unpredictable movements in the horizontal direction to follow play. There is little movement in the vertical direction as the width of the pitch can be covered without much camera movement. When erroneous transformations are detected, the previous transformation is reapplied as an approximation to the camera movement. In
5.2 Nearest Neighbour

A tracking algorithm tracks a representation of an object through some feature space. The state of an object at time $t$ is a point in the feature space. At time $t + 1$ a set of candidate states are obtained from the evidence. These form a set of points in the feature space. Nearest neighbour algorithm links the tracked object to a candidate state by selecting the candidate with the smallest Euclidean distance to the previous state in the feature space (Section 2.4.2).

The evidence is the set of player objects, obtained from frame $t + 1$ by the process documented in Chapter 4. A feature space must be defined to represent the state of a player object. Tracking is lost when the wrong player maps closest to the previous state in the feature space. An occlusion occurs when two objects map to (or within the covariance of) the same point in the feature space. The choice of feature space is important to avoid occlusions and loss of tracking.

Using only properties of a player object that vary slowly over time will ensure that a players’ state distribution over time has a low covariance in the feature space. Including player properties that will distinguish players is also important. This will keep candidate player regions apart in the feature space to avoid occlusion and loss of tracking. When player regions are inseparable in the image they are represented as a single player component in the evidence passed to the tracking algorithms. Depending on the dimensions of the feature space this composite player object may map to a different region of the feature space.

We develop a basic feature space consisting of the $(x, y)$ co-ordinates of the player regions centroid. In a moving camera sequence the previous player co-ordinates will be with respect to the old reference frame. The previous state (in homogeneous form) is pre-multiplied by the camera motion homography, translating it into the current reference frame prior to applying the nearest neighbour association.

5.2.1 Evaluation and Discussion

Figure 5.2 shows this method performs well in basic tracking instances where players maintain a suitable degree of separation. The accuracy of the tracker is measured by considering the Euclidean difference (in the image reference frame) between the tracker output and a
Figure 5.2: Graph showing the accuracy of tracking a player using the Nearest Neighbour algorithm. Three scenes were tested, accuracy is lost when a player becomes occluded or a player region is missing due to occlusions. The High error values for occlusion and player misclassification show where tracking was lost. The red line terminates at frame 23 where tracking was lost.

manually marked ground truth. The small inaccuracies are caused by players’ centroids not matching the centre of the players body.

Figure 5.2 demonstrates this method is susceptible to recall errors in player classification. When a player is misclassified as a non player region the evidence for that players’ new state is missing. This causes the tracker to associate to the next closest player region. Once the tracker adopts a different players’ state it no longer holds any state for the original player. This results in loss of tracking. Precision errors in player classification, causing non player regions to be identified as players, do not impact as heavily on tracking. These are only a problem if a misclassified object maps closer the previous player state than the new player state. In this case the tracker will associate with the misclassified region causing a loss of accuracy. If a similar misclassification does not occur in the next frame the tracker can usually recover, unless a different player has moved closer to the misclassified region during these two frames.

The tracker has lower accuracy in cases of occlusion. Occluded player objects (player objects representing two or more players) map to the same region of the feature space as the individual players would have mapped to. This causes the tracker to associate with the composite player region when a player becomes occluded. If two tracked players occlude, both trackers associate with the composite player object, adopting the same state. The centroid co-ordinate of an occluded player region represents the centre of the two occluded
CHAPTER 5. TRACKING, CAMERA MODELS AND BEHAVIOUR ANALYSIS

players, not the centre of the individual player regions. This causes the loss in accuracy in Figure 5.2.

When an occlusion is resolved into separate players all trackers with the occluded state will follow the same player out of the occlusion. This will be the player that has moved least from the centroid of the composite player region. Given an occlusion of $n$ players there will be a loss of tracking with probability $n - 1/n$. If more than 1 tracker enters an occlusion all but one are guaranteed to lose their player. Changing the feature space cannot resolve this issue. All trackers, tracking the occluded region, will have the same state. When the occlusion is resolved they will all make the same, deterministic, association irrespective of what player properties constitute the state. This issue is caused by trackers not maintaining their own independent state. In this algorithm a tracker may only adopt one of the states provided by the evidence. There is no tolerance to compensate for noisy measurements, such as those caused by recall errors or occlusions. The Kalman and Condensation trackers provide mechanisms to compensate for noisy measurements, maintaining more accurate and reliable tracking.

Loss of accuracy from occlusions in the feature space is negligible compared to the error caused by noisy evidence. An occlusion in the feature space would require the wrong player to be closest to the previous state, without the two player regions occluding in the image. As changing the feature space will have no effect on errors from player recognition or noisy (occluded) measurements no further feature spaces were tested.

5.3 Kalman Filters

The previous section identified the principle limitations of the nearest neighbour algorithm. These are the intolerance to noisy measurements (such as those from occlusions or recall errors in player classification) and inability to resolve occlusions. This is caused by trackers only adopting the states measured from the evidence. When two trackers converge on an occluded player region they adopt the same state, becoming identical. There is no state in the tracker to resolve the original player after occlusion. If a measured state contains noise, this is transferred directly into the tracker.

Kalman filters provide better noise tolerance and occlusion resolution. Trackers maintain their own state. This is influenced by the measured state but can deviate to filter noisy measurements and resolve occlusions. The Kalman filter also defines a confidence value $P_t$ for the confidence of the kalman state at time $t$. The Kalman state $X_t$ is computed from
the measured player state $z$ and a predicted state $X_t^+$. $X_t^+$ is calculated by applying a stochastic transformation model $A$ to the previous state $X_{t-1}$.

$$X_t^+ = AX_{t-1}$$  \hspace{1cm} (5.3)$$

$$X_t = X_t^+ + k(z - cX_t^+)$$  \hspace{1cm} (5.4)$$

Where $c$ is a “measurement vector”, projecting the measurable elements from the predicted state vector.

$k$ defines a weighting, used to ‘mix’ the predicted and measured states. This is known as the **Kalman Gain**. It is influenced by the confidence in the previous state, the confidence in the prediction transformation and the confidence in the measurements.

$$k = cP_t^+ * (cP_t^+ c^T + R)$$  \hspace{1cm} (5.5)$$

where $R$ is the error in the measured state and $P_t^+$ is the confidence in the prediction. $P_t^+$ is computed from the confidence in the previous kalman state $P_{t-1}$ and the error $Q$ between the actual transformation and the transformation model $A$.

$$P_t^+ = AP_{t-1}A^T * Q$$  \hspace{1cm} (5.6)$$

The confidence in Kalman state $X_t$ is computed from the confidence in the predicted state $P_t^+$ and Kalman Gain $k$:

$$P_t = (I - kc)P_t^+$$  \hspace{1cm} (5.7)$$

As with nearest neighbour, state is represented as a point in the feature space. The same identification and occlusion factors influence the choice of space. When using a Kalman filter it must be possible to define a stochastic model $A$ to compute a predicted next state from any current state point in the feature space. Including player properties that may vary unpredictably between frames will lead to inaccuracies in a Kalman filter.

The initial feature space uses players’ centroids’ $(x, y)$ co-ordinates (with respect to the image reference frame) and a speed and acceleration component for each direction:

$$X_t = [x_t \ y_t \ v_x \ v_y \ a_x \ a_y]^T$$  \hspace{1cm} (5.8)$$
It is assumed a player moves under an approximation to first order motion. A new player state at a time $\delta t$ can be predicted from a players state at time $t$ using the first order motion equations applied to the x and y directions separately:

\[
d_{t+\delta t} = d_t + v_t \delta t + \frac{1}{2} a \delta t^2
\]

\[
v_{t+\delta t} = v_t + a \delta t
\]

where $d_t$ and $v_t$ are displacement and velocity at time $t$. These can be obtained from the players state.

These equations can be encoded into the matrix:

\[
A = \begin{pmatrix}
1 & \delta t & \frac{\delta t^2}{2} \\
0 & 1 & \delta t \\
0 & 0 & 1
\end{pmatrix}
\]

(5.11)

This is the stochastic state transformation required by Equation 5.3 (in one dimension). Pre-multiplying a one dimensional state $[d_t \ v_t \ a]^T$ by $A$ predicts a new state for time $t + \delta t$. Expanding the matrix multiplication shows the new displacement and velocity are generated by exactly the motion equations 5.9 & 5.10. This is trivially scaled to two dimensions. The predicted state co-ordinates will be with respect to the previous image reference frame. The co-ordinates are converted to homogeneous form and the camera motion homography is applied before the predicted state is used in Equation 5.4. The matrix $c$ should project out the values that can be measured from the state vector. For this feature space the matrix $c$ is defined as:

\[
c = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0
\end{pmatrix}
\]

(5.12)

There are two extra parameters $R$ & $Q$ in a kalman filter that were not in the nearest neighbour tracker. $Q$ is a measure of the error in the transformation model. Player movement is modelled as first order motion. The actual movement of the players is not always under constant acceleration. In particular sudden changes of pace or direction are not modelled by this system. $Q$ should quantify this discrepancy, factoring it into the weightings. When a player will make an abrupt change in direction or velocity is undecidable and the dif-
CHAPTER 5. TRACKING, CAMERA MODELS AND BEHAVIOUR ANALYSIS

Figure 5.3: Graph demonstrating how varying the transformation error parameter $Q$ effects accuracy of Kalman Filter. Raising the value above 10 produced no further improvements.

The difference between this movement and the motion model can vary. This makes choosing a suitable value for $Q$ difficult. Figure 5.3 shows how the accuracy of the tracker is affected by varying the value for $Q$. Following these tests a value of $10I$ was chosen.

$R$ represents the error in the measured states. $R$ should be set high when there is error in the measurements. Possible errors include occlusions and missing players regions. Setting $R$ to a large number will affect the weighting such that the tracker biases more toward the predicted state. A metric to detect noisy measurements must be developed. In cases of occlusion or the wrong player being identified, due to misclassification, there is likely to be a variation in the size of the player region. Variation in the size of the bounding box is a good indicator of error in the measurements. This indicator will also detect smaller changes in bounding box. Such variations may be caused by players extending arms or legs. These can result in the region centroid not matching the centre of the actual player. This was the cause of the minor inaccuracies in the nearest neighbour algorithm. $R$ is computed as:

$$
R = \begin{pmatrix}
    b_x & 0 \\
    0 & b_y
\end{pmatrix}
$$

(5.13)

where $b_x$ and $b_y$ are the differences between the players’ bounding box and the time averaged player bounding box. The zeros represent the decision that error in measurements for one direction should not affect the weighting of the other direction. Empirical studies show it is common for errors to exist in only one direction.
5.3.1 Evaluation and Discussion

Figure 5.4 demonstrates the Kalman filter provides higher accuracy tracking than the nearest neighbour algorithm. The error is in general lower, even in the basic non-occluded state as the filter averages the evidence with a prediction to reduce the effect of player region centroids not matching a players actual centre. The player tracks are visibly smoother when plotted on the pitch.

The prediction also allows the Kalman filter to maintain higher accuracy through noisy sequences. The occlusion result was measured on a video sequence where two players occlude and remain occluded for approximately 40 frames. The occlusion begins on frame 7. For the first 15 frames of the occlusion the predicted state helps to counter the inaccuracies in the measurements. As the occlusion persists the state of the tracker slowly converges to that of the noisy occluded measurement. The errors accumulate and propagate forward until the noisy state is adopted. Post frame 27 the prediction reflects the noisy measurement and the error is similar to that in the Nearest neighbour tracker.

The inaccuracies in the un-occluded measurements at frames 13 and 15 correspond to occasions where the player makes an abrupt stop and change of direction. The player is not obeying a first order motion model. This causes the state transformation $A$ to be inaccurate. This inaccuracy is not modelled in the Kalman filter. A constant transformation error $Q$ was defined. To resolve this inaccuracy, $Q$ should be set dynamically, similar to the measurement noise $R$. In this case the discrepancy between prediction and centroid is due to poor prediction. In other cases it is due to noisy measurements. It is not easy to distinguish between these two cases.

The inaccuracies from players not obeying first order motion are emphasised in the blue plot on Figure 5.4. Here two players occlude and suddenly stop. Although there is a loss in accuracy the prediction helps to resolve the trackers to their respective players post occlusion. Applying nearest neighbour to this sequence resulted in loss of tracking.

Kalman filters are extended to track multiple players by running more instances of the algorithm. Each tracker has the same accuracy as a single Kalman filter. In some cases multiple trackers may associate to the same player after occlusions. This is because the system consists of multiple, object trackers not a multiple-object tracker. No communication is passed between each tracker to avoid multiple tracking instances following the same player.
5.4 Condensation

A Kalman filter only maintains a single state and prediction. It is not possible to accurately predict when a player might not obey a first order motion model thus this cannot be modelled in the predictive step. Using multiple predictions allows multiple hypotheses for the players movements to be predicted. Section 2.4 describes two different tracking methods that maintain multiple hypotheses. Interacting Multiple Model (IMM) maintains a set of kalman filters to model different players movements between frames. Condensation uses a set of particles and a fitness function to model and track unpredictable player movement. The Condensation algorithm is introduced in Section 2.4.5. Unlike Kalman filers and IMM, Condensation does not restrict state transitions to linear functions (i.e first order motion) and makes no assumptions about Gaussian state distributions. This section applies a Condensation tracking algorithm and evaluates the results. An extension is described to extend the Condensation tracker to track multiple objects.

As with Nearest Neighbour and Kalman filters, Condensation tracks points through a feature space. Choice of feature space is important to disambiguate player regions and avoid occlusions. Unlike Kalman and Nearest Neighbour, Condensation maintains a set of possible states in the feature space. Each state has an associated weighting. Measured states, from the evidence, are not projected into the feature space. Instead a Fitness Function computes a weighting for how well a given state describes the evidence. This is inline with the Baysian framework for probabilistic tracking (Section 2.4.3). Initially a basic feature space of player co-ordinates with respect to the image reference frame is used.

Figure 5.4: Graph demonstrating accuracy of Kalman filter under different conditions. Accuracy is marked against a ground truth.
5.4.1 Algorithm

This section describes the 4 step condensation cycle through one iteration at time \( t \). This is as implemented and based on the original condensation algorithm, presented in Figure 2.10:

- **Sample:** At time \( t \), the sample step generates a new sample set \( \{ s_{t}^{(1)}, ..., s_{t}^{N} \} \) by randomly sampling a prior density function. The prior density function is derived from the previous sample set and weightings \( \{(s_{t-1}^{(i)}, \pi_{t-1}^{(i)})|i = 1, ..., N\} \). The weighted sampling factors the probabilistic weightings of the previous samples into the new sample set. Those samples that best described the previous evidence are biased and more likely to be propagated forward.

- **Predict:** This step transforms the samples to new states in the feature space. This models the state transformation to predict the new state of the tracked object. Applying different transformations to particles models multiple hypotheses for the actual state transformation. A transformation consists of a drift and diffusion. The drift is a deterministic operation, similar to the prediction in the kalman filter. As this is deterministic, identical elements (sampled from the same sample of \( t - 1 \)) will undergo the same drift. The diffusion is a random movement, the particles move with independent (Brownian) motion. This breaks apart identical samples, covering more of the feature space. It is important to cover as much of the feature space, within the possible distribution of the new state. This is balanced against biasing point transformations to the most likely regions for the new state. Higher sampling rate in the most likely player regions will increase the probability of finding a sample that produces a high weighting. Covering too much of the feature space could discover invalid states that produce an erroneously high weighting. The transformation and fitness function must be carefully defined to avoid this. The number of samples is another important consideration that effects the sample rate. 100 samples was observed to produce a good sample rate across the distribution. Additional samples increase the space and time complexity of the algorithm. Extra samples were observed to have no measurable impact on the posterior density. Applying different transformations is discussed in Section 5.4.2 Following the transformation a new set of samples \( \{ s_{t}^{(1)}, ..., s_{t}^{N} \} \) have been generated, independent of the old sample set.

- **Evaluate:** The evaluation step generates a weight \( \pi_{t}^{(i)} \) for each sample \( s_{t}^{(i)} \) from an observational density \( p(Z_{t}|X_{t}) \). This is achieved using the Fitness function \( f \):

\[
\pi_{t}^{(i)} = f(s_{t}^{(i)}, Z_{t})
\]  (5.14)
Where $Z_t$ is the player evidence obtained from frame $t$. Given a state and the player measurements from the image processing phase, the fitness function computes a value for how well the state describes the evidence. The fitness function is described in Section 5.4.4. The samples and corresponding weights form the required representation of state density for time $t$.

- **Update**: A tracked objects’ state can be updated from the sample set $\{s^{(1)}_t, ..., s^N_t\}$ and weightings $\{\pi^{(i)}_t | i = 1, ..., N\}$. This system uses the highest weighted sample as a predicted state. A weighted mean could be computed however this can lead to inaccuracies following an occlusion. If samples are clustered around two player states in the feature space a weighted mean will move the predicted state toward the mid point of the two player states. The correct state should be within one of the clusters not the average. The fitness function described below should correctly weight the particles such that the highest weightings correspond to the correct player.

### 5.4.2 State Transformations

A first approximation to the state transformation applied white Gaussian noise to each component of the feature space. The random noise is scaled by a constant to vary the covariance. Empirical studies found players’ centroids varied by a maximum of approximately 15 pixels between frames and variances were greater in the $x$ direction of the image reference frame. Scale values were chosen such that samples were transformed into this region, and spread across the whole region with high probability. Values 7 and 5 for the $x$ and $y$ direction respectively, were found to produce the desired distribution. The Gaussian noise models small movements being more likely than larger movements. Figure 5.5(a) demonstrates a typical distribution of 100 samples, sampled from the prior density function for the previous frame then transformed using a white Gaussian noise transformation. The $(x, y)$ co-ordinates from the sample states are plotted on the image frame for visualisation purposes. This achieves a reasonable prediction as all the new player positions are within the distribution of samples. This is achieved without the variance being too large. A variance is too large if samples become closer to other players (causing an occlusion in the feature space). Without a sufficient fitness function this could lead to a tracker associating to the wrong player. If another player centroid falls within the covariance of this distribution it is highly likely the two players will occlude in the image before there is a chance for an occlusion in the feature space.

Small player movements are more likely than large player movements. To obtain the largest
weighting with high probability, a sample should perform a small transformation with higher probability. This is achieved using white Gaussian noise however the previous sample is with respect to the previous reference frame. In Figure 5.5(a) although the players in blue circles have not moved the camera motion has caused only outlier samples to represent good predictive states. There is a low sample rate at the player region. Figure 5.5(b) demonstrates applying a transformation to model camera movement prior to the random drift phase. This centres the distribution around the location of a non moving player, generating a higher sample rate at the player regions. This increases the number of samples with high weightings for the next iteration. This represents more hypotheses being propagated forward. Too few samples at a player region throttles the number of hypotheses, reducing the effect of condensation. If the number of weighted samples is throttled, the random drift phase ensures new hypotheses are initialised at the next step.

Kalman Filter as Transformation

Applying the camera transformation to compensate for camera movement maximised the sample density where a stationary player would be located. A random drift is used to model a prediction of players’ movements. Using white Gaussian noise represents small movements having a higher probability than large movements. This section experiments with using a motion model to provide a better density at player locations than white Gaussian noise.

The motion model, as used in the Kalman Filter, predicts a player location using the first order motion equations. The feature space is extended to include velocity and acceleration values for each direction. The motion model transformation $A$ (as in the Kalman filter)
is applied to samples before the random drift. This should maximise samples in the most likely player region. The random drift models non linear player movements such as abrupt changes of pace and direction. This was not possible in the single prediction kalman model.

In a kalman filter the predicted state is updated by ‘mixing’ it with the player measurements. This updates the velocity and acceleration values such that the transformation converges to the actual transformation (under the first order motion hypothesis). In the condensation model, predicted states are not updated from the evidence. Player evidence is used to assign a weighting to each prediction. Bad predictions achieve low weightings, these are less likely to be sampled in the next iteration. This phases out bad samples over time. Without updating the velocity and acceleration state from player measurements, an accurate motion model is never achieved. This causes a large variance in the transformation and results in lower density for the most likely player region.

It is possible to break with the Condensation model and refine the velocity and acceleration states from the player measurements, as in a kalman filter. Player evidence is now used twice, in the prediction step and the fitness function. Each sample \( s_t^{(i)} \) maintains a confidence \( P_t^{(i)} \) as in a Kalman filter. At each iteration \( t \) the transformation is computed from the previous state \( s_{t-1}^{(i')} \) (where \( i' \) is the index of the sample \( s_t^{(i)} \) was sampled from) and the player evidence \( Z_t \) by the Kalman update equation (Equation 5.4). The new confidence, \( P_t^{(i)} \), is computed by Equation 5.7. This method predicts a players’ location and refines the velocity and acceleration. This enables a better motion model to be constructed. Following this transformation, a random drift is applied to break apart samples and model non first order motion.

When applied to tracking, the small transformations and the levels of random drift required to model non first order motion reduce the effect. No improvement on the density of samples at player regions was realised. This method also exposes the transformation phase to some of the inaccuracies in a Kalman filter. The disappointing improvements in sample rate, additional complexity and additional risk involved with this method justify the decision to use a basic prediction based on a camera transformation and a random drift.

### 5.4.3 Multiple Player Tracking

When using a Kalman filter, multiple players were tracked using multiple instances of the filter. No information was shared between these instances. In some cases, this led to multiple trackers associating with the same player following occlusions. Dearden et al [8] report similar issues when tracking players using multiple instances of the Condensation
tracker.

This issue is addressed by building multiple object tracking into the algorithm rather than running multiple, single object trackers. A set of states and associated weightings are maintained for each object tracked. Each step of the algorithm is applied to each set of samples in turn. Following the predict step, the new set of samples for each tracker is added to the evidence $z$. This ensures knowledge of all the trackers’ states is available to the fitness function. Multiple trackers associating with the same player can be avoided by assigning low weightings to states which occlude with other trackers in the feature space. A mechanism for achieving this is given in Section 5.4.4.

5.4.4 Fitness Function

Given the player evidence from Chapter 4 and a predicted player state, the fitness function must define a measure of how well the state describes the evidence. This should provide a scale of ‘player-ness’ rather than a binary classification. This function should be tolerant to noisy measurements, in particular player objects containing more than one player. Following an occlusion, a trackers’ sample states will cover all the resolved players. This is the result multiple hypotheses. The fitness function must ensure only the correct player region receives high weightings. Samples corresponding to incorrect player states should receive low weightings such that the predictions converge to the correct player region over time.

Given state $X = [x \ y]^T$ a basic measure of how well this co-ordinate describes a player can be computed by defining a window around the co-ordinate in the image frame and counting the number of player pixels within the window. This is a good measure as points in and around (within a window of) a player will get weighted and co-ordinates in the centre of a player region will achieve the highest weightings. A window size of 15x21 was observed to give the desired results. This is large enough that only the centre of player regions achieve the top weight. Larger windows increase the number of weighted samples, causing larger distributions. Larger windows also cause more outlier samples to get weightings from other player regions. Figure 5.6(a) demonstrates high and low weighted samples using this method. This is a good measure of how well a state fits the hypothesis that it is a player. In particular, unlike Kalman and Nearest Neighbour there is no dependancy on a centroid measurement which has been shown to be inaccurate in cases of occlusion.

When two players occlude, the measurements for either player would provide equal weighting to samples. Incorrect player regions providing a high weighting will prevent sample states converging around a single player state following an occlusion and may cause the
tracker to associate with the wrong player. To prevent the incorrect player region providing a sample with a high weighting, a mechanism to distinguish the tracked player is required.

A high proportion of occlusions are between players on opposing teams, caused by tackles and close marking. Team data is added to the feature space. This is a discrete number, representing the players’ team. All samples in a tracker are given the same classification. As a team classification is constant, no transformation is applied during the prediction step. Section 4.2.4 describes a method for team classification. This method can be applied to the evidence in the states’ window. If the team classification for the evidence does not match the team classification in the samples’ state, a penalty term is applied to the samples’ weighting. The weighting is scaled down with respect to confidence in the classification. This ensures samples that provided a good match to the wrong team receive a low weighting. If the classification was inconclusive the sample still achieves some weighting. Figure 5.6(b) demonstrates samples with wrong team classification receiving lower weightings to resolve occlusions.

This method will not prevent an incorrect player on the same team providing a high weighting. A secondary objective specifies the system should identify individual players. The current system does not achieve this functionality. Should this be developed in future work, it could be used to negatively weight an incorrect player. This would be similar to the team classification technique above. If both players are tracked, the technique to resolve occlusions between two trackers (below) will help resolve the occlusion.

In addition to the player information from the image sequence, the fitness function also has access to each trackers’ state information. From this, the centre of each trackers’ state distribution in the feature space is computed as an approximation to the trackers’ state. If two trackers’ state distributions overlap, it may result in the two trackers converging to the same player. To avoid this, samples closer to another tracker (computed as the Euclidean distance to the centre of each trackers’ distribution) are given a zero weighting.

Figure 5.6: (a) Fitness function applying high weightings to samples with high number of player pixels in surrounding window. (b) Using team classification to impose penalty term on samples. (c) Applying penalty term to samples closest to the wrong tracker, preventing occlusions.
This ensures all trackers remain un-occluded in the feature space. This prevents the issue of trackers following the same player, described by Dearden et al [8]. Figure 5.6(c) demonstrates samples closer to another tracker receiving a negative weighting, ensuring a degree of separation is maintained.

If a player measurement is missing, perhaps due to recall error in player classification, it is possible all samples will receive a zero weighting. In this case all samples are given an equal weighting in the prior density function.

5.4.5 Initialising and Terminating Tracks

The requirements specify the system should terminate tracks when players leave the scene and instantiate new tracks when a player enters. If the predicted state for a tracker falls outside the frame region for two consecutive frames a player is considered to have left the pitch. Two frames is specified in case a player rejoins the scene in the next frame. Increasing the cut-off had little effect on re-identifying players. The missing player causes samples to get equal weightings and the distribution expands in search of evidence. This increases the chance of the track associating with an incorrect player and reduces the probability of the samples identifying the re-emerging player.

Player regions are identified by the player recognition system. At the end of a Condensation iteration it is possible to identify all the player regions extracted from the frame that are not explained by any state across all the trackers. These are considered untracked players. Not restricting this search to players at the edge of the screen allows tracking to be automatically reinitialised if tracking on a player is lost. New trackers can be initialised by the same process as the original trackers.

The major problem with this method is imprecise player classifications. These cause non-player regions to be classified as players. Although it has been shown these errors do not have much impact on the accuracy in tracking they lead to the creation of many unwanted trackers. In general, such misclassifications only appear for one or two frames and do not persist through the video sequence. New trackers are initialised on all new player regions. Trackers initialised on misclassified non-player regions, will lose their evidence once the region is correctly classified. This is represented in the tracker by all samples receiving a zero weighting. If a new tracker has a weighting of all zeros within the first 4 frames it is assumed this tracker was initialised on a misclassification and the track is killed. If this was a valid initialisation with the zeros caused by a recall error in player classification, a new tracker will be initialised on the player region in the next frame. If erroneous trackers
are not killed sufficiently quickly they will attach to player regions, creating unwanted tracks and loss in accuracy. New tracks are not initialised within a radius of player regions to prevent misclassifications such as the ball or fragmented player regions initialising and immediately associating with the player objects.

5.4.6 Evaluation and Discussion

Figure 5.7 demonstrates the accuracy of a Condensation tracker applied to a three video sequences under different conditions. The results demonstrate the accuracy of this method. The higher accuracy in the basic sequence is a result of the player pixel count providing a more accurate measure of a players’ centre than region centroid measurements used in both Nearest Neighbour and Kalman. There is still a degree of error due to the sampling and variation in the most dense player region. The most dense region varies most during occlusions, resulting in the occasional small increases in error seen in the occluded plot on Figure 5.7. The large error peak in the non-occluded sequence corresponds to a recall error in player classification. Here there is no evidence to support any of the prediction states. They are all assigned an equal weighting causing inaccuracy. The Figure demonstrates the systems’ ability to recover from such an error, immediately returning to high levels of accuracy.

Condensation is more reliable for accurately tracking players through occlusion. Team classification and tracking data are utilised successfully to distinguish between players in the fitness function. Unlike the Kalman filter, which slowly converges to the noisy measurements, high accuracy is maintained throughout the occlusion, resulting in the correct players being tracked when the occlusion is resolved.

Figure 5.8 demonstrates the accuracy of Nearest Neighbour, Kalman Filter and Condensation applied to a video sequence containing a number of occlusions, a recall error in player classification and motion not obeying a first order model. This demonstrates how Condensation achieves higher accuracy and better occlusion resolution than both Nearest Neighbour and Kalman Filters. Condensation is the most accurate and reliable tracking system of those tested. Chapter 6 presents the results from comprehensive testing of the full system using the Condensation algorithm for tracking.
CHAPTER 5. TRACKING, CAMERA MODELS AND BEHAVIOUR ANALYSIS

Figure 5.7: Graph demonstrating the accuracy of the condensation algorithm through three video sequences with varying conditions.

5.5 Behaviour Analysis

Behaviour Analysis aims to ‘explain’ the raw player data, extracting useful information and statistics from the players movements. There is a vast amount of information that can be obtained from analysis of the player data. Such data provides useful statistics for broadcasters and coaches.

A secondary objective of the system is to provide some basic behaviour analysis. Two basic behaviours are measured from the raw track data. Section 7.2.2 provides discussion on future work to improve this analysis. It identifies additional behaviours which could be analysed and extensions to enable more information to be extracted.

5.5.1 Attacking Team

This section aims to derive which team is attacking and provide a measure for the pace of the attack. In football, the team with the ball are said to be attacking if they are progressing up the pitch toward the opponents goal.

If a team is attacking then a number of their players will be moving toward the opponents goal. Each teams’ attacking direction is input by the user. The $x$ direction basis vector (in the image reference frame) runs parallel to the field of play. A players direction (away or toward own goal) is given by the sign of the players velocity in the $x$ direction. The absolute value of a players velocity provides a measure of how ‘attacking’ a player is. An attacking value for a team $i$ can be computed as:
Figure 5.8: Graph demonstrating the comparative accuracy of the three tracking algorithms. Discontinuities represent loss of tracking.

\[
attacking^{(i)} = \frac{1}{n^{(i)}} \sum_{i=1}^{n^{(i)}} \left| \text{vel}_{\text{attack}}^{(i)} \right| \text{players}^{(i)}
\]  

(5.15)

where \( \text{vel}_{\text{attack}}^{(i)} \) is the velocities of those players in team \( i \) travelling in the direction of the opponents goal, \( n^{(i)} \) is the number of players in \( \text{vel}_{\text{attack}}^{(i)} \) and \( \text{players}^{(i)} \) is the number of team \( i \) players in the frame. Dividing by the number of team \( i \) players prevents a single player running against the direction of play creating an unrealistic attacking result. The team with the highest attacking value is considered to be attacking. The magnitude of this attacking value gives a measure for the pace of the attack.

Applied to footage containing different levels of attack, from slow build up to fast counter attacks this method performed well at correctly identifying the attacking team. The attacking metric was observed to vary from 2/3 to 14/15 pixels per frame for very fast counter attacks.

Misclassifications were observed predominantly during slow build up plays. If the defending team are moving out to meet the attackers faster than the attackers are progressing this can lead to misclassification. Low attacking metrics (approximately 2/3 pixels per frame) were always associated with such misclassifications. It is arguable whether such slow progression constitutes an attack. These misclassifications could be prevented by creating a third state ‘no attack’ or ‘build up play’. All attacks less than a threshold, approximately 4 pixels per frame, can be classified as ‘build up play’.

CHAPTER 5. TRACKING, CAMERA MODELS AND BEHAVIOUR ANALYSIS

5.5.2 Unmarked Players

This section aims to identify which attacking players are being marked by members of the defending team and which players are free to receive the ball. An attacking player is said to be marked if there is a defending player within a fixed radius. The radius size is a parameter to the system. Player co-ordinates and team classifications are available from the tracker output.

Player co-ordinates represent the player centroid with respect to the image reference frame. Ideally the marked radius should be considered with respect to a pitch plane reference frame. A transformation is required to map image reference co-ordinates to the pitch plane. The current system does not compute such a transformation. Section 7.3.3 discusses finding camera to pitch transformations and accurately resolving players’ pitch positions. In this investigation, marking measurements are considered in the image reference frame. Converting the system to work in the pitch frame is a trivial adaptation should a camera frame to pitch plane transformation be computed in future work.

The attacking team is identified by the process described in Section 5.5.1. For each player in the attacking team, the distance from each non-attacking team player is computed. If any distance is less than the radius then the player is classed as ‘marked’. Otherwise the player is classed ‘free’. This is visualised by plotting the radius on the frame in blue for ‘free’ players and red for ‘marked’ players. Figure 5.9 demonstrates the attacking team identification and subsequent ‘marked player’ analysis applied to a sequence in which the red team are performing a rapid counter-attack.

The results of applying behaviour analysis to different scenes is evaluated in Chapter 6.
5.6 Scene Detection

Although a high proportion of football footage is shot from the wide angled view required for tracking, occasionally close-ups, slow motion replays and off screen shots are added to the video sequence (Section 2.5.3). A secondary requirement of the system specifies tracking should be terminated at scene cuts. If the new scene is also suitable for tracking then trackers may be reinitialised.

Section 2.5.3 discusses the challenges of applying standard scene cut classification methods to football footage. Using the camera motion model to perform classification is investigated.

In general, scene detection works on a two phase principle, score and threshold (Section 2.5.3). Score phase should produce a probability that a scene detection occurs between two frames. Thresholding should identify scene cuts from the scores. For each frame in the sequence, a homography is discovered to perform a transformation from the previous frame to the current frame. In the case of a scene cut, such a transformation does not exist. A simple error check was developed to detect bad transformations. This monitored the vertical movements of the camera (Section 5.1). This provides a basic scoring technique for scene detection. Vertical variance will produce a high score when there is a scene change however bad transformations will also produce high scores. Figure 5.10 demonstrates the vertical camera movement in the camera motion model for a sequence containing a bad transformation and a sequence containing a scene cut. A mechanism is required to distinguish bad transformations from scene changes.

When a transformation for frame $t-1$ to $t$ exceeds the vertical movement threshold, a transformation from frame $i-2$ to frame $i+1$ is sought. If a good transformation cannot be found between these, it is concluded there is a cut after frame $t-1$. This method assumes a bad transformation could be caused by frame $t-1$ or $t$. As the transformation between frames $i-2$ and $i+1$ represents a longer period of time there is scope for more camera movement between these frames. Modelling the larger movement also increases the possible error in the transformation. To compensate the threshold for vertical movement is increased. Accurate classification has been achieved setting this threshold to 130. Figure 5.10 shows a scene change is likely to produce a vertical movement larger than 130. It is possible to extend this method by iterating through frame pairs $(i-n-1), (i+n)$ in search of a valid transformation. It was found that the error in transformations between 2 frames more than 3 frames apart increased the necessary vertical threshold to a variance that could correspond to a scene change.
CHAPTER 5. TRACKING, CAMERA MODELS AND BEHAVIOUR ANALYSIS

Figure 5.10: (a) Computed vertical camera movement per frame in sequence containing a bad transformation. (b) Computed vertical camera movement per frame in a sequence containing a scene change.

Another method may consider the transformation between frame $t-1$ and $t+1$. It is more likely that frame $t$ is the cause of error as frame $t-1$ must have been used to define the previous transformation $t-2$ to $t-1$. Applying this did not achieve any better results. It was found transformations between $i-2$ and $i+1$ were in general more accurate.

Section 7.2.2 discusses how this classifier could be extended and combined with scene classification to extend the functionality of the system.
Chapter 6

Results

This Chapter documents the results of a full system test applied to the whole football tracking system. The test plan aims to cover a wide range of scenarios, varying as many independent variables as possible. The system is tested against a range of broadcast footage taken from a number of games involving different teams playing in different stadiums with different environmental conditions and camera configurations. Sequences contain fast and slow moving footage with different levels of congestion and occlusion.

6.1 Test Plan

Section 3.7 describes the important metrics for computing the accuracy and reliability of a tracking algorithm. Accuracy is measured by computing the distance in pixels between the tracker output and a manually marked ground truth. Reliability measures the extent to which the system remains accurate through a video sequence and across a range of sequences with differing independent variables. Reliability is measured in terms of the average accuracy through a sequence and the number of occasions where tracking was lost.

There are a number of other important metrics that can be obtained from the sequence to measure the success of the tracking system. Calculating the number of tracks initialised can be used to measure the success of automated track initialisation. Section 5.4.5 describes the player initialisation process. A new track is associated to every player object unexplainable by any of the sample states. Some player objects may be misclassified non-player regions. Such objects will initialise tracking, this is subsequently killed during the first 4 frames. Such tracks should be discounted to provide an accurate measurement of track initialisation.
CHAPTER 6. RESULTS

Following the execution and evaluation of the tracking, the success of behaviour analysis is tested. The attack classification is measured against a human evaluated interpretation of who is attacking. Attack sequences are manually ranked by the ‘pace’ of each attack. The ranking is compared to the pace metric to evaluate the accuracy in this metric.

The following independent variables are identified to be tested. Ideally each independent variable should be tested on a video sequence in which the variable being tested is all that varies. It should be ensured that all other factors remain constant. The nature of these variables and the football domain make it very difficult to obtain such footage. Without the resources to generate such footage, video sequences exhibiting the desired properties are carefully selected from broadcast footage. Rerunning these tests on more suitable footage is identified for future work.

Independent variables:

- Camera Angle
- Team Colours
- lighting / weather conditions
- Camera Movement
- Congestion

Lighting / weather conditions and team colours only impact on the player extraction component of the system. If it can be shown that the player identification phase is robust to changes in these variables there will be no impact on tracking. Congestion, camera angle and camera movements may have implications for the system beyond player recognition.

6.2 Full System Test Results

To begin testing, a full system test was applied to a long (200 frame) video sequence. This sequence contains camera movements, high congestion and many players entering and leaving the frame.

Results of the tracking test are given in Figure 6.1(a)&(c). The graph (a) demonstrates the accuracy of 8 players through the sequence. The results show there is a high level of accuracy. The green and black peaks represent two players leaving the scene. Once a player leaves the scene, samples disperse in search of player evidence, resulting in lack of accuracy.
Figure 6.1: (a) Graph demonstrating the accuracy of the full tracking system, applied to a 200 frame video sequence. (b) Team attacking statistics, obtained from the tracking data. Positive values represent red attacking, negative represent yellow attacking. (c) Table of important reliability metrics for tracking and automatic tracker initialisation.
CHAPTER 6. RESULTS

If a player only leaves the field briefly, persisting the trackers for a short period may allow tracking to continue. This is represented in the red peak. A player leaves the frame, there is a loss of accuracy for approximately 3 frames whilst there is no player evidence. When the player returns, tracking continues, immediately returning to high accuracy.

The remaining peaks are caused by losses in tracking and trackers momentarily associating to the wrong player. These are all caused by untracked players. The 4 missed initialisations (Figure 6.1(c)) resulted in four players without associated trackers. In the case of an occlusion or player classification recall error, the untracked region can provide a high weighting, causing an association with the wrong player. This creates a loss of accuracy and in 4 occasions (Figure 6.1(c)), a loss of tracking. If these regions were tracked, the penalty terms in the fitness function would have prevented the incorrect associations.

Four players failed to be tracked as they entered the frame. This represents 55% recall in automatic player initialisation. This is caused by congestion and occlusions. Section 5.4.5 justifies why player tracks are not initialised too close to existing player regions. One player entered the scene too close to a player already being tracked. Tracking was initialised a number of frames later when there was sufficient space for tracker initialisation. The remaining players entered the frame occluded. If two players enter in one player object, only 1 tracker is initialised. Upon occlusion resolution only one player from the occlusion will be tracked until there is sufficient space for additional trackers to be initialised. These results demonstrate how automatic tracker initialisation impacts on the accuracy of tracking.

Precision errors in player classification caused 6 incorrect objects to be tracked. The misclassification check (Section 5.4.5) resulted in only 1 incorrect region being tracked for longer than four frames. This was an unusual misclassification, caused by a balloon on the pitch. Using the track killing system for tracks that lose their evidence within the first 4 frames resulted in 92% precision for automatic tracking initialisation.

Figure 6.1(b) demonstrates behaviour analysis applied to the tracking data obtained. The graph demonstrates the attacking metric plotted for each frame. Positive values represent the red team attacking. Negative values represent yellow team attacking. In the sequence there is no clear attack, initially the red team push forward then the yellow team get the ball. It is hard to classify which team is attacking. This is represented by the small attacking metrics and the number of intersects with the x axis. This is a good example of the no attack state discussed in Section 5.5.1. To avoid noisy, highly fluctuating attacking measurements this should be classified as ‘no attack’. At the end of the sequence there is the beginning of a red attack. This is reflected in the ‘more positive’ values. There is still
CHAPTER 6. RESULTS

a large amount of noise from inaccuracies in tracking and the camera model.

When players are only moving along the pitch a small number of pixels per frame, factors such as errors in the camera motion model and players precise locations become more prevalent. This results in noisy measurements, seen in Figure 6.1(b). If there was more player movement, minor inaccuracies have less impact on the results. The high attack values correspond to occasions where there is inaccuracies in tracking. This shows how the accuracy of behaviour analysis is dependant on accurate tracking data. Marked or free classification only failed on occasions where players were marked by untracked players. Accurate marked player classification is dependant on all players being accurately tracked. Accurate automatic track initialisation is vital to achieve this.

6.3 Camera Movement

This section investigates the effect of camera movement on the accuracy of tracking. Two sequences from the same football match are selected. In the first there is not much ‘action’ (no team is attacking) and the camera position remains constant. In the second sequence the red team are making a fast counter-attack. The camera pans rapidly to follow the action. There is also a variation in zoom through the sequence. A mosaic demonstrating the camera movement and screen shots of the tracking output are given Figure 6.3. The results of applying the tracking system to these sequences are shown in Figure 6.2.

The results in Figure 6.2 show camera movement does not have much impact on the accuracy of tracking. The average error (in pixels from a ground truth) was 4.7 for moving camera and 4.6 for non moving camera. Figure 6.2(c) demonstrates there was no significant difference in the precision recall metrics. Figure 6.2(b) demonstrates the behaviour analysis applied to the moving camera sequence. In this sequence the red team are making a fast attack. This is shown by the negative values. Ranked compared to the other test cases this is the fastest attack. The high values show this is a faster attack than the sequence in Figure 6.1(b).

Using the camera motion model enables accurate tracking to be maintained on a moving camera sequence. This produces accuracy comparable with static camera sequences.
CHAPTER 6. RESULTS

102

Figure 6.2: Results of applying the tracking system to moving and static camera sequences. (a) demonstrates the average accuracy for the two sequences. (b) The attacking metric for the moving camera sequence. The sequence is of red team attacking. This should produce a negative result. (c) Precision Recall metrics for player segmentation and player tracks lost through the sequence.

### 6.4 Congestion

This section investigates how congestion in the video sequence affects the accuracy of the tracking system. Congestion is the density of players in a particular region of the scene. More players in a small area increases the likelihood of occlusions and trackers associating with the wrong player. It is important to test the robustness the system to such events.

To test how congestion affects the system, three short video sequences with increasing congestion levels are used. These are all sourced from close together in the same sequence of play. Figure 6.4 demonstrates the results of applying the system to these sequences. Examples of the congestion and system output are given in Figure 6.5.

These results demonstrate how congestion affects the accuracy and reliability of the system. The average distance from a ground truth is 2.9 pixels for a non congested scene, 5.8 for medium congestion and 5.7 for high congestion. This inaccuracy is introduced by player regions misclassified as non-players in some cases of occlusion. This is demonstrated by the lower recall metrics in Figure 6.4. For an occluded region to be classified as a player the composite shape must form a sufficient approximation to that of a player. In the highly
congested scene occlusions consist of more players. In medium congestion, the maximum number of players in an occlusion was 3. In the highly congested scene up to 6 players may be occluded at one time. The recall results show occlusions of many players are as likely to be classified ‘player’ as occlusions of only 2 players. Although recall error in player classification caused lower accuracy for medium congestion, no players tracks were lost. This demonstrates the systems’ robustness to recall errors.
Despite better recall in the High congestion scene, 3 player tracks were lost. Although there were less recall errors, the missing regions describe the evidence for more players, creating greater uncertainty in the system. This led to one loss of tracking. The fitness function relies on a tracker for each player region. Any untracked player region on the same team will produce a high weighting (Section 5.4.4). When a tracked player occludes with an untracked player on the same team, the tracker may associate to the incorrect player. This caused the subsequent losses in tracking.

The increased congestion in the scene was detected by marked player analysis. On average the relative number of marked players increased through the congestion. The results show that congestion caused a small loss of accuracy. The system is robust through most congestion however occluded player misclassification can cause tracking to be lost in the most congested scenes.

### 6.5 Camera Angle

This section compares results when applying the system to two sequences of the same play, shot from different camera angles. With more resources, sequences from angles between
these two configurations should be tested. This would enable more conclusive investigation into the optimal camera angle. One sequence is taken from a zoomed out camera, mounted at the top of the stands. The second uses a zoomed in camera from a less aerial perspective. The different camera angle impacts on the size of the players and the extent of the pitch visible in the shot. This predominantly affects the image processing and player identification.
modules. There may be indirect effects on other factors such as camera movement as
the zoomed in model must perform more, faster camera movements to follow the action. Figure [6.6] demonstrates the two camera angles.

The highly zoomed out camera (b) results in significantly smaller player regions. Player regions are approximately 15 pixels high, compared with 32 pixels high in camera position (a). This resulted in player regions getting removed by the morphological erosion operator in pitch segmentation. To compensate, the initial erosion was removed and the size of the dilation and erosion were reduced to 3 and 1 respectively. This is a minor modification. A tool in the testsuite helps to identify and test good structuring elements.

Less of the football scene is visible in the zoomed in sequence. This creates less features for the camera motion model to use in computing the homography. Occasionally this creates a bad homography and a scene cut detection. Scene detection was temporarily disabled to ensure the full sequence was processed. Section 7.2.2 discusses improvements to avoid such misclassifications. The previous camera model, as discussed in Section 5.1, was applied in cases of bad transformations. Figure [6.6] demonstrates the camera motion models for both sequences.

Figure [6.7] demonstrates the results of applying the tracking system to the two sequences. Examples of the tracker and behaviour analysis output are given in Figure [6.8]. Although there are more players in the zoomed out scene, players constitute a smaller proportion of the frame, resulting in fewer occlusions. The players’ shapes (following erosion and dilation) are more consistent in the zoomed out sequence. In particular, occlusions between smaller players provide a better approximation to the shape of a single player. Despite this, there is lower recall in the zoomed out sequence due to small players not being distinguished.

**Figure 6.6**: Mosaic images demonstrating the viewpoint and camera movement for the two sequences (a) Zoomed In (b) Zoomed Out. The result demonstrates the greater movement and occasional errors in the zoomed in sequence.
CHAPTER 6. RESULTS

Figure 6.7: Results of applying the tracking system to the high and low camera angle sequences. (a) demonstrates the average accuracy for the two sequences. (b) The attacking metric for the tracked sequence. The sequence is of black team attacking. This should produce a negative result. (c) Precision Recall metrics for player segmentation.

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low, Zoomed In (a)</td>
<td>99%</td>
<td>98%</td>
</tr>
<tr>
<td>High, Zoomed Out (b)</td>
<td>90%</td>
<td>82%</td>
</tr>
</tbody>
</table>

from the pitch. The tracker accuracy (Figure 6.7(a)) demonstrates how the condensation tracker is tolerant to this error, maintaining accurate tracking. The difference between the most dense player region (considered best player position in condensation fitness function) and the actual player centre is on average less for smaller players, particularly in occlusions. This results in the slight difference in accuracy in Figure 6.7(a).

Figure 6.7(b) demonstrates the result of applying behaviour analysis to the two sequences. In the sequence the black team are attacking, this should result in a negative attack metric. Both sequences demonstrate negative values on average. There are a number of erroneous blue attack results. There were less incorrect classifications from the zoomed out sequence.

These results demonstrate the system is robust to different camera angles. High accuracy was obtained through both sequences. Player identification is less accurate on zoomed out footage, but still within the tolerance of the condensation tracker. The zoomed out footage achieves more accurate camera models and less scope for errors between a players computed centre and actual centre.
6.6 Lighting / Weather Conditions

Pitch segmentation is achieved using colour information from the scene in HSV space. Lighting and weather conditions may impact on the effectiveness of this method. Accurate and reliable pitch segmentation is vital for identifying player regions. These form the evidence for tracking. Bad segmentation, from the effect of lighting, may lead to inaccuracies and loss of tracking. As lighting only affects the system through player evidence, it is sufficient to measure the impact on player classification. An ideal test requires the same sequence to be recorded in different lighting and weather conditions. In absence of the suitable resources required to obtain such footage the system has been applied to broadcast footage exhibiting the desired conditions. A match was identified in which the weather conditions change from bright sunlight, with shadows across some pitch regions, to torrential rain. Three similar sequences from this match, displaying sunny, overcast and wet conditions were tested. Every effort has been made to ensure other factors such as levels of occlusion, size of players and camera angle are as consistent as possible.
Figure 6.9: Results of applying the system to three sequences from the same match, displaying different weather and lighting conditions. The red region represents pitch and crowd regions. Black regions represent non pitch regions classified ‘non-player’. Unmasked regions represent player regions.

<table>
<thead>
<tr>
<th>Conditions</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Lighting</td>
<td>86%</td>
<td>95%</td>
</tr>
<tr>
<td>Low light, Torrential Rain</td>
<td>95%</td>
<td>98%</td>
</tr>
<tr>
<td>Bright Sunlight, Shadows</td>
<td>81%</td>
<td>97%</td>
</tr>
</tbody>
</table>

Figure 6.9 shows a frame from each sequence and its segmentation. The precision recall statistics demonstrate the success of the system in each case. The results demonstrate the system is robust to changes in lighting conditions. By rebuilding the pitch model per frame, the system is able to adapt to the change in conditions through the match. Even with the shaded pitch regions (bright sunlight test) the Eigenmodel correctly distinguished pitch pixels from non pitch pixels. The recall statistics were high in all three cases. All recall errors were caused by occlusions. This was mainly the goal keeper occluding with the goal region. In general, precision errors were caused by misclassified line regions. In the wet, low light sequence the lines are less distinguishable from the pitch in HSV space. This results in less lines classified ‘non pitch’, to be misclassified. This creates the higher precision in the wet sequence. Similar regions are misclassified in the other two sequences. The variation in precision is due to fewer non pitch objects in the bright sunlight sequence. This causes each misclassified region to have a greater impact on the precision. Stronger
conclusions could be drawn given the resources to create the more scientific testing data described above.

Precision and recall metrics were high in every test. The system is robust to various lighting conditions.

### 6.7 Team Colours

Player classification is identified as an important intermediate result for the tracking system. Accuracy metrics for player classification (precision and recall) are given explicitly in many of the above examples. Team classification of samples is another important result. If team classification from the evidence does not match the classification of the sample then the sample is subject to a penalty term and a low weighting from the fitness function. Inaccurate team classification may result in tracking errors. Figure 6.10 demonstrates the accuracy and robustness of team classification, applied to three different pairs of teams.

It is important for the system to distinguish between all different football kit combinations. Some teams may have multicoloured shirts or shirts that don’t match the shorts. In some cases the two teams may share a common colour. The 3 sequences in Figure 6.10 test these conditions:

- (a) Team 1: red, Team 2: yellow
- (c) Team 1: red and white stripe, Team 2: blue
- (e) Team 1: White (burgundy shorts), Team 2: blue (white shorts)

The results in Figure 6.10(a)(c)(e) show only one player is misclassified. This is highlighted by a black ring in (e). In this case the player is on the floor after a tackle. The white shorts are more predominant than the blue shirt. As the shorts match the oppositions shirt this has caused a misclassification.

Figure 6.10(b)(d)(f) demonstrate the results of applying team classification to a sample located at every pixel around a player region. As the player region will be sampled, it is important that all pixels around a player region produce the correct classification.

In the case of occlusion between players from opposing teams, it is important that the classification changes where the players meet. There are 3 examples of this in Figure 6.10(b). In Figure 6.10(d) one player only has a small region with the correct classification. This
Figure 6.10: (a)&(c)&(e) Demonstrate team classification of a single sample on each player region. The radius of each semi-circle demonstrates the confidence of a sample belonging to each team. A sample is classified as the greater confidence. This is denoted by a coloured ‘x’ above the sample. The black ring in (e) demonstrates the only misclassification. (b)&(d)&(f) Demonstrate the classification for each pixel in a window around each player.

is due to poor segmentation (caused by the low definition of the players’ outline from camera movement and a low bit-rate video encryption). When tracking was applied to this sequence, the red area was sufficient to get sampled. Figure 6.10(f) demonstrates there is a region that will get the correct classification for the player misclassified in (e).

Figure 6.10 demonstrates the effectiveness of team classification. The change in classi-
CHAPTER 6. RESULTS

fication where two occluded players meet (as in Figure 6.10(b)) demonstrates how team classification is used by the fitness function to resolve occlusion. All results were obtained using a learning sample of 6 players from each team, built from different frames to those tested.

6.8 Additional Applications

Developing a tracker that could have application beyond the football domain was specified as a secondary objective. This section investigates applying the system to other tracking applications. The system is applied unmodified to tracking rugby players on broadcast footage and tracking pedestrians on a precinct.

6.8.1 Tracking Rugby Players

Rugby has many parallels to football. It is filmed from a similar camera perspective, the pitch is grass and opposing teams play in directions parallel to the x basis vector in the camera plane. Teams dress in distinctively coloured shirts and perform movements similar to those in football.

Figure 6.11 demonstrates the results of applying the system to the rugby sequences. Figure 6.11(a) is a sequence of ‘free flowing’ rugby where players are running with the ball. The tracker has successfully tracked 11 players through the sequence. These are below the red line on the figure. Above the red line 9 players are tracked with less success. Error is introduced by the advertising slogan painted on the pitch. Pitch segmentation is based on an assumption that pitch pixels can be represented by a colour model such that non pitch pixels do not associate with this model. This coloured pitch region does not fit the colour based pitch model. The slogan region is identified as a foreground object and classified non player. When the players (above the red line) occlude with this region they are also classified non pitch. This results in loss of accuracy. If a player remains in the region for a number of frames tracking is lost. A GMM pitch model was tested to incorporate the advertising region in the pitch colour model. The comparatively small number of pixels in this region fail to generate sufficient representation in the model for accurate segmentation. Unlike the green pitch, colours in this region also appear in player regions. Any colour based model that represents the advertising region must result in misclassified player pixels. It is challenging for a human to identify a player on the advertising region from a single frame. A more sophisticated pitch model using more information than colour from a single
Figure 6.11: (a) (b) Results of applying system to two sequences from broadcast footage of rugby. (c)&(d) A frame and player identification, demonstrating challenges of adapting system to rugby. The scrum of players is classified non player. The advertising region is not segmented as pitch. Not all line information is identified and segmented by the hough transform.

frame is needed to accurately segment the advertising region whilst leaving players intact.

In addition to tracking players, Figure 6.11 shows 4 line regions have been tracked. The rugby pitch contains more line information than a football pitch and lines are not painted as distinctively on the pitch. Line identification must be improved to filter this line information. Additional image processing should be performed to emphasise line information prior to computing the hough transform.

Figure 6.11(b) demonstrates the system applied to a second video sequence. The ball has been kicked forward and 7 players are chasing it. No players occlude with pitch advertisements in this sequence. The system achieves better accuracy and reliability. Accuracy is comparable to that in football tracking. Useful information such as relative players speeds can be obtained from these short sequences even if the system cannot be applied to a whole match. Behaviour analysis applied to this sequence correctly identified that the blue team are ‘attacking’. In rugby, the notion of attacking team is, in general, considered to be the team with the ball. As the ball must be passed back there is less correspondence between the overall movement of players toward either end and the attacking team. This metric does still provide useful information. It is a measure of which team is pushing forward or ‘closing down the opponent’, the most.
CHAPTER 6. RESULTS

These examples considered ‘free flowing’ rugby, where players are running with the ball. Much of a rugby game involves a large number of players congregating in small areas. The system experienced less success during these sequences. It is very difficult to resolve players during congested scrums, mauls and rucks. This is reflected in the broadcasts, the footage will cut to a close up, ‘pitch side’ camera to cover these sequences. Figure 6.11(c) demonstrates a frame from a scrum. The dense mass of players is misclassified as a non player region. The large number of players involved generates high uncertainty across the system and tracking is lost. Often players may be completely occluded, at the bottom of a pile of players for extended periods, 30 seconds or more. In football, players remain totally occluded for only a small number of frames. More advanced tracking and player recognition techniques would be required to accurately track players through these sequences. Future work could consider multiple camera techniques to resolve the occlusions caused by the high congestion.

This section demonstrates the system is applicable to other sports footage. When players can be tracked, the accuracy is comparable to that of football. The pitch advertising and highly congested scenes add additional complexity to the original problem. Further work is needed to make the system reliable.

6.8.2 Tracking Pedestrians

The system is applied to two video sequences of pedestrians walking around the university campus from different perspectives. Footage was shot with a standard video camera from an elevated position, similar to that from the spotter cam in football. The camera is static however it is subject to a small amount of ‘jitter’.

Figure 6.12: Output tracks from applying the system to video sequences of pedestrians walking on a precinct.
Figure 6.12 demonstrates the output from applying the tracking system to these scenes. Some promising results have been produced. The system is generic enough to track objects through this very different tracking application.

Unlike the rugby domain, there are some major differences between tracking pedestrians on a precinct and tracking football players. To compensate the tracker was configured with different settings to those used in football and rugby.

- **Background:** Figure 6.13(a) demonstrates the results of using the background subtraction technique (Section 4.1.1) to identify a suitable training set of background pixels. Non masked pixels represent those which are used to train the background model. This set still contains most the building pixels. In the football scene, non pitch background (crowd) was very noisy and contained movement. This filtered non pitch background from the model. As the non pavement background in this scene is static, these pixels are classified background. This produces a poor model. To compensate, a sample of pavement is input to the system. Future work could investigate extending the system to automatically bootstrap the background model in this domain.

  It was shown (Section 4.1.2) the football pitch can be approximated to a Gaussian distribution in HSV space. In this scene the background is bath-stone paving slabs. Figure 6.13(b) demonstrates this distribution is clearly multi-modal. Figure 6.13(c) demonstrates a segmentation using an Eigenmodel. Clearly the Eigenmodel is insufficient for representing this distribution. Figure 6.13(d) demonstrates segmentation using a Gaussian Mixture Model of the background. A Gaussian mixture model is required for accurate segmentation. Figure 6.13(f) & (h) demonstrate two more frames segmented using a Gaussian mixture model.

- **Non Tracking Regions:** In football tracking, the pitch region is identified to reduce the trackers’ operating space. The pitch forms a convex shape from all camera angles. A convex hull of the grass pixels is computed to segment non pitch regions. Similarly, the pedestrian scene contains non-pavement background regions such as buildings and fences. Figure 6.13(e) demonstrates using a convex hull to identify the pavement region. This has correctly segmented the buildings at the top of the frame however the small railing region (middle left) has not been removed. In pedestrian scenes, the tracking region is not a consistent shape (unlike the football pitch) and shapes are not necessarily convex. This is not a major concern, no tracking regions can be lost using a convex hull for segmentation. By definition, the convex hull must contain
CHAPTER 6. RESULTS

the whole pavement region. Future work could improve segmentation by applying a better shape fitting method. This should segment more non pavement background where the pavement region is not convex.

- **Line Information**: There is no line information in the pedestrian scenes. Line detection was disabled to avoid erroneous line detections.

- **Team Kits**: The football tracker relies on the assumption that all tracked objects wear coloured shirts representing their team. This information is used by the condensation fitness function to resolve occlusions. This assumption does not hold in pedestrian tracking. Team classification is disabled and removed from the tracking feature space. The fitness function must resolve occlusions based on the temporal tracker information. This reduces the reliability of tracking through occlusion. In figure 6.12(b) the 4 occluded tracks remain distinct however there is visibly greater inaccuracy than in the football domain. When the 3 figures in Figure 6.12(c) occlude, tracking is lost.

- **Larger tracking objects**: The pedestrians in the second sequence (Figure 6.12(c)) are approximately 50% bigger than average football players in broadcast footage. Figure 6.13(g) demonstrates fragmentation in the larger player regions. To compensate for this, the size of the morphological operators, applied to the thresholded background segmentation, are increased. A dilation of 9 and erosion of 6 were found to reduce fragmentation.

All other components of the system are applied with the same configuration as in football tracking. Player classification, camera motion, fitness weightings (except team penalty term) and sample transformations remain unchanged.

The results show the larger objects are subject to greater inaccuracy. This is due to variation in the players’ most dense point. The perspective view in Figure 6.12(a) demonstrates how the noise in the tracks increases as the people get larger. As the pedestrians move slower than footballers inaccuracy in the plotted track is exaggerated. Larger objects also increase the probability of two trackers associating to the same player object. Figure 6.12(c) demonstrates two trackers associated with the same pedestrian. Combined with no team information this reduces the reliability of tracking through occlusion. In highly congested scenes such as Figure 6.13(f) temporal tracker information is insufficient to accurately resolve occlusions. Many occluded pedestrians can still be distinguished by colour. An extension for future work could investigate learning a pedestrian’s colour distribution and adding this information to the tracking feature space.
In football, a player leaves the scene when they leave the frame. Tracks are terminated once a players’ predicted position is outside the frame window. In Figure 6.12(c) the tracking space does not extend to the edge of the frame. A pedestrian may leave the scene whilst the predicted position is still within the frame window. A trivial extension for future work could extend the track termination to terminate tracks when predicted player position falls outside the tracking space (the convex hull).

These examples demonstrate the feasibility of applying the system to tracking problems beyond football. Some extensions and minor modifications are required to improve reliability however this section provides evidence that the concepts and techniques used are extensible and not specific to the football domain.
Chapter 7

Conclusions & Future Work

This Chapter evaluates the results of Chapter 6 and considers how the system compares to the original objectives outlined in Chapter 3. Future work to improve the results and extend functionality are discussed.

7.1 Evaluation of results

This section evaluates the success of the system from the results in Chapter 6. Improvements that could increase accuracy in future work are identified. Improvements to the algorithms and other computer vision methodologies that could be applied are considered.

Before the results are evaluated, a conclusion is drawn on the results gathering process. The results were achieved by testing carefully selected broadcast footage which exhibited the property being tested (i.e. high congestion or poor lighting). Care was taken to ensure other factors (such as player size and camera angles) remained constant. Using clips from the same match or from similarly coloured teams helped to achieve this. To produce more scientific results and draw stronger conclusions the above tests should be run on more consistent footage. The experimenter needs greater control than is available using broadcast footage to ensure variables not being tested remain constant. The tests were designed such that given the resources to obtain better footage the tests can simply be re-run.
CHAPTER 7. CONCLUSIONS & FUTURE WORK

7.1.1 Condensation

The results show the success of the condensation algorithm for tracking football players. It was shown that this algorithm produces more accurate and reliable results than nearest neighbour tracking and Kalman filters.

Accuracy

The greater accuracy is a result of how the fitness function measures a player's position. Condensation samples a player region and chooses the sample with the most player pixels in a surrounding window as the player's position. This is the sample with the highest player density. Nearest neighbour and Kalman rely on a noisy, player region centroid measurement. This centroid position may move around the player region dependant on a player's shape (in particular occlusion). The results show condensation produces a more consistent prediction of player locations.

Although density measures are more accurate than region centroids, this measurement is still subject to a degree of noise. The results show this was an average of approximately 4 pixels per frame. This is due to variation in a player's most dense point (Figure 7.1(b)). This could be improved by applying an algorithm (such as a Kalman Filter) to the outputted player track to perform a smoothing. This would produce higher accuracy by averaging out small, noisy fluctuations. Weighting samples based on player density is a good measure as it ensures sampling is centred around the 'most player' region. This increases the probability of producing a sample with a high player weighting at the next time step. Predicted player measurements need to identify a consistent point on a player region not the 'most player' region. Future work could investigate using more information than just sample weighting to predict player locations. One such investigation could attempt to mix the sample weights with an edge detection measurement to identify a consistent point on a player where shirt meets shorts.

One case when accuracy is poor occurs when a player leaves the scene. Examples of this can be found in Section 6.2.2. Tracking is persisted for 2 frames once a player has left the scene. This may enable tracking to resume if the player only leaves the sequence temporarily. As there is no player evidence once the player has left the scene a prediction is chosen at random. This is not a good approximation to the players location. If the player does not rejoin the scene the final predictions could be removed as the player was already off the scene. If a player rejoins the scene, a better prediction could be obtained for the missing
frames using interpolation. This would make a trivial extension for future work.

Reliability

It is hard to identify any tracking errors in testing that were a direct result of a failing in the condensation tracking algorithm. The fitness function resolves occlusions between players on opposing teams and prevents trackers occluding. Incorporating these features and maintaining multiple hypothesis (through multiple samples) provided a more robust tracking system than nearest neighbour and Kalman Filters. Noisy measurements, such as occlusions, were shown to cause unreliability in Kalman and nearest neighbour. The results in Chapter 6 demonstrate condensation is able to maintain tracking through noisy measurements and occlusion. The results (Section 5.4.6) demonstrate the condensation tracker can maintain accurate tracking through prolonged noisy measurements. Using Kalman filters the state slowly converges to that of the noisy measurements over time.

All tracking failures in the results are caused by erroneous player classification and untracked players. There is not a direct, one to one relationship between tracking errors and misclassification/untracked player errors. Condensation is tolerant to small degrees of error. This is seen in the medium congestion testing (Section 6.4). There are many misclassified players (87% recall) but tolerance in condensation prevents loss of tracking. In particular, tracking is maintained through player recall errors involving one or two players, provided they are not misclassified for more than 2 consecutive frames. There is a small loss of accuracy but high accuracy is resumed once the error has been resolved. When error in player initialisation occurs, high accuracy tracking can be maintained until an untracked player region occludes with a tracked player region on the same team.

Greatest improvements in the reliability of the condensation tracking would be achieved by improving the reliability of player track initialisation and reducing player misclassifications. Condensation was shown to be very reliable when these errors did not occur. Alternatively, tracking could be improved by reducing the dependancy on all players being tracked. This could be achieved by not using temporal tracker information to resolve occlusions between players on the same team. Should player identification be developed, Section 5.4.4 outlines a method for resolving occlusions using player identification. This could improve reliability by reducing the dependancy on all players being tracked.
7.1.2 Track Visualisation

The results show the condensation algorithm is robust in cases of occlusions and misclassified player regions. In these cases there is a small loss of accuracy then high accuracy is resumed. Figure 7.1(d) demonstrates an extreme case where a player is misclassified in 4 consecutive frames. Following the error, high accuracy is resumed. The error has a severe effect on the visualisation output. Figure 7.1(b) demonstrates how a small inaccuracy in player location measurements can affect the visualisation. This is exaggerated by stationary or slow moving players. If tracking is lost, a track can ‘jump’ between player regions or associate with an existing player region, as in Figure 7.1(e). This creates erroneous, sometimes erratic, tracks on the visualisation output. Track smoothing, considered in Section 7.1.1 to filter noisy measurements and bad track culling, suggested in Section 7.1.2 will have a great impact in resolving these issues. A trivial extension to resolve bad tracks without any improvements to the system could allow the user to manually select which tracks to visualise in the output.

7.1.3 Player Initialisation

Player initialisation is identified (above) as a principle cause of unreliability in the tracking system. The fitness function in condensation relies on all player regions being tracked. When samples from one tracker get too close to another tracker in the feature space, a penalty term is applied. This is all that differentiates occluded players on the same team. The results demonstrate several occurrences of trackers associating to the wrong player in occlusions between tracked and untracked players. Players may be untracked due to two failings in player identification:
• **Congestion:** Section 5.4.5 describes how player fragmentation prevents new players getting initialised within a small radius of existing player regions. Congestion at the edge of a frame can lead to new players not getting initialised. If this player occludes before there is space to initialise tracking, tracking may be lost.

• **Occlusion:** Only one tracker will get initialised if multiple players enter the scene occluded. This may cause loss of tracking if an untracked player occludes before entering enough space for tracking initialisation.

Both these errors are caused by unsatisfactory player identification. Player initialisation is reduced to the problem of player identification. Player fragmentation and failure to identifying individual players in occlusions prevent ideal player initialisation. This results in the errors observed in testing. If player identification were to be improved in future work this could allow the congestion restriction to be removed and trackers to be initialised for every player in an occlusion. This would resolve both player identification failings.

Without improvements to player classification a 3 pass tracker (Section 7.3.1) could be used to improve results. When a new player region is found, away from the edge of the scene, tracking could be applied in reverse time direction to identify where and when this player entered the scene. A new tracker can be initialised at this point and tracking re-run from here.

The second issue in player identification is tracks initialised on non-player objects. Section 5.4.5 describes a process to terminate tracks where player evidence disappears within the first four frames. This removes tracks initialised on misclassified non player regions. Section 6.2 demonstrated this produced good results. 5 out of 6 incorrect tracks were killed. This issue can be reduced to precision in player classification. Improving precision reduces incorrect non player classifications, reducing the number of incorrect trackers initialised. To improve performance without improvements in player classification, the track termination system, outlined in Section 7.3.2, could remove tracked non player regions that persist more than 4 frames.

Despite these minor issues caused by player classification errors, players entering the scene under normal conditions are identified and initialised correctly. High accuracy tracking is obtained immediately.
7.1.4 Player Identification

Player identification is fundamental as it identifies player regions. These form the evidence for the tracking algorithms. Pitch segmentation, proceeded by line segmentation and non-pitch region segmentation, accurately and reliably locate player connected components. In general, the histogram threshold method (Section 4.1.3) and the morphological operators (Section 4.1.4) successfully remove pitch regions without losing or fragmenting player regions. The main challenge in this section of the system is the size for the structuring elements in the morphological operators. Tests on camera angles (Section 6.5) identified a problematic sequence from a highly zoomed out camera. Erosion elements were too large for small player regions in this sequence. This causes player information to be lost. In the current system, structuring elements are a fixed size. Given this sequence, the user must correct the structuring element sizes manually. There is a tool to aid this process in the testsuite. An improvement for future work could automatically vary the size of structuring elements from scene information. Morphological operators are applied prior to player classification. Structuring element sizes must be set without the player classification for this scene. If camera zoom changes, player sizes may vary through the sequence. Such an automatic operator could be used to vary the structuring elements dynamically through the sequence.

Errors in condensation and player initialisation can be reduced to problems in player classification. Although both systems are robust to small amounts of misclassification, more accurate classification is required to produce better results. The results demonstrate there is a lower recall in congested scenes, caused by occluded players being misclassified as non-players. Where many players were occluded, this caused a large uncertainty in the system and in some cases, a loss of tracking. Section 5.4.2 documents 4 of the feature spaces tested in the search for the best feature space. Future work could investigate finding a better feature space. If a sufficient feature space cannot be found, more properties could be considered. In addition to extra shape properties, introducing region colour properties might help in resolving players from lines.

Figure 7.1(c) demonstrates an occlusion from the low camera angle results (Section 6.5). Here a player has occluded with the advertising boards. Identifying a feature space that would accept this region, whilst rejecting advertising board regions with no player occlusions is unlikely. This demonstrates a fundamental problem with the approach employed to identify players. The player identification system represents a bottom up approach to classification. Pitch segmentation begins at a pixel level. The results are combined into
shapes and the shapes are classified as players. Small noisy measurements (such as the pixels connecting the grass to the advertising board) or occlusions can affect the building process, creating drastically different results. An alternative, top down, solution searches the frame for evidence that fits a player model.

The neural network classifier represents a top down approach to player classification. The preliminary neural network investigations in Section 4.2.3 demonstrate this is a promising approach to accurate and reliable player classification. Applying a more complex neural network and testing different configurations of nodes, layers and transfer functions could produce a more accurate method for player classification. Breen et al [7] demonstrates using a supervised classifying neural network to classify balls in sports footage. Feature vectors are defined for each object to be classified and neural networks are trained for each object. The network is trained from training data for the object, training data for different objects and a noise training set. Such a system could be used to develop the neural network in Section 4.2.3. Using the top-down, search ideology, neural networks could identify individual players in occluded regions. Section 7.1.4 identifies how this would improve the player initialisation process.

### 7.1.5 Team Classifications

Section 4.2.4 & 6.7 demonstrate the success of team classification. The main source of unreliability (shown in Section 6.7) occurs when image segmentation fails to identify enough of the players’ colour information for an accurate classification. This is very rare occurrence. Such an error produces the same effect as a player misclassification. Condensation has been shown to be robust to this error.

To improve this process in future work, a system to automatically bootstrap the team model could be developed. The current system builds a model by considering every player pixel for each team in a HSV feature space. A model is fitted to the distribution. Multi-modal kit distributions prevent plotting all player pixels and using k-means to automatically cluster the distribution into two teams. Comparing the distribution of a player to other players should give a measure of how likely they are to be on the same team. Applying k-means to this data may provide a method for automatically bootstrapping the model.

The current system uses two team models for the two team kits. Another extension could expand this model to classify the referee and goal keepers. These also wear distinctively coloured kits thus the system can be trivially expanded to include referee and goal keeper models. Less training data is available for these kits as there can only be one occurrence
of each per frame. This should not be a problem as a model requires a training set of just 6 players.

7.2 Delivered System vs. Objectives

This section compares the delivered system to the objectives from Chapter 3.

7.2.1 Primary Objective

The principle objective was to build a system capable of accurately and reliably tracking multiple players through a football scene from a monocular viewpoint video sequence. The results show a condensation tracker produces accurate tracking across a range of footage. In cases of occlusion, team and temporal tracker information is used to reliably track players.

7.2.2 Secondary Objectives

- **Quantifiable Metrics.** Using an accuracy measure, based on deviation from a ground truth, nearest neighbour, Kalman and condensation were objectively compared. The success of tracking through different conditions (such as camera angle and congestion) was also measured. Precision/Recall metrics were used to measure the success of classification tasks, enabling direct comparisons between different player classification and thresholding techniques. A vertical displacement measure was used to quantify error in camera motion. This was used to detect bad transformations and scene cuts.

- **Evaluating different approaches.** At each step in the process, different techniques have been tested and evaluated. Section 4.1.3 implements and evaluates four different thresholding techniques. Section 4.2.3 implements and evaluates three different player classification methods. Nearest neighbour, Kalman filter and condensation tracking solutions are implemented and evaluated. Precision/Recall and ground truth measurements are used to quantify performance and evaluate each technique.

- **No user input.** A small amount of user input is required. Player classification requires a set of training data from which to build a player shape model. In general, players shapes do not vary significantly between matches. A model is provided with the system. This is sufficient in most cases. Team classification also needs training
data. Multi-modal shirts prevent an algorithm such as k-means automatically bootstrapping the model from scene information. As every pixel in a player region contributes to the model, only a small training sample (approximately 6 players per team) is required. Training data for the pitch model is automatically bootstrapped, using background subtraction to identify a grass sample. Tools are provided to generate all training data by simply clicking appropriate regions through a training sequence. User input is also required to plot ground truth paths, this is unavoidable.

- **Classifying tracked players into teams.** Section 7.1.3 concludes the success of team classification. Section 6.7 demonstrates the accuracy in this method and it’s great success in resolving occlusion between players on opposing teams.

- **A camera model.** Section 5.1 documents using Harris corners and RANSAC to produce a camera model. This is used in the tracking algorithms and to transform previous track co-ordinates into the current reference frame. Section 6.3 demonstrates how this method results in accuracy on a moving camera comparable to that from a static camera. Occasionally a bad transformation is produced. This can be due to bad non deterministic choices in the RANSAC algorithm or failure to identify a sufficient set of points in the two frames. A simple sanity check based on vertical camera movement provides an accurate method for identifying bad results. In the case of bad transformations the previous transformation provides an approximate camera model. Section 2.5.2 discusses an optical flow system used by Beetz et al [6] when no line information is available. This could be used to produce a better camera model when bad transformations occur.

- **Generalising the system to work for other sports and tracking applications.** Section 6.8 documents applying the tracking system unmodified to rugby and pedestrian applications. Over 50% of objects in both scenes were tracked with accuracy comparable to that in football. Section 6.8 discusses modifications and extensions to achieve reliability comparable to that in football.

In rugby, line identification and pitch segmentation must be improved. Tracking players through highly congested sequences, such as scrums, mauls and rucks is an unrealistic objective. It is not possible to accurately mark players manually from the ‘spotter cam’ position. This is reflected by scene cuts, to close up camera viewpoints in broadcast footage. A multi camera system would help resolve some occlusions in these cases.
The pedestrian scenes represent a rather different tracking domain. Section 6.8.2 outlined how the system must be configured to compensate for these variations. In the pedestrian scene, the absence of team classification reduces the systems’ reliability in cases of occlusion. A dynamic colour model, learnt per object, was proposed as a generalisation of the colour based classification system. A number of smaller modifications were also outlined for future work. In particular, to compensate for non-pavement background in the background model and to better segment the non convex tracking space. This experiment demonstrates the flexibility in the tracking system and provides evidence that the system is applicable beyond the football domain. The results show the system is built with minimal use of football specific hypotheses. This produces a generic, extensible platform from which future work may investigate a range of other tracking applications.

**Scene Detection and Classification.** Section 5.6 developed a system for detecting scene cuts using a scoring system based on the camera motion model. This can be used to terminate tracking at the end of a scene. In general, this method achieved high accuracy in classification. Erroneous classifications were observed in the highly zoomed, feature sparse view from the camera angle test. To resolve this issue scene classification must use more than camera models in its scoring. Classifications could be improved by combining camera information with tracking information. In a scene change, it is likely most of the trackers will lose their player evidence. This can be detected as very low sample weightings. In cases of bad motion models, the previous camera motion model is re-applied and tracking continues undeterred. This information could added to the scoring phase of scene cut detection to improve classifications. Adopting additional techniques, such as SALSA [19], identified in Section 2.5.3 may also improve scene classification.

Not all scenes are appropriate for tracking. Broadcast footage contains many close up shots and off screen shots of the managers and crowd. To reinitialise tracking, the system must distinguish between different types of shot. No methods for achieving this classification were investigated in the current system. Section 2.5.3 outlines a number of possible methods to investigate in future work. A scene classification method would enable the system to be applied to a whole match, without editing to select only those sequences suitable for tracking.

**Behaviour Analysis.** Section 5.5 documents two behaviours extracted from the tracking information. The first behaviour computes which team is attacking, using player movements along the pitch. A metric for how fast a team is attacking is com-
computed from the players’ velocities. Section 6.3 demonstrates this method produces
good results when applied to a scene with a clear attack. Brief pauses in teams’ at-
tacks, defenders coming out to meet attackers and minor inaccuracies in the camera
motion model can cause incorrect classifications. This causes noisy results in which
the attacking team fluctuates. Attacks in football are relatively consistent, the at-
tacking team will change at most every few seconds. Results should not change every
two or three frames. Future work could apply a smoothing filter to compensate for
incorrect classifications. In some sequences, the ball is passed around and neither
team is attacking. An extension could introduce a new state ‘no attack’. Sequences
in which the attacking team fluctuates and attacking metrics are low may be classified
‘no attack’.

The second behaviour measurement distinguishes marked players from unmarked
players. Space is an important concept in football. This metric provides an indica-
tion of good attacking and poor defending. This method was applied in the camera
reference frame. If the camera to pitch transformation (section 7.3.3) is developed,
an extension could improve this model by detecting unmarked players in the ground
plane, a more useful statistic. Errors in this method are exactly those in tracking
and attacking team analysis. Any other improvements would be achieved through
improved tracking or improved attacking team analysis.

There are many different behaviours and statistics that could be obtained through
analysis of the tracking data. These are vital for extracting as much information from
the data as possible. A simple extension could investigate relative player displace-
ments and velocities. This could be used to detect players that are too slow or fail to
react fast enough to opponents movements. Given a camera to pitch transformation
(Section 7.3.3) player metrics such as distance travelled and velocity would provide
very useful statistics for coaches and managers. These would be a trivially computable
given the transformation. If the ball was tracked, metrics such as time on the ball
would also provide useful statistics. More advanced behaviour analysis could analyse
player and team tactics. Tactical models could be defined or learnt from patterns in
the tracking output. Measuring players against the different tactical models could
analyse how a team or individual players conform to the tactical structure. Such
a model is demonstrated by Needham [23]. He uses a competitive, learning neural
network and GMM’s to model player positions, learnt from the video sequence in an
unsupervised manner.
Analysis of tracker output can be used to detect and classify match highlights (for example; ‘red: free kick’, ‘blue: corner’, ‘red: shot on goal’, ‘blue: good save’). This could be used to summarise the action, automatically index the footage, and create semantic tags for fast information retrieval. Corners and free kick information can also be used to calculate match statistics. Bertini et al [2] use Hidden Markov Chains (HMM) to detect such highlights, using camera motion and player positions as observations to the model. As both of these are output from the football tracker such a system could be incorporated in future work. Ekin et al [9] demonstrate a novel approach to classification. Cinematic features of the broadcast footage are exploited to detect events such as goals. To classify a goal, scene detection must identify action replays, player close-ups and crowd shots for at-least 30 seconds. If scene classification is developed, such heuristics could easily complement the HMM method. If the system is to exploit cinematic features of broadcast footage in behaviour analysis, another solution could detect goals and key events by monitoring the on-screen graphics. Techniques, such as OCR, can be used to extract useful information from these.

• Tracking the ball. The current system does not investigate tracking the ball. It is common for the ball to be extracted as a non pitch region. Consistency of this segmentation is dependant on the quality of the cameras and recording. In some games, poor cameras and poor video encryption fail to resolve the ball when it is travelling quickly. There is also a higher frequency of occlusion between the ball and player regions. Often the ball is totally occluded. This adds additional challenge to tracking the ball. Where the ball can be consistently identified, similar techniques to player tracking can be applied. Variance measures should be increased to account for the ball travelling faster. The number of samples may need to be increased to accurately sample this new variance. Figure 7.1(a) demonstrates the current system tracking the ball due to player misclassification. This demonstrates the feasibility of this extension for high quality footage.

• Identifying individual players. This was not investigated in the current system. A method using team classification and OCR is identified in Section 2.3.5. How this method could be incorporated into the condensation fitness function to improve reliability is discussed in Section 5.4.4. This feature would enable the system to relate tracks in different scenes to the same player. This could be used to build a profile of a players’ positions and behaviours over the whole match.
7.3 Extensions to Functionality

This section identifies extensions to the systems’ functionality, not listed in the initial objectives.

7.3.1 Multiple Pass Tracking

The current tracking system iterates through each frame in a single pass. Only information from the preceding frames is utilised. Future work could investigate extending the system to perform multiple iterations through the sequence. This enables the tracker to utilise future temporal information. This could be exploited to improve accuracy and reliability. A simple application of this idea could perform a smoothing operation to player tracks, reducing the error caused by noisy player measurements (as discussed in Section 7.1.1). Another simple application could apply the tracking algorithm forwards then backwards and average the two measurements to produce a more accurate and reliable track. A more complex tracker could use the extra information to improve occlusion resolution, help resolve player tracks (as opposed to tracks initialised on misclassified non players) and improve player initialisation in cases of occlusion and congestion. As condensation performs an iterative sample, predict, evaluate, update process a more advanced tracking algorithm may be required for this spatio-temporal model.

7.3.2 Terminating Bad Tracks

If a tracker loses its player the prediction may associate to another player region or ‘jump’ between small regions of misclassified lines and pitch. This leads to an erroneous track in the output (Section 7.1.2). A system to automatically identify and terminate bad tracks would improve the visual output and increase accuracy. It is important that this system only terminates bad tracks and does not terminate trackers that are successfully tracking players.

In bad cases of player misclassification a player may be misclassified for two or three consecutive frames and condensation can still recover. Absence of evidence alone, (i.e low weightings) cannot be used to identify bad tracks. This was the method used in track termination for incorrect player initialisations. Absence of evidence will generate larger distributions in the feature space as there is a more even weighting across all samples. This distribution may not be Gaussian. Plotting the variance in the distribution over time, we
would expect a steadily increasing variance for a bad track. Player misclassification would cause a spike in variance. Heuristics applied to the variation in sample distribution over time would create a possible method for identifying bad tracks.

### 7.3.3 Camera to Pitch Plane Transformations

Section 5.1 develops a camera model by defining a homography from one camera plane to another. A similar model can be used to build a camera to pitch-plane transformation. To build the homography, at-least four well known pitch points (such as pitch line intersections) must be identified. Selecting points on the pitch ensures they are all on the same plane. Points could be marked manually or identified automatically. In broadcast footage from low, zoomed in cameras, identifying four common pitch points may not be possible. In such scenes, the football pitch has high feature sparsity. It is not necessary to produce a direct camera to pitch transformation for every frame. Composition of linear transformations corresponds to matrix multiplication. The inverse camera model (new reference frame to old reference frame) can be pre-multiplied by the previous camera to pitch transformation to create a new camera to pitch transformation. The main issue foreseen with applying this method is accuracy. The perspective camera view results in low resolution for players on the far side of the pitch. Here 1 pixel can correspond approximately 0.5 meters in real world pitch plane co-ordinates. When a new pitch model is computed from a previous model the errors are accumulated. Even if a new camera to pitch model is computed on a regular basis an average 4 pixel error in tracking could correspond to several meters in real world co-ordinates.

To plot players’ positions on the pitch, the system must also compute a better approximation of a players location. The tracking system tracks the most dense player region. This is good for achieving a high sample rate at the player region. A players’ position on the pitch plane should be the midpoint of the players’ feet. For non occluded players this can be approximated to the midpoint on the base of the bounding box. When players occlude, the base of the bounding box may correspond to another players’ feet. One solution could add the bounding box dimensions to the feature space. When a player occludes an approximation to the players’ feet can be computed from the players’ state. This method is used by Xu and Ellis [33] to produce partial observations through occlusions in their Kalman filter.

There are many more extensions that could be applied to the system. Introducing multiple cameras could improve accuracy and occlusion resolution. Tracking and camera motion
information could be exploited to produce 3D reconstructions and synthesised viewpoints. The running time could be improved by porting the system into a more efficient language, optimising the code and removing visualisation and evaluation routines. The Matlab implementation, running on a standard personal computer, processes at approximately 2 seconds per frame. Based on investigations into removing visualisation code and results from other condensation based systems, this could be vastly improved to yield close to real time performance.

The system provides an extensible basis from which to investigate further extensions into sports footage analysis and other tracking applications.
Bibliography


**URL:** http://www.bbc.co.uk/rd/projects/virtual/piero/


URL: http://www.prozonesports.com/


Appendix A

Project Plan
## APPENDIX A. PROJECT PLAN

<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>Start</th>
<th>Finish</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Project Dissection</td>
<td>Mon 06/10/07</td>
<td>Tue 15/10/07</td>
</tr>
<tr>
<td>2</td>
<td>Background Reading + Lit Rev</td>
<td>Mon 06/10/07</td>
<td>Mon 16/10/07</td>
</tr>
<tr>
<td>3</td>
<td>1st Iteration Simple Background 1st &amp; Nearest Neighbor</td>
<td>Mon 22/10/07</td>
<td>Tue 25/10/07</td>
</tr>
<tr>
<td>4</td>
<td>Implementation of Core Objectives</td>
<td>Tue 25/10/07</td>
<td>Fri 15/12/07</td>
</tr>
<tr>
<td>5</td>
<td>Implementation of Feature 1</td>
<td>Fri 15/12/07</td>
<td>Mon 05/01/08</td>
</tr>
<tr>
<td>6</td>
<td>Implementation of Feature 2</td>
<td>Mon 06/01/08</td>
<td>Fri 21/02/08</td>
</tr>
<tr>
<td>7</td>
<td>Poster</td>
<td>Mon 06/01/08</td>
<td>Mon 10/02/08</td>
</tr>
<tr>
<td>8</td>
<td>Results &amp; Evaluation</td>
<td>Fri 21/02/08</td>
<td>Fri 21/02/08</td>
</tr>
<tr>
<td>9</td>
<td>Wake Up</td>
<td>Fri 21/02/08</td>
<td>Mon 03/04/08</td>
</tr>
</tbody>
</table>